Towards geometric and semantic change detection for mixed reality experiences

Olivier Roupin

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE PAYS DE LA LOIRE –
IMT ATLANTIQUE

ECOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Informatique*

Par
Oliver ROUPIN

Towards geometric and semantic change detection for mixed reality experiences

Thèse présentée et soutenue à l'IMT Atlantique, Brest, le 22/09/2023
Unité de recherche : Lab-STICC
Thèse N° : 2023IMTA0362

Rapporteurs avant soutenance :

Pascal VASSEUR	Professeur, Université de Picardie Jules Verne
Alain PAGANI	Senior Researcher, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Composition du Jury :

Président :	Vincent CHARVILLAT	Professeur, Institut Mines-Télécom
Examinateurs :	Pascal VASSEUR	Professeur, Université de Picardie Jules Verne
	Alain PAGANI	Senior Researcher, DFKI GmbH
Dir. de thèse :	Guillaume MOREAU	Professeur, IMT Atlantique
Encadrants :	Maja KRIVOKUĆA	Principal Scientist Research and Innovation, InterDigital
	Jean-Marie NORMAND	Professeur, Ecole Centrale Nantes

Invité(s)

Matthieu FRADET	Demo & training engineer, Broadpeak

Acknowledgements

I would like to express my gratitude to all my thesis advisors, Matthieu Fradet, Caroline Baillard, Guillaume Moreau, Maja Krivokuća and Jean-Marie Normand who –despite the many outside challenges we faced during those three years– helped me see it through to the end.

I would also like to thank my colleagues for making me feel part of the team even when I was isolated in my work, especially Koki, Sylvain, Rodrigo, Tomás and Gustavo, as well as our musical next-door neighbours.

I am also grateful to Tim and his family for their warm welcome in Rennes and for helping me make the most of my stay here. I extend this gratitude to all my friends who kept in touch during this thesis, most of which are now going through this experience themselves.

Lastly, I am grateful to have such a great family, who all supported me always, and without whom I could not have made it this far.

Abstract

"Mixed Reality" (MR) can be used to describe Augmented Reality (AR) which primarily focuses on achieving photo-realism when augmenting the real world with virtual content. MR is applicable to a variety of immersive experiences, running the gamut from visual entertainment to simulation-based training, and for previsualization tasks with a emphasis on design.The realistic visual integration of virtual elements in a scene requires an accurate and up-to-date model of the real scene's geometry and semantic contents. The 3D structure of the environment impacts rendering tasks such as occlusion detection and physically-based lighting, but also influences the spatial layout of the virtual content and its interactions with the real world. Given the hardware and power constraints of widely available AR devices, it is impractical to perform an extensive geometry scan whenever these devices are used in a known environment. Still, even in a dynamic scene, a lot of structural information (e.g., solid walls) will remain constant over time and can be stored between sessions, only needing local updates to accurately represent the scene's current state.

The aim of this doctoral study has been to provide the means to identify and correct the areas of change through the detection of inconsistencies between a prior representation and an up-to-date view of the scene. In this thesis, we relate our contributions towards achieving this goal. In particular, we present the completion of a lightweight reprojection-based geometric change detection framework, as well as the elaboration of a semantic scene model representing the properties of the scene relevant to the MR experience using a graph structure. Following this, a scene analysis and registration method is implemented that showcases said graph model representation capabilities, and its performance as the basis of a robust localisation framework. The maintenance of an accurate geometric and semantic model of a real scene can ultimately be achieved with limited AR devices using the graph structure to interpret preliminary change detection results as changes in the scene's objects' properties.

Résumé en français

Introduction

En 2016, le regain d'intérêt pour la Réalité Virtuelle (RV) suivant la sortie de l'Oculus Rift et de l'HTC VIVE, a été partagé par la Réalité Augmentée (RA) pour ses capacités d'immersion [40]. Répondant à cette demande, en 2017 Microsoft a sorti le visiocasque Hololens et Apple a publié ARKit, un outil de développement permettant de produire des applications de RA sur leurs appareils. Avec la sortie de ARCore pour Android l'année suivante, une grande variété d'applications de RA grand public ont pu être développées. L'essor du télétravail [156] a également renouvelé l'intérêt pour la RA en tant qu'outil de collaboration et communication [142], étant intrinsèquement moins isolante que la RV. La RA soulève néanmoins des questions concernant l'incrustation réaliste de contenu virtuels dans une scène réelle [50].

Dans cette thèse, on utilisera le terme "Réalité Mixte" (RM) pour désigner les applications de RA cherchant à intégrer des éléments virtuels dans une scène réelle de manière photoréaliste. Le photoréalisme a été cité comme facteur d'immersion, quand les éléments virtuels se comportent également de manière réaliste [77]. En effet, la RM a été utilisée dans des applications d'enseignement et de formation quand des situations réelles sont difficilement procurables [147, 128], ou présentent des dangers ou enjeux trop importants [178]. De manière plus générale, le photoréalisme peut être bénéfique dans toutes les applications RA de prévisualisation ou simulation [1, 114, 66], ou quand il y a des enjeux esthétiques [48, 35, 199].

Les premiers travaux en RM [94, 64] s'attelaient au problème de réduction du temps de calcul pour accomplir trois tâches principales : l'estimation des paramètres de la caméra, la reconstruction de la géométrie de la scène, et l'estimation de son illumination [94, 68]. Contrairement à la RV, où la scène est entièrement modélisée au sein de l'application, une grande portion d'une scène de RA doit être reconstruite à partir d'observations partielles du monde réel. La qualité de cette reconstruction impacte directement le photoréalisme de tâches comme la gestion d'occlusions [206], la profondeur de champ [134], et l'éclairage simulé [107, 1]. Par ailleurs, la modélisation de la scène devra également inclure des informations sémantiques pour garantir un réalisme "comportemental" [77], affectant par exemple le placement des objets virtuels [111], leur

comportement [32], ou la simulation des interactions physiques [24]. La création d'un tel modèle en temps réel serait hors de la portée de la majorité des appareils de RA grand public, et dans une scène dynamique, le temps requis à chaque session de RA pour la scanner exhaustivement serait rédhibitoire.

Pour les applications de RM grand public, la solution est donc de conserver le modèle de la scène entre chaque session afin de pouvoir démarrer l'expérience immédiatement. Cependant, le modèle devra être régulièrement mis à jour afin de rester fidèle à la scène réelle, et trois problèmes seront à résoudre :

- créer une représentation de la scène permettant d'obtenir le niveau de réalisme souhaité;
- détecter les changements entre la scène actuelle et le modèle, à partir d'observations partielles;
- modifier le modèle à partir des changements détectés en maintenant son degré de fidélité.

Le but de cette étude doctorale est donc de trouver une représentation de scène utile et maintenable pour la RM, et un processus à faible coût de calcul [174] pour détecter les changements entre ladite représentation et des observations partielles de la scène réelle.

Il existe des travaux dont le but est de détecter des changements entre deux modèles d'une même scène, et d'agir en conséquence. En général, la nature de ces modèles dépend du domaine d'application de la méthode, car ce dernier impose des contraintes sur les moyens et le délai acceptables pour les produire. Certaines méthodes en navigation autonome [59, 145] et robotique [192, 149, 51, 4, 97, 197] comparent deux représentations 3D de l'environnement pour produire des résultats en 3D. Ces approches nécessitent cependant des capteurs spécifiques (LiDAR, laser) ou une puissance de calcul importante (algorithmes de *shape-from-motion*) pour fournir leurs résultats en temps réel, et sont difficilement applicables à la RM grand public. D'autres domaines d'application évitent ces coûts en utilisant des représentations 2D de la scène. En télédétection [38, 39, 67, 42, 30], des images aériennes sont utilisées et les résultats de la détection de changements sont similairement donnés en 2D au pixel-près. Pour des images produites au sol, notamment dans le domaine des villes intelligentes [71, 198], des approches ont recours aux réseaux de neurones siamois pour être d'avantage robustes aux "pseudo-changements", introduisant ainsi la notion sémantique de pertinence des changements. La plupart des ces résultats ne sont cependant pas exploitables en RM sans avoir recours à une conversion coûteuse de la 2D à la 3D. Dans le contexte de la RM, où la quantité et qualité d'information disponible en temps réel est inférieure à celle contenue dans le modèle de référence, on peut utiliser des méthodes à données d'entrée hétérogènes [188, 151, 136]. Ces méthodes

comparent des observations 2D à un modèle 3D afin d'obtenir directement des résultats sur la géométrie de la scène à moindre coût en termes de matériel ou calcul.

Parmi les travaux de détection de changements avec données hétérogènes, les approches photo-géométriques, qui comparent des images actuelles à un maillage 3D de référence, sont applicables au cas de la RM. Cependant, on remarque que ces méthodes, notamment celle proposée dans [136], sont souvent biaisées vers la détection d'objets apparaissant dans la scène, au détriments de ceux qui disparaissent. Notre première contribution sera donc la généralisation de cette méthode [136] vers une détection plus exhaustive des changements. Cette approche photo-géométrique est basée sur l'étude des erreurs introduites lors de la reprojection des images à différents points de vue à l'aide du maillage désuet. Les résultats 3D donnent une estimation de la forme et la position des objets ayant été introduits ou retirés dans la scène. Sur la base de cette première approche, on conclura que ce choix de représentation ne nous permettra pas d'obtenir une expérience photoréaliste en RA, et que le processus de détection de changements ne permet pas de la mettre à jour avec suffisamment de précision. Notre deuxième contribution sera ainsi l'introduction d'un modèle construit par analyse sémantique de la scène, fournissant une représentation segmentée par objets. On notera également l'évolution des appareils de RA grand public, auxquels auront été ajoutés de nouveaux capteurs 3D [205]. Enfin, notre troisième contribution sera un processus de génération et exploitation de ce modèle, un "graphe de scène", pour la tâche de localisation dans une scène. Nous évaluerons si le graphe généré est fidèle à la sémantique de la scène réelle, afin de montrer que cette qualité du modèle peut être améliorée sans compromettre la précision de la localisation, ou sa robustesse aux changements dans des scènes dynamiques.

Détection de changements photo-géométrique

La méthode décrite dans [136] a été développée pour le domaine de la robotique et fonctionne en temps "interactif". Elle fait l'hypothèse de capteurs limités pendant l'observation de la scène : la détection de changements est faite entre un maillage 3D non-texturé désuet, et 5 images à jour, localisées relativement au maillage. A l'aide de cette localisation, on peut projeter les images d'un point de vue à un autre en utilisant la géométrie du maillage. Cependant, si des changements ont eu lieu, la reprojection aura des erreurs que l'on pourra détecter en comparant l'image projetée à une image de référence pour le point du vue étudié. Cette approche n'est pas efficace pour la détection des objets supprimés : ils sont présent dans le maillage mais pas dans les images, et donc n'introduisent pas de nouvelles textures qui pourraient donner lieu à de fortes incohérences photométriques. Notre contribution sera donc de permettre la détection des

insertions et suppressions d'objets dans la scène, en améliorant tout d'abord l'algorithme de reprojection.

La reprojection de la méthode de l'état de l'art était faite en transformation "directe", où chaque pixel de l'image à transformer était déplacé dans l'image finale. Cela résulte en de nombreux pixels non coloriés dans l'image finale, qu'il faudrait interpoler : on utilise donc une transformation "inverse", où à chaque pixel de l'image finale est attribué un pixel de l'image à transformer. Certaines régions de l'image transformée resteront cependant non coloriées car elles correspondent à des régions 3D non visibles dans l'image de départ (car obscurcies par des objets au premier plan, ou en dehors de l'image). Déterminer si des régions sont visibles ou non se fait à l'aide d'un test de profondeur, et on remarque qu'en inversant ce test on peut donner à ces pixels la couleur des objets qui les masquent au premier plan. Cette reprojection alternative donne lieu à des images qu'on nomme "ombres texturées", et on remarque qu'en comparant ces ombres texturées à l'image de référence pour un point de vue donné, les régions **cohérentes** (en couleurs) correspondent aux régions des objets **supprimés**.

Pour un point de vue donné on peut donc produire une carte de détection de suppressions 2D. On prend une des 5 images comme référence, et on projette les 4 autres à ce point de vue (avec le test de profondeur inversé). Pour mesurer la cohérence, on crée 4 "delta maps" qui donnent la distance euclidienne pixel par pixel entre la valeur RGB dans la référence et celle dans une des 4 ombres texturées. Pour obtenir la position réelle des changements en 2D, on projette les pixels de ces delta maps au "premier plan", d'après le maillage et la caméra du point du vue étudié. On fusionne ces 4 delta maps de "premier plan" afin de réduire le bruit : on prend la valeur minimale de chaque pixel (les pixels non coloriés ayant une valeur maximale). Pour obtenir les suppressions d'objets en 2D, on fait un seuillage par valeur de pixel de cette delta map fusionnée. Contrairement à la méthode de [136], qui utilisait une valeur de seuil arbitraire, on effectue un seuillage automatique comme décrit dans [214], le seuillage triangle.

Les pixels de la carte de suppression 2D sont groupés par régions par dilatation et érosion des pixels de changements. Une fois qu'une carte des régions de changements est générée pour les 5 points de vue, on peut obtenir les régions 3D en identifiant, à l'aide du maillage, quelles régions 2D correspondent au même changement 3D dans la scène. Une fois les régions identifiées, on crée des ellipsoïdes 3D décrivant la position et la taille des suppressions d'objets dans la scène, en utilisant l'algorithme de triangulation de [136].

La méthode résultante peut être combinée avec l'approche de [136] avec un impact négligeable sur le temps de calcul, car les ombres texturées peuvent être générées en parallèle des images transformées de la méthode originale. On peut également améliorer la précision de la détection d'insertions en filtrant les résultats 2D avec nos résultats de détection de suppressions. Une implémentation de ce système complet a été réalisée sur

GPU à l'aide de *shaders*, améliorant d'avantage la précision (la résolution des images transformées) et la vitesse.

Afin d'évaluer la qualité de la détection de suppressions, on utilise les métriques 2D proposées dans [136] : les ellipsoïdes sont projetés au point de vue de chaque image de la scène, et on calcule les *Intersection over Union (IoU)* et *True Positive Rate (TPR)* des régions de changements estimées avec celles de la vérité terrain. On modifie également le dataset d'origine afin d'introduire des suppressions d'objet dans les scènes (des objets sont rajoutés au maillage 3D), et on modifie des scènes d'un autre dataset : ScanNet [37]. Sur toutes ces scènes, notre méthode est capable de détecter des suppressions d'objets avec plus de précision et sensibilité que la méthode d'origine (respectivement 36% et 70% de *IoU* et *TPR* contre 12% et 22%). On évalue également positivement la qualité du choix de seuil automatique, à l'aide des courbes *ROC (Receiver Operating Characteristic)*, comme décrit dans [188]. Notre méthode photo-géométrique est donc supérieure à l'état de l'art dans la majorité des cas, et est capable de différentier deux types de changements. Cependant, elle ne permet pas encore la mise à jour du maillage.

Description sémantique d'une scène

Les résultats de détection de changements de l'approche photo-géométrique ne sont pas suffisamment précis pour mettre à jour le maillage 3D de référence de la scène. Une analyse sémantique de cette scène permettrait de segmenter son contenu par objet pour améliorer l'approche. D'une part, les cartes de changements 2D pourraient être rendues plus précises en utilisant des contours d'objet obtenus à partir d'une segmentation 2D des images actuelles de la scène. D'autre part, un maillage 3D segmenté par objet serait plus simple à mettre à jour, en particulier pour le cas de la suppression ou du déplacement d'objets dans la scène. On note également que les résultats de l'analyse sémantique seraient exploitables pour localiser les images par rapport au maillage.

Au delà de l'amélioration de l'approche photo-géométrique, un modèle sémantique de la scène pourrait adresser le problème du réalisme comportemental des objets virtuels [77] et leur attribuer des propriétés utiles en plus de leur géométrie (poids, matériaux, modèle photométrique). Une détection de changements ne peut être exhaustive pour toutes les applications de RM, et définir un "vocabulaire" pour la scène à l'aide de l'analyse sémantique permet d'identifier quelles propriétés sont pertinentes.

Afin d'organiser l'information sémantique, on utilise un **graphe de scène**. Un graphe de scène basique peut représenter les objets présents par ses sommets, et les relations entre ces objets par ses arrêtes. La représentation est cependant flexible et peut inclure des attributs et labels sur les sommets et arrêtes, ou avoir plusieurs couches représentant différents niveaux d'abstraction (les sommets pouvant représenter d'autres

propriétés de l'environnement). Selon le domaine d'application, ces graphes peuvent être utilisés pour générer des scènes virtuelles (en rendu graphique [33, 75]), ou pour décrire des scènes réelles (en robotique [100, 213]). En RA, ces deux cas d'usage sont pertinents, et l'utilisation de deux graphes permet également de faciliter les interactions entre le réel et le virtuel [163].

Avec les avancées dans le domaine de la RA suivant le développement de notre première approche, on peut faire l'hypothèse que l'appareil de RA est capable de produire une carte de profondeur en plus de l'image RGB et de sa pose relative fournies par la caméra, et il existe des approches en robotique exploitant ces données d'entrée. La méthode décrite dans [100] génère des graphes (locaux) pour chaque image en entrée avec Factorizable-Net [116], et les fusionne dans un graphe 3D (global) en cherchant des sommets similaires dans ces graphes 2D. Cette mesure de similarité repose majoritairement sur le classe et la position absolue de l'objet représenté par le sommet, rendant son utilisation pour une relocalisation (ou détection de changements) entre différentes sessions de RA difficile. Les méthodes de localisation par graphe utilisent souvent uniquement ses attributs sémantiques et sa topologie pour comparer les sommets, cependant la topologie des graphes générés par [100] n'est pas assez informative. En effet, les graphes générés par Factorizable-Net [116] et d'autres méthodes 2D similaires se focalisent sur la description des actions dans la scène plutôt que sa structure, et les quelques arrêtes décrivant la structure 2D ne sont pas utilisables dans le graphe 3D ("à gauche de", "derrière"). Enfin, la génération de graphes 2D est trop instable d'une image à l'autre, et donnerait lieu à trop de faux positifs pour la détection de changements.

Suivant ces observations, on décide d'étudier les méthodes de génération de graphes utilisées pour la tâche de localisation plutôt que celle de description de scène, afin de favoriser la stabilité des graphes d'une session RA à la suivante. En conséquence, on choisit une représentation de graphe simple : les sommets ont deux attributs, un label et un vecteur 3D de position, et les arrêtes sont non-directionnelles et basées sur un seuil de proximité entre les sommets. Pour notre application, les méthodes de localisation par graphe devront pouvoir identifier les changements dans une scène, et pas seulement les ignorer pour des besoins de stabilité, mais également produire des graphes suffisamment fidèles à la scène pour être utiles en RM. On identifie 4 problèmes à résoudre pour l'utilisation de graphes sémantiques en détection de changements pour la RM:

- détecter les changements entre deux graphes de scène complets;
- détecter les changements entre un graphe partiel et un graphe de référence complet;
- créer un graphe partiel à partir des observations à l'aide de la référence complète;
- mettre à jour la géométrie du modèle de scène à partir de changements sémantiques.

Graphes pour l'alignement de modèles de scènes sémantiquement denses

Afin de résoudre le premier des 4 problèmes énoncés précédemment, et tester le modèle de scène choisi, on s'intéresse au problème de la localisation relative de deux vidéos par graphe. Dans ce problème, la pose locale de la caméra est connue dans le référentiel d'une vidéo donnée, mais la transformation 3D nécessaire pour passer du référentiel d'une vidéo à celui d'une autre est inconnue. Pour le résoudre, chaque vidéo est convertie en graphe de scène et les sommets (objets) similaires entre les deux graphes sont identifiés, et leur position relative dans le référentiel de chaque vidéo est utilisé pour calculer la matrice de transformation 3D.

La similarité entre deux sommets est quantitativement évaluée à l'aide de **descripteurs**, qui encodent pour chaque sommet la classe de l'objet qu'il représente (le label), ainsi que les classes des objets voisins d'après la topologie du graphe. Certains descripteurs se limitent aux voisins directs [180], mais d'autres explorent la topologie locale du graphe par parcours aléatoire [61, 124, 203] ou exhaustivement [72]. L'approche exhaustive est cependant limitée par la taille du descripteur qui croît en l^d avec l le nombre de labels utilisés lors de l'analyse sémantique, et d la profondeur d'exploration ($d-1$ "pas" depuis le sommet d'origine). L'estimation de la transformation 3D est généralement évaluée avec l'erreur de translation par rapport à la vérité terrain, mais il existe des métriques plus informatives appliquées au problème d'alignement de nuages de points, notamment la *RMSD (Root-Mean-Square Deviation)* [132] qui moyenne l'erreur de position à chaque point du nuage.

Pour le cas des scènes en intérieur, qui sont sémantiquement riches et où l'erreur de translation n'est pas suffisante, on va donc contribuer un nouveau descripteur de sommet non-aléatoire dont la taille croît en $l \times d$, ainsi qu'une métrique dérivée de la *RMSD* sur nuages de points. Enfin, on propose un nouveau processus de génération de graphes afin de produire des graphes de scènes exploitables au delà de la tâche de la localisation, comme représentation fidèle de la scène. Notre système complet, similaire à ceux décrits dans [72] et [61] a 3 parties : l'extraction de graphes à partir de vidéos, la génération de descripteurs et l'isomorphisme de sous-graphes pour les graphes générés, et enfin l'estimation de la transformation 3D basée sur les sommets sémantiquement similaires.

Pour la génération de graphes, la vidéo en entrée est composée de cartes de segmentation par objets issues d'une analyse sémantique préalable, ainsi que des cartes de profondeur et poses de caméra (dans le référentiel de la vidéo) correspondantes. On peut générer des points 3D correspondant aux objets visibles dans une image en calculant les barycentres 2D des régions sémantiquement uniformes, et en extrapolant leurs positions

3D dans la scène à partir de la carte de profondeur et de la caméra. D'une image à la suivante, les mêmes objets restent souvent visibles, donc [72] ignore les nouveaux points 3D si ces derniers sont trop proches d'un point 3D précédemment observé avec le même label, d'après un seuil de proximité. En conséquence, la première observation d'un objet donne sa position 3D (son attribut spatial dans le graphe de scène) et est souvent approximative car généralement les objets sont d'abord partiellement visibles aux bords de l'image, et de plus, les objets plus grands que le seuil de proximité sont fragmentés en plusieurs sommets.

Notre approche alternative consiste à ne pas ignorer les nouveaux points 3D similaires, mais à les regrouper sous un même "super-sommet", dont l'attribut spatial est la moyenne de ces points 3D. Par ailleurs, on peut alors utiliser tous ces points 3D pour le test de proximité, et ainsi les grands objets ne sont plus fragmentés, et ont un attribut spatial plus proche de leur véritable barycentre 3D. La génération du graphe se termine avec l'ajout d'arêtes selon un autre seuil de proximité automatiquement calculé à partir de la distance moyenne entre les sommets. Le graphe est donc représenté par une matrice d'adjacence A binaire, symétrique et à diagonale nulle, et par un vecteur de labels, respectivement de taille $n \times n$ et n, le nombre de sommets.

Pour l'alignement des deux graphes, les descripteurs sont générés à partir des puissances A^k de la matrice d'adjacence, où $A^k_{i,j}$ donne le nombre de chemins de longueur k entre les sommets i et j du graphe. Pour chaque $k < d$, on va encoder l'histogramme des labels des sommets atteignables depuis le sommet étudié dans un vecteur de longueur l. On va également encoder le label du sommet en question avec un vecteur *one-hot* (de taille l), qu'on peut concaténer avec les $d-1$ histogrammes de voisinage, obtenant ainsi un descripteur de taille $d \times l$.

Les méthodes de parcours aléatoire [61] introduisent des règles d'exploration du graphe pour favoriser l'encodage d'information pertinente dans les descripteurs : on peut simuler ces règles en utilisant des variantes de la matrice d'adjacence. On crée les matrices $B^{(k)}$ qui approximent une exploration du graphe avec la règle "on ne peut pas passer deux fois par la même arrête". On pondère également les contributions des histogrammes de voisinage par l'inverse de la profondeur k qu'ils représentent, afin de réduire les erreurs dues aux approximations des $B^{(k)}$ ou à l'homogénéisation des A^k pour les grandes valeurs de k. Les calculs matriciels sont accélérés en utilisant des opérateurs de logique booléenne, car les descripteurs ne sont pas affectés par le nombre de chemins entre deux sommets, seulement par leur existence.

Lors de la recherche d'isomorphisme de sous-graphes, la similarité entre deux sommets est donnée par le produit scalaire de leurs descripteurs respectifs. Une fois les sommets les plus similaires entre les deux graphes identifiés, on peut calculer la transformation 3D entre les référentiels de leurs vidéos respectives par régression à partir

des attributs spatiaux (les coordonnées 3D des objets). Les couples de sommets dont les coordonnées sont des données aberrantes pendant la régression (erreurs d'extraction, changements dans la scène) peuvent être filtrés par algorithme de RANSAC [12] pour améliorer la localisation. Cependant, n'ayant pas de vérité terrain pour l'isomorphisme de sous-graphes, on rend cette option optionnelle afin de mieux évaluer la performance des descripteurs à partir de la vérité terrain de localisation.

L'erreur de localisation est calculée à partir d'une intégration formelle de la RMSD sur le volume de la scène, plutôt d'une sommation des erreurs sur les points du maillage ou nuage qui la représenterait. Une telle méthode est décrite dans [90] pour l'alignement de cerveaux représentés par des sphères, prenant ainsi en compte les possibles erreurs de translation et rotation. Nous adaptons cette méthode pour l'appliquer à la localisation dans des scènes en intérieur, qui sont alors modélisées par des parallélépipèdes à 3 dimensions. L'erreur correspond à l'intégrale sur ce volume de l'écart entre un point transformé par notre estimation et le même point transformé par la vérité terrain : elle est donc indépendante du contenu de la scène mais dépend de ses proportions.

On évalue notre méthode de localisation sur deux datasets de scènes en intérieur : ScanNet [37] et ChangeSim [139], contenant respectivement des scènes réelles et des scènes d'entrepôts virtuelles. Pour ScanNet [37], on calcule la vérité terrain d'alignement, et on utilise la vérité terrain de segmentation par instance d'objet pour comparer la fidélité des graphes de sommets de [72] et de nos super-sommets à la scène réelle. Pour ChangeSim [139], on calcule les trajectoires de caméra, et on teste la robustesse de notre système à la présence de changements dans la scène. La qualité des graphes est quantitativement évaluée avec le *ARI (Adjusted Rand Index)*, et l'utilisation de nos super-sommets produit des graphes plus fidèles à la réalité (0.705 de *ARI* contre 0.599 pour [72], 1 indiquant une corrélation maximale et 0 aucune corrélation). Pour la tâche de localisation, nos nouveaux descripteurs donnent de meilleurs résultats que ceux de [72] sur les deux datasets, pour des temps de génération et comparaison (et taille en mémoire) bien inférieurs. On note enfin que l'utilisation de super-sommets ne dégrade pas la performance de localisation, et ces derniers pourront donc servir pour cette tâche aussi bien que pour d'autres propres à la RM.

Conclusion et perspectives

Dans cette thèse, nous avons présenté de nouvelles méthodes pour s'atteler au problème de la détection de changements pour les applications de Réalité Mixte. Ces travaux ont été menés afin de fournir un processus de maintenance du modèle d'un environnement réel, qui contiendrait les informations nécessaires à l'atteinte du niveau de réalisme souhaité pour des applications de RM.

Dans une première partie, nous avons souligné les différences entre les données requises par les applications de RM, et celles qui peuvent être acquises à l'aide d'un appareil de RA, en implémentant une méthode de détection de changements qui compare deux états d'une même scène représentés par ces données hétérogènes. Nous avons représenté l'état "passé" de la scène par un maillage 3D non-texturé (requis pour un rendu photo-réaliste mais difficile à scanner avec un appareil de RA) et l'état "présent" par une série d'images (nécessitant un simple caméra). Dans ce contexte, la représentation 3D peut être mise à jour en détectant uniquement la géométrie qui aurait été ajoutée ou supprimée entre les deux états. Nous avons utilisé notre propre approche basée sur la reprojection d'images, qui corrige le penchant [151] des méthodes similaires [136, 188] pour la détection de l'ajout de géométrie, au détriment de la suppression.

A la fin des ces travaux sur la comparaison du maillage 3D à une série d'images, nous avons noté que notre analyse photo-géométrique ne pourrait pas fournir des résultats suffisamment précis pour mettre à jour la géométrie du modèle de scène. Nous avons donc conclu dans une deuxième partie qu'une connaissance du contenu sémantique de la scène, et une segmentation 3D et 2D (par objet) des données du modèle associé, nous permettraient de maintenir la fidélité de ce modèle vis-à-vis de l'environnement réel après sa mise à jour. Nous avons choisi de représenter la scène par un graph sémantique, où les sommets représentent des objets et les arrêtes les relations (spatiales) entre ces objets. Ce graphe de scène est mis à jour par une analyse sémantique de la scène, qui nous permet de définir quels changements sont "pertinents" (ceux qui affecteraient l'expérience de RM) et d'améliorer le réalisme comportemental des objets virtuels.

Enfin, nous avons introduit dans une troisième partie, une méthode pour aligner ces graphes sémantiques par l'identification de sommets similaires. En partant du principe que les appareils de RA modernes seraient maintenant équipés d'un LiDAR ou de caméras stéréoscopiques (contrairement à la première partie) nous avons présenté une nouvelle méthode de génération de graphes à partir de séries d'images RGB-D. Dans la même partie, nous avons également introduit un nouveau descripteur de sommet qui permet de mesurer la similarité entre les objets (sommets des graphes de scène) extraits de deux différentes séries RGB-D. A l'aide des méthodes proposées, deux modèles d'une scène peuvent être comparés et alignés par comparaison sémantique de leurs graphes, et du contenu virtuel persistant peut être réintégré à la scène à chaque session de RA.

En conclusion, à l'aide des nouvelles hypothèses introduites dans la deuxième partie, l'approche photo-géométrique de la première partie de détection de changements pourrait être utilisée sur un maillage segmenté par objet, ouvrant une opportunité pour sa maintenance sans perdre en fidélité. De plus, avec la présence de ces objets sous forme de sommets dans le graphe de scène, la localisation des images à jour, nécessaire pour le bon fonctionnement de l'approche photo-géométrique, pourrait être prise en charge

par le système de localisation de la troisième partie. En cas de changements dans la scène, les objets retirés peuvent être simplement supprimés d'un maillage de scène segmenté, et les objets qui se déplacent dans la scène peuvent être similairement déplacés dans le maillage. Si l'appareil de RA est capable d'obtenir des images RGB-D pour la génération de graphe (comme dans la troisième partie), nous avons également accès à des informations géométriques 3D pour mettre à jour le maillage en cas d'ajout d'objet. En principe, une fois le problème de la détection de changements sémantiques dans la scène résolu, un modèle de scène segmenté pourrait être entièrement mis à jour pour des applications de RM dans des scènes changeantes.

Ce problème de la détection de changements sémantiques a été partiellement adressé par notre système d'alignement de graphes. En effet, l'approche décrite dans la troisième partie nous permet de comparer et coupler les sommets de deux graphes de scène, en mesurant leur similarité à l'aide des descripteurs. Pour obtenir une véritable détection de changements, le système d'alignement devra être capable d'identifier les sommets rendu "uniques" (qui n'ont pas d'équivalent dans l'autre graphe) par des changements dans la topologie du graphe de scène. La qualité d'un tel système devra être évaluée à l'aide d'une vérité terrain d'isomorphisme de graphe plutôt que d'alignement de scène. Par la suite, pour vérifier sa viabilité dans le contexte de la RA, le système de localisation devra être capable de fonctionner avec une vue limitée de la scène actuelle (à comparer à une référence globale). Cette perspective peut être simulée si les séries d'images RGB-D utilisées dans la troisième partie sont tronquées, ne montrant ainsi que la portion de l'environnement visible pendant une courte période. Cette hétérogénéité entre les observations et la référence conduira à un gain d'importance de cette dernière lors de l'analyse sémantique et géométrique de la scène, afin de définir les régions d'intérêt pour la détection de changements, basées sur les données contenues dans le modèle. Enfin, si on peut faire l'hypothèse que le modèle initial de la scène est "parfait" dans sa représentation des données requises pour la RM (le maillage, la topologie du graphe, les attributs des sommets...), un ultime problème serait de le garantir après plusieurs mises à jour successives. Les méthodes de détection de changements et le modèle de scène devront donc être choisis pour préserver la qualité de ce dernier sur plusieurs itérations, ou être capables de comparer deux états de la scène sur une plus longue période si le modèle d'origine (parfait) est toujours utilisé comme référence.

Une fois ces défis relevés, il sera possible d'envisager différentes extensions pour le modèle de scène, afin d'améliorer les interactions entre les éléments virtuels et réels, ainsi qu'avec l'utilisateur final. Notre modèle de scène ayant été inspiré par des avancements dans le domaine de la robotique, il pourrait également y être utilisé comme medium de traduction homme-machine pour la perception de l'environnement. Plus généralement, ce modèle est un outil pour communiquer l'information concernant une scène, et malgré

le volume important des données 2D et 3D, le contenu sémantique et son organisation sont représentés efficacement par un graphe. En outre, un modèle prévu pour surveiller les changements dans une scène peut s'avérer efficace pour transmettre des instruction pour la modifier (i.e. prescrire des changements plutôt que les décrire). Avec une compréhension exhaustive de la scène au niveau sémantique, une grande partie des changements géométriques et sémantiques nécessaires à la description de ses portions dynamiques peut être résumée en une liste de changements sémantiques.

Contents

1 Introduction **23**

2 Related work **29**
2.1 Change detection with 3D data . . . 30
2.2 Change detection with 2D data . . . 32
2.3 Change detection with heterogeneous data . . . 34
2.4 Conclusion . . . 35

3 Photo-geometric change detection **37**
3.1 Context and inputs . . . 37
3.2 Image-based object removal detection . . . 39
3.2.1 Image reprojection and occlusions handling . . . 40
3.2.2 Photo-consistency in occluded pixels . . . 43
3.2.3 Photo-consistency with multiple points of view . . . 44
3.2.3.1 Foreground projection . . . 44
3.2.3.2 Combination of projected delta maps . . . 45
3.2.4 3D localisation of changes . . . 47
3.2.4.1 Segmentation by region . . . 47
3.2.4.2 Region matching . . . 48
3.3 Experimental evaluation and results . . . 49
3.3.1 Quantitative evaluation . . . 50
3.3.1.1 IoU and coverage with automatic thresholding . . . 50
3.3.1.2 ROC curves for different thresholds . . . 53
3.3.2 Qualitative evaluation . . . 54
3.3.2.1 Filtering of the insertions detection results . . . 54
3.3.2.2 Scenes from the Palazzolo et al. dataset . . . 55
3.3.2.3 Scenes from ScanNet dataset . . . 59
3.3.3 Computation time . . . 66
3.3.4 Qualitative results from GPU implementation . . . 67
3.3.5 Limitations . . . 70

3.4 Conclusion . . . 70

4 Semantic description of the scene **73**
4.1 Semantic analysis and motivations . . . 73
4.1.1 Object-level modelling . . . 74
4.1.2 Feature-based localisation . . . 75
4.1.3 Explicitly defining the relevancy of changes . . . 75
4.2 Graphs to structure the scene information . . . 77
4.2.1 Linking semantic information to photo-geometric data . . . 77
4.2.2 Accounting for unknown objects . . . 78
4.2.3 Acting as an intermediary representation . . . 79
4.2.4 Conclusion: Representation requirements for MR and CD . . . 79
4.3 Survey: Semantic Graphs for scene understanding . . . 80
4.3.1 Prescriptive graphs for scene authoring . . . 80
4.3.2 2D Descriptive graphs and relationship extraction . . . 81
4.3.3 3D Descriptive graphs and graph fusion . . . 82
4.3.4 Object similarity for node matching . . . 83
4.3.5 Graph subsumption . . . 85
4.4 Conclusion: Scene Graphs for Change Detection . . . 86
4.4.1 Experiments with the scene representation . . . 87
4.4.2 Problems to solve to achieve the thesis goals . . . 88

5 Graph-based model registration for semantically rich scenes **91**
5.1 Introduction . . . 92
5.2 Related Work . . . 94
5.2.1 Graph-based localisation and descriptors . . . 95
5.2.1.1 Neighbourhood vectors . . . 95
5.2.1.2 Random walk . . . 95
5.2.1.3 Walk histograms . . . 96
5.2.2 Quality metrics for scene registration . . . 96
5.3 Proposed method for scene registration . . . 97
5.3.1 Pipeline . . . 97
5.3.1.1 Input data . . . 98
5.3.1.2 Point Cloud extraction . . . 98
5.3.1.3 Graph construction . . . 99
5.3.1.4 Graph matching . . . 99
5.3.1.5 Registration . . . 99
5.3.2 Super Nodes for graph generation . . . 100
5.3.3 Adjacency descriptor . . . 104

5.3.3.1 Basic descriptor using the adjacency matrix 104
5.3.3.2 Improving the descriptor 107
5.3.4 Comparison with the walk histogram 110
5.4 Experiments 110
5.4.1 Datasets 111
5.4.1.1 Preparing the ScanNet data 112
5.4.1.2 Preparing the ChangeSim data 115
5.4.2 Error metrics for model alignment 117
5.4.3 Results 119
5.4.3.1 Graph extraction 119
5.4.3.2 Variants of the adjacency descriptor 121
5.4.3.3 Adjacency descriptor vs walk histogram 124
5.4.3.4 Generation and matching times 131
5.4.3.5 Impact of the depth 132
5.4.3.6 Impact of graph quality 133
5.5 Conclusion 138

6 Conclusion and Future Work **139**
6.1 Future work 141

A Photo-geometric change detection dataset **173**
A.1 Scenes from Palazzolo et al. dataset 174
A.2 Scenes from ScanNet dataset 181

Chapter 1

Introduction

After the resurgence of public interest in Virtual Reality (VR) following the 2016 launches of the Oculus Rift and HTC VIVE devices, the field of Augmented Reality (AR) also saw an attention gain despite earlier commercial failures [13] for its own immersive capabilities [40]. This eventually resulted in the 2017 release of Microsoft's head-mounted display (HMD), the HoloLens, and Apple's ARKit development toolkit for their existing hand-held products –and its Android counterpart ARCore the following year– leading to the development of AR applications with broad appeal. The increase in remote work in the last few years [156] has also renewed interest in AR as a collaboration and communication tool [142] with its innate feature of letting reality through, distinguishing it from the more isolating VR experience. One of the concerns for AR applications is the seamless integration of virtual content into a real scene, such as realistically behaving user avatars [50].

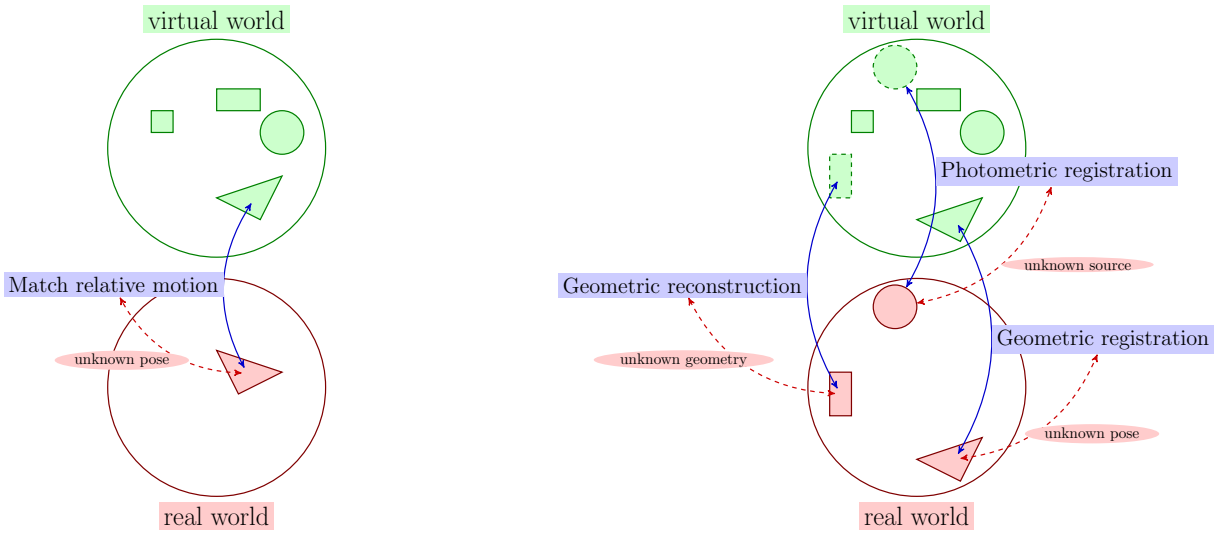
In this thesis, we will use the term "Mixed Reality" (MR). Among its many definitions [177], we will consider MR as a branch of AR with a focus on immersivity through the blending of real and virtual elements in a photo-realistic manner. Photo-realism has been claimed to engage AR users by giving them a sense of "being there" as opposed to more stylized representations [77]. Nevertheless the immersion is more easily broken as realistic behaviour is also expected [77]. However, when both visual and behavioural realism are achieved, so-called Mixed Reality becomes a powerful and versatile tool. It has notably given birth to applications such as teaching in fields where real world experiences cannot be conveniently provided, such as in the study of far-away or otherwise unreachable spaces [147, 128], or in dangerous and high-stakes occupations. In medical training for instance, where there was once a need for dedicated tools for either realistic visualization or accurate mechanical simulation because of technological limitations, it has long been possible to merge the two tasks into a more encompassing teaching medium [178]. More generally, photo-realism can be desirable in any AR application [1] involving previsualization and layout planning such as machining [114] and construction [66]. It is also relevant when there are aesthetic concerns e.g., in visual effect production [48], for

interior design [35] or to showcase products in retail experiences [199]. The work in this thesis is motivated by MR uses in both outdoor urban environments, such as for the study of cultural heritage and architecture [80], and in indoor scenes, such as for remote collaboration [127].

In early works [94, 64], photo-realism in AR was framed as a problem of balancing rendering time and accurately performing three main tasks: geometric registration (estimating the camera's parameters), geometric reconstruction (estimating the scene geometry), and photometric registration [94, 68] (estimating the scene illumination). Indeed, compared to a VR experience where every asset is designed by or known to the author and goes through the same rendering pipeline, parts of the AR experience belong to the physical world and their unknown properties must be inferred through partial observations (see Figure 1.1). Specifically, when AR users are free to move within the environment, rendering effects such as occlusion handling [206] and depth of field [134] are more realistically achieved with knowledge of the 3D geometry of the space, as well as the light sources and surface materials within it [107, 1]. The results were then evaluated on an image-basis [64], not accounting for some of the perceptual issues inherent to AR like virtual object presence and depth perception [104]. These works would also leave the task of geometry reconstruction to the user [65], allowing them to use the AR display as a 2D window for the insertion and manipulation of geometric primitives –roughly representing the objects in the scene– in order to build a so-called "scene-graph" and calibrate the camera for the geometric registration.

With more standardized hardware and extensive methods to achieve unassisted geometric registration [34, 200], the focus of research in MR rendering shifted towards illumination simulation [107] through analysis and modelling of the environment. These processes were increasingly complexified, starting with representing real objects with rough "phantom" models to accurately cast virtual shadows [78, 141], then expanding the scene representation to include sky boxes and virtual light sources [141], and eventually attempting to discard physical probes –such as reflective balls and fiducial markers [94, 93]– by relying on real-time observations [68, 146, 89].

For the virtual elements' behaviour to match their visual counterparts in terms of realism [77] (e.g., object placement [111], agent interactions [32], physics simulation [24]), the scene representation should include semantic information, requiring a low or high-level interpretation of its contents. While such a "thorough" representation of the scene can be hand-crafted in static scenes and controlled environments [112, 107], dynamic environments would require real-time exhaustive 2D and 3D structure scanning and processing of the raw observations to produce it. This can be performed using additional sensors, such as time-of-flight or laser scanners [112, 107], or RGB-D cameras [68], which might be absent or too rudimentary in a widespread (hand-held) consumer-level AR



(A) VR experience.

(B) AR experience.

FIGURE 1.1: Differences in knowledge between VR and AR experiences. Cameras are represented as triangles, scene geometry as rectangles and light sources as circles. In the AR scene, the camera pose needs to be known relative to the real (red) and virtual (green) scenes. Real scene assets also need to be simulated (dashed shapes).

setup. Alternatively, vision-based methods (such as structure from motion [184]) can be used by exploiting simple RGB images [133], but are computationally expensive and do not account for the difficulties of recovering geometry in areas not visible from the point of view of the AR device [64]. Indeed, [157] resorts to using multiple external cameras and a personal computer to perform the computations then transmitted to the device, to answer for its lack of power. Finally, the OpenRooms framework was introduced in [118] to provide the means to generate a photo-realistic indoor scene dataset, with the aim to train machine learning models to perform inverse rendering and scene understanding for robotics and MR. However the synthetic scenes it can produce have yet to be used in that latter field.

Were these technologies to become more accessible in the future, the time taken to thoroughly examine the scene would still be prohibitively long for most applications, if performed at each immersive session or for every change in very dynamic scenes. Consequently, a representation of the scene needs to be preserved between each session for the MR experience to be able to (re-)start immediately. While this strategy assumes

that the representation is sufficiently consistent with the current scene's state to simulate the effects of what is out of view [64], it will eventually require updating (as in Figure 1.2). The problem of providing an up-to-date representation of the environment to be used in MR applications can thus be broken down into three main issues:

- designing said representation to contain sufficient data about the scene to achieve the desired level of realism from its exploitation;
- detecting the relevant differences between the current and previous scene states from limited current observations and a more exhaustive but outdated representation;
- modifying the representation based on the detected changes to maintain its original level of fidelity to the scene.

There is a need for a representation of the scene prior –the reference state of the scene– that facilitates its own maintenance and use in an MR context. In this doctoral study, the goal is to find such a representation and design a process to detect changes between the prior and the real scene, to then update said prior with only a partial perception of the current state at low computational cost [174].

There are existing works that deal with the detection of changes between a prior representation of a scene and a current one, whose results can be used to act on those inconsistencies. Some works which apply to surveillance or monitoring tasks [191], only deal with anomaly detection. Alternatively, other works are geared towards making regular updates to the scene representation, but separate the tasks of Change Detection (CD) and scene model maintenance [189]. In this case, the maintenance step is performed by re-examining the areas of change with specialized equipment, while the detection step can be accomplished with a simpler device. In general, the types of representation that are used for the prior and current scene states depend on the target application field, as the application imposes constraints on the means and acceptable time required to build them. For the purposes of MR, changes must be detected using the limited data available at runtime: the current scene's state is only partially observable, and the "quality" of its representation is poorer than that of the prior. Said quality can be defined as the quantity of raw data that can be captured and its reliability, and is governed by the AR device's sensors and processing power.

The rest of this thesis will be organised as follows. Chapter 2 will be dedicated to the exploration of the most relevant CD methods as they relate to MR, as well as fields where CD is a central issue. To that purpose the chapter will be divided by the types of representation (3D geometry, 2D geometry, semantics, etc.) used to compare the reference and current scene states.

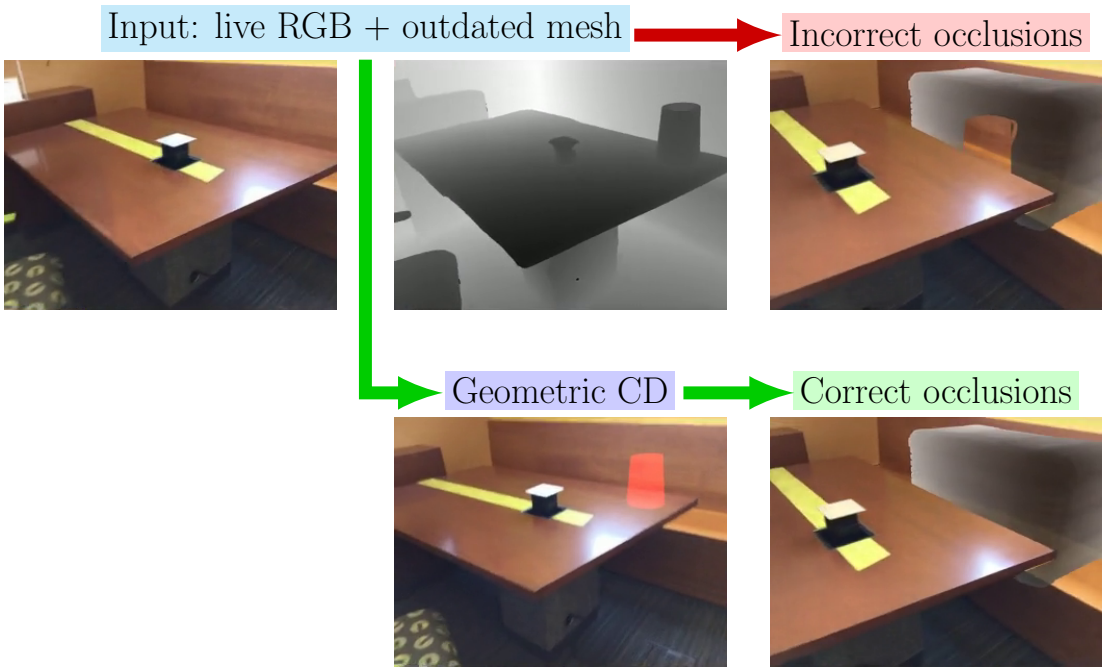


FIGURE 1.2: Geometric Change Detection for MR. Using an outdated 3D representation of the scene (top-center) causes an unrealistic integration of virtual elements (top-right) into the current scene view (top-left). Detecting changes between the past and current scene states (bottom-center) can allow us to correct this (bottom-right).

Some of the mentioned prior works deal with CD using heterogeneous representations [136] (e.g., a 3D mesh prior against an up-to-date image sequence), and while the detected changes cannot be directly used to update the prior, the underlying speed and hardware constraints imposed on the method are close to our MR use-case. For this reason, this photo-geometric approach [136] will be expanded into a more complete CD framework, described in Chapter 3. The new method will resolve some of the limitations of the previous works in terms of types of changes detected –namely object removal– while testing its efficiency in the field of AR in changing environments. The core concept of the photo-geometric CD is the reprojection of up-to-date images (observations) aided by an outdated untextured 3D mesh (prior representation) into different points of view, and the evaluation of photometric inconsistencies in the resulting images introduced by the changes in scene geometry between the observations and the prior. The results will be quantitatively evaluated thanks to the pixel-wise ground truth introduced through

the simulation of changes in an existing real indoor scene dataset.

From these results, it will be determined that this preliminary choice of scene representation is not able to yield photo-realistic results in all cases, and more importantly is impractical to update using the detectable changes. Therefore in Chapter 4, we will investigate alternative representations that are better suited to our needs, by observing that semantic analysis allows for a more practical object-level understanding of the scene, and that mainstream AR devices have been equipped with hardware such as LiDAR sensors [205] since the beginning of this thesis.

Chapter 5 will be dedicated to the process of generating the semantic scene graph and its application to geometric registration. We will evaluate the fidelity of graph scene models to the real environment, and show that it can be improved without compromising the registration accuracy, or its robustness to changes in non-static scenes.

Finally, we will present our conclusions in Chapter 6, and show how geometric and semantic CD can allow us to maintain a scene model for MR applications using our contributions:

- a complete geometric CD framework –for insertion and removal of geometry– that operates at interactive time with minimum hardware requirements,
- a semantic scene graph model tailored to the CD task in an AR setting emanating from a survey of such representations in other fields,
- a scene graph generation and registration framework that produces a multi-purpose graph representation without compromising the precision of scene alignment.

We will also describe the future works that can be built upon the results presented in this thesis.

Chapter 2

Related work

As mentioned in the previous chapter, the main challenges of MR are:

- geometric registration: aligning the virtual and real contents of the scene,
- geometric reconstruction: understanding the real scene's geometry,
- photometric registration: detecting and modelling the light sources in the real scene.

The task of Change Detection has been used as a proxy to tackle similar challenges in many different fields: previous results can be updated rather than reproduced. However, outside of the context of MR, CD can refer to a variety of processes, involving the comparison of different types of input data and detection results tailored to specific applications.

For instance, in medical analysis or manufacturing monitoring, CD can be synonymous with anomaly detection between two images, and the expected result can be a caption –verbal description– of the changes [120]. Furthermore, in applications such as land surface remote sensing, where the changing nature and borders of terrains are extensively studied [99, 30, 22, 42], in order to apply CD to scene analysis, the changes are expected to convey precise geometric information (e.g., a pixel map aligned with the input aerial images). Indeed, these works often deal with aligned image pairs, as the conditions of observation can be reproduced over time.

In contrast, the most susceptible fields to address the challenges of MR are those of ground-based applications, as 3D geometry and semantic analysis become more relevant when scanning [151] –or rendering– is performed from the ground: the perspective most AR users are expected to have. There are numerous works dealing with CD as it applies to smart cities [198, 109, 197, 151], autonomous vehicles [3, 59, 145] and human-robot collaboration in a shared space [125, 149, 51, 4], as it makes localisation more robust and improves adaptability and interactivity. The following tables showcase the various input

data types used that are compared by the CD algorithms of each previously mentioned fields (Table 2.1), as well as the nature of the CD results (Table 2.2).

Table 2.1: Common types of input data for CD algorithms

Field	Input data
Medical, disaster	images [120, 138, 95]
Remote sensing	images [67, 38, 39, 42, 30]
Smart cities	images [198], point clouds [109, 197], both [151]
Autonomous vehicles	images [3, 16], point clouds [59], both [151, 145, 188]
Robotics	3D map and RGB-D images [4, 51, 149, 97]

Table 2.2: Common natures of results of CD algorithms

Field	Results
Medical, disaster	caption [120, 138, 95]
Remote sensing	pixel map [67, 38, 39, 42, 30]
Smart cities	pixel map [198, 151], objects [109], vertices [197]
Autonomous vehicles	pixel map [3, 16, 151, 145], vertices [59], voxels [188]
Robotics	3D objects [4, 51, 149, 97]

Of note, these works assume semi-static input scenes, as opposed to dynamic ones. This means that detectable changes only occur between observations [192], and CD is performed between two static states: a prior –or "reference"– and a current –or "query"– state. Depending on the type of scenes, we have identified three prevailing combinations of input data used for semi-static CD: homogeneous 3D representations for both states, homogeneous 2D representations, or heterogeneous representations between the two states. The following sections will give an overview of existing CD methods grouped according to those input data types, and assert their applicability to MR applications.

2.1 Change detection with 3D data

In the fields of self-driving vehicles [59, 145] or robotics [192, 149, 51, 4, 97, 197], CD is often performed directly between 3D representations of the environment in order to directly obtain 3D results. This is achieved by equipping the autonomous device with laser or LiDAR sensors or RGB-D cameras, and CD is usually used to maintain an accurate or otherwise "useful" map of the environment. More generally, many such

methods use point clouds [59, 4] or voxel grids [51] to compare observations of the environment in different states –with short [145], long [4] or indeterminate [167] elapsed time between them. When the observations are global (i.e., not partial) the data may still require alignment, which is achievable through gravitational registration [197]. If the data are point clouds, temporal changes can be detected using a displacement threshold on the points' coordinates [197]. While this approach gives an exact localisation of the changes –the displaced points– it requires an exact matching of all points between the compared point clouds, and therefore processing of the input 3D data if it is not synthetic. Assuming that the observations are already aligned, other works use Growing Least Square reconstruction [137] on the clouds which do not necessarily share the same exact points due to the random nature of laser scanning. A segmentation of the scene into "objects" is performed and then compared at different times, but the method is not meant to run in real time –despite speed improvements using downsampling– and requires noiseless and global clouds.

However, in the aforementioned fields, the more recent models of the scene are expected to be partial due to the limited field-of-view of the observation device. The registration of the data is done through localisation of the observer, using for instance SLAM [51], pose graphs [149] or normal distribution transforms on a voxelized map of the scene [97]. For example, by converting the input data to said 3D grid representation, [97] is able to quickly perform CD and robust localisation by comparing the occupation of the voxels in the prior and current states. In the work, this allows the robot to compare a dense point cloud to up-to-date 3D range data from an onboard Kinect camera, but despite good performance on localisation, changes of small size will be undetected if the voxels are oversized due to speed requirements. For a more granular representation of the scene, a Truncated Signed Distance Function is often used [149, 51, 4] and offers more precise CD results than the simple occupation of the grid's voxels. The CD results are further improved and filtered in [149] by clustering semantically consistent 3D input data into "objects", providing the robot with an up-to-date and segmented representation of its working environment to accurately navigate and perform tasks. When scanning the scene, said objects are stored in a library, which allows them to be recognized when they are encountered at a later time (using iterative closest point). This tracking of the movements of unique objects greatly improves the precision of the updated 3D map, but comes at a great computational cost as the number of objects increases. In earlier works [51, 4], the application of CD to map reconstruction differs slightly as it serves to filter out dynamic objects and obtain a "static map" of the scene. As a result, their methods are less computationally intensive than that of [149], but much less versatile or robust to dynamism in the scene, and produce more reconstruction artefacts.

Expanding the scope of this section to works dealing with "dynamic" scenes, we can

also find relevant methods which still compare successive 3D scene observations, albeit on a shorter time scale. In [167], different point clouds are turned into voxel grids, and the identification of "see-through" voxels provides the localisation of dynamic points. In this context, see-through voxels contain points at a given time but do not prevent the device from capturing points located behind them at a different time, inducing the removal of their content. The authors also introduce the notion of "point shadow" that represents the ability of a 3D point to occlude parts of a scene from a given point of view. This multi-angle approach is however very costly in terms of memory, as it requires the storage of multiple points clouds at different known vantage points, since merging them into a single 3D map before CD would not allow to method to check for see-through voxels. The method described in [145] detects dynamic objects in a point cloud as well, but also takes advantage of an RGB camera and real-time depth information to track patches of similar color in RGB or similar depth in a depth map. Although this makes the method perform in real time, the work does not focus on comparison to a prior map, and is not directly usable as a map reconstruction or maintenance framework.

Finally, machine learning tools have more recently been introduced to process 3D data, such as PointNet [21] and its variants [5, 115] and RPM-Net [212]. Convolution networks have therefore been used for CD with point clouds [109], taking into account the color and geometry of the compared object pairs in the environment. However these methods remain demanding computationally and hardware-wise. While some of these issues are addressed in [122], where the current 3D observations are derived from stereo-vision over several frames, there exist more lightweight representations and methods for CD. Indeed, if 3D geometric information is very relevant to MR applications –for photo-realism concerns– its scanning and manipulation comes at too great a computational cost in most situations. Moreover, these works are not themselves focused on producing a realistic model of the scene, and therefore rarely deal with the textures of the scene, or its light sources.

2.2 Change detection with 2D data

Avoiding the need for specialized 3D capture equipment, many CD techniques rely on the comparison of two pictures, particularly in the remote sensing field [38, 39, 67, 42, 30]. Many modern 2D methods are based on the use of Siamese Neural Networks, which can provide pixel-wise CD results. In [38], such networks are employed to compare two co-registered RGB or multi-spectral aerial images. More recent work focuses on developing robustness to pseudo changes [22]. This notion is significant for the street-level applications in the smart city field [71, 198], where images are less "flat": neighbouring pixels can represent 3D regions separated by large distances. In this situation, even slight

lighting or viewpoint changes are likely to introduce noticeable –yet likely irrelevant– changes. Other works build on the foundation of Siamese networks to generalize the CD process, such as in [71], where picture registration is not mandatory, and in [198], where object segmentation is performed, making the approach more robust to changes in weather or scene illumination. The method from [71] is also mostly robust to large viewpoint changes, being able to differentiate noise from meaningful changes. It is however not due to a 3D understanding of the images' contents but rather a robustness inherent to the large receptive field of its Convolutional Neural Network, and the focus on semantic analysis during training.

Efforts have also been made to further describe the nature of the changes themselves –rather than the objects affected. For instance, in [16] the notion of "directional change" is introduced, which describes whether the change is detected due to the removal, the insertion or the exchange of pixels belonging to foreground objects. This is indeed more challenging using 2D data, since voxel grids or point clouds already represent geometry, and insertion and removals can be inherently inferred after CD. Moreover, several semantic-based 2D methods have been developed, which focus on the nature of the elements of a scene, for satellite [30, 39, 67] or ground-level images [165]. This approach is also put to use for unsupervised training [42], where such pictures are artificially altered with patches of different nature. The nature of change is further explored in change captioning methods, and more generally approaches used for captioning pairs of images. In [95], the nature of the semantically identified objects informs the nature of the change, whereas in [138] both are independently identified. Still, the works based on semantic understanding from learning are most effective on the data they were trained on. For this reason, they cannot directly be used to thoroughly detect all the changes that would impact the scene geometry, as those could arise from objects unrecognizable to the networks.

In all of these works, inferring 3D CD results from the 2D images remains a challenge, yet can be necessary to correctly understand the scene. Indeed, to improve the captioning of 3D relationship change (and camera-robustness), [120] introduces scene graphs to represent the scene composition, as concept that will be furthered explored in Chapter 4. Finally, despite providing CD results in the shape of a 2D pixel map, [3] infers sparse 3D representations from the prior and current image sequences it compares, and uses said 3D models to reproject the 2D images to better align them prior to comparison. If the prior was assumed to be a 3D representation, this reprojection approach could be used without resorting to an estimation of the scene's geometry.

2.3 Change detection with heterogeneous data

While image-based CD methods are less computationally expensive than 3D-based ones, this is irrelevant when real-time interpretation of the 2D results is required to (re-)construct the 3D geometry of the environment. This has led to the development of hybrid methods that use heterogeneous data as input.

Methods described in [188] and [195] aim at monitoring the evolution of a urban environment. This is achieved through the comparison of up-to-date images with an outdated 3D mesh [188] or with a dedicated 4D model [195, 196] that represents the evolution of 3D data over time –considered the fourth dimension. The detected changes are provided in the shape of a 3D grid of change probabilities. For instance, in [188]: for every voxel in the grid, the corresponding pixels of the images it appears in are observed, then the probability of change is lowered if the colour is consistent between images (the Euclidean norm of the RGB vector). This approach primarily focuses on the structure of the environment as opposed to its texture and in practice, after this preliminary CD, some specialized equipment –such as vehicle or aerial laser scanners [84, 44]– is deployed to the locations of detected changes in order to more precisely update the mesh.

These methods rely on an offline processing of the images and are still too computationally expensive to use on mainstream devices [174]. In [136] a fast approach based on image reprojection is proposed for the purpose of autonomous exploration of an environment by a robot. In the absence of changes, the current 2D images should be identical when reprojected to the same point-of-view using the reference 3D mesh (not including the occluded areas). However, differences in the scene geometry between the capture of the mesh and that of the images would introduce inconsistencies in the reprojection which can be used to infer said geometry changes. While performing at an interactive rate, the technique is strongly biased toward the detection of object insertion in a scene rather than removal. This issue of false negatives induced by photometric homogeneity in the images is addressed in [151], which similarly reprojects current 2D observations using a 3D point cloud, but has to further process the latter as it contains "holes", unlike a mesh. Indeed, inconsistencies in the reprojection are only visible if the reprojected contents have great photometric variations. Removed objects are therefore harder to detect since the inconsistencies they introduce are entirely dependent on the background textures revealed by their removal, which are more likely to be uniform (i.e., a wall or the sky). In [151] this issue is dealt with by propagating 2D CD results using both color and semantic labelling, but this assumes a preliminary segmentation of the images.

2.4 Conclusion

For MR applications, CD should be able to provide results on the 3D geometry of the scene in order to preserve the realism of the experience. In Section 2.1, we found that attempting to directly scan the new geometry of the scene and updating the model imposes many hardware requirements (i.e., sensors, computing power) that mainstream AR devices cannot yet meet. Conversely, Section 2.2 showed that while an image-based approach could be taken to CD, the results of the detection would not be directly usable. We conclude that a 3D model is needed to represent the prior state of the scene, otherwise the updated geometry of the scene would have to be entirely reconstructed from the new observations.

TABLE 2.3: Applicability of CD algorithms input types to MR applications

Data types	Benefits	Drawbacks
3D vs 3D	Access to 3D geometry/changes	Specialized sensors
2D vs 2D	Simple RGB sensor (camera)	Entire geometry reconstruction
2D vs 3D	3D changes from camera	Some reconstruction (changes)

In this light, we find that the CD methods using heterogeneous input data presented in Section 2.3 are more applicable to the field of MR. As outlined in Table 2.3, they closely match the assumptions we can make for MR: an existing 3D model of the scene is available, but requires updating using the limited hardware of the AR device –sometimes limited to a camera and motion sensors. Using heterogeneous CD, the only geometry that needs to be reconstructed in the model, is that which corresponds to the detected changes. However, as outlined in Section 2.3, the bias towards changes that introduce the most photometric disturbances has to be addressed in order to obtain a complete CD framework.

Chapter 3

Photo-geometric change detection

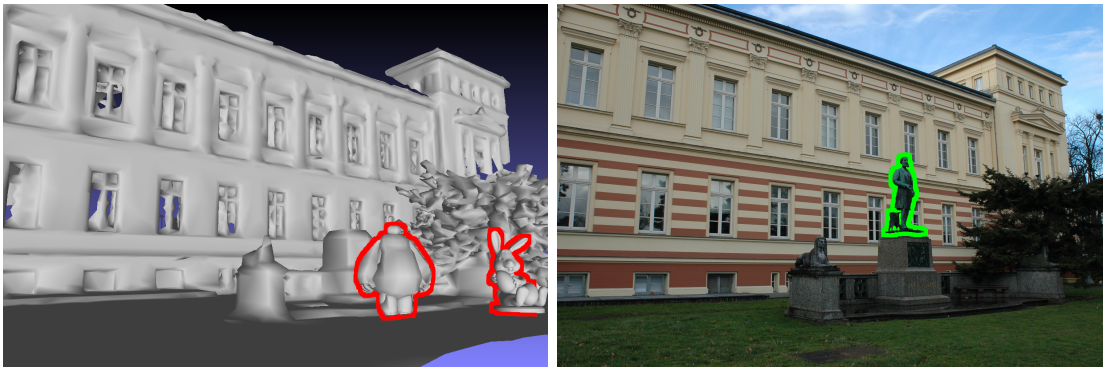
In this chapter we propose a method for detecting the removal of objects in a scene using heterogeneous input data to represent its past and present states. Similarly to the other approaches that use such inputs [188, 136, 151] mentioned in Chapter 2, we will use a 3D reference model –an untextured mesh– and 2D observations –a sequence of images registered to said mesh. The removal detection is accomplished by studying the impact of ignoring all foreground objects from the scene during the projection of images onto the mesh, as to avoid the false negatives resulting from photometric homogeneity with regular reprojection [151].Indeed, a foreground object that is still present in the images will produce inconsistencies if ignored during the projection, while a removed object will not to the same extent. The removed objects can be found by highlighting the regions of most consistency between the projected images and a reference one, with an interactive processing time. The method properties are summarized in Table 3.1.

Table 3.1: Change detection using reprojection

Prior model	Current observations	CD results
3D untextured mesh	Registered image sequence	3D ellipsoids

3.1 Context and inputs

With the popularization of MR applications in an expanding number of fields, there is an increasing need for accurate and cost effective 3D model building techniques [55, 57].The naive approach to keep these models updated is to perform a regular and comprehensive 3D scan of the environment, which is time consuming, expensive [8, 47, 54, 84, 155], and requires appropriate equipment [44, 88, 164]. A more efficient process is to compare an existing but outdated 3D mesh to current images of the environment to locate the 3D locations of changes and limit the updates to those areas [188].



(A) Outdated 3D mesh. (B) Up-to-date scene photograph.

FIGURE 3.1: Insertions (green) and removals (red) over time.

Using images to describe only the up-to-date –and not the prior– state of a scene allows the detection to be independent of the illumination, poses, and devices used during the captures, since these factors generally do not affect the geometry of the scene and therefore the reference mesh. However, this asymmetry in the types of data used to represent the past and current states of the environment makes the identification of the nature of the changes more difficult. The attachment of such semantic information to changes often relies on the ability to match elements or locations of the scene at different times [138].

For the purposes of updating the geometry of a 3D mesh, we can categorize changes in a scene as either "insertion of matter" or "removal of matter", since the displacements can be regarded as a combination of the two. Notably, insertions and removals also account for the "deformation of objects", as there is no object separation or texture information in the mesh to maintain: all 3D geometry is broken down into vertices and triangles. In this chapter, we propose a method for change detection which specifically focuses on the detection and the localisation of "matter", or objects, that has been removed from a scene using a reference mesh and images taken at a later time. The novelty of the solution is to focus on the parts of the scene that should be occluded by some foreground.
The main contributions of this chapter are:

- An image warping algorithm that generates textured shadows for the study of occluded areas in an image.
- An improved object removal detection method that uses the aforementioned algorithm.

- A complete fast change detection framework that combines our improved removal detection method with an existing change detection algorithm at no significant computational cost.

Our proposal is documented in Section 3.2, and the results are discussed in Section 3.3. Finally, our conclusions are summarized in Section 3.4.

3.2 Image-based object removal detection

As in [136] and [188], a scene is represented by an outdated 3D mesh (see in Figure 3.1a) and changes are detected using pairs of up-to-date images registered to the mesh (see in Figure 3.1b). All images are taken within a narrow time frame, and we therefore assume that there are no structural or lighting changes between them.

Firstly, changes are evaluated in 2D from the point of view of each image: starting from a reference image, every other image is reprojected from the reference point of view. Note that this reprojection must take into account the original 3D scene to handle occlusions. Colour differences are computed between the reference image and each reprojected image and stored in *delta maps*. The plurality of delta maps per point of view is used to reduce noise and retrieve more accurate changes. Secondly, 3D changes are deduced by matching 2D changes together across multiple points of view. Location and size of the changes are estimated based on the delta maps.

To achieve removal detection, our approach differs from the state of the art in the choice of regions of interest for 2D change detection. The projection method used when warping images generates regions of occlusions as seen in Figure 3.2a which are ignored in other approaches but are the primary focus in ours.
In summary, the proposed method is comprised of 5 steps:

1. For each image of the sequence, create reprojected copies to fit the points of view of the other images.

2. For each point of view, render the delta maps between the corresponding sequence's image and each accordingly reprojected image.

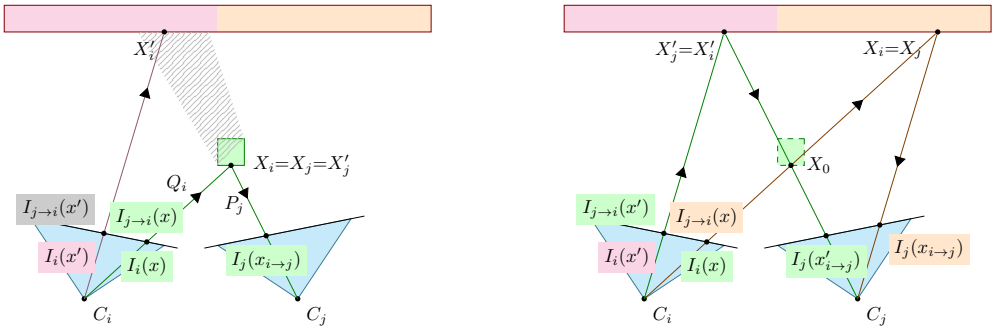
3. For each point of view, combine the delta maps into a single delta map to reduce false positives.

4. For each combined delta map, filter and group the pixels of detected changes into 2D areas of changes.

5. Match the 2D areas from one point of view to the other to infer the 3D locations and size of detected changes.

3.2.1 Image reprojection and occlusions handling

In this chapter, "reprojection" is not strictly used in its conventional meaning: it here amounts to back-projecting pixels onto the reference mesh and rendering them using another projection. This process is formalized in the next paragraphs, insisting on the impact of occlusions.



(A) Reprojection $I_{j\to i}$ with no change.

(B) Reprojection $I_{j\to i}$ with an added object.

FIGURE 3.2: Left: Reprojection from viewpoint j to i: foreground object (green) produces an occlusion (in black) of the background (yellow/pink). Right: Inserted object at X_0 (i.e. absent from mesh) introduces inconsistencies: $I_i(x) \neq I_{j\to i}(x)$, $I_i(x') \neq I_{j\to i}(x')$.

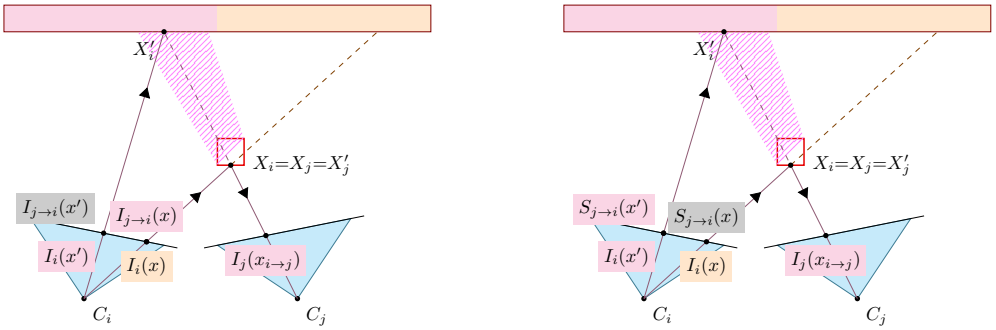
Let P_j and P_i be the projection matrices of cameras C_j and C_i. Then, a pixel x rendered by C_i using P_i can also be back-projected to the closest 3D point X_i of the mesh. The back-projection function of C_i is called Q_i:

$$X_i = Q_i(x)\,. \tag{3.1}$$

Using Q_i and P_j, any pixel x from point of view i can be associated to a pixel $x_{i\to j}$ from point of view j:

$$x_{i\to j} = P_j X_i\,, \text{ with } X_i = Q_i(x)\,. \tag{3.2}$$

This process is illustrated in Figure 3.2a: every pixel x from point of view i is back-projected to its corresponding X_i and then projected to $x_{i\to j}$ in the point of view j. If performed on all the x pixels in i, it can be used to assign a pixel value $I_i(x)$ to their corresponding $x_{i\to j}$ in j and render a "reprojected" image $I_{i\to j}$. Alternatively, the exact same transformation can be used to assign to each x in i a unique pixel value $I_j(x_{i\to j})$ and render $I_{j\to i}$ (see Figure 3.4c) with Algorithm 1.



(A) Reprojection $I_{j\to i}$ with a removed object. (B) Textured shadows $S_{j\to i}$ with a removed object.

FIGURE 3.3: Removed object (i.e. absent from RGB image) is *textured* by the occluded background (pink) and introduces mild inconsistencies: $I_i(x) \neq I_{j\to i}(x)$. Its *textured shadow* is consistent with reference: $I_i(x') = S_{j\to i}(x')$.

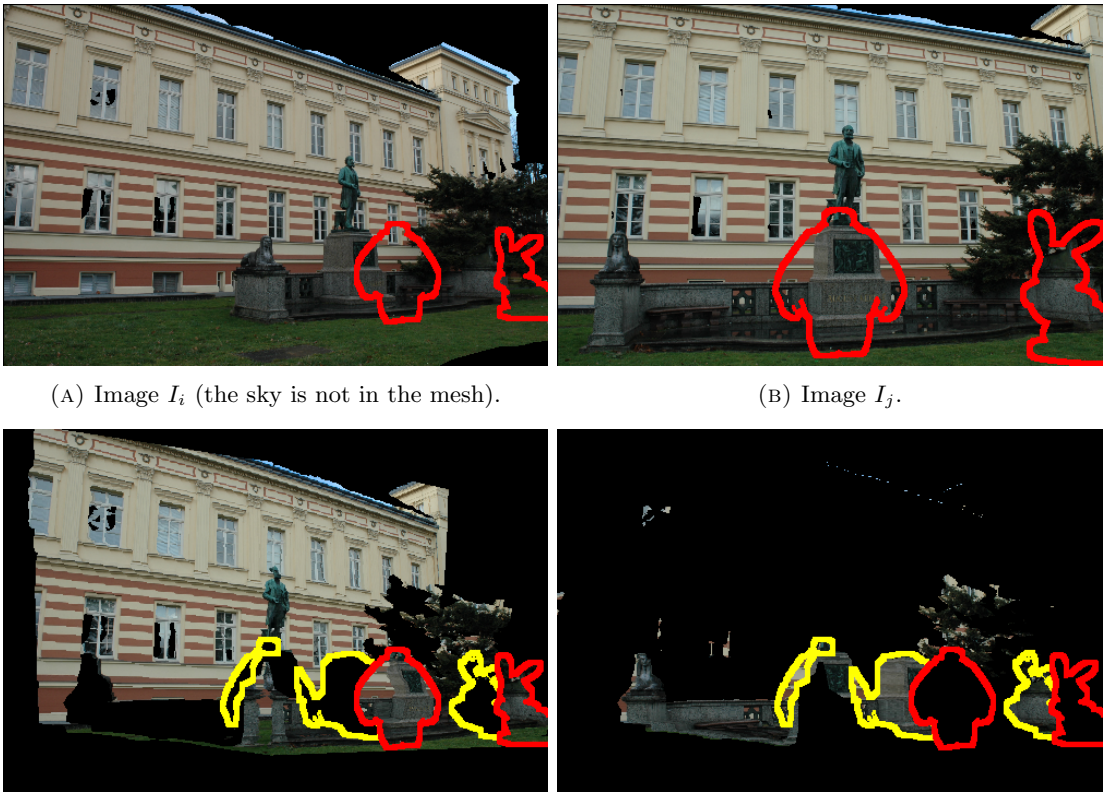
Using this alternative, every point of the "reprojected" image is given a unique value, obviating the need for a depth-buffer and the necessity to interpolate any pixels that would have remained blank after the transformation. Indeed, multiple x can be reprojected to the same $x_{i\to j}$, but all x reproject to some $x_{i\to j}$ if a corresponding 3D point can be found in the mesh.

When every pixel of the reprojected image is computed, some pixels are associated with 3D points that were occluded in the original view and therefore have no RGB value (see Figure 3.2a). We can check for such cases during the application of the transformation. For X_i as defined in Equation 3.2:

$$X_i \text{ occluded} \Leftrightarrow \|X_i - C_j\|_2 > \|X_j - C_j\|_2 \ , \text{ with } X_j = Q_j(x_{i\to j})\,. \tag{3.3}$$

In Equation 3.3, X_i and X_j can be different from one another as the latter is obtained by back-projecting $x_{i\to j}$ using C_j, and is by definition the closest point to the camera. Since they are on the same axis, X_j not occluding X_i means they are equal. In contrast, in Figure 3.2a $X'_i \neq X'_j$, meaning X'_i is occluded.

In existing methods, occluded 3D points will be discarded when computing 2D changes (see Figure 3.4c). Conversely, our method systematically assigns an RGB value to these points: the one associated with the occluding point. Rendering those points with these RGB values has the effect of creating a *textured shadow* $S_{j\to i}$ as in Figure 3.4d and 3.3 of the occluding points. Using Equation 3.2 and Equation 3.3 we can render $S_{j\to i}$ with Algorithm 2.



(A) Image I_i (the sky is not in the mesh).

(B) Image I_j.

(C) Image $I_{j\to i}$ (I_j seen from camera C_i).

(D) Textured shadows $S_{j\to i}$.

FIGURE 3.4: Reprojecting image I_j on image I_i (removed objects in red, corresponding occlusions in yellow).

Algorithm 1: Base pseudo-code of our image "reprojection" process

CreateReprojectedImage (i, j)
- **inputs :** points of view i and j; 3D mesh; image I_j
- **output:** "Reprojected" image $I_{j\to i}$
- $I_{j\to i} \leftarrow \emptyset$;
- **foreach** *x from point of view i* **do**
  - back-project x to its corresponding X_i ;
  - project X_i to $x_{i\to j}$ in the point of view j ;
  - $I_{j\to i}(x) \leftarrow I_j(x_{i\to j})$;
- **return** $I_{j\to i}$;

Algorithm 2: Pseudo-code of our "textured shadows" rendering process

CreateTexturedShadows (i, j)
- **inputs :** points of view i and j; 3D mesh; image I_j
- **output:** "Texture shadows" image $S_{j\to i}$
- $S_{j\to i} \leftarrow \emptyset$;
- **foreach** *x from point of view i* **do**
  - back-project x to its corresponding X_i ;
  - project X_i to $x_{i\to j}$ in the point of view j ;
  - **if** *X_i is occluded* **then**
    - $S_{j\to i}(x) \leftarrow I_j(x_{i\to j})$;
- **return** $S_{j\to i}$;

3.2.2 Photo-consistency in occluded pixels

When reprojecting, removed objects that are still present in the mesh have the effect of back-projecting the colours of the points they mask onto their surface (see Figure 3.4c, where the statue's podium is projected onto the middle red shape) and leaving those points untextured. Rendering the regions of occlusion using Algorithm 2 avoids this back-projection effect (see Figure 3.4d where the podium is correctly placed).

More generally, the *textured shadows* of removed objects will be photo-consistent [110] with the reference image i.e., they will fill holes in the warped image with accurate data (compare the yellow *shadows* in Figure 3.4d to the reference in 3.4a). However for unchanged objects, such back-projections will not be textured by occluded points but by the object itself, and therefore will not be consistent with the reference image (see

the lion statue's *shadow* to the left of Figure 3.4d). Our approach consists of looking for the regions of least change between the reference image and the textured shadow images from different points of view.

The delta maps $\delta_{j\to i}$ (see Figure 3.5a) are computed using the norm 2 distance between the RGB values of each rendered pixel in the textured shadow and in the reference images [63]. In order to account for the inaccuracies of the warping process, either in the camera pose or in the 3D mesh, the reference pixel's colour is compared to the colour of all pixels in its neighbourhood $\mathcal{N}$ in the warped image [187] and the minimum value is chosen:

$$\forall x \in \delta_{j\to i}, \quad \delta_{j\to i}(x) = \min_{y\in\mathcal{N}_x} \|S_{j\to i}(y) - I_i(x)\|_2 \,, \text{with } \mathcal{N}_x = \{y \in S_{j\to i} \mid \|y - x\|_1 < d/2\} \,, \tag{3.4}$$

where d is the neighbourhood size and $y \in S_{j\to i}$ if $S_{j\to i}(y)$ is rendered.

3.2.3 Photo-consistency with multiple points of view

As we will further detail in the following paragraphs, a single delta map per point of view will generally not contain enough information to accurately retrieve the 2D location and shape of a change. Firstly, the regions of least change computed in Subsection 3.2.2 are located within the *shadows* of the foreground objects rather than in their actual positions in the frame. Secondly, only the parts of a removed object that cast such *shadows* will be detectable, which is why using several points of view can enable the method to more thoroughly retrieve the shape of the object, by uncovering new parts of it with each additional view.

3.2.3.1 Foreground projection

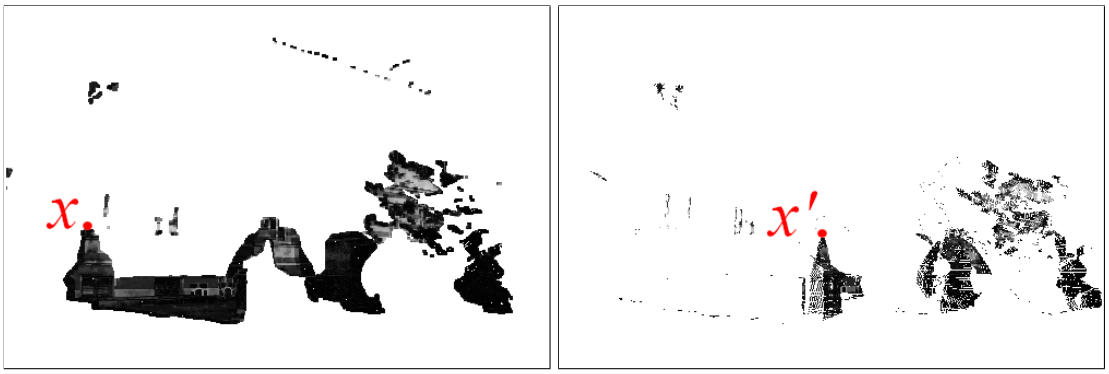
Before combining the multiple delta maps for different points of view, we first project the detected removals onto the foreground (see Figure 3.5b). For any pixel x of the delta map $\delta_{j\to i}$, we can obtain the corresponding pixel $x_{i\to j}$ in the original point of view j:

$$\forall x \in \delta_{j\to i} \,, \quad x_{i\to j} = P_j X_i \,, \text{ with } X_i = Q_i(x) \,. \tag{3.5}$$

Moreover, if X_i is occluded, back-projecting $x_{i\to j}$ will return one of its occluding points X_j. More specifically it will be the closest one to camera C_j:

$$\forall x \in \delta_{j\to i} \,, \quad X_j = Q_j(x_{i\to j}) \,, \tag{3.6}$$

which corresponds to the pixel $x'(= x_{i\to j\to i})$ in point of view i:



(A) Delta map $\delta_{j\to i}$. (B) Projected map $\Delta_{j\to i}$.

FIGURE 3.5: Once photo-consistency is evaluated in the occluded parts of the image, the potential changes are located in the foreground.

$$\forall x \in \delta_{j\to i}\,, \quad x' = P_i X_j\,. \tag{3.7}$$

We adapt Equation 3.3 to fit the point of view i and avoid rendering occluded objects (such as the bench to the left of Figure 3.4b visible in I_j and $S_{j\to i}$, but not in I_i):

$$X_j \text{ visible} \Leftrightarrow \|X_j - C_i\|_2 \leq \left\|X_i' - C_i\right\|_2 \text{, with } X_i' = Q_i(x')\,. \tag{3.8}$$

X_i' is the closest point to C_i that could occlude X_j. Equation 3.8 checks whether they are the same or not. We note that, as opposed to the reprojection process (Algorithm 2), this transformation is not reversible due to the repeated use of the back-projection, which always returns the closest point to the camera. In practice this means that not every pixel of the foreground is assigned a value (see the white *holes* inside the shapes of Figure 3.5b), while some pixels are given multiple values. The correct value is chosen using a depth buffer D_j associated with C_j and the foreground-projected delta map $\Delta_{j\to i}$, is then rendered using Algorithm 3.

3.2.3.2 Combination of projected delta maps

In $\Delta_{j\to i}$, not every pixel is assigned a value (see Algorithm 2 and 3 and the white pixels in Figure 3.5b). Therefore, we can define for each $\Delta_{j\to i}$ a binary mask $M_{j\to i}$:

$$M_{j\to i} = \{x \mid \Delta_{j\to i}(x) \text{ is assigned a value}\}\,. \tag{3.9}$$

Algorithm 3: Pseudo-code of our foreground projection process

ForegroundProjection (i, j)
- **inputs :** points of view i and j; 3D mesh; delta map $\delta_{j \to i}$
- **output:** Projected map $\Delta_{j \to i}$
- $\Delta_{j \to i} \leftarrow \emptyset$;
- clear(D_j) ;
- **foreach** *x from point of view i* **do**
  - back-project x to X_i ;
  - project X_i to $x_{i \to j}$ in point of view j ;
  - back-project $x_{i \to j}$ to X_j ;
  - project X_j to x' in point of view i ;
  - **if** *X_j is visible in point of view i* **then**
    - **if** $\|X_j - C_j\|_2 < D_j(x')$ **then**
      - $D_j(x') \leftarrow \|X_j - C_j\|_2$;
      - $\Delta_{j \to i}(x') \leftarrow \delta_{j \to i}(x)$;
- **return** $\Delta_{j \to i}$;

In order to uncover new parts of a potentially removed object, the combination of two projected delta maps with the same point of view involves the union of their binary masks. As for their values, the maximum per pixel of the two is chosen in order to reduce false positives. The maximum value is used to be more selective in the detection process. Foreground objects that are not removed could still share some RGB values with the background they occlude for a particular point of view (i.e., be photo-consistent), but it is unlikely they would for every point of view.

This approach is similar to the intersection process described in [136]. There, in a single delta map, when an object is inserted in a scene, changes are detected at the correct position of the object in the reference image and at an erroneous position resulting from the projection from another point of view (see Figure 3.2b). Since the actual 2D location of the change is the former, the erroneous positions are removed by intersecting two different delta maps, requiring 3 points of view in total. However, in this previous paper [136], the intersection process shrinks the area of the combined mask with every new point of view, instead of expanding it. This further reduces the chances of false positives but is not practical for the study of occluded regions, which potentially do not overlap for every point of view, even after foreground projection.

Using every available point of view, we define the combined delta map Δ_i as:

$$\forall x \in \bigcup_{j \neq i} M_{j \to i}\,, \quad \Delta_i(x) = \max_{j \neq i} \{\Delta_{j \to i}(x) \mid x \in M_{j \to i}\}\ . \tag{3.10}$$

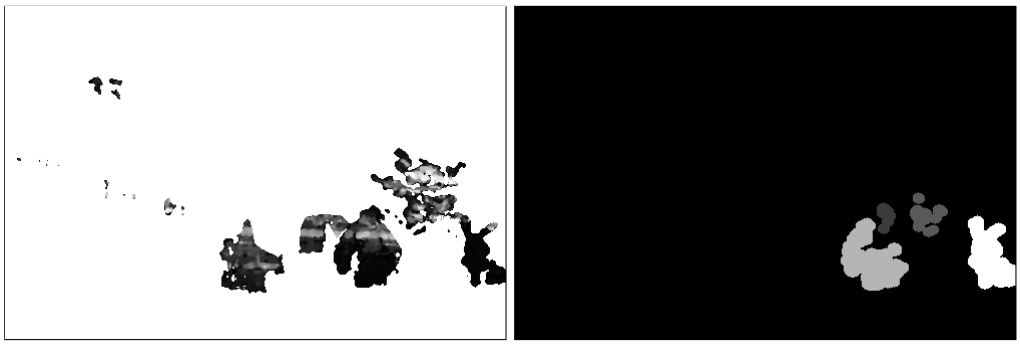
If x is not in any of the $M_{j \to i}$, then $\Delta_i(x)$ remains unassigned.

3.2.4 3D localisation of changes

Similarly to the process described in [136], the 3D localisation relies on the segmentation of the combined delta map Δ_i into 2D regions of detected change, and the matching of those regions from one point of view i to another. From these matched regions' locations in the images, we can infer the 3D location of the change and its spread, as detailed in the following paragraphs.

3.2.4.1 Segmentation by region

The segmentation is achieved similarly to [136], namely the generation of the regions' contours using [183] on a binarized Δ_i. Our contribution to this process is a more generalized binarization step that relies on a *triangle threshold* method described in [214], rather than an arbitrary constant threshold value. This *delta map binarization threshold* is the main sensitivity threshold for the change detection algorithm. The darkest pixels in Δ_i are selected as candidates for the removal detection since they describe the regions of least change in the *textured shadows*. Isolated pixels are then removed through erosion and the contours are generated. A final threshold on the area of the regions is used to remove the smallest changes [136] (see Figure 3.6b).

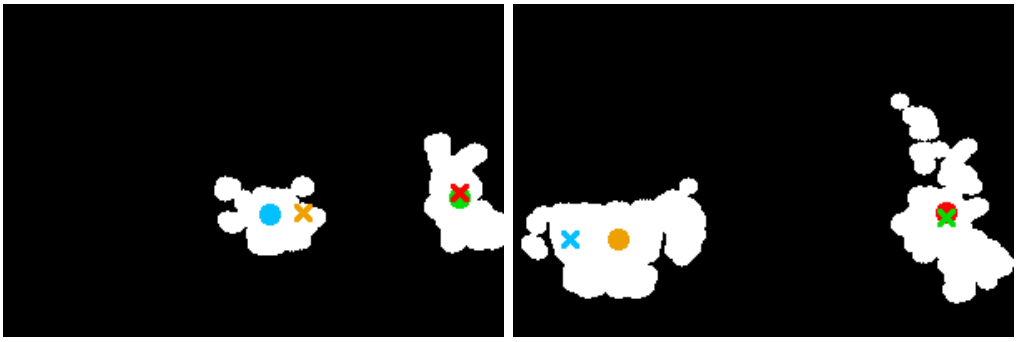


(A) Δ_i. (B) Detected change regions.

FIGURE 3.6: Segmentation in regions from point of view i.

3.2.4.2 Region matching

A 3D region of change that is visible for two or more points of view should have its projected 2D regions represented in several segmented delta maps. Although the 3D location of the changes could be retrieved through the use of back-projection, in practice the segmented maps are an approximation of the 2D projected changes, and a pixel-wise depth estimation would be inaccurate. Indeed, every reprojection introduces slight numerical errors, as they are based on a discretized 3D model of the scene (i.e. the depth maps). Therefore, even taking into account the previously mentioned erosion process, neighbouring pixels in an region could belong to mesh faces separated by large distances. This is why we use moments [85] to compute the (2D) centroids of each region and then the same triangulation process described in [136].



(A) Point of view i (detail). (B) Point of view j (detail).

FIGURE 3.7: Region matching for two points of view. Circles: centroids of the regions, Crosses: reprojected centroids.

Our method differs in the criteria used for matching the regions: instead of computing and comparing the HSV histograms of the regions in the corresponding images –as is done in [136]– we rely on the back-projection of the centroids on the mesh. This change is necessary because of the focus on the detection of removed objects, which by definition are present in the mesh but not in the images. This is a factor in the bias of detection toward inserted objects in [136]. If the back-projected centroid of a region in i is projected inside a region in j (or "reprojected" from i to j), and vice-versa, then the two regions are matched as in Figure 3.7: the orange and blue centroids belong to matched regions, as do the red and green ones.

The triangulation step produces 3D ellipses based on the size and location of the matched regions [136]. This final output is presented in Figure 3.8.

FIGURE 3.8: Left: Scene model without the "removed objects". Right: 3D removal detection: red ellipsoids mark the estimated location and size of changes.

3.3 Experimental evaluation and results

The algorithm from Section 3.2 was implemented in C++11 using the source code from [136] as a basis. Meshes and camera poses were handled using GLOW (OpenGL Object Wrapper) [9], mathematical operations were computed with Eigen [70], and images were processed with OpenCV [17]. The code is available at:

`https://github.com/InterDigitalInc/ObjectRemovalDetection`

Our method was evaluated on several scenes that presented some changes: at least one object is removed, and in some cases, some are inserted. The sequences are made up of five pictures that display the location of the removals, which are introduced by adding 3D objects to an existing accurate mesh of the scene (as seen in Figure 3.8). Conversely, insertions are simulated by removing objects from that mesh. Meshes were taken from two sources: the dataset introduced in [136], and the ScanNet [37] dataset[1], which also provide estimated camera poses for the images.

The scenes in the dataset from [136] already showcase one or more insertions. Each mesh is already associated with five pictures that display the inserted objects. However, this dataset has some limitations: there are approximations in the meshes that can be detected as changes and the camera distortion coefficients are not available for every camera used which leads to inaccurate projection.

In contrast, the ScanNet [37] images are noisier but have been corrected distortion-wise. The scenes do not contain any insertion, and all meshes are associated with a

[1] All images from the datasets are available in the appendix Appendix A.

video with thousands of frames, from which we picked five with the least motion blur possible to ensure that the camera poses were accurate.

Since we compare our method with the one from [136], the images were also chosen to showcase the location of the removed object we added to the mesh. For the sake of this comparison we also scaled the pictures to a width of 500px and used an area threshold of 50. Delta maps were computed using neighbourhoods $\mathcal{N}$ of size $d = 3$.

3.3.1 Quantitative evaluation

In order to evaluate the change detection quality, we used the same criteria as in [136] and [188]: for each image point of view, we have a corresponding 2D ground truth which we compare to the results of our 3D detection. The 3D ellipses are rendered in 2D using each camera pose and numerical results are averaged across multiple points of view. The following numbers were computed for the evaluation:

- IoU: area of the intersection of the ground truth and the 2D ellipse, divided by the area of their union,
- *coverage*: area of the aforementioned intersection, divided by the area of the ground truth, i.e. *true positive rate* (TPR),
- *false positive rate*: area of the intersection of the ground truth complementary and the 2D ellipse, divided by the area of the ground truth complementary.

For the scenes that contain inserted objects, we also took into account that the method described in [136] detects changes of all natures indiscriminately. Therefore, we subtracted the shapes of the inserted objects from the image comparisons between the ground truth of removals and the 2D ellipses. Consequently, any insertion that was correctly reported by the algorithm is not to be considered as a false positive for object removal detection.

3.3.1.1 IoU and coverage with automatic thresholding

The chosen criteria favour detection that is accurate in 2D but not necessary correctly localised in 3D i.e., if there are several 3D regions of change accurately detected in 2D but incorrectly matched together. Since this does not happen with our removal detection, using a 3D-based criteria could improve the performance of our method comparatively to the one in [136].

In most cases, our method is the most accurate for both criteria. The IoU is often greater than 40% but there are particularly difficult scenes where it will drop below

30% while the algorithm from [136] does not detect anything (such as in Figure 3.18). Generally these scenes will have a 3D mesh that is incomplete or too dissimilar from the images in areas that should have remained unchanged.

As explained in Subsection 3.2.3.2, our approach to the combination of delta maps is based on mask union rather than intersection. This makes the detection more robust for objects near the edge of the frame.

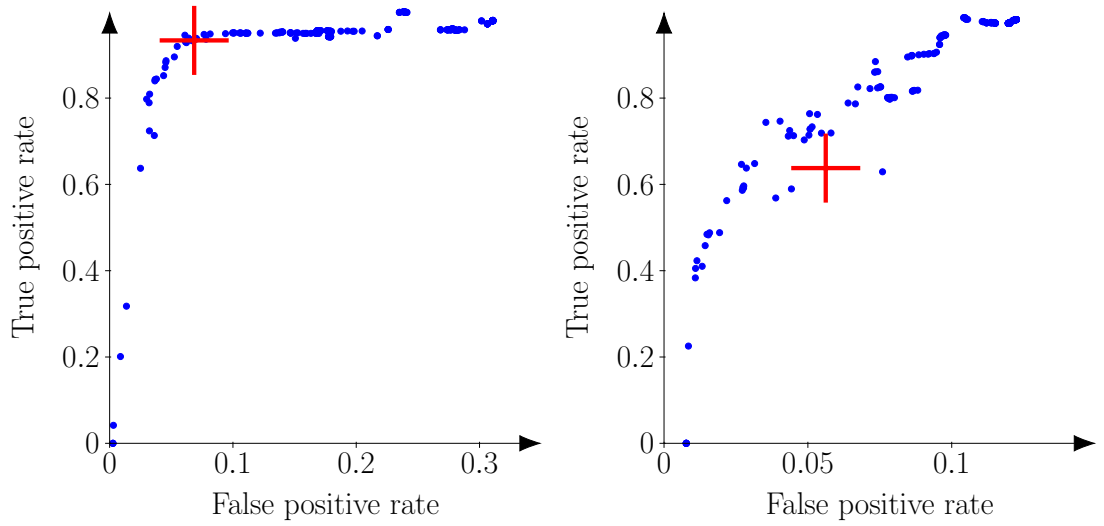
TABLE 3.2: Removal detection: quantitative results (shown in %).

Scene	Ours		Palazzolo et al.	
Palazzolo et al. dataset	IoU	TPR	IoU	TPR
container-shelf	**31**	**80**	0	0
container-shelf2	**39**	**100**	20	27
playground-car	**16**	**64**	5	34
statue-robot	**48**	**93**	14	17
statue-robot-bad-exp	**51**	**94**	3	3
statue-robot-bad-temp	**53**	**95**	1	1
statue-robot-temp	**59**	**95**	7	9
toilet-stone	**17**	**97**	0	0
Dataset average	**39**	**90**	6	49
ScanNet dataset	IoU	TPR	IoU	TPR
0000_00+plant	**11**	**20**	7	15
0000_01+box	**45**	**57**	0	0
0000_01-insert+box	**42**	**53**	1	1
0000_02+statue	**42**	47	22	**64**
0001_00+dollhouse	**6**	**18**	0	0
0001_01+table	**27**	**36**	21	31
0002_00+chair	**40**	**80**	8	20
0002_01+extinguisher	**15**	**66**	0	0
0003_00+cat	0	0	**23**	**47**
0003_01+desklamp	**61**	**69**	27	30
0004_00+ghost	**44**	**83**	8	9
0005_00+bucket	**62**	89	52	**89**
0005_01+pitcher	**27**	**100**	0	0
0006_00+lamp	**57**	**98**	38	82
Dataset average	**34**	**58**	15	28
Global average	**36**	**70**	12	22

3.3.1.2 ROC curves for different thresholds

The ROC (receiver operating characteristic) curves in Figure 3.9, are used to compare the *true positive rate* and *false positive rate* of a binary operator for different discrimination thresholds [188], which in our case is the binarization threshold in the delta maps mentioned in Subsection 3.2.4.1. The automatic threshold is highlighted on the curves to evaluate its performance and the other threshold values are all the integers between 0 and 255. A threshold of 255 means that only the most consistent pixels of a delta map are considered (near the origin of the graphs).

We note that the *false positive rate* never reaches the maximum value of 1 in the presented curves. This is due to the fact that ellipses are only generated for objects that cast *shadows* during reprojection, which generally only represents a small part of any given image. The value obtained for a threshold of 0, when any pixel in the delta map's mask is categorized as "changed" regardless of value, is the de facto maximum.



(A) statue-robot. (B) playground-car.

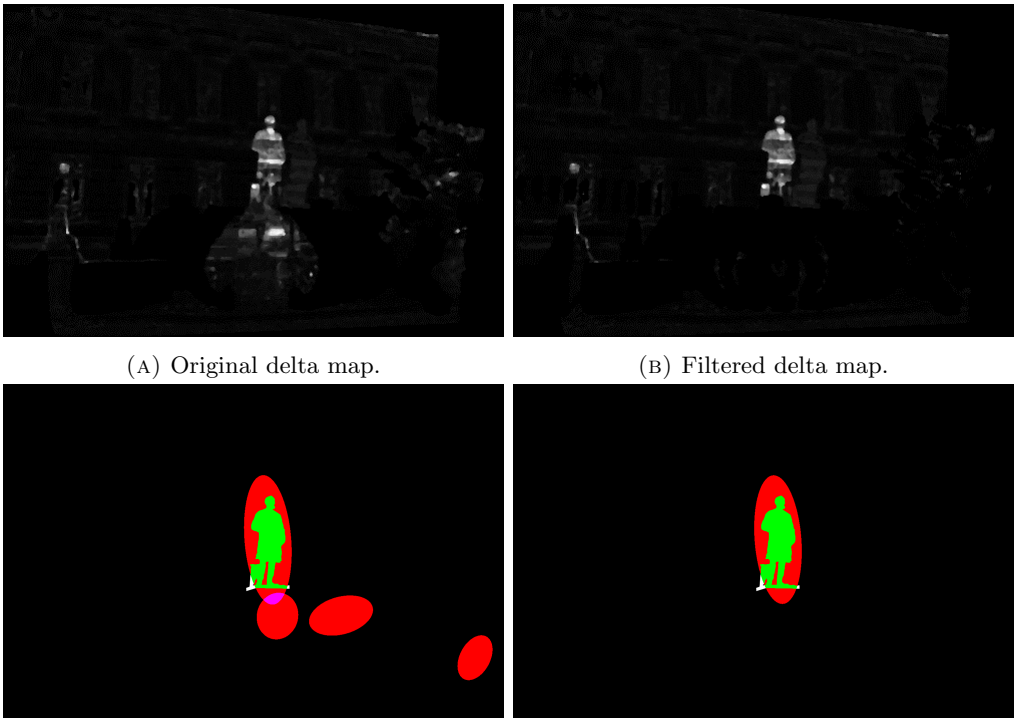
FIGURE 3.9: ROC curves. Blue points represent different threshold values, the red cross is the automatic threshold.

In these curves, the best results are located to the top left. The automatic threshold value is generally chosen among those ideal values, but there are instances such as in "playground-car" (Figure 3.9b) where it is at a local minimum. These discontinuities are a consequence of the thresholding by the changes' areas and the following segmentation into 2D regions.

3.3.2 Qualitative evaluation

3.3.2.1 Filtering of the insertions detection results

While the method described in [136] cannot reliably detect removed objects in the scene, it is also unable to differentiate them from inserted objects when detection occurs. To improve the performance of **insertion** detection, we can use the delta map produced by our removal detection process to filter the insertion delta maps. A naive approach consists of simply erasing the pixels from the insertions delta map where the corresponding removals delta map is positively identifying changes (as seen in Figure 3.10). In a real scenario, both an insertion and a removal could be occurring in the same 2D region of an image –the removed and added objects are aligned with the camera– but given enough different points of view, inserted objects would eventually be detectable. This is occurring in Figure 3.10b, where part of the insertion detection is removed due the filtering (the bottom part of the statue).



(A) Original delta map. (B) Filtered delta map.

(C) Final results without filter. (D) Final results with filter.

FIGURE 3.10: Part of the insertions detection pipeline, without (left) and with (right) filtering of the removals detection results in 2D (top). Filtering reduces false positives in 3D (bottom).

3.3.2.2 Scenes from the Palazzolo et al. dataset

In the following figures, on the left is shown the ROC curve for removal detection in a scene while the right show the 3D results for the automatic threshold.

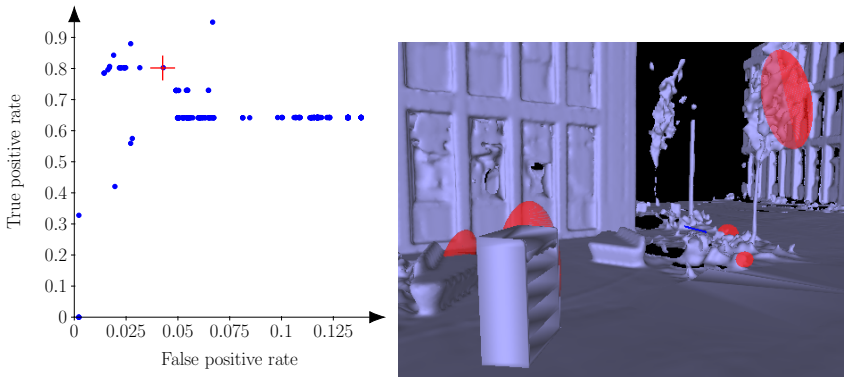


FIGURE 3.11: Removal detection for container-shelf.

The removed object is very close to the edge of the screen, making it harder to detect and leading to false positives on the trees in the background, whose geometries are approximated. The large inserted object does no affect the detection of removals.

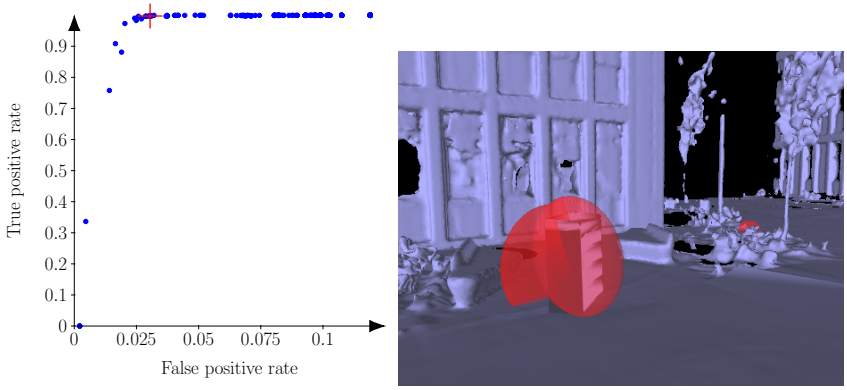


FIGURE 3.12: Removal detection for container-shelf2.

The same image sequence is used but the object is more centered in the frame, resulting in a much more accurate detection. The ROC curve is also less fragmented. The fragmentation is partly due to the grouping of changed pixels into regions, which can be discarded if their surface area falls under a certain threshold. The fragmentation occurs when there are multiple areas of detection (the trees and the object in the previous sequence).

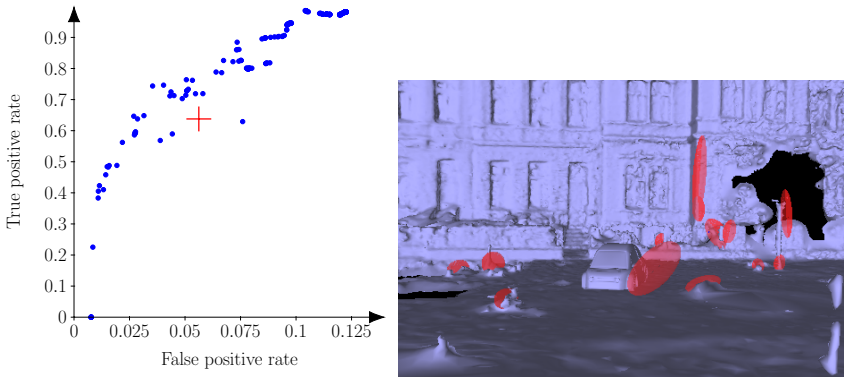


FIGURE 3.13: Removal detection for playground-car.

In the scene, an inserted object is in front of the removed object. As a result the object is only partially detected. The complexity of the scene also generates a lot of false positives because of the approximations in the mesh. In this particular scene, the automatic threshold is at a local minimum of performance. Also the results for the automatic threshold do not coincide with ones for any fixed threshold, as it can be different for each image of the sequence.

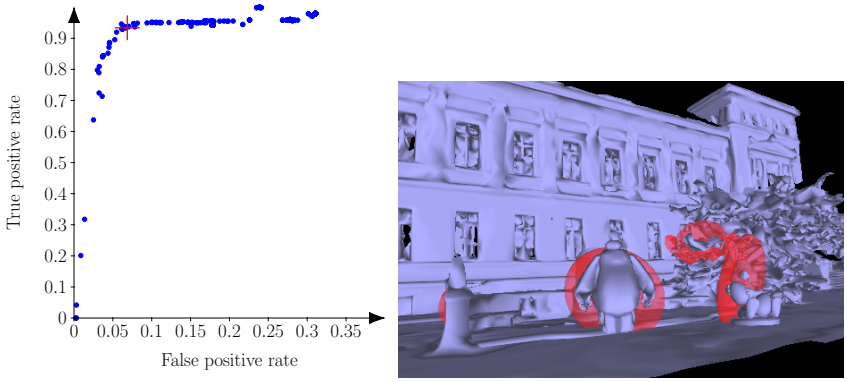


FIGURE 3.14: Removal detection for statue-robot.

In this scene there are two disjointed removed objects. A few false positives occur on the trees on the right side of the frame and the statue on the left. The false positives can only occur on foreground objects, which cause occlusions.

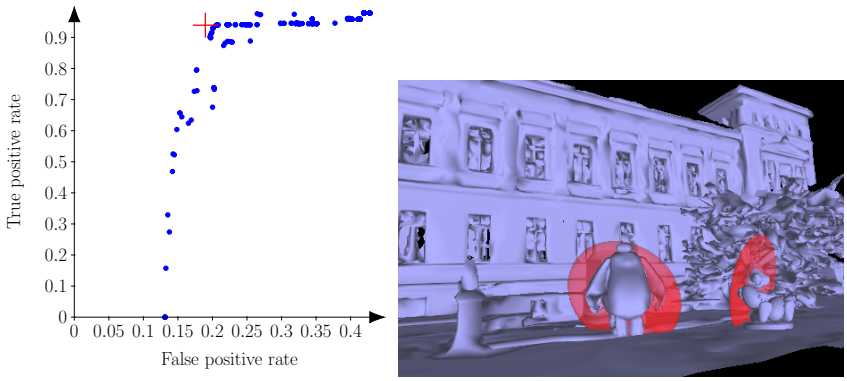


FIGURE 3.15: Removal detection for statue-robot-bad-exp.

This scene uses the same mesh and poses as the previous one, but the exposure in each image has been independently altered. This does not greatly affect the detection: the false positive rate is higher but it is the result of overestimating the size of the objects rather than detecting other objects altogether. The automatic threshold results are at a local maximum of performance: there is no other point to the top and left.

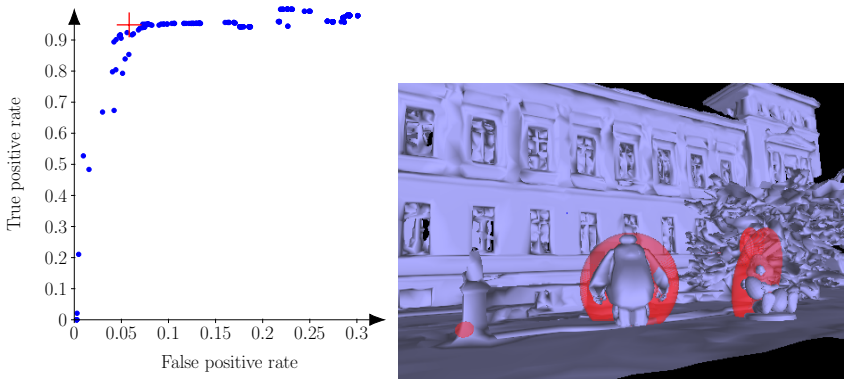


FIGURE 3.16: Removal detection for statue-robot-bad-temp.

This scene uses the same mesh and poses as the previous one, but the colour temperature in each image has been independently altered. This does not significantly impact the detection, and once again the automatic threshold results are at a local maximum.

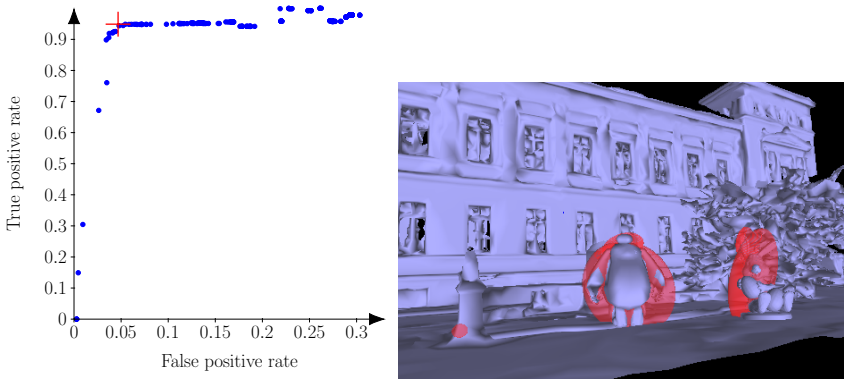


FIGURE 3.17: Removal detection for statue-robot-temp.

This scene uses the same mesh and poses as the previous one, but the colour temperature and exposure have been made more consistent across the images. This improves the results slightly.

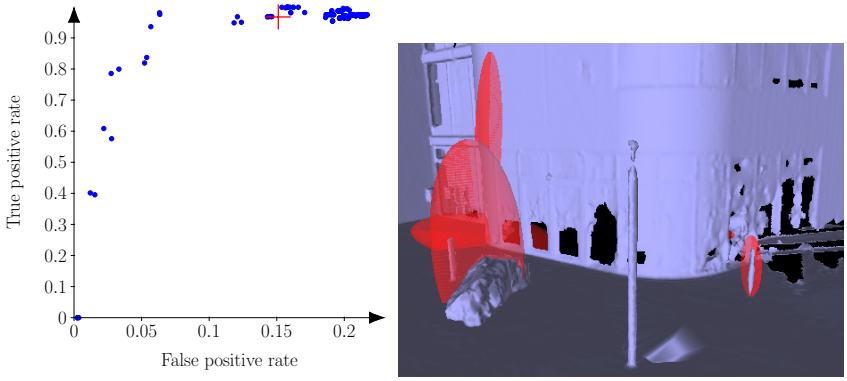


FIGURE 3.18: Removal detection for toilet-stone.

The object is detected but it is located in front of an inaccurately modelled part of the building, which greatly increases the false positive rate. It should be noted that a lower threshold would significantly improve the result.

3.3.2.3 Scenes from ScanNet dataset

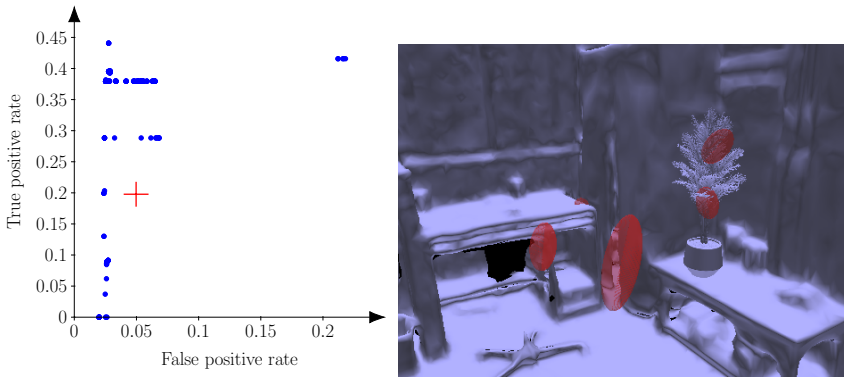


FIGURE 3.19: Removal detection for 0000_00+plant.

There is little parallax between the points of view as the camera is mostly rotating on it-self, and parts of the scene are inaccurately modeled. The true positive rate does not go over 0.45 signifying that some parts of the object cannot be detected.

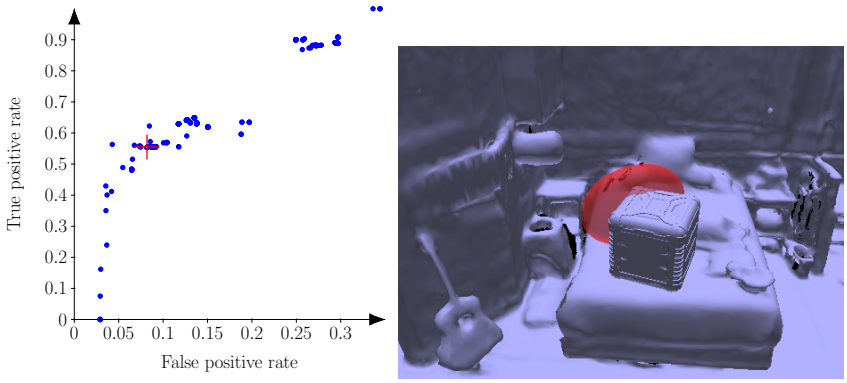


FIGURE 3.20: Removal detection for 0000_01+box.

The top of the object occludes the wall which is far away, while the bottom rests on top the bed. The parts of the object that are closer to the background are harder to detect.

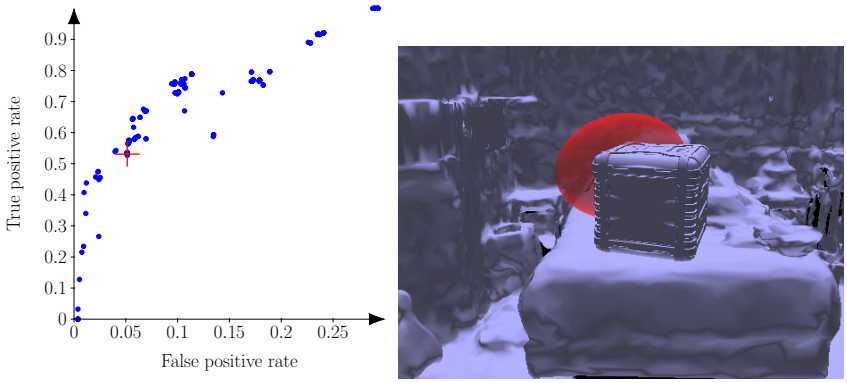


FIGURE 3.21: Removal detection for 0000_01-insert+box.

The presence of an inserted object does not significantly affect the removal detection.

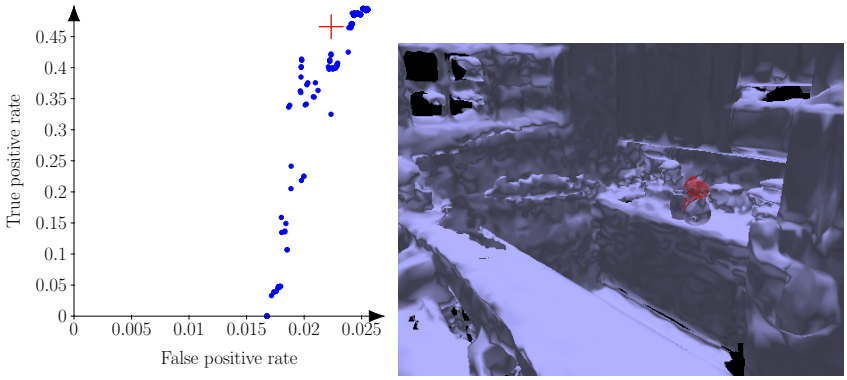


FIGURE 3.22: Removal detection for 0000_02+statue.

Even at the highest sensibility, some parts of the object cannot be recovered, they do not produce occlusions.

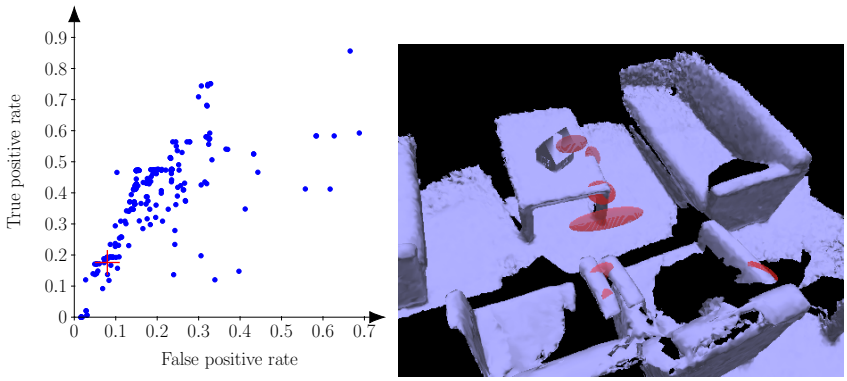


FIGURE 3.23: Removal detection for 0001_00+dollhouse.

The top of the object is the most subject to parallax and is the only portion detected. The false positive are caused by the photometric similarities between the objects of the scene and the background.

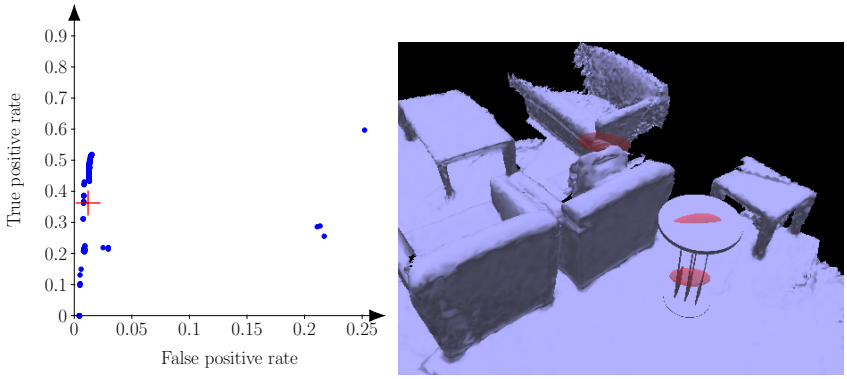


FIGURE 3.24: Removal detection for 0001_01+table.

The bottom of the table is merged with the background and cannot be recovered. However the legs are detected despite their narrowness.

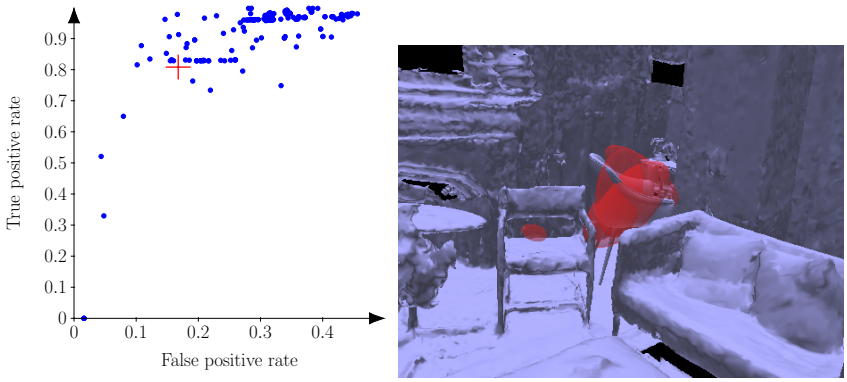


FIGURE 3.25: Removal detection for 0002_00+chair.

One arm of the foreground chair is wrongfully detected as removed since it does not perfectly align between the images and the mesh.

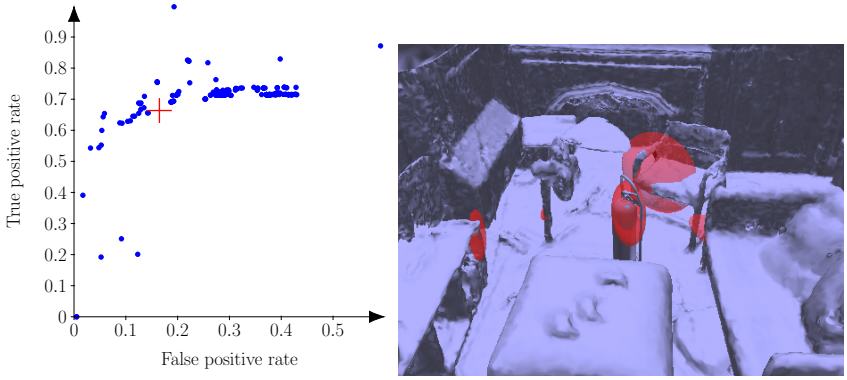


FIGURE 3.26: Removal detection for 0002_01+extinguisher.

Once again, one arm of the foreground chair is wrongfully detected.

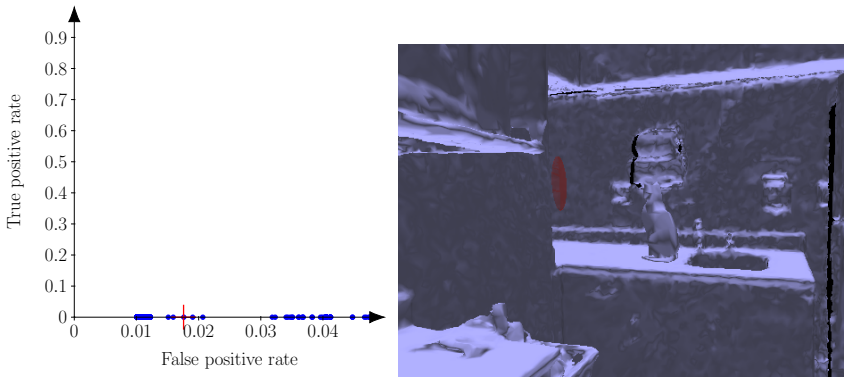


FIGURE 3.27: Removal detection for 0003_00+cat.

The object does not produce any occlusion because of the camera is mostly rotating on itself, creating little to no parallax.

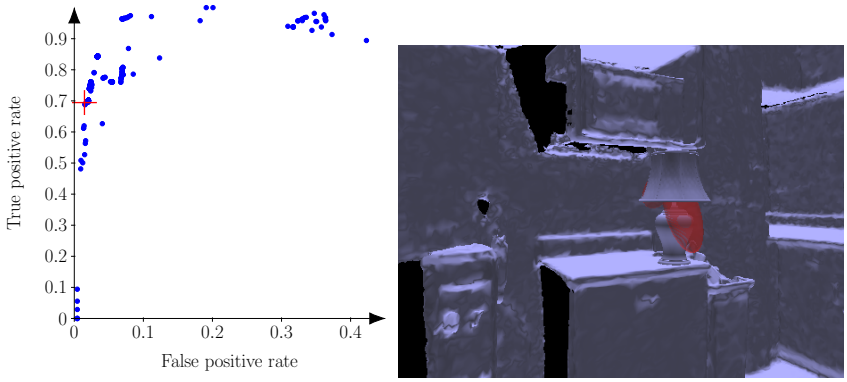


FIGURE 3.28: Removal detection for 0003_01+desklamp.

The object is rarely seen in full, which makes evaluating its size more difficult.

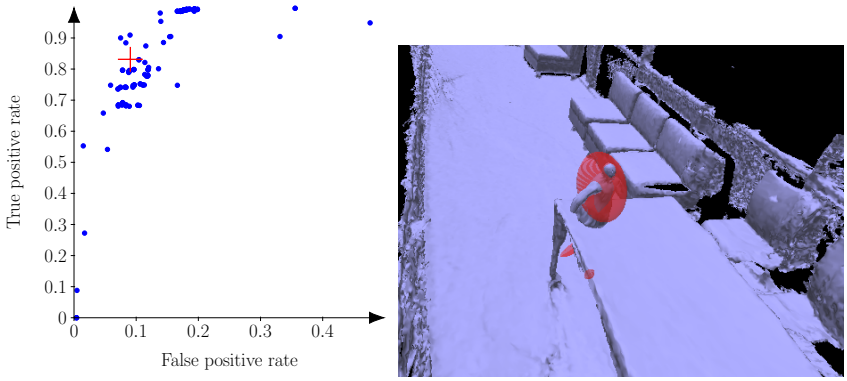


FIGURE 3.29: Removal detection for 0004_00+ghost.

The downward angle of the scene makes the base of the object harder to detect.

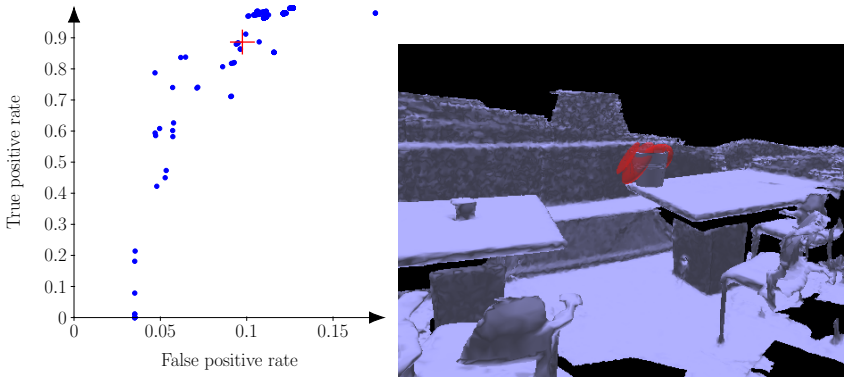


FIGURE 3.30: Removal detection for 0005_00+bucket.

The object is detected despite the incorrect camera pose estimation on the last 3 pictures.

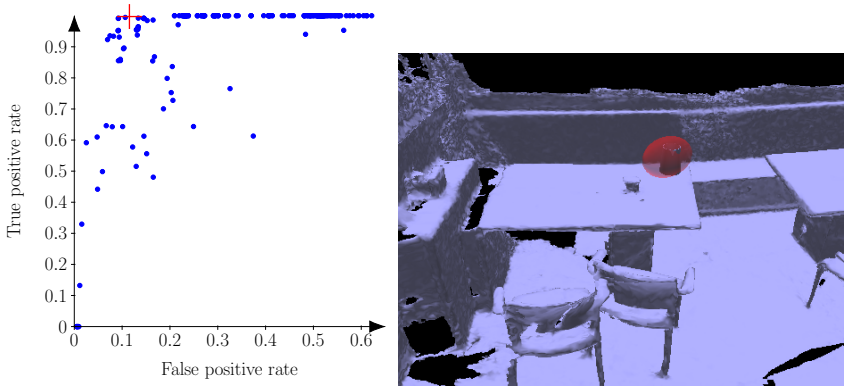


FIGURE 3.31: Removal detection for 0005_01+pitcher.

The power socket on the surface of the table does not produce a false positive, despite its very approximate 3D capture. However the size of the object is overestimated.

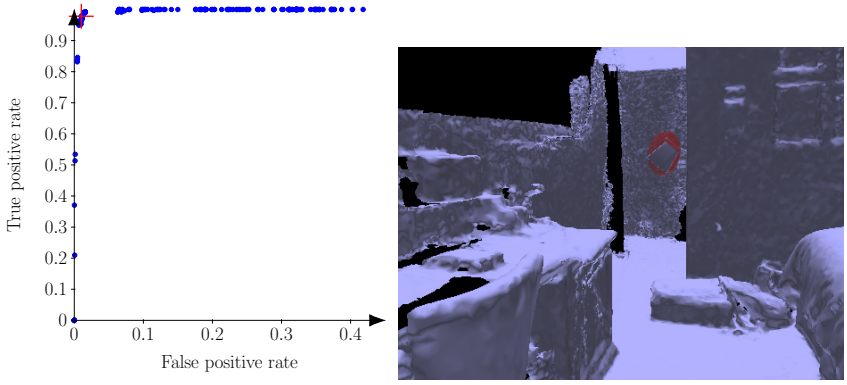


FIGURE 3.32: Removal detection for 0006_00+lamp.

In this scene the geometry of the object is very simple, and the object itself is in front of a distant background. Most threshold value are able to retrieve the object in its entirety.

3.3.3 Computation time

The methods were run on a CPU rather than a GPU to better reflect the limited hardware of portable AR devices. The execution time is on the same order of magnitude as

the one reported in [136] as *interactive time*. The processes of generating the delta maps and triangulating the changes in 3D never takes longer than a few seconds.

We note that of the two processes mentioned above, delta map generation is more computationally expensive. The reprojection operations as well as difference calculations on the images are each performed by looping once over the pixels each image. Empirically, we measured that the computation time was indeed proportional to the image resolution.

The computation time of the triangulation process cannot be evaluated as reliably, since it depends on the number of 2D areas detected in the first step. However, for a constant area threshold it will become less negligible when compared to the time of delta map generation, as the number of individual detected changes increases with the resolution. When the area threshold is increased in accordance with the resolution, this effect is less pronounced. The overall computation time is also tied to the number of images in the sequence, as every image is compared to every other one (i.e. for n images, $n \times (n-1)$ reprojections are made).

In a virtual machine with 16GB of RAM and four 2.60Ghz processors, both [136]'s method and ours process sequences of five 500 pixel-wide images in less than 3 seconds. Compared to the implementation by [136], the computation time of our method remains low despite the addition of the removal detection process. This is because reprojection –the most time consuming operation– can be performed for both occluded (textured shadows) and visible pixels in a single pass by storing the rendered pixels in the corresponding images (respectively $S_{j\to i}$ and $I_{j\to i}$). In other words: most of the computations necessary to produce the textured shadows were already made in the original method from [136], but their results were not exploited, since only the visible pixels are used. In practice, both the algorithm from [136] and ours can be used at the same time in order to detect both insertions and removals with more accuracy (as mentioned in Subsection 3.3.2.1). It should also be noted that our reprojection process uses less memory thanks to the absence of a depth-buffer as described in Subsection 3.2.1. The reprojected image also has sharper features since no interpolation is required as shown in Figure 3.33.

3.3.4 Qualitative results from GPU implementation

For the reprojection and delta map rendering processes, we have reimplemented the existing algorithms using shaders directly applied to the mesh instead of functions run on the depth maps derived from said mesh. In this configuration, the execution time becomes tied to the precision of the mesh rather than the image resolution, because no depth map has to be generated. When such shaders were run an NVIDIA GeForce RTX

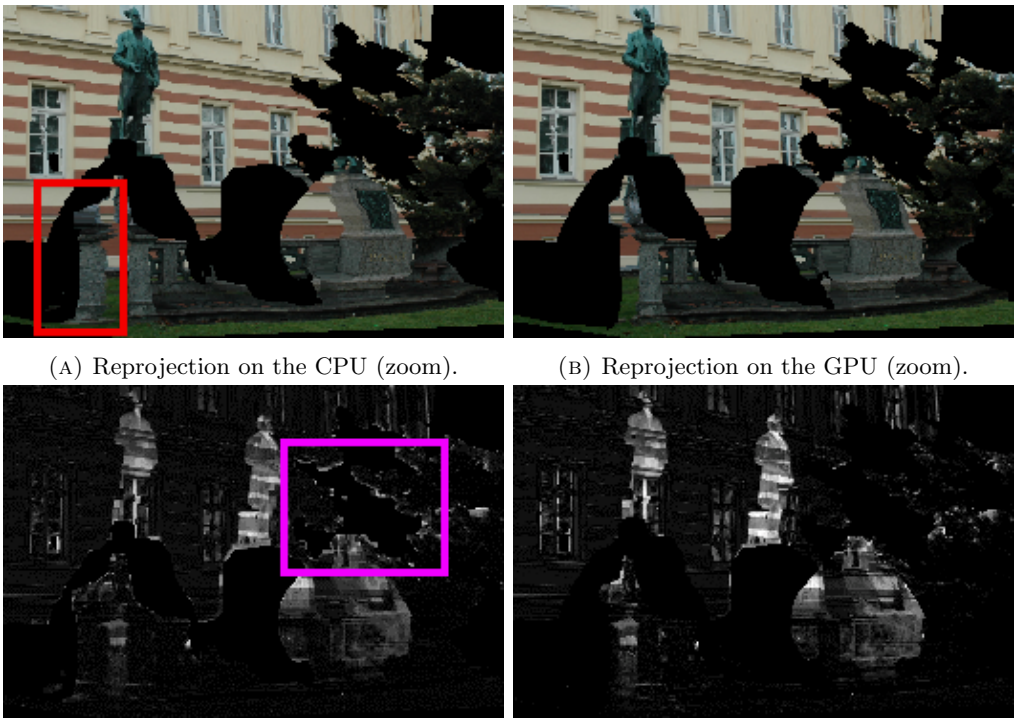
(A) Reprojection from [136], using direct transform. (B) Ours, using inverse transform (Algorithm 1).

FIGURE 3.33: The reprojection step being performed with different algorithms. The direct transform results in interpolated or missing pixels.

2060 GPU the computation time of the aforementioned processes was reduced by up to 95%.

Working directly with the mesh also eliminated the artefacts introduced by the pixelisation of the depth information. These include imprecisions over the computation of reprojected coordinates, as well as loss of definition in the geometries contours. The impact of this change is illustrated in Figure 3.34 for the object **insertion** detection pipeline, as it is more noticeable.

The foreground projection process is also more precise using the mesh, and allows us to avoid removal detection false negatives due to the presence of objects between the removed object and the background. In this case, as illustrated in Figure 3.35, the (consistent) textured shadows of the removed object are replaced by the (inconsistent) textured shadows of the objects behind it, making it undetectable. This can be avoided through a separate process of foreground projection of the reprojected images followed by the generation the delta maps. The inconsistencies between those delta maps provide the location of those removed objects at additional computational cost.



(A) Reprojection on the CPU (zoom). (B) Reprojection on the GPU (zoom).

(C) Insertions delta map on the CPU (zoom). (D) Insertions delta map on the GPU (zoom).

FIGURE 3.34: Parts of the insertions detection pipeline, as performed on the CPU and the GPU. During reprojection on the CPU, occluded geometry is wrongfully included (red square), and on the delta maps the contours of objects are more likely to produce false positives (purple square).

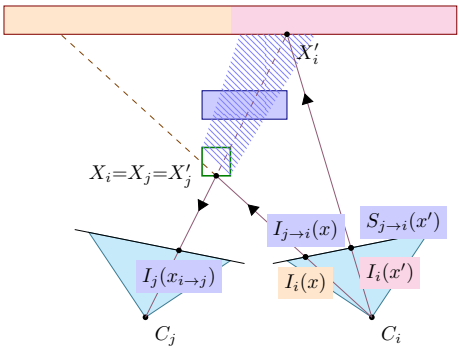


FIGURE 3.35: The removed object in the foreground is undetected because the object behind it creates inconsistencies in the textured shadows.

3.3.5 Limitations

Only studying regions occluded by a foreground object has a few side-effects. For instance, false positives can only be detected on objects that produce such occlusions. These false positives will only occur on the objects that are the most photo-consistent with their background (see "0001_00+dollhouse" Figure 3.23). Moreover, any removal accurately detected in 2D will be accurately localised in 3D. This differs from insertion detection where accurate 2D regions of change can be wrongly matched with other regions from a different change in another point of view.

In a scene, reflective surfaces might still pose a challenge. Instead of being detected as false positives as in [136], they can make removed objects situated in their foreground difficult to detect i.e., generate false negatives.

Change size estimation can be an issue, as the detection size is proportional to the occlusion size of the removed object (see "0000_02+statue" Figure 3.22). If a removed object does not produce any occlusion it will not be detected by this method. Such an object could be detected using the method from [136] if it is far enough from its background to greatly distort its textures.

Not every image of the ScanNet dataset's sequences is perfectly aligned with its corresponding 3D mesh (see "0005_00+bucket" Figure 3.30). While this has not severely impacted the results of the detection in our experiments, in theory a misalignment of an image will generate 2D false positives and negatives for its point of view. On the one hand, false positives are still dealt with by using the other images of the sequence. On the other hand, false negatives can negatively impact the detection by reducing the areas of detection and lead to 3D false negatives or inaccurate size estimation. However, they occur less often, since they require that a removed object aligns with an object still present in the image.

Appendix A contains the image sequences used for all the scenes in the datasets and the resulting 2D change detection maps from which the errors are calculated.

3.4 Conclusion

In this chapter, we introduced a new approach for detecting the removal of objects between an outdated 3D mesh and a set of up-to-date pictures of a scene. The technique is based on the projection of the pictures' foreground textures to the background as defined by the 3D geometry and the pictures' points of view, and the comparison of the resulting images to the unmodified set of pictures. The definition and study of the foreground make our approach distinct from other reprojection-based approaches, which are not as effective at removal detection due to the low photometric disturbances caused by this type of change. The focus on object removals also simplifies the process

of translating the results back into the 3D world, as we know their geometry is still present in the outdated mesh. It therefore allows us to make inferences on 3D geometry from images in which said geometry is absent, without relying on expensive depth-estimation methods [56]. The technique is able to perform well even in environments with uniform textures or changes of different natures (the adding or removal of geometry) while remaining as fast as the state of the art methods that are meant to run on devices with low computational power. Thanks to its short computing time, this process can be repeatedly used over time, leading to an extension of the set of images used to describe the current scene. This would yield cleaner results –filtering out false positives– or uncover different layers of changes in the scene –observe the environment from new vantage points.

The results could be improved by basing the 3D shapes of the changes on the geometry of the mesh, rather than relying on standard shapes like ellipses. With more accurate shapes, the mesh could directly be altered to reflect the detected changes. This shape-level segmentation of the scene could be estimated through the study of the mesh's geometry and its properties (faces' sizes and normals, convexity, distance between vertices etc.). However it could be made part of the initial scene representation, and rely for instance on a semantic description of the scene's contents [151].

With removals now specifically identified, it is possible to ignore their impact in the change detection algorithm proposed in [136], as the slight photometric disturbances they introduce during reprojection can be filtered out. As shown, this improves the detection of inserted objects in scenes, but also allows for the categorization of changes according to their nature: "insertion" or "removal". This categorization can be expanded on by including the notion of "displacement" of an object, when it is both removed and inserted into a scene. This would necessitate the ability to track said object over time, which could be accomplished photo-geometrically or through semantic analysis, as will be discussed in the next chapter.

Chapter 4

Semantic description of the scene

With the implementation of a full 3D CD framework based on the work presented in [136], we introduced a new reprojection method [161] in Chapter 3 and used geometry-based matching in order to specifically detect and localise the removal of objects. This was accomplished without compromising the efficiency of the original work [136], as the new reprojection exploits computations that were already necessary but not fully taken advantage of. This focus on removals also allowed us to filter the results in the original CD process, resulting in a more accurate object insertion detection. The method returns as results 3D ellipsoids, which are produced through the detection and matching of 2D areas of changes across fives images.

While the matching and 3D localisation of the regions resulting from object removals can be performed with knowledge of the relevant 3D geometry in the scene, the process is still texture-based for inserted objects, making it less reliable than its removed object counterpart. Moreover, in both cases, the 2D region extraction also relies on the assumption that different objects in the scene have different textures. Parts of objects that are too photometrically similar to the background will not be detected if they were just inserted into the scene, or will falsely be identified as "removed" if they were present in the previous state. Inaccurate 2D or 3D estimation of the shape of changed areas prevents us from correctly updating the scene model, even when said shapes are improved through techniques such as active contours [2, 108], as those still require reference 3D geometry, which is unavailable for inserted objects.

4.1 Semantic analysis and motivations

In practice, the changes are generally correctly localised in 3D using our improved method [161], but the ellipsoids used to represent their shapes are generally not precise enough to accurately modify the mesh. We attempted to improve the geometric precision

of the removal detection by using a depth-based contour detection and flood-filling approach on the intermediary 2D results of the method (shown in Figure 3.7 of Chapter 3). While this made the CD framework perceptive to previously undetectable removed 3D geometry, it also led to more false positives, as it became an issue of correctly segmenting the geometry of the scene solely based on the depth maps. Likewise, insertion detection would not be improved with a preliminary color-contour detection, if the uniformity of the textures is the main issue. Therefore, in order to obtain a better object-level understanding of the scene, we decided to rely on a semantic analysis of its contents.

4.1.1 Object-level modelling

As mentioned in Chapter 2, the rise of Machine Learning and particularly Convolutional Neural Networks (CNNs) has led to the achievement of pixel-wise CD [22, 38, 71], where photometric techniques would have struggled. Such tools allow for captioning [138] or pixel-wise labelling of the nature of the changes [16, 30], of the objects affected [198, 42], or both [39]. However we also saw in Chapter 2, that they do not provide results pertaining to the 3D geometry of the scene, and that the cost of estimating said geometry from 2D inputs without a 3D reference is prohibitive –hence the heterogeneous approach taken in Chapter 3.

Given the amount of work dedicated to semantic analysis of images [154, 82, 102, 208], this can also be used as a preemptive step to change detection, effectively translating the scene into a more easily comparable representation. For instance, in [95] the segmented maps are compared instead of the original images, and in [42] CD is performed in the feature-space of a network pre-trained for image segmentation. It should also be noted that CNNs have more recently been used for 3D point cloud processing [21, 5, 115].

In order to improve the 3D results of our method [161] in Chapter 3, the 2D maps provided by image segmentation networks [154, 82, 102, 208] could be used to filter out the noise –or false positives– in our 2D detection maps, or as a contour extractor for inserted objects. Being able to successfully identify objects across different scene states could also alleviate the need for accurate geometry estimation of the removed objects, if the 3D prior were appropriately segmented. Estimation and insertion of new geometry would remain difficult with few images, but the results would be more accurate. Finally, an object-level understanding of the scene and of its changes provides many features relevant to the final MR application, such as physical interactivity [24, 25], agent interactivity [32], and improved object positioning [111].

4.1.2 Feature-based localisation

In most MR applications, the user only observes a partial view of the world at any given time, that must be localised with respect to the prior representation said world (i.e., its state when previously scanned) to add content or fix inconsistencies through CD. Previously mentioned CD works make use of GPS data [189] followed by a refinement process, assume co-registered representations [137] while sometimes allowing for slight misalignment [71], or ignore the issue of localisation [136]. However, with semantic analysis of the scene, registration becomes easier.

There exists literature on feature-based localisation, particularly for SLAM problem, that involve CD as a means for robustness [131, 45]. Related works also include the place recognition process cited in [182], which is able to match images based on their contents using a pre-trained CNN classifier. Some localisation works also make use of a graph representation of the scene: for instance [150] uses the Kuhn-Munkres algorithm on graphs generated from the objects present in each image to detect loop closure, i.e., check if an area is being revisited to correct accumulated estimation errors in the pose. Other works use pose graphs in order to organise a multitude of relative pose estimations [159, 87] and infer localisation from it.

With the parallelization of the CD and localisation tasks, there is an opportunity to improve both processes by avoiding a negative feedback loop: localisation without CD can wrongly assume which landmarks are displaced between sessions, leading to incorrect pose estimation and following CD. It should nevertheless be noted that in those works, CD is a by-product of the main task of localisation, and is prone to false positives. Indeed, the purpose is only to find reliable landmarks in the environment, similarly to the static maps [51, 4] mentioned in Chapter 2. The scene representation required for MR applications will necessitate a more exhaustive description of its geometry, and ideally of its semantic contents.

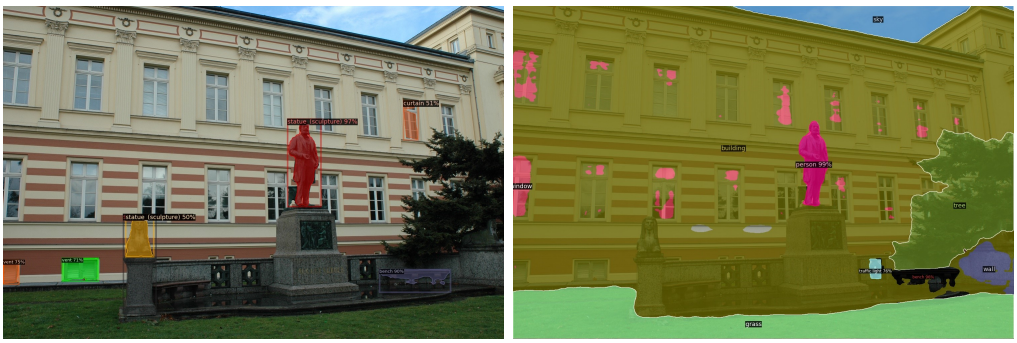
4.1.3 Explicitly defining the relevancy of changes

Using semantic analysis, we can set the level of detail or define the changes that are relevant to the maintenance of our scene representation [79]. Indeed, we have seen that CNN-based CD methods [71, 198] use semantics to filter out photometric inconsistencies due to view-point or illumination variation between scene states. The ability to distinguish between different sources of photometric (pixel value) changes in images can improve the quality of detection when focusing on any of those sources (illumination, geometry, texture, semantics etc).

Through object-level understanding, suspected geometric changes that result from the inaccuracies of the MR device's sensors can be ignored. This is particularly useful

where there is a large difference in data quality between the reference model and the current observations, or when the computational cost of CD and updating is non-negligible. Indeed, with the development of LiDAR technology and multi-view stereo on consumer devices, 3D geometry can be directly monitored, but the quality of the representation would inevitably degrade as the environment changes and updates are applied based on the low-detail readings of the devices' sensors. If changes can be understood as the manipulation of objects in the scene, the same can be done to its representation, preventing the loss of detail in the original photo-geometric contents, and allowing for faster updates and lighter storage.Scanning the scene directly –in 2D or 3D– would only be required to capture previously unknown objects.

In the context of MR, any change in geometry that affects the blending of virtual and real elements must be accounted for (photo-realism), as well as higher-level changes that will affect the MR experience (behavioural realism). This "exhaustivity" constraint is where the semantic analysis works mentioned in this chapter fall short, as their vocabulary is limited by their training data –as seen in Figure 4.1 where object detection is not exhaustive (left) and "panoptic" segmentation [102] (segmentation of both object instances and semantically consistent areas) lacks precision (right). To achieve exhaustive CD, we will need to structure the available semantic information in a way that allows for the identification and tracking of unknown object instances, when all that is available is their appearance.



(A) Object detection trained on the LVISv0.5 dataset [73].

(B) Panoptic segmentation [102] trained on the COCO dataset [121].

FIGURE 4.1: Object-based semantic image segmentation, limited by the data the networks are trained on.

4.2 Graphs to structure the scene information

When semantic segmentation tools such as Detectron2 [208] are used to improve our geometric understanding of a scene, we are ultimately limited by the number of object classes that can be detected. In order to avoid those limitations, we now focus on characterizing the contents of a scene using the objects' identifying features and relationships to one another, as recorded in a Scene Graph [91] (SG). A Scene Graph can be used to represent a 3-dimensional scene state, which itself is described by natural language, sensory readings, or visual representations such as images and 3D models.

4.2.1 Linking semantic information to photo-geometric data

In the works mentioned in the previous section, the semantic analysis results were provided in the shape of pixel-maps or 2D bounding boxes corresponding to each object or regions identified. For instance, in the ScanNet [37] dataset of 3D annotated rooms, the semantic segmentation information is directly stored as indexed color maps of multiple angles of each room, which are then used as textures for a 3D annotated mesh. A much lighter representation was introduced in [91], called Scene Graphs: graphs where the nodes are objects in the scene, often with attributes, and the edges are semantic relationships between those objects, such as the ones in the Visual Relationship Detection dataset [126]. While there are CNNs designed for relationship extraction [215], other architectures [210, 116, 186] directly produce graphs from images, making use of the Visual Genome dataset [105].

In Section 4.3 we will explore the many applications of such data structures, but for our purposes, SGs are a means to efficiently store and analyse semantic information as it relates to the geometry of the scene. This can ultimately lead to better means of interactivity between the virtual and real actors, and provides the medium to achieve the lightweight geometric maintenance mentioned in the previous section.

4D graph representations which add a time attribute [18, 74] also allow for the tracking of entities (areas of land in [18] and semantic data in [74]) over several iterations, where 3D representations can only be used to compare the current and previous states. Being able to monitor changes in the scenes over longer periods of time means that repeatedly disappearing and reappearing geometry can be reused without rescanning. Overall the CD task would have a larger prior to base its results on, and infer the variability of particular objects [125].

4.2.2 Accounting for unknown objects

In a Scene Graph, the nodes representing objects can contain more information than the object's label, allowing us to differentiate between multiple instances of a similar object in a scene through appearance or context. This is beneficial for tracking specific objects between states, going beyond the simple model of insertion and removal of geometry, but also to provide information to differentiate between semantically unrecognisable or novel objects. Indeed, when nodes need to be compared in semantic graphs, the similarity metric can be a combination of the similarity of the nodes' attributes [100, 213] and that of their neighbours [180, 61, 124, 72]. Incremental learning and anomaly detection could expand the semantic vocabulary of the scene analysis method, but it could lead to catastrophic interference [130] (abrupt forgetting of previously learned information), and therefore be unreliable over long periods of time.

In order to fully segment the scene, methods tailored to specific object classes can be more effective, and their results can be directly stored in the SGs. For instance, plane detection in 2D [152, 76, 41, 124, 174], 2.5D [176] or 3D [52] can be used to detect the boundaries of an indoor environment: the walls, floor and ceiling; using semantics, vanishing points or perspective correction methods. These surfaces can be hard to classify and localise using object detection networks due to their uniformity and size. It should also be noted that by detecting objects and planes separately, each process could improve the others' result. Masking the planes from the images during object detection could increase the precision [103] or improve the efficiency [204], as noisy textures could be removed and smaller objects become visible. Conversely, 3D data could be filtered to remove the objects' points before the plane detection.

In the absence of semantic labelling for some 2D or 3D geometry in the scene, the shape of the underlying objects can be inferred through observation of their known surroundings and be tracked by their geometric attributes. Depending on the available data, more general photometry [209] or geometry-based [194] segmentation tools would help in reducing the size of the unknown entities, making them more trackable. The resulting geometry can be understood as "stuff" or "things" [19] in the context of semantic understanding, and the concept of "blank node" [207] is used to describe such unknown entities in Resource Description Framework (RDF) graphs. RDF graphs have been introduced to organise data for semantic web applications, and blank nodes serve as structural elements to organise pieces of knowledge relatively to each other. Recent works in robotics have started to apply this concept to scene graphs [175].

4.2.3 Acting as an intermediary representation

Finally, another benefit of SGs is their ability to be easily parsed by humans and machines, and to be used as a "translation medium" between different representations. This will be further illustrated in Section 4.3, as their applications to scene authoring and Human–machine Interaction (HMI) will be shown, which are both related to the MR field.

In graph-based localisation methods, such as X-View [61], the graph structure is used as a means of converting RGB-D data captured from different perspectives –by a flying drone or a ground-based vehicle– into comparable spatial representations of the scene. In robotics, it is used as the intermediary representation between a controller's commands and the internal understanding of the scene by the agents [213, 100, 160].

SGs can be used to generate text from images [211], or caption changes [120], while remaining close to the disposition of real and virtual assets in the scene. The focus on interactivity and clarity makes the SG particularly suited to scene representation in MR applications, where the goal is always to provide a human user with an immersive and convenient experience. This ability to serve as a intermediary for switching between different scene representations (images, text, 3D geometry...) further benefits applications with narrative or educational goals. SGs are able to represent scene states gathered from heterogeneous data –be it gathered from various sensors or synthetic– and making such states comparable for tasks such as localisation and CD.

4.2.4 Conclusion: Representation requirements for MR and CD

For MR applications and based on recent technological advances, we can now assume that the capture device is able to capture RGB images and depth maps with their associated camera intrinsic and extrinsic parameters, relative to where the MR session was started. The prior scene representation against which changes are detected would contain the scene geometry necessary to achieve **photorealism**, and associated semantic annotations for **behavioural realism** [77]. The corresponding information for the current state of the scene would have to be inferred from the limited perspective of the capture device, whose interpretation could be facilitated by prior knowledge.

Both the semantic and geometric layers of the scene representation can be organised using a SG, where nodes describe objects' and edges' spatial or general relationships. Some the objects could be semantically labelled using 2D [154, 82] or 3D [21] processing, but unknown objects would still be identifiable through stored geometric and photometric attributes, as well as their neighbours' (defined by the edges). The relationship described by the edges have to be viewpoint-independent, in order to be usable in 3D.

In conclusion, SGs can effectively describe which changes are considered relevant with regards to the application. On the one hand, changes in the scene that impact the experience should be represented by modifications to the graph's topology or to the nodes' attributes. On the other hand, every element present in the graph should be maintainable using the available information on the current state of the scene, as to not diverge from reality.

In the following section we will survey works relating to scene graphs (generation and application) that are relevant to solving the problem of CD for MR applications.

4.3 Survey: Semantic Graphs for scene understanding

In the field of Augmented Reality, SGs may represent two different types of scenes: the real one and the virtual one. As outlined in [163], a "scene graph" can be used to refer to the representation of the latter: an organised collection of virtual assets to be inserted into the scene. The graph used to represent the real world is referred to as the "world graph" and organises the various "trackables" (fiducial markers, objects, landmarks, etc.) in the scene in order to, for instance, localise the device. Both graphs can contain "anchors" which are used to interface between the real and virtual contents: they are locations defined relatively to the trackables in the world graph, and around which assets are organised in the scene graph.

Our change detection task relates to the study of the real scene and therefore to the "world graph", however, works relating to either category (virtual or real scenes) are relevant to our application. These works all refer to the graphs they use as "scene graphs", since "world graphs" are a notion specific to the AR field. Consequently, the following survey will also refer to them as "scene graphs" (SGs) when no distinction is required.

4.3.1 Prescriptive graphs for scene authoring

SGs were introduced as a database searching tool in [91, 60, 153], but have been used to manipulate [46, 181] or generate [92] images. Their structured nature make them an efficient scene authoring tool when semantic constraints are placed before strict geometric ones. The graph representations used for this purpose are prescriptive (i.e., a set of instructions) rather than descriptive [36].

In the context of 3D building software [33] or physics engines [75], graphs –and specifically "trees"– may be used to organise the different assets of a virtual scene. For example, spatial coordinates or material properties may be passed down the branches of an asset management tree, but the object-nodes may also be connected by edges

describing relative transformation matrices. The graph is simply a representation of a set of constraints between files or models as defined by the user in the scene [36].

By definition, these SGs provide an exhaustive representation of the scene while relying purely on geometric and photometric data [179, 98]. Similarly, updates to the topology or the nodes' attributes are directly reflected in the scene, and there is little need for performing CD, beyond version control [20]. The study of prescriptive graphs still informs our choice of graph representation in two ways. Firstly, it helps in creating a world graph that is better interfaced with the virtual scene graphs for MR applications. Secondly, the design of these graphs makes explicit many of the attributes that are relevant to track to achieve realism (textures, materials, meshes, light sources etc. [98]), and that should therefore be present in our own SG.

Finally, we note that manually created SGs have been used to search for object patterns in a scene for AR applications [185]. This can be seen as a case where the anchors of the (real) scene graph and (virtual) world graph are the trackables [163], and the SG ends up describing the scene, although not in an exhaustive manner.

4.3.2 2D Descriptive graphs and relationship extraction

Building on the idea that scene structure can help identify objects (as in Subsection 4.2.2), Relation R-CNN [27] (Region-based CNN) is a 2D object detection network which computes semantic and spatial relation features to improve its sensibility and precision. Although Relation R-CNN only returns the objects' classes and bounding boxes, there exist works that specifically detect the objects' relationships, yet do not directly provide their results in the shape of a graph [215].

The works based on the Visual Genome [105] and Visual Relationship Detection (VDR) [126] datasets provide similar interaction-based relationships. Another dataset, namely, SpatialVOC2K [11], was developed for the purpose of spatial relationships annotation in images [10]. This has led to the implementation of methods that employ mono-depth estimation [15] and k-means clustering [14] to improve the quality and quantity of the detected relations. It should also be noted that object pairs should be able to share multiple semantic relationships to fully describe the scene structure.

For image-based scene understanding, there are many methods for directly generating scene graphs as well: Graph R-CNN [210], Factorizable Net [116] and Align R-CNN [186]. These methods are derived from object detection R-CNNs [154, 82] where additional branches were added to provide information on the relationships between the objects. Such 2D scene graphs have proven useful as a translation tool between the natural language and geometric or photometric measurements [166], as well as the applications mentioned in the previous subsection [91, 60, 153, 46, 181, 92]. However, for all these

applications, the contents of the scene are usually described with a focus on actions or interactions focus rather than on the structure of the scene [116, 210].

Unlike 3D models or 2D floor plans, "realistic" images of scenes (i.e., ground-based photography) are not suited to spatial planning and often contain active subjects. The uniqueness and qualities of an image are often better conveyed through its implied narrative rather than the spatial arrangement of its contents. Indeed, converting such action-based captions into accurate visual layouts is an active area of research [49]. Some relationships detectors are designed to extract more structural information about the scene, such as support relationships [173] (pairings of objects to the surfaces they are supported by), however most 2D SG generators are not. In practice, the resulting graphs usually contain a reduced number of edges with complex semantic meanings, and there have been efforts to increase the density of the graph by producing more edges per node pairs [186]. However most of these relationships are dynamic or rely on the presence of active subjects, which makes them unlikely to be stable enough for long-term change detection and representation in semi-static MR scenes.

4.3.3 3D Descriptive graphs and graph fusion

When SGs are used to describe 3D environments, the focus usually shifts from recording the narrative of the scene to recording its layout, as they are meant to be used for interactivity and navigation rather than captioning. Semantic graphs can be directly constructed from 3D data [201], using Machine Learning tools such as PointNet [21] and Graph Convolutional Networks [101], but the semantic analysis is generally performed with 2D images.

One approach consists in generating local 2D graphs from a video or image sequence [100, 213], which are incrementally merged into a global 3D graph in a process that will be further described in the next subsection. The method described in [100] uses graphs produced by Factorizable Net [116] and therefore requires filtering of the dynamic edges (actions) or nodes (actors) that do not serve to describe the scene structure, which further decreases the density of the local graphs before their merging. The density issue is compounded by the removal of 2D spatial relationships that do not translate directly into a 3D representation without ambiguity [148], and often only vertical structure is preserved as this axis is assumed to be aligned with the gravity vector (i.e., the normal to the ground).

In order to increase the number of structural edges with 2D semantic analysis, graph generators can use simpler metrics –such as proximity or contact– as opposed to learning-based methods, therefore decoupling the node and edge creation tasks. This is the approach taken in generating some "layered" graphs representations [6, 160]. These data structures have been introduced to facilitate the instruction interpretation task

in robotics, and to further organise data of differing scale or nature, and they employ "vertical" edges to correlate the information found on each layer, be it geometry, camera viewpoints, actors, or building layout.

In a similar fashion to multi-view object detection [28, 190, 43, 202, 217, 117, 144, 29], semantic analysis is improved by correlating the results found on different 2D points of view. In [6], object detection is performed on panoramic images registered to the 3D model, which is in turn textured using the segmentation results. Alternatively, [160] uses Kimera [159], which generates the geometry from a stereo-camera and an inertial measurement unit, and provides semantic analysis through panoptic segmentation of the left-side camera. The 2D segmentation is then converted into a 3D voxel grid of label probabilities, which is finally used to texture the mesh with the most likely labels using marching cubes.

As we have seen, the exact structure of 3D graphs vary with their method of construction or the intended application. Some favour processing efficiency over human readability, and none are specifically tailored to the task of comprehensive scene change detection, or monitoring on a human-scale. However, the 2D graph fusion methods requirements are the closest to those of the MR scenario.

4.3.4 Object similarity for node matching

In Subsection 4.3.3 we introduced the process of constructing a global 3D SG from the fusion of local 2D graphs [213, 100]. This fusion requires the association of the nodes present in the local graph with the ones contained in the reference: the nodes that refer to the same object in the scene are matched and merged. Conversely, the ones that refer to new entities are simply attached to the global graph with the edges seen in the local graph. To determine if there is a match between a local node and a global node, there needs to be a means of measuring similarity between nodes and ascertain if it is significant enough.

The similarity metrics used in graph fusion methods are based on the object nodes' attributes, and a threshold value is set to indicate similarity. In [213], the nodes' label and 3D position are compared, as the photometric aspects of the objects are considered too unstable over time. The label score is positive if the two nodes share the same label and is equal to the maximum of the nodes' "confidence score" for said label –the probability of the semantic labelling to be accurate reported by the 2D graph extractor. The position score is proportional to the nodes' distance normalized by the diameter of the largest of the two nodes. [100] uses more information: the position and size are stored as a 3D Gaussian distribution, photometric attributes are stored as a color histogram and the label is a probability vector over the whole set of object classes. The similarity metric is based on the 3 most likely labels for each object, the histograms' intersection

and the Gaussian distribution overlap. All these individual values are weighted and summed into an overall similarity score used during the fusion.

A closely related approach to this similarity metric can be found in place recognition methods that use landmarks [182]. The second-to-last layer of an object classifier network [106, 81] produces a vector-like descriptor that can be used to identify specific objects: there is enough information remaining to recognize them as individual instances rather than as members of a category. For the place recognition task, a database of the most recognizable, reoccurring and stable landmarks can be constructed containing said descriptors. The similarity metric can then simply be the cosine distance between two of those vectors –their scalar product divided by the product of their norms. If it were to be applied to graph fusion, this metric might need to be used in conjunction with other measurements, such as relative position, in order to compensate for the generic nature of most of the objects in the scene. However, this descriptor can act as a more relaxed substitute for the node's label probability vector, especially when the object is of an unknown class.

Switching to a graph structure for localisation [69], collecting landmarks of any nature in the environment and attaching them to their respective spatial points of capture, can also prove to be effective. These landmarks can be any type of sensory measurement (i.e., trackables: temperature, radar, LiDAR...) or miscellaneous properties such as the presence of WiFi hot-spots or connected items. During a later exploration of the scene, the graph is searched to find the spatial coordinates which correspond the most to the currently visible landmarks. In this graph, the **directed** edges correspond to the paths between landmarks pairs, and they store the distance and direction along those paths. The work in [69] notes that landmarks sometimes need to be disambiguated to be accurately matched, and this is achieved using a "belief" score based on the same relative spatial properties.

While changing the nodes' attributes could improve the performance of graph fusion methods [213, 100], their node-matching processes still ignore the relative spatial knowledge provided by the graphs' topology. The local and global graphs are practically reduced to a list of objects with attributes, and edges are only reintroduced when new nodes need to be attached the reference. As mentioned before, this can be attributed to the lack of stable relationships extracted from the images when forming the 2D graphs, which would make the edges an unreliable means of identification for the few nodes they connect. However, with a more robust way of generating the edges, the topology of the graph could be a determining factor in graph fusion processes.

Ignoring the graphs' topologies may provide the necessary time gain for those methods [213, 100] to produce and merge the local graphs at a sufficient frame rate so as to guarantee sufficient overlap in the content of the analysed images for real time analysis

(such as for MR or robotics). However, matching individual nodes cannot sufficiently discriminate between multiple instances of the same objects in the scene, and ignores the benefits of a multi-view observation of the scene to determine spatial relationships. If the pose estimation of the capture device was to become unreliable or unavailable –due to, for instance, overly quick movements or loss of GPS information– similar objects at different places could be merged, leading to wrong edges to be drawn without ever being removed. Attempting to preserve graph topology during fusion can provide some stability during fusion and retain the spatial information contained in the edges. Only with said stability can changes be accurately detected.

4.3.5 Graph subsumption

A graph isomorphism is an equivalence relationship between two graphs for which there exists a bijection between the nodes of each graph which preserves adjacency. When two graphs are different –and therefore no bijection exists for all nodes– only subgraph isomorphism is possible and is referred to as "graph subsumption". In the context of semantic graphs, the constraint of matching the nodes' and edges' attributes is added on top of the preservation of topology. Graph-based localisation and "graph matching" are analogous when one graph –the query, which represents the current point of view– is much smaller than the other –the reference, which represents to global environment. Graph matching tools already are the backbone of previously mentioned applications, such as image database searching [91]. For instance, the searching process was improved upon by using a catalogue graph that links all the existing types of nodes and edges to the images in which they appear [153]. These works [91, 153] are seemingly applicable to the field of MR as their visual-semantic graphs also use nodes to tally the objects of a scene and edges to describe their interactions or relationships.

In order to efficiently compare two graphs with attributes, "node descriptors" can be used to encode the local topology of the graphs around each node. When several nodes share similar attributes in the same graph, this helps in differentiating between said nodes. For SGs, this helps in recognizing specific instances of the same type of object in the scene, as they would share the same label. In the field of large-scale re-localisation in semi-static environments, where nodes correspond to semantically consistent contiguous areas, there exist several approaches for exploring and encoding the local topology around each node. Works such as [61] ("X-View") and [124] employ an arbitrary number of "random walks", which start at the studied node and take a constant number of steps from it along the edges. Alternatively, [180] and [72] exhaustively explore the local topology, but with a usually smaller number of steps –in fact [180] only checks each node's direct neighbours. The processes involved in the generation and comparison of each descriptor will be further developed in Chapter 5.

We notice that in the above-mentioned localisation methods, the edges of the graphs are considered unlabelled, as only the nodes' and their neighbours' attributes are stored in the descriptors. Edges merely reflect whether or not the relative distance based on two nodes' spatial attributes falls under an arbitrary threshold. Attempting to store the nature of the relationships in exhaustive descriptors such as those of [72] may be too costly in terms of memory –which is why the number of steps is limited in the first place. The random walk approach would not be impacted as severely but the increased amount of classes involved could necessitate a higher number of walks to better represent the local topology, especially if multiple relationships per node pair are allowed. Additionally, both [72] and [61] are designed for outdoor scenes, which are more prone to environmental –photometric– changes but are less semantically dense and dynamic than indoor ones. Outdoor scenes present larger and more spread out objects, which makes each partial view of the environment more unique and camera pose estimation errors –specifically errors in translation– relatively less significant. Translating those methods to indoor applications requires consideration for the increased number of classes and relative size of the query graphs to the reference, and if the current definition of edges is suitable.

Finally, we must note that [61] and [72] both take as input sequences of relatively registered RGB-D frames: the relative movements of the camera between each frame within a sequence are known, but the sequences do not share the same "global" coordinate system, hence the need for localisation. They therefore include their own graph fusion methods to produce the query and reference graphs, which rely on 2D semantic analysis and are exclusively node-based. The fusion process is not required to perfectly merge same-object nodes across frames, and the number of classes detected is limited to accommodate the descriptor used, i.e., to limit memory consumption and processing time. Therefore, contrary to the graph generation methods from Subsection 4.3.3, these localisation methods do not produce graphs useful for scene description, and [150] even remarks that object classification can be wrong as long as it is consistent.

4.4 Conclusion: Scene Graphs for Change Detection

Like the localisation task, the change detection task requires the existence of a reference –or prior– which can be of a different nature from the current view of the environment. In the case of X-View [61], both the query and reference are built from the same rudimentary graph fusion process, whose inconsistencies can be ignored given the robustness of the final matching algorithm. This is true of most localisation methods, especially in dynamic scenes: the underlying goal of the task is to find enough of the most reliable –recognizable and stable– features of the scene to match the query to the reference [45].

With these tools, performing change detection by listing the unmatched features of the query and prior would result in many false positives and untracked changes. The detected changes would also need to be translated back into a meaningful representation of the scene, which could include a filtering step but would not retrieve the missed changes. In conclusion, the state of the art methods in SGs generation and matching are not designed to adequately perform the scene CD and model maintenance tasks that are required for MR applications using current semantic graph tools.

4.4.1 Experiments with the scene representation

Using the capabilities of a standard MR device, the accessible data to produce the query graph is the RGB image and a corresponding depth map provided by the cameras with some positional information: as a consequence, we can use image-based graph generation. With this assumption, we first altered the graph fusion pipeline proposed in [100] (previously mentioned in Subsection 4.3.3) to include the localisation and comparison to a graph prior generated by the original algorithm. We found that its node matching process is reliable given an accurate pose estimate, but it is limited by the object detection capabilities of Factorizable-Net [116]. The detection is inconsistent from frame to frame and most objects in the scene are unaccounted for, even when only considering the classes the network was trained on. To improve the process, another 2D SG generator was experimented with, Graph R-CNN [210], which gave much more consistent results along the sequence but still recognized a limited number of objects. Both 2D SG generators also produced very few relationships edges, which were further filtered out during fusion. Switching to alternative generators designed to augment the number of edges –such as Align R-CNN [186]– would not fix this issue, due to their focus on describing the events occurring in the image rather than the scene layout. Therefore, our graph representation should include edges provided by geometric analysis of the nodes' attributes [61, 71], rather than 2D semantic analysis, as the former provide a more stable representation of the scene layout –thanks to the reliable extraction of nodes' geometric attributes– and would be better suited to CD.

For the purpose of evaluating a change detection method, it is more practical to assume that the prior is an exhaustive and accurate representation of the scene, from which the perceived current state differs. The problem of detecting the mistakes made during the construction of a prior using newly acquired information is more akin to robust scene representation generation, which could be tackled with or without actual changes. Using the knowledge that the scene prior is "perfect", the shortcomings of the change detection method can only come from the sensing of the scene's current state or failure to interpret the sensor's readings.

In summary, we will make the following hypotheses:

- the AR device is equipped with sensors allowing it to produce a depth map in addition to the corresponding RGB image and camera pose estimation,
- these readings can be semantically analysed to identify the objects in the scene, which are the labelled nodes of our SG representation with geometric attributes,
- the edges of the SG are produced from geometric analysis of the nodes' attributes, giving the graph a topology based on the scene layout rather than object interactions,
- when evaluating a CD method, the prior representation must be accurate and exhaustively describe the properties that are tracked by the method.

4.4.2 Problems to solve to achieve the thesis goals

After focusing on a photo-geometric CD method in the previous chapter [161], we have now chosen to work with a SG representation to deal with the scene's semantic contents. With the new hypotheses made in Subsection 4.4.1, we have identified four problems to solve to achieve our current goal of change detection for MR with scene graphs:

1- *Semantic change detection between two complete scene graphs.*

We assume that the current state of the scene is described in a similar manner as the prior: it covers the same spatial regions with the same level of detail. The two graphs must first be aligned using graph subsumption, where the most similar nodes are matched. Once this is achieved, we can detect changes of two forms: the modification of graph topology (addition or removal of nodes) and the modification of node attributes (besides their label). This second form of change is the result of the flexibility of the matching algorithm: nodes with different attributes can be matched because we should not assume that properties such as illumination will remain constant over time. In addition, simply regarding a node's change of attributes as replacement by another node –consecutive removals and additions– would not allow for the tracking of changes of a specific object instance. Finally, we notice that spatial and visual attributes (colors, shape, material etc.) are necessary to track changes in unclassified nodes.

2- *Semantic change detection with a partial query and a complete prior.*

This problem will occur in MR applications, where change detection may need to be performed live without preemptively scanning the whole scene. An added challenge from the first problem lies in the complexity of the graph matching step, since less information will be available from a partial view of the scene than a global one. The criteria for matching nodes representing large objects (e.g., walls) also changes, as the

areas they cover are unlikely to be fully visible. Once the query graph is directly localised in the prior, there is then the issue of differentiating between actual changes and the absence of nodes due to the limited field of view or occlusions. Occluded or out-of-view objects still matter during MR experiences, as they can affect the behaviour of virtual agents (on-screen or off-screen) or the rendering of virtual content (physically-based lighting, reflections, shadows etc.) and therefore cannot simply be removed from the scene model. Limiting the scope of the CD to the partial scene may require the reconstruction of the camera's point of view –effectively finding the relevant sub-graph in the prior– based on current node visibility and the relative change in size and shape of nodes representing large areas (partial visibility). We note that this problem may be unsolvable if the information on the current state is too sparse, in which case the problem will reside in minimizing this minimum data requirements (i.e., the time and effort required from an MR user at the beginning of a session).

3- *Local scene graph extraction guided by the global prior.*

If we take into account the limitations of hardware of the MR device at the time that CD is performed, the use of the prior data is required to fill the knowledge gaps and accurately detect changes. Attempting to recreate the exhaustiveness of the prior from low quality observations will result in many missed or misidentified objects, where the use of the prior may provide additional information in the shape of regions of interest. The prior also defines contents of the scene relevant to the MR application, and failure to explicitly track their changes will therefore lead to issues during the experience.

However, if we were to use the prior to generate the local query, the matching process becomes two-fold: a **preliminary query** must be proposed and localised in the global prior to be able to use the local properties of the prior and to **subsequently** generate a more accurate query before CD. This may only be required at the start of the MR session or when the pose estimation becomes unavailable, and a preliminary query graph may be accurate enough to significantly reduce the number of potential matches in the prior.

4- *Geometric update of the model from semantic CD.*

In the three previous problems, we avoid the issue of the initial construction of the graph prior, since its efficient and accurate generation does not directly relate to the change detection task. However, the tools used to produce this prior graph and its contents will ultimately dictate the nature of the changes that can be detected and how the scene representation may be updated. We must design a prior and method to allow the conversion of semantic results of the change detection into geometric or photometric updates of the scene model. This can for instance be done using a layered SG representation [6, 160], or any prescriptive graph-inspired structure that would allows us

to link the relevant photo-geometric data to the nodes and their attributes (as explained in Subsection 4.2.1).

The only data that would need to be retrieved by the MR device would be that which relates to novel or altered objects in the scene. The textures and geometric data could be recovered and refined over time using the available sensors [58], but accurate semantic analysis would allow us to duplicate such data from identical objects in the scene, or from an existing database [170].

In the following Chapter 5, we will consequently tackle the first of the above-mentioned problems by introducing our graph generation and registration pipeline. In order to have graphs of similar nature for the query and reference, they will both be produced from sequences of relatively registered RGB-D frames, as to emulate the available sensors from an AR device. Finally, the registration of the SGs will be performed using node matching –based the graphs' topologies and the nodes' semantic labels. The elaboration of a robust node matching process for the registration should ultimately lead to an effective object tracking method to be used for CD.

Chapter 5

Graph-based model registration for semantically rich scenes

In an AR experience, persistent virtual content is content whose state and locations in the real world are preserved between AR sessions. When changes occurs in the real scene between sessions, the persistent content can be integrated in the new scene layout based on its spatial relationships with real world objects rather than absolute coordinates, for a more realistic behaviour [77, 111]. As shown in Chapter 4, the detection of such objects and relationships can be achieved through semantic analysis of the scene as perceived by the AR device, the results of which are stored in a semantic Scene Graph (SG) as nodes and edges respectively. Using graph matching, this semantic information is enough to register two instances of a scene –from a previous AR session and the current one– without additional extraction or reconstruction of 3D geometry. In this chapter, we propose a novel node descriptor to compute node similarity during the matching, which encodes the node's label (i.e., the object's class) as well as local topology (i.e., objects' relationships). The size of this new descriptor increases only linearly with the depth of the SG exploration as opposed to the exponential growth seen in the state-of-the-art, without sacrificing matching performance. We also introduce a graph generation algorithm that provides an object-level representation of the scene to improve its semantic understanding for AR applications, without compromising the registration performance. We prepare and use subsets of ScanNet [37] and ChangeSim [139] as scene registration datasets, and propose a specialised metric for evaluating said registration. The method properties are summarized in Table 5.1.

Table 5.1: Graph-based registration from image sequences

Raw data	Output representation	Registration results
RGB-D sequences	Semantic SG	3D transform

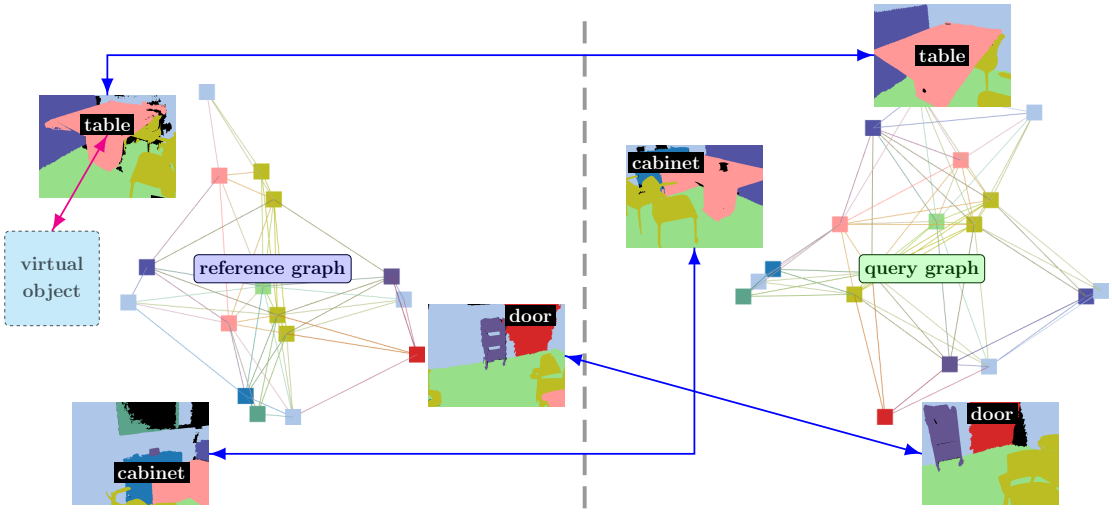


FIGURE 5.1: In an AR session, the contents of the scene can be semantically and spatially represented by a scene graph so that virtual objects can behave in a realistic manner. Left: A reference graph captured during a past session and a virtual object attached to a "table" node (magenta arrow). Right: A query graph generated from the current state of the scene and must be registered to the reference to restore the virtual object. The registration is achieved through semantic node matching (blue arrows).

5.1 Introduction

In the previous Chapter 4, we noted that in a number of emerging AR applications such as photo-realistic rendering [157, 146, 140] and environment-guided storytelling [26, 113], being able to rapidly detect changes in the scene from one AR session to the next is critical in order to enable a realistic immersive experience for the user. This can be efficiently achieved when the real world is already segmented into objects or areas that are relevant to the application. For such applications [111, 22], the positioning of virtual objects in relation to the ones present in the real world is more relevant than their absolute coordinates [83]. This can offer more flexibility in the placement of virtual objects to improve the readability and facilitate user interaction [31], or to help increase the realism of their interactions with real objects [119]. Previously, fiducial markers have been used to place virtual objects or to help user navigation [96, 135], but semantic analysis of the scene has also been able to provide usable landmarks [123]. This has led to the use of semantically-oriented representations of the AR scene, which allow convenient storage and processing of the virtual and real objects and their relationships [185]. These

representations often take the form of a scene graph [91], where the nodes represent objects or areas and the edges represent spatial or more general relationships between those objects. Such graphs provide a lightweight and versatile representation of the scene, especially on portable consumer devices that lack processing power or complex sensors.

At the start of a new AR session in a previously visited real environment, any persistent virtual content needs to be reintegrated in its correct location despite potential layout changes in the real world. For instance, virtual objects that appear to be resting upon a real table should follow the movements of said table between sessions (see in Figure 5.1). Regardless of the presence of such changes, the AR device will likely be started in a different position and orientation, and the video feed it captures will showcase different parts of the scene every time. In this chapter, we aim to register two image sequences of a real scene, which represent the AR device video feeds of two different sessions, using the same graph representation whose purpose is to integrate the AR content in the real scene. Our proposed registration method makes use of information that is already necessary to the execution of the AR application –i.e., the scene graph– and will ultimately simplify the tasks of detecting changes and updating this representation of the scene. From this process we obtain both a 3D transformation aligning the two scene instances' coordinate systems, as well as a list of matching graph nodes –i.e. a list of consistent real objects between the two sessions– onto which to anchor the virtual content (see Figure 5.1). We also noticed in Chapter 4 that the use of such semantic graph-based localisation methods is usually limited to outdoor environments [72, 61], so we evaluate the pertinence of our method for indoor scenes from the ScanNet dataset [37], which contains a high variety of objects in spatially restricted environments. We also observe its behaviour in larger changing environments, such as the photo-realistic virtual warehouses present in the ChangeSim dataset [139].

In our method, a scene graph is generated from each of the two image sequences captured at different times, and individual nodes are matched between the two graphs to identify corresponding areas in the scenes. The nodes of a graph represent the real objects in the scene using a semantic label –the object's class– and a 3D point –the object's center's local coordinates in the session– while the edges give a simplified account of the scene's structure by connecting objects close to each other. From the labels and edges, a node descriptor can be generated for each node in one graph and compared against the node descriptors in the other graph, to measure their similarity. Once nodes are matched based on their similarity, the 3D points of the matched nodes can be used to estimate the transformation between the sequences (scene instances). The quality of this estimation derives from the accuracy of the matching process, which in turn relies on the effectiveness of the descriptors used. The registration results from our method are

compared to a state-of-the-art descriptor-based approach [72] to evaluate the usefulness of the descriptors. Compared to this earlier work, the size of our descriptor only increases linearly with the depth of graph exploration when generating the descriptor, or the number of different object classes, as opposed to exponentially. Furthermore, we tailor our approach to AR applications, where the SGs generated can be used for purposes beyond the registration –such as interactivity between real and virtual agents– and where additional node attributes can be provided by other processes –such as color or material for photorealistic rendering in MR experiences. As such, we propose a variant of the graph generation algorithm where the object segmentation of the scene (into graph nodes) more closely matches the real world instance segmentation.

Our contributions are the following:

- A new lightweight and effective node descriptor based on the adjacency matrix of the scene graph.
- An alternative scene graph generation algorithm from image sequences, which better preserves object instance segmentation.
- A bounding box-based Root-Mean-Square Deviation (RMSD) metric applied to the evaluation of indoor scenes registration.
- Methodologies to process and use ScanNet [37] and ChangeSim [139] as semantic model alignment datasets.

Section 5.2 is dedicated to an overview of the related graph-based scene registrations methods and the existing processes for evaluating said registration. Our proposal is documented in Section 5.3, and the preparation of the experiments and results are explained in Section 5.4. Finally, our conclusions are summarized in Section 5.5.

5.2 Related Work

In Chapter 4, we surveyed the works related to the use of semantic graphs for scene understanding. Notably, we found that such graphs would often be generated and compared against each other through node matching to provide quick and robust localisation within said scene. In this section, we will focus on the means by which these works can compute node similarity for matching, and their limitations in the context of indoor scene registration.

5.2.1 Graph-based localisation and descriptors

In this chapter, graph matching refers to a weaker variant of graph isomorphism, previously defined in Chapter 4. Instead of searching for a bijective mapping between the nodes of two graphs, the focus is on finding pairs of nodes which are similar according to a predefined metric. This process is closer to graph subsumption because there is no assumption that a perfect pairing can be found between the nodes or that the graphs have the same size. Therefore, the edges of the graph become the mean to identify individual nodes through the observation of their surroundings. In order to process this topological information efficiently when comparing nodes, it is encoded into node descriptors.

5.2.1.1 Neighbourhood vectors

The use of SGs as an retrieval tool from image database [91, 60, 153] allows them to be effective in visual place recognition [143]. Such a problem was tackled in [180], where landmarks of the environment were stored as labelled nodes in a graph connected by edges weighted by the number of times the landmarks appeared together.

Inspired by the Weisfeiler-Lehman isomorphism test [171], the graph matching method used in [180] substitutes each node's label with a vector representing its direct neighbourhood. The vector is sized according to the number of labels, and acts as a histogram where each bin is the sum of the weights of the edges going from the node to one of its neighbours' labels.

The similarity between two nodes can then be measured using a normalized dot product of the nodes' corresponding vectors, which serve as node descriptors. However, the topology is only recorded with a depth of 2, i.e., one step away from the node described.

5.2.1.2 Random walk

In order to record the local topology around nodes with an arbitrary depth, random walks can be used [124, 203]. To describe the nodes, X-View [61] generates a fixed number of random walks with a fixed number of steps, and stores the explored labels in a 2D array. Each row corresponds to a specific walk and exploration rules can be added to avoid duplicated walks or to limit the number of times nodes can be visited, in order to provide more useful descriptors. Contrary to the vector approach [180], the finite number of walks introduces an element of randomness during the generation of descriptors. Moreover, the measure of similarity between descriptors involves a time consuming process of finding identical walks which also limits the depth and number of walks that can be used.

5.2.1.3 Walk histograms

Solving the aforementioned issue of randomness, [72] performs an exhaustive tallying of the walks from each node and stores the result as a vector-like histogram. The search can be performed with any depth, but the number of steps in the walks is limited to 2 in the paper. This is due to the rapidly expanding size of the stored descriptors with increasing depth: like the neighbourhood vector [171], the size of the walk histogram increases with the number of labels in the scene. Since the walks can be seen as label triplets, the histogram's length is the number of labels cubed.

The exhaustiveness of this method allows for the use of the normalized dot product as a similarity metric, therefore combining the convenience of [171] with the descriptive powers of [61]. However, this representation of the topology would be impractical in scenes with a large number of object labels, as the histogram's size would exponentially increase while most of its bins would remain empty.

In this chapter, we propose an alternative representation of the local topology around each node with only a linear increase in size with the depth or number of labels, thanks to the elimination of the redundant information contained in the walk histogram. This new adjacency matrix-based descriptor can also be compared using the normalized dot product, being a vector-like histogram. We therefore keep the advantages of the deterministic walk histogram approach while having more control over the size of the descriptors.

5.2.2 Quality metrics for scene registration

Point clouds are often used to represent the geometry of scenes as they can be directly obtained from real environments using the fusion of RGB-D images [37], high-resolution laser scanning [7] or multi-view stereo reconstruction [168]. As such, there are many works in scene registration whose performances are evaluated based on the use of clouds as input data, yet these quality metrics can also be applied to our graph-based method.

The registration of two clouds can be evaluated through the calculation of the rotation and translation errors compared to a ground truth transformation. However, [53] underlines that having two separate metrics or an arbitrarily weighted sum of the two is not convenient to compare registration methods. [53] cites RMSD [132] (or RMSE) as a suitable metric which can be computed by taking all "homologous" point pairings between the reference and the transformed source and measuring their squared distances, averaging the result and taking the square root. Homologous points in a point cloud pair, are points which correspond to the same region of space in both clouds. To avoid the need for finding such point pairings, [129] proposes to model the clouds as platonic solids, and only measure the distance for their vertices. RMSD can also be computed

using only the points of the source, by comparing the cloud transformed using the estimation to one transformed using the ground truth [216]. Finally, an earlier work [90], deals with the registration of brains by abstracting such objects as "volumes", and replacing the numerical summation of distances over the points of a cloud –or a solid– by a formal integration of the RMSD function over a sphere of a given radius. Since the method in this chapter is focused on indoor scenes, we can perform a similar abstraction by using the scene's bounding box as its shape, and formally integrate the RMSD over the resulting volume. In effect, this will make the error independent of the scene's layout –i.e., the error is based on the scene's boundaries (walls) rather than its contents (objects)– which we assume to be desirable in changing environments.

5.3 Proposed method for scene registration

5.3.1 Pipeline

Since this work is mostly concerned with the elaboration and evaluation of the new node descriptor and graph extractor, the scene registration pipeline is similar to the ones described in [61] and [72] as shown in Figure 5.2. The process takes as input two semantically segmented representations of the same scene, and outputs an estimation of the transformation between the two corresponding world coordinate systems.

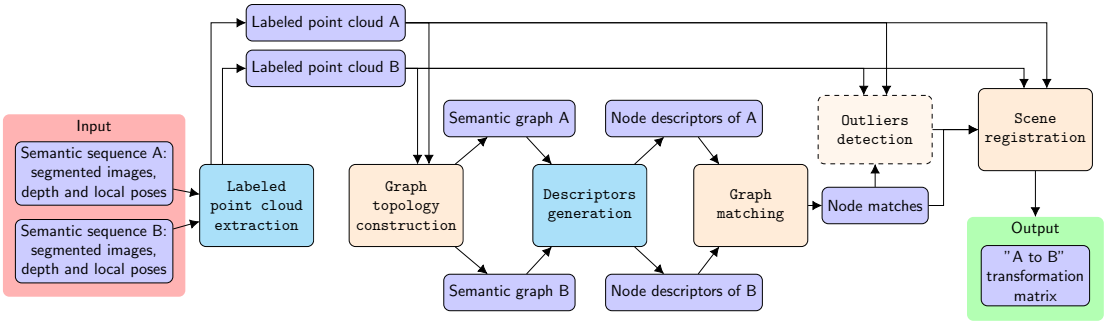


FIGURE 5.2: Scene registration from semantic information. Two instances "A" and "B" of the same scene are each represented by a sequence of semantically segmented images, with corresponding depth maps and camera poses. The two sets of poses do not use the same world coordinate system, so the semantic information from the images is used to register A and B through the construction and matching of two semantic graphs.

5.3.1.1 Input data

Each scene instance is represented as a semantically segmented image sequence with associated depth and pose information. These sequences are chronologically sorted sets of images taken at short regular intervals, a simulation of the footage captured by an AR device as an AR session is running. For the ScanNet [37] sequences, the frame-rate of capture is 30Hz.

The images store the labels of the visible objects pixel-wise (as seen in Figure 5.1), and 3D geometric information can be recovered from the depth maps in conjunction with the intrinsic and extrinsic matrices of the capturing device for each frame.

5.3.1.2 Point Cloud extraction

The SGs are first generated through the extraction of 3D points from the sequences, corresponding to the graphs' nodes. For each segmented image, the barycentres of uniformly labelled regions whose area exceeds a fixed threshold are computed using a four-way flood-fill algorithm. These 2D barycentres are turned into 3D points using the depth and pose, and added to a sparse "labelled" point cloud –an edgeless graph of nodes with spatial attributes and a label each– unless they are filtered based on their proximity to existing points.

For instance, in [72] new points with the same label and within a 3D distance threshold T_{object} of already present points in the cloud are ignored. This threshold allows for small regions/objects seen across multiple frames to be recorded as single nodes. As a side effect, objects whose diameter exceeds T_{object} are broken up into several nodes, and since the objects' volume are not recorded, using multiple nodes helps in representing the structure of the scene. However, this is undesirable if the instance segmentation of the scene is to be preserved in the final graph, and increases the influence of the camera path during scene exploration. Indeed, the spatial attributes of a node will be inferred from the first detection of its corresponding region/object in the image sequence, which will likely be at the edge of the frame in question. The object may consequently only be partially visible for that first detection and the depth estimation may be less accurate far from the frame's center: the spatial attributes will not accurately reflect the position of the object in the scene.

In Subsection 5.3.2, we describe an alternative point processing method which addresses both the object fragmentation and localisation concerns, while preserving the necessary scene structure information. The resulting graph has two layers: a bottom layer containing the aforementioned –now unfiltered– 3D points, and a top layer used to group those points into object instances using T_{object}. We call the resulting nodes in the top layer "Super Nodes" (SN). SNs are similarly used for the semantic graph matching

as the nodes produced by the algorithm from [72] –now referred to as "Legacy Nodes" (LN)– but represent whole objects in the scene rather than semantically consistent areas of arbitrary size.

5.3.1.3 Graph construction

To build the graph, every pair of nodes (i.e., SN or LN) whose distance falls under a threshold T_{edge} based on the scene's size, is connected by an edge. When using LNs and setting this threshold higher than T_{object}, large objects can always be represented by several interconnected nodes. In practice, we use an automatic threshold that scales with the average distance between nodes pairs in the graph based on their spatial attributes. T_{edge} is set to avoid orphan nodes while not making the graph too dense, which would make the local topologies around each node less unique. Empirically, we found that setting T_{edge} to 75% of the square root of the average squared node pair distance in the graph yielded 40% to 60% connectivity in the resulting topology. In this context, connectivity percentage refers to the number of existing edges in the graph over the total number of possible edges.

The edges give a simplified account of the spatial structure of the scene, where the distance between two objects can be represented by the number of steps they are separated by. The graphs' topologies are stored in symmetric binary adjacency matrices with zeros on their diagonals. They are considered "simple graphs": edges are undirected and unweighted, nodes cannot be connected to themselves, and node pairs can be connected by a single edge at most. Simple graphs with n nodes have at most $n(n-1)/2$ edges.

5.3.1.4 Graph matching

Once the graphs are generated, the local topology around each node is exploited to produce a node descriptor based exclusively on the labels and edges. This process is further detailed in Subsection 5.3.3. The descriptors are compared using a vector normalized dot product, similarly to [72]. This results in a list of the most similar nodes in both graphs according to the semantics and simplified scene structure. This can also be seen as a list of matching 3D points between the two clouds.

5.3.1.5 Registration

The transformation between the two image sequences is estimated by minimizing the error when transforming nodes from one graph to their matches in the other.

The previous work [72] also uses the ICP-RANSAC algorithm [12] in order to filter out outliers that negatively affect the registration. The presence of those outliers is

unavoidable even with a theoretically perfect graph matching. Indeed, as mentioned in Subsection 5.3.1.2, due to the sequential extraction of 3D points, when using LNs, the location of the objects is defined by the parts that first appear in the sequence. This can cause correctly matched nodes to be located at points that are separated by –at most– the diameter of the object they represent. The effect is not as pronounced using SNs, but slight changes in the scene or imprecisions in the 3D point extractions will still lead semantically sound matches to be spatially inaccurate.

The graph matching quality is ultimately evaluated by the closeness of the estimated transform to the ground truth from the dataset. For this reason, node filtering through RANSAC can be disabled, therefore forcing the registration algorithm to use **all** the matches found, allowing for a better comparison the matching performance of the node descriptors.

5.3.2 Super Nodes for graph generation

For every frame in the input sequences, the regions of the segmentation map –which store the object labels (see Subsection 5.3.1.1)– are converted into 3D points based on their 2D barycentre and the associated pose and depth map. Since this is performed at every frame, with a high enough frame rate, the same regions in space are likely to be observed several times. Even when filtering out similar-looking consecutive frames, large objects and structures would likely be often visible, and the camera will eventually capture the same areas over time.

In order to limit the size of the resulting scene graph, and get a better object-level understanding of the scene, previous graph extraction methods such as [72] do not add new LNs if there is already a node with the **same label** within a distance T_{object} (see Subsection 5.3.1.2). However, constructing the graphs with these LNs leads to the previously mentioned issues: a fragmentation of large objects, and spatial attributes that do not match the objects' actual 3D barycentre.

To solve this, we introduce SNs, which can contain a multitude of 3D points rather than a single attribute vector of 3D coordinates. Like LNs, every SN has a single label, which is the same label shared by all the points belonging to it. We define the distance between a 3D point and a SN as the smallest distance between the former and any point belonging to the latter. Consequently, the distance between two SNs can be defined as the smallest distance between one SN and any point belonging to the other. Using these metrics, a new 3D point is identified as a "duplicate" when its distance to an existing SN with the same label falls under T_{object}. When this occurs, rather than being ignored, the new point is added to the corresponding SN. If the new point were to be added to multiple SNs using this criteria, they all are merged into a single SN –containing all their respective points and said new point. This greatly alleviates the problem of

object fragmentation, as the delimitation threshold can be tested at the extremities of large objects, which are updated as they include more points, matching their spread in the real world. The volume of large objects can be inferred from the 3D points within their corresponding SN, rather than multiple LNs of the same label separated by T_{object}. Consequently, it is the distance between two SNs that is used to create the graph topology.

The use of SNs also allows for filtering out *a posteriori* some of the errors introduced by the sensors. Indeed if the pose estimation momentarily fails or artefacts appear in the depth map, a few extracted points may be incorrectly identified as new objects and added to the graph. Once the extraction is over, those anomalies can be removed using a minimum threshold T_{super} on the number of points a SN must contain. If we require the object to be visible for at least 5 frames to be added to the graph as a SN, we should set T_{super} higher than 5, since several points can be extracted per frame. In practise we use $T_{super} = 10$.

Figure 5.3 shows how LN and SN extraction processes differ for a specific object: a "counter", which only appears once in the scene. The counter is first detected in frame 165, and the corresponding LN and SN are added to their respective LN-graph and SN-graph. Initially, both the LN and the SN have the same spatial attribute, which corresponds to the 3D location of the object's first sighting (the SN contains a single 3D point). In frame 413, the location at which the "counter" is detected is further than T_{object} away from its original detection, therefore a new LN is added to the LN-graph. The object has now been fragmented into multiple LNs, each localized at the extremities of the "counter" seen in frame 165 and 413. Conversely, in the SN-graph, the 3D location of the SN is simply averaged from all the counter's detection locations since the creation of the SN. The 3D points contained in said SN all have a neighbouring 3D point within a T_{object} distance, and the two "counter" LNs in the LN-graph correspond to the extremities of the cloud formed by these 3D points at frame 413. Frame 978 is the last "counter" detection in the image sequence, no LNs have been added since frame 413, but the location of the SN has been updated based on all the object's sightings. In conclusion, rather than representing the object's size by breaking it into two LNs –neither giving an accurate location of the object's center position– a single SN can be used for a more faithful object-level representation of the scene.

It should be noted that generating a graph using SNs is momentarily more memory intensive than using LNs, as more 3D coordinate vectors are stored. The resulting graphs will however end up with fewer labelled nodes to be used during matching due to the diminished fragmentation, and therefore fewer descriptors will be generated. Moreover, the registration step described in Subsection 5.3.1.5 requires the SNs to have their own 3D coordinates, as using every single point within them to compute the 3D transform is

too computationally expensive. Therefore, the final step of graph creation using SNs is the computation of the SNs' spatial attribute –their location in 3D space– by averaging the 3D coordinates of the points they contain. The resulting attribute should be closer to the real objects' barycentre, especially for small objects. Finally, following the creation of the edges, the SNs' 3D points are no longer required –only the SNs' labels and spatial attributes– allowing us to free the corresponding allocated memory if necessary.

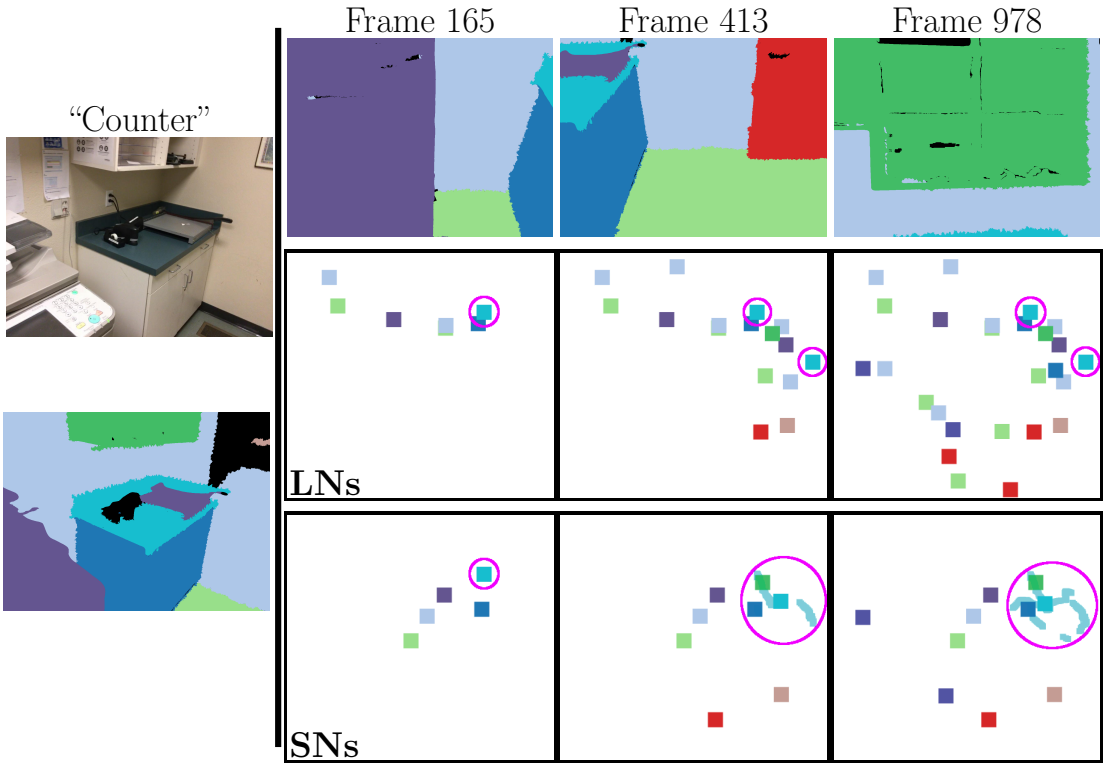


FIGURE 5.3: Node extraction for a "counter" (left side) shown in cyan in the segmentation maps (right side, top row), in the LN-graphs (middle row) and in the SN-graphs (bottom row). Frame 165 is the first detection of the object in the image sequence (right side, left column), while 978 is its last (right column). Frame 413 shows an occurrence of object fragmentation: a second LN is added to the LN-graph to represent the same object (right side, middle column). For clarity, 3D points are only displayed for the "counter" node (circled in magenta) in the SN-graphs (light cyan), and all other SNs are already shown at their final location.

5.3.3 Adjacency descriptor

Previous node descriptors relied on tallying the paths that can be taken from each node to describe its surrounding topology. The maximum number of unique possible walks of depth d –corresponding to $d-1$ steps– among nodes with one of l labels is l^d. While only a fraction of these unique walks will be present in any graph, in the random walk approach from [61] the number of walks in each descriptor must still be large enough to avoid cases where a lack of identical walks among descriptors makes matching impossible. For the histogram method of [72], the size $s_H(l,d)$ of each descriptor will always be l^d, but most of the bins will remain empty: since a given node has only one label, only a maximum of $l^{(d-1)}$ bins are ever used, and for every label not present at depth k, there are an additional $l^{(d-k)}$ empty bins.

In order to describe the neighbourhood around each node with a larger depth and in scenes with a high diversity of objects, we propose a descriptor that tallies only the end label of each walk. Based on the adjacency matrix of the SG, the descriptor's size only increases linearly with the depth and number of labels.

5.3.3.1 Basic descriptor using the adjacency matrix

Given the adjacency matrix A of a graph $\mathcal{G}$ with n nodes, the element (i,j) of A^k indicates the number of walks with k steps between two nodes i and j. Based on the symmetry of the adjacency matrix, each i-th column or row of A_k will therefore give a description at depth $k+1$ of the neighbourhood of node i.

We can fit this description into a histogram $V_{k,i}$ of size l using the list of nodes' labels L as a look-up table. L is of size n, and with values between 1 and l, mapping each node's index to its label's index. In practice, for every node j, if $A^{(k)}_{i,j} = (A^k)_{i,j}$ is positive then $V^{(k,i)}_{L_j}$ is incremented.

Since $V_{k,i}$ only contains information for depth $k+1$, we compute it for every k between 1 and $d-1$ (d being the desired exploration depth) to get a full account of the local topology. We also encode the node's label using a vector $V_{0,i}$ of size l that is null except for its L_i-th element that is set to 1. For a node i, its descriptor $D_{d,i}$ at depth d is obtained by concatenating the $(V_{i,k})$ for k between 0 and $d-1$. By tallying only the end points for walks of depth 1 to d, each $D_{d,i}$ is only of size $s_D(l,d) = l \times d$, and no space is wasted accounting for walks that cannot be taken in $\mathcal{G}$.

Since A is already computed as the representation of the topology of $\mathcal{G}$, increasing the depth of exploration comes at no great computational cost as only one symmetrical matrix product of size n is required to update all n descriptors.

The basic algorithm for computing all the descriptors for $\mathcal{G}$ is described in Algorithm 4 and an example with a small graph is shown in Figure 5.4.

Algorithm 4: Pseudo-code for our descriptor generation algorithm. The i-th element of a vector X_j is noted as $X_i^{(j)}$

DescriptorGenerator (A, d, L)

inputs : Adjacency matrix A; Depth d; Label vector L

output: List of n descriptors $(D_{d,i})_{1 \leq i \leq n}$ of size $d \times l$

for $i \leftarrow 1$ **to** n **do**

$D_{d,i} \leftarrow 0_{d \times l}$;

$D_{L_i}^{(d,i)} \leftarrow 1$;

$A_1 \leftarrow A$;

for $k \leftarrow 1$ **to** $d-1$ **do**

$\boxed{A_{k+1} \leftarrow A_k A}$;

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

$D_{kl+L_j}^{(d,i)} \leftarrow D_{kl+L_j}^{(d,i)} + \left(A_{i,j}^{(k+1)} > 0\right)$;

return $(D_{d,i})_{1 \leq i \leq n}$;

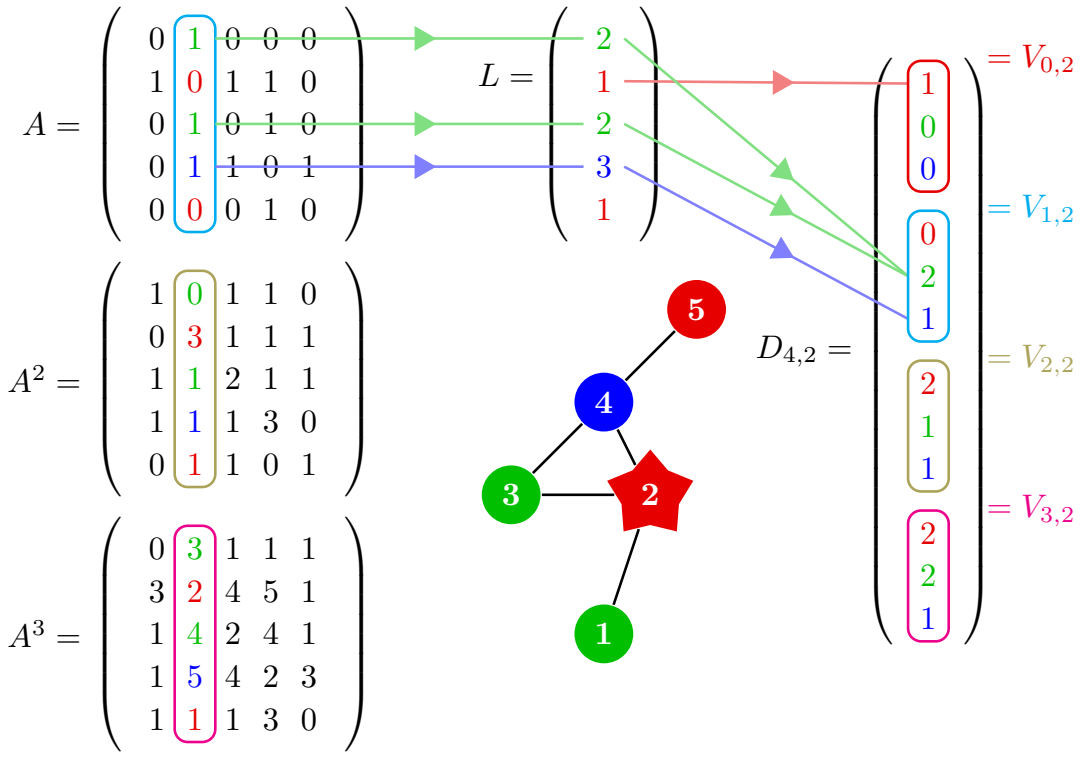


FIGURE 5.4: Generating the descriptor $D_{4,2}$ of depth 4 for node 2 of a graph with 5 nodes and 3 different labels (red has index "1", green has "2" and blue "3").

5.3.3.2 Improving the descriptor

As mentioned in [61], different exploration strategies can be followed when producing random walks, to make the limited number of walks more descriptive. These include the exclusion of duplicate walks or forbidding back-and-forth motions between two nodes. While our approach is not limited by a sample size from a random distribution of walks, these strategies can be employed to fix some issues with our basic descriptor, by altering the framed line in Algorithm 4.

For instance, by using $A^k_{i,j}$ to count the number of walks of depth $k+1$ between i and j, we include the walks that involve repeated moves to reach the desired depth. In practice, this means that if d is divisible by x, every cycle of length x starting from i will be tallied in $D^{(d,i)}_{km+L_i}$, making the $(V_{i,k})_{k\geq 1}$ histograms more correlated.

To avoid this correlation issue, we define a series of "no-return" adjacency matrices $B^{(k)}$ whose elements (i,j) give the number of walks of depth $k+1$ between i and j that do not pass through the same edge twice. We find for depths 2 and 3 that:

$$B^{(1)}_{i,j} = A_{i,j}, \quad \text{and} \quad B^{(2)}_{i,j} = \begin{cases} A^{(2)}_{i,j} & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

For depths $k+1$ greater than 2, the computation of $B^{(k+1)}$ is not trivial but its values can be approximated –without exhaustively exploring the graph– using the cardinality of the following sets:

$$\mathbf{B} = \bigcup_{p_1 \dots p_k} (i \overset{1}{\to} p_1 \overset{2}{\to} \dots \overset{k}{\to} p_k \overset{k+1}{\to} j), \tag{5.2}$$

$$\mathbf{L} = \bigcup_{p_1 \dots p_k} (i \to p_1 \overset{1}{\to} \dots \overset{k-1}{\to} p_k \overset{k}{\to} j), \tag{5.3}$$

$$\mathbf{R} = \bigcup_{p_1 \dots p_k} (i \overset{1}{\to} p_1 \overset{2}{\to} \dots \overset{k}{\to} p_k \to j), \tag{5.4}$$

$$\mathbf{C} = \bigcup_{p_1 \dots p_k} (i \to p_1 \overset{1}{\to} \dots \overset{k-1}{\to} p_k \to j), \tag{5.5}$$

$$\mathbf{D} = \bigcup_{p_1 \dots p_k} (i \overset{1}{\to} p_1 \overset{2}{\to} \dots \overset{k-1}{\to} p_k \overset{1}{\to} j), \tag{5.6}$$

$$\mathbf{E} = \bigcup_{c=1}^{k-1} \bigcup_{d=1}^{k-1} \bigcup_{p_1 \dots p_k} (i \overset{c}{\to} p_1 \overset{1}{\to} \dots \overset{k-1}{\to} p_k \overset{d}{\to} j), \tag{5.7}$$

where $(i \to p_1 \to \dots \to p_k \to j)$ is the set of walks of depth $k+2$ from i to j passing

through nodes $p_1 \dots p_k$ and numbered arrows denote unique edges in the walk. Since our graphs are simple, these particular sets contain either 0 or 1 walk. We find that:

$$\mathbf{L} \cap \mathbf{R} = \mathbf{B} \cup \mathbf{D}, \qquad \mathbf{B} \cap \mathbf{D} = \emptyset, \tag{5.8}$$

$$\mathbf{C} = \mathbf{L} \cup \mathbf{R} \cup \mathbf{E}, \qquad (\mathbf{L} \cup \mathbf{R}) \cap \mathbf{E} = \emptyset, \tag{5.9}$$

$$|\mathbf{B}| = |\mathbf{L} \cap \mathbf{R}| - |\mathbf{D}| = |\mathbf{L}| + |\mathbf{R}| - |\mathbf{C}| - |\mathbf{D}| + |\mathbf{E}|. \tag{5.10}$$

Therefore:

$$B^{(k+1)} = AB^{(k)} + B^{(k)}A - AB^{(k-1)}A - |\mathbf{D}| + |\mathbf{E}|. \tag{5.11}$$

Using equations (5.1) and (5.11) allows us to estimate $B^{(k+1)}$ through recursion for any value of k. However, since we use B in a similar fashion as A in Algorithm 4, the only information we require is whether or not its elements are positive.

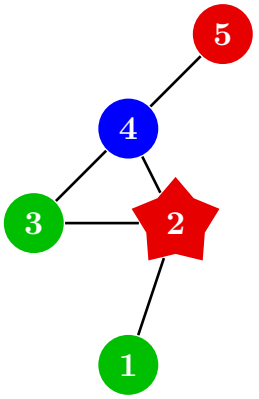
If we consider A as a matrix of boolean values rather than integers, and the scalar product and sum as logical *and* and *or* in the matrix multiplication (or use the Roy-Warshall algorithm [162]), we notice that the powers of A will give the same information necessary to Algorithm 4. This not only speeds up the calculation of A^k but also alleviates any issue of overflowing for large values of k.

Finally, we need to account for the homogenization of the A^k matrices for high values of k, since they are eventually filled with `true` except for the rows and columns of orphan nodes. To achieve this we simply divide the values added to $V_{k,i}$ by $k-1$ when $k > 1$. This results in closer neighbours having more weight during the matching and also mitigates the effect of the increasing inaccuracies in the estimation of $B^{(k)}$ for large values of k. The validity of our estimation of $B^{(k)}$ can nonetheless be tested for the smaller values, as the correct matrix can be constructed from exhaustive exploration of the graph rather than matrix multiplication. This exploration process is similar to the one used to create the walk histograms H_d at a depth d but the results are stored in the smaller adjacency descriptors D_d, and we keep track of the edges already visited as to not reuse them in a given walk.

We will refer to this "exact" but computationally expensive version of the "no-return" adjacency matrix as $C^{(k)}$. For simplicity, we will also refer to the adjacency descriptors generated with weights as $A_w^{(k)}$, $B_w^{(k)}$ and $C_w^{(k)}$, even if the matrices themselves are not weighted, but rather their contribution to the descriptors (it is the $(V_{i,k})$ that are divided by $k-1$, not the $A^{(k)}$, $B^{(k)}$ or $C^{(k)}$). The effect these improvements have on the adjacency descriptors is illustrated in Figure 5.5 which shows the same small graph example as in Figure 5.5. The resulting descriptors can then be compared using a normalized dot product.

$$B^{(1)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad L = \begin{pmatrix} 2 \\ 1 \\ 2 \\ 3 \\ 1 \end{pmatrix}$$

$$B^{(2)} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$



$$D_{4,2} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 0.5 \\ 0.5 \\ 0.67 \\ 0.67 \\ 0.33 \end{pmatrix}$$

$$B^{(3)} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

FIGURE 5.5: Generating the improved descriptor $D_{4,2}$ (red has index "1", green has "2" and blue "3"). Changes in the adjacency matrices introduced by a "no-return" exploration strategy and descriptor values affected by depth-wise weighting are shown in red.

5.3.4 Comparison with the walk histogram

Our adjacency descriptor $D_{d,i}$ at depth d for node i can be seen as a compressed version of the walk histogram $H_{d,i}$ from [72]. For each i, $V_{d-1,i}$ corresponds to the merging of the bins of $(H_{d,i})$ into bins of the label of the walk's last step, effectively reducing a histogram of size l^d to a size of l. The descriptiveness of $H_{d,i}$ lost in this reduction, is regained through the $V_{k,i}$ with $0 \geq k < d - 1$.

While $D_{0,i}$ and $H_{0,i}$ are identical, their sizes grow linearly and exponentially, respectively, with regards to the depth, making a fair comparison of their effectiveness difficult since we have more granular control over the size of the adjacency descriptor. In order to produce results for more values of the descriptors' sizes for the walk histogram, without resorting to interpolation, we added a "merging" parameter m to its implementation, which reduces the number of histogram bins by merging labels on the last step.

Given a value of m that is a divisor of l, the last step of a walk is recorded as one of l/m "hyper-labels" –groupings of m labels. This results in a descriptor size $s_H(l, d, m)$ of l^d/m. We note that changing the value of m for any given depth d does not greatly affect the complexity of the exploration algorithm, only the size of the histogram in which the results are stored.

Comparing the two approaches will determine if the trading of the descriptor's precision at a given depth for a lighter descriptor which explores the neighbourhood further is worthwhile. We describe our experimentations in the following section.

5.4 Experiments

We are looking to evaluate the performance of our novel node descriptor for the task of graph matching in an indoor environment with a high diversity of objects. The results of the graph matching are then used for the registration of two instances of the same scene captured with different world coordinate systems. It is the measure of the quality of the registration that ultimately allows us to compare different node descriptors, as the parts of the pipeline following descriptor generation remain unchanged (see Figure 5.2).

We are also checking if the new graph extractor using SNs gives as good results for the registration task as the original LN-based one, while also providing a representation that gives a better object-level understanding of the scene structure. The latter can be evaluated by measuring the performance of the extractors for clustering the 3D points extracted from the images that represent the same object into the same node in the graph.

Our code to generate the adjacency descriptors and SN-based graphs from the ScanNet [37] and ChangeSim [139] datasets, as well as optimized code to generate the walk histograms and LN-based graphs from [72] is made available here:

https://github.com/InterDigitalInc/GraphBasedSceneRegistration

5.4.1 Datasets

To test our registration algorithm in semantically rich indoor scenes, we use the ScanNet dataset [37], which contains 1513 models of scenes along with the localised RGB-D sequences used to generate them. The dataset includes semantically segmented versions of the models and images, convertible to the 40 labels of the NYU Depth dataset v2 [173] (as seen in Figure 5.1). Finally, the dataset also provides instance-segmented image sequences, which can be used to evaluate our graph extractors clustering ability.

We also use the ChangeSim [139] dataset to check our method's robustness to changes, namely: "new", "missing", "replaced", or "rotated" objects. This dataset contains 10 simulated photo-realistic warehouses scenes, which are explored twice by a virtual drone before and after changes occur (for a total of 40 sequences, not accounting for those with added visual disturbances). Similarly to ScanNet, localised RGB-D and label maps are included.

ScanNet [37] contains 707 unique scenes, but many have been scanned and added multiple times in the dataset in order to reach the 1513 count. Those scenes with several instances can be used to perform registration in pairs, and Table 5.2 describes the dataset's composition with regards to instance pairs.

TABLE 5.2: Number of instance pairs to perform registration

Instances in dataset	Number of scenes	Resulting pairs
1	221	0
2	211	211
3	246	738
4	19	114
5	5	50
6	4	60
7	1	21
total	**707**	**1194**

Given that one can make $\binom{x}{2}$ pairs out of a scene with x instances, ScanNet's contribution to our registration dataset is of maximum size 1194. For each of these instances,

the scene is represented by its semantically segmented RGB sequence, its depth map sequence with accompanying camera poses, as well as the camera's intrinsic matrix.

For each ChangeSim [139] scene, the 2 scans before changes occur are referred to as Ref, and the 2 scans after as $Query$. Each of the 2 Ref scans can be compared to is corresponding $Query$ scan, making for a total of 20 additional instance pairs for our registration dataset.

5.4.1.1 Preparing the ScanNet data

In order to use our pipeline on ScanNet's subset of interest, we need to find the ground truth for the alignment of any two instances of the same scene. The original dataset does not include such information but has for each model an `alignTransformMatrix` which permits the user to transform the scene's coordinates such that the origin is in the middle of the room's floor, and the walls are aligned with the X and Y axes (the floor's normal is already aligned with Z).

Assuming that two instances of the same scene showcase the same portions of the world, they should share the same room center. If we apply each `alignTransformMatrix` to their corresponding 3D models, the two instances should be aligned or be rotated around the Z axis by $\frac{\pi}{2}$, π or $\frac{3\pi}{2}$, since the walls are aligned with X and Y.

In order to check which of the 4 rotations (including the identity) will result in the alignment of the two instances, we choose a pose P from one of the instances and render a reference image I_{ref} using the corresponding semantically segmented model $\mathcal{M}_{ref}$. We then convert this pose into the world coordinates of the other instance using the two matrices and a rotation matrix R_θ (with θ being one of the 4 possible angles of rotation) to render an image I from the other model $\mathcal{M}$. As I and I_{ref} are segmentation maps (and not RGB images from a camera) we can count exactly how many of their pixels values are identical, and use the percentage of identical pixels as a similarity metric. We also assume that unlabelled objects (which appear as black pixels) are different from on instance to the other –they do not count as identical pixels– so that we only rely on known objects for similarity evaluation. If I and I_{ref} are similar enough, we assume the rotation being tested is the correct one, otherwise we try the other rotations and then move on to another pose. The exact procedure to determine the correct orientation is described in Algorithm 5 and illustrated in Figure 5.6 (where we use a similarity metric between 0 and 1 rather than a percentage).

Between two instances, the dissimilarity between I and I_{ref} when the rotation is correct can come from a translation error (the rooms have different centres) or a different semantic segmentation of the content. For this reason –and because of the unlabelled areas in the scenes– we cannot use a similarity threshold of 100% identical pixels in the images. Empirically, we found that using an similarity threshold of 80% would result

in 450 instance pairs with a sufficiently reliable ground truth. 5 pairs were manually removed for having very different centres despite having cleared the threshold thanks to their lack of semantic variation.

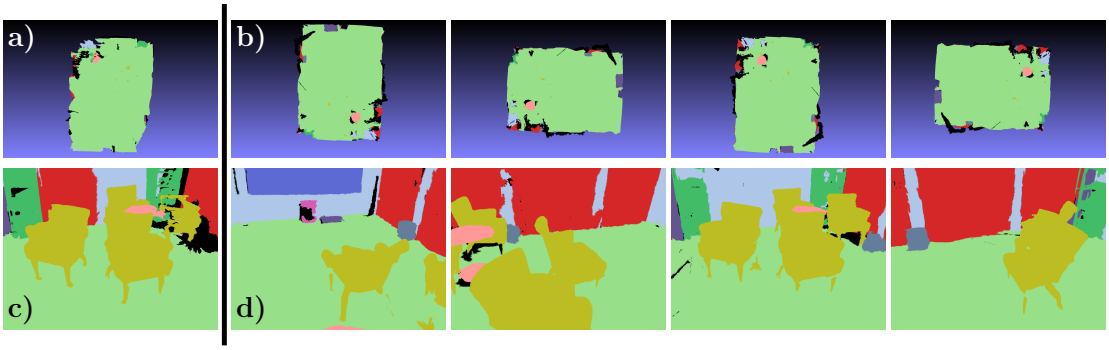


FIGURE 5.6: a) Bottom view of the aligned reference mesh, which can be registered to b) another aligned mesh from the same scene using successive rotations of $\pi/2$ around the Z axis. c) An image generated from the reference mesh is compared to one d) generated from the other mesh rotated by 0, $\pi/2$, π or $3\pi/2$. A rotation of π registers the meshes.

Algorithm 5: Base pseudo-code of our instance alignment algorithm. *Render* produces an image given a mesh and a camera pose.

AlignInstances ($\mathcal{M}_{ref}$, $\mathcal{M}$)
- **inputs :** Meshes $\mathcal{M}_{ref}$, $\mathcal{M}$; Alignment matrices A_{ref} , A; Similarity threshold T; Number of frames i_{max}; Poses $(P_i)_{1 \leq i \leq i_{max}}$
- **output:** Rotation θ_{best}; Similarity t_{best}
- $\theta_{min} \leftarrow -1$;
- $t_{best} \leftarrow 0$;
- **for** $i \leftarrow 1$ **to** i_{max} **do**
  - $I_{ref} \leftarrow Render(\mathcal{M}_{ref}, P_i)$;
  - **for** $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$ **do**
    - $P' \leftarrow A^{-1} R_\theta A_{ref} P_i$;
    - $I \leftarrow Render(\mathcal{M}, P')$;
    - $t \leftarrow mean(I = I_{ref})$;
    - **if** $t > t_{best}$ **then**
      - $t_{best} \leftarrow t$;
      - $\theta_{best} \leftarrow \theta$;
      - **if** $t_{best} > T$ **then**
        - **return** $(\theta_{best}, t_{best})$;
- **return** $(\theta_{best}, t_{best})$;

5.4.1.2 Preparing the ChangeSim data

Contrary to ScanNet [37], in ChangeSim [139] the matching *Ref* and *Query* sequences are already aligned, meaning that the registration ground truth is an identity matrix I_4. However, we found that the provided `trajectory.txt` files do not match the virtual camera trajectories used in the images sequences, requiring additional preparation of the dataset before it can be used.

For the *Query* sequences with over 1000 frames, the inconsistencies in the trajectory files are caused by a sorting error, likely occurring during the merging of the individual files for each frame. This was solved by recreating them from the individual pose files provided in the `pose` folder with a Python script that would order them by filename. For the *Ref* sequences, the trajectory files do not contain the correct number of poses, and cannot be simply recreated due to the absence of a `pose` folder. Instead, they have to be extrapolated from the "raw" data shipped with the dataset.

Indeed, each *Ref* scan contains the aforementioned RGB, depth and label map sequences, as well as a `raw` folder containing additional RGB and depth map sequences. The raw sequences do not have the same number of frames as the "final" sequences, and are accompanied by a `poses.g2o` file which gives positional data for the virtual drone at certain keyframes. Those keyframes are themselves a subset of the raw frames and their poses were directly written into the `trajectory.txt` file, despite them not matching the frames in the final sequences. Finally, since the raw data does not include label maps, the keyframes cannot be used as input to the registration pipeline, making the generation of a new `trajectory.txt` file for the final data mandatory.

To generate the final poses, we notice that the final sequences follow the same trajectory as the raw sequences, but with different sampling (i.e., frames were not generated at the same times). With the exception of the warehouses scenes "2" and "4", there are always more raw frames than final ones. We therefore use a 3-step process, which we will describe in detail in the following paragraphs, based on the comparison of the raw and final depth maps:

1. Match each final depth map to a raw depth map, while preserving the sequence order,
2. Copy the poses for every matched raw frame that is a keyframe in the `poses.g2o` file,
3. Interpolate the missing poses for the matched raw frames using the closest known poses.

In to order to match the frames, we sequentially go through each depth map of the final sequence and find the most similar one in the raw sequence according pixel-wise

L2 distance. The raw pixel values must be divided by 200 to match the final scale – converting them to metric measurements, as we found experimentally. To preserve the sequence order, the search for a match is restricted to a selection of frames around the previous match. We define the reduction ratio r from the raw to the final sequences, as the number of frames of the former divided by the number of frames of the latter. We search for the next matching frame from $\lfloor 1r \rfloor$ before the previous match to $\lfloor 6r \rfloor$ after. We use that search window as it can successfully find matching frames for every scene save "2" and "4", and a larger window would result in a longer computation time. This approach does not work for "2" and "4" because of the too small r values, and both scenes have to be ignored due to their limited numbers of raw frames, which lead to issues with the interpolation step.

Once the keyframe poses are copied for the relevant raw frames, the missing ones must be interpolated from other –copied or not– keyframes. This process is illustrated in Figure 5.7. Poses in ChangeSim [139] are given as a 3D vector t for translation and a quaternion q for the rotation, and keyframes are rarely separated by more than 10 frames. Therefore for a given raw frame, the interpolation is simply performed between the closest anterior and posterior keyframes poses: we use linear interpolation (Lerp) for the translation and spherical linear interpolation (Slerp) for the rotation [172]. This assumes a constant speed between consecutive keyframes, and the interpolation parameter is simply given by the difference between the current raw frame index and the previous keyframe index, over the difference between the two consecutive keyframes indexes (as seen in Figure 5.7).

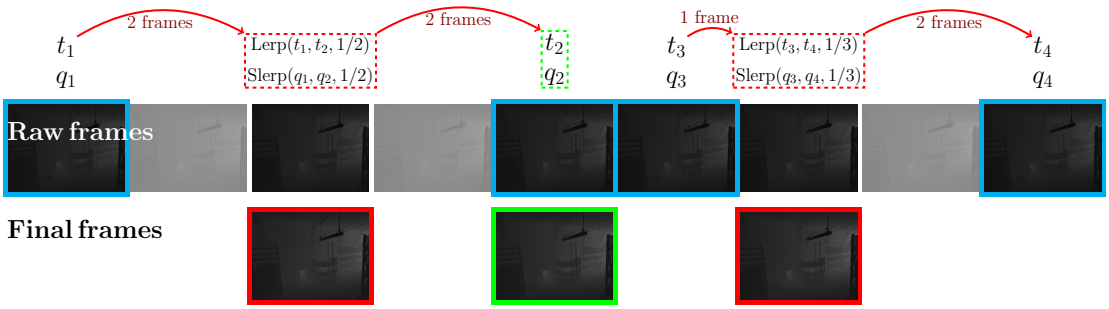


FIGURE 5.7: Finals frames (bottom row) are matched to raw frames (top row). When a raw frame has a known pose ("keyframe", blue contour), it can be used for a matched final frame (green contour). When a raw frame without known pose is matched to a final frame (red contour), said pose is interpolated from the last and next known poses.

In our implementation, the 3 steps are performed in parallel thanks to the sequential matching. This process allowed us to add 16 instance pairs to register to our dataset.

5.4.2 Error metrics for model alignment

As mentioned in Section 5.2, as well as using the translation and rotation errors between the estimated transformation and the ground truth, we use the RMSD E_{RMS} to provide a unique metric to compare the methods.

Firstly, if t and R are the estimated translation vector and rotation matrix and t_{GT} and R_{GT} the corresponding ground truth, then E_t and E_R are defined as:

$$E_t = ||t - t_{GT}||_2\,, \qquad E_R = ||R_{GT}^{-1}R - I||_F\,, \tag{5.12}$$

where $||.||_2$ and $||.||_F$ are the Euclidean norms for vectors (L_2) and matrices (Frobenius) respectively.

Secondly, we call T and T_{GT} the transformation matrices associated with those translation vectors and rotation matrices:

$$T = \left[\begin{array}{ccc|c} & R & & t \\ 0 & 0 & 0 & 1 \end{array}\right], \quad T_{GT} = \left[\begin{array}{ccc|c} & R_{GT} & & t_{GT} \\ 0 & 0 & 0 & 1 \end{array}\right]. \tag{5.13}$$

Following the approach given in [90], we formally integrate the transformation error over a volume $\mathcal{V}$ modelling the reference. The offset introduced by the error between an estimated transformation T and the ground truth T_{GT} for any given point p in homogeneous coordinates is:

$$\Delta p = Mp, \qquad M = T_{GT}^{-1}T - I_4. \tag{5.14}$$

Therefore the RMSD over the volume $\mathcal{V}$ is given by:

$$E_{RMS} = \sqrt{\frac{\int_{\mathcal{V}} |\Delta p|^2\, dp}{\int_{\mathcal{V}} dp}}, \qquad |\Delta p|^2 = p^T M^T M p. \tag{5.15}$$

[90] then calculates the integrals in (5.15) by assuming that $\mathcal{V}$ is a sphere with its radius as a parameter and working in spherical coordinates. In our dataset of indoor rooms, most of the scenes can be reasonably modelled as their 3D rectangular bounding boxes, with 3 dimensions in a Cartesian space (width, length and height):

$$\mathcal{V} = [-X, X] \times [-Y, Y] \times [-Z, Z], \tag{5.16}$$

For ScanNet [37] scenes, `alignTransformMatrix` can be used to align the bounding box's axes to the world's axes. By representing the scene in this way and integrating the distance error over every point inside the volume, we define a metric that is independent from the contents of the room.

In order to compare the estimated transform to the ground truth for our indoor datasets, we only need to measure the dimensions of the scene $P = (X, Y, Z)$ and find its center, using the 3D models provided in ScanNet [37] and ChangeSim [139]. If we first assume that the box is centered on the origin, once T is estimated, we can use T_{GT} – provided by the method in Subsection 5.4.1.1 for ScanNet– to calculate A, b (submatrices of M) and D defined as:

$$M = \left[\begin{array}{ccc|c} & A & & b \\ \hline 0 & 0 & 0 & 0 \end{array}\right], \quad D = diag(A^T A) = -2 \begin{bmatrix} A_{1,1} & 0 & 0 \\ 0 & A_{2,2} & 0 \\ 0 & 0 & A_{3,3} \end{bmatrix} \tag{5.17}$$

then with v, corresponding to p in non-homogeneous coordinates:

$$|\Delta p|^2 = v^T A^T A v + 2 b^T A v + b^T b. \tag{5.18}$$

and

$$\int_{\mathcal{V}} b^T b \, dp = b^T b \int_{\mathcal{V}} dp, \qquad \int_{\mathcal{V}} 2 b^T A v \, dp = 0. \tag{5.19}$$

and finally

$$\int_{\mathcal{V}} v^T A^T A v \, dp = \frac{1}{3} P^T D P \int_{\mathcal{V}} dp. \tag{5.20}$$

Therefore the equation for the RMSD for a centered box is:

$$E_{RMS} = \sqrt{b^T b + \frac{1}{3} P^T D P}. \tag{5.21}$$

As the origin of our aligned models is actually set to the center of the scene's floor, we need to move the box to a new center v_c by updating M:

$$M' = M M_c, \qquad M_c = \left[\begin{array}{ccc|c} & I_3 & & v_c \\ \hline 0 & 0 & 0 & 1 \end{array}\right], \tag{5.22}$$

then $A' = A$ and $b' = b + A v_c$, therefore:

$$E_{RMS} = \sqrt{(b + A v_c)^T (b + A v_c) + \frac{1}{3} P^T D P}. \tag{5.23}$$

An illustration of the errors metrics for a rectangle is shown in Figure 5.8. In this 2D example, the relationship between the RMSD and the translation and rotation errors is made apparent, as well as the impact of the studied box's dimensions. We also note that E_t does not directly appear in the $3D$ formula for the RMSD Equation 5.23 because

the translation error is measured at the centre of the room's floor –the origin of the coordinate system of an aligned ScanNet model– whereas the RMSD uses the 3D center v_c of the model. The vertical offset between these two definitions of the room's centre is not present in the 2D case from Figure 5.8.

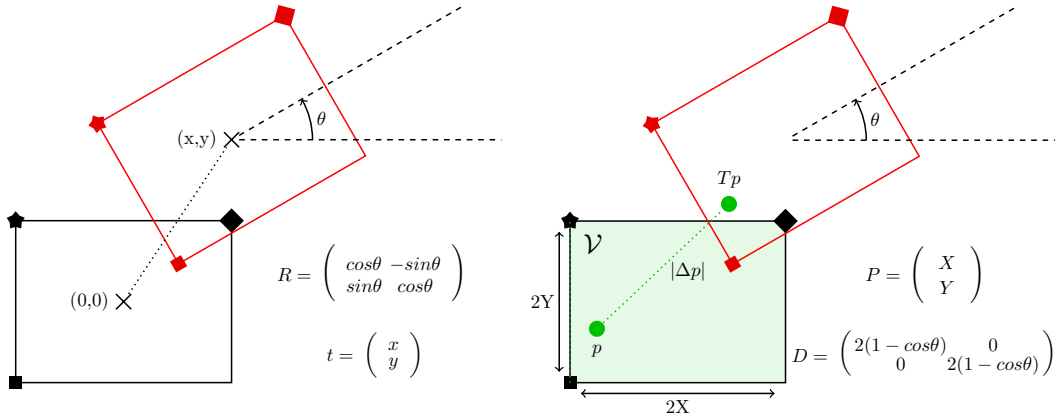


FIGURE 5.8: Error metrics for a 2D transformation. The ground truth is $T_{GT} = I_4$ and the box's center v_c is at the origin. The estimated transformation T differs from T_{GT} by an horizontal offset x, a vertical offset y, and a rotation θ. Left: the translation error is $E_t = \sqrt{x^2 + y^2}$ and the rotation error is $E_R = 2\sqrt{1 - cos\theta}$. Right: the RMSD is $E_{RMS} = \sqrt{E_t^2 + \frac{1}{6}E_R^2(X^2 + Y^2)}$.

5.4.3 Results

Since our dataset includes indoor scenes with a smaller scale than the outdoor scenes from the SYNTHIA [158] or KITTI [62] datasets used in [72], we reduce the node fusion threshold T_{object} from Subsection 5.3.1.2 from 10 meters [61] to 1 meter.

5.4.3.1 Graph extraction

In order to compare the LN-based [72] and SN-based graph extractors' abilities to preserve the object instance segmentation of the scenes, we use the Adjusted Rand Index (ARI) [86]. The ARI is designed to compare the similarity between two clusterings of data, and has been used in previous work for evaluating point cloud segmentation [23]. In our case we are clustering the 3D points extracted from the regions in the label maps.

We will be comparing to which object instance they belong in the ground truth, and to which node in the graph they are associated during extraction.

The reference clustering is simply obtained by looking up the index of each region in the instance segmentation maps provided in ScanNet [37], as they are aligned with the label maps. During extraction, as explained in Subsection 5.3.2, the 3D points are converted into nodes for the graphs:

- Using LNs, only a single 3D point is used to produce a new node, while the "duplicates" are ignored.
- Using SNs, all points identified as duplicates of each other are bundled into a single node.

Therefore the estimated clustering –from graph extraction– can be inferred from which nodes are identified as duplicates of each other. For SNs this is directly detailed by the final graph structure, and for LNs we can keep track of the ignored nodes during the extraction to monitor the underlying clustering.

The ARI results over the ScanNet dataset [37] are detailed in Table 5.3. An ARI of 1 indicates identical clusterings –indices' permutations notwithstanding, which are indeed irrelevant in our case– while an ARI of 0 occurs for independent clusterings. Additionally, the ARI may be negative if the estimated clustering is "worse than random" (the clusterings are not independent). In this light, our SN-based extractor better preserves the 3D point clustering inferred from the instance segmentation maps. In practice this means that objects in the scene are less likely to be fragmented or merged with one another into nodes during graph extraction.

We also provide in the table the results for a Ground Truth-based (GT) extractor, which directly uses the instance segmentation maps to create the SN, therefore getting an ARI of 1. This shows that, on average, while the scenes from ScanNet [37] contain 31 objects, the SN extractor recovers 19 and the LN extractor 50: meaning that the SN extractor tends to merge objects of the same label into a single node. The table shows a slight discrepancy in the number of points used between the SN and GT extractors, which is due to the filtering of nodes which have less than $T_{super} = 10$ points (see Subsection 5.3.2).

Also in Table 5.3 are the number of nodes created and 3D points used on average per graph for ChangeSim [139], for which we do not have the instance segmentation GT (therefore no ARI results). For LNs, the number of points is the same as the nodes since only one point is required to the computation of spatial attributes. For SNs, the number of points per node is 652.89 on average for ScanNet and 120.50 for ChangeSim, assuming that all scenes are weighted the same –which differ from the values obtained by dividing the number of points by the number of nodes, as those give larger weight to

TABLE 5.3: Average results of the graph extractors over the datasets

ScanNet [37]	Number of nodes	Number of points	ARI
GT extractor	*31.09*	*12374.80*	*1.000*
LN extractor	49.99	49.99	0.599
SN extractor	18.80	12376.66	**0.705**
ChangeSim [139]	Number of nodes	Number of points	
LN extractor	767.59	767.59	
SN extractor	172.28	17790.72	

scenes with more nodes. Therefore a lot more memory is required when generating the spatial attributes for SNs.

However, using SNs significantly reduces the number of nodes in the graph, which reduces the number descriptors to generate. Assuming the optimal descriptors H_3 described by [72] with the 40 NYU labels [173]: each descriptor is of size 64000, while each point only stores 3 coordinates. In conclusion, the SN extractor produces graphs that more closely match the scene instance segmentation, and the total memory cost must be evaluated by taking into account the descriptor types and depths used to achieve the desired quality of registration.

5.4.3.2 Variants of the adjacency descriptor

In this section, we evaluate which of the modifications introduced in Subsection 5.3.3.2 improve registration results. The ScanNet dataset [37] is processed using 4 variants of the adjacency descriptor with depth d (equivalent to walks of $d-1$ steps) going from 2 to 19 and then from 20 to 160 with increments of 20. The results are presented in Figure 5.9: the variant can either use the powers of the standard (Boolean) adjacency matrix (A^k) or the "no return" (Boolean) matrices ($B^{(k)}$), and it can either have constant weights or be inversely proportional to the depth (indicated by the w index).

As well as the 4 main variants, we also plot the results for depths 2 to 4 using the non-approximated "no return" matrices ($C^{(k)}$) and ($C_w^{(k)}$) to validate our approximation. When comparing the variants, depths 1 and 2 result in the same descriptors, and we do not plot the $d = 1$ case, for which the node descriptor is simply its label. The registration error is plotted against the descriptors' size (using a logarithmic scale). We use the median over the dataset rather than the arithmetic mean in order to not let failed registrations influence the results too greatly, as the translation error and RMSD can get arbitrarily large due to numerical errors. We also avoid using the geometric or

harmonic means for this reason: if an error metric for a single instance pair gets too close to 0, it nullifies the geometric and harmonic means over the dataset for that metric.

When using LNs, $(B_w^{(k)})$ consistently gives better results than the other variants, especially after $d = 4$. All variants also seem to stabilize for higher values of d except $(B^{(k)})$. This is likely due to the compounding inaccuracies introduced by our estimation of $(B^{(k)})$, which is however accurate to $(C^{(k)})$ for the depths it was computed at.

When using SNs, better results are obtained at low depths, after which the registration deteriorates then stabilizes (except for $(B^{(k)})$). This is likely due to the smaller size of the SN-graphs. The degradation is more pronounced for $(A^{(k)})$, as the same paths are likely being taken repeatedly when extending the descriptors, giving the relevant information less weight during matching. Despite this, the results are generally better using SNs –even at less than optimal depths– and $(B_w^{(k)})$ outperforms the other variants using either graph extractor.

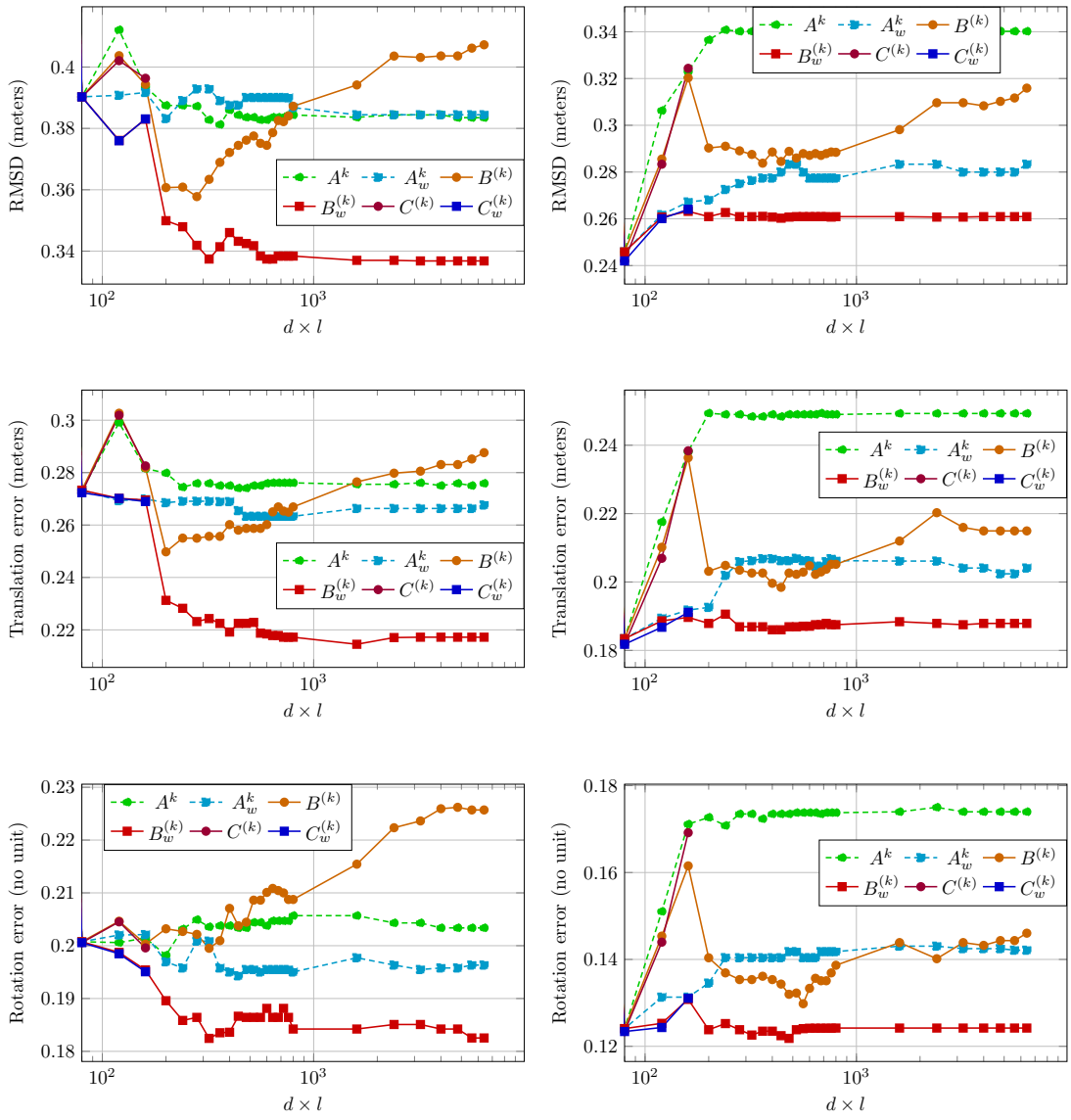


FIGURE 5.9: Median errors metrics for ScanNet [37] vs. $d \times l$ for 6 variants of the adjacency descriptor, starting at $d = 2$. Left: using LNs, right: using SNs. Lower is better.

5.4.3.3 Adjacency descriptor vs walk histogram

The weighted "no return" variant ($B_w^{(k)}$) of our descriptor is compared to the walk histogram descriptor [72] (H_d) on the datasets. We run the pipeline for (H_d) for depths 1 to 4, and for $d > 1$ we use a merging m of 20, 10, 8, 5, 4, 2 or 1, all divisors of the number of classes $l = 40$.

We again calculate the registration error against the descriptor size: $s_D(l, d)$ for our descriptor and $s_H(l, d, m)$ for the walk histogram. The results for ScanNet [37] are presented in Figure 5.10, and for ChangeSim [139] in Figure 5.11.

Each method is evaluated with and without RANSAC filtering of the outliers, and we can see that the results for both methods are significantly improved when RANSAC is used, underlying the shortcomings of the point cloud extraction algorithm as mentioned in Subsection 5.3.1.5. The exception is ChangeSim using LNs, where ($B_w^{(k)}$) manages to get as good of results with or without RANSAC filtering (see Figure 5.11 top row). However, studying the methods without filtering will be our focus, since their performance is more indicative of the quality of the graph matching process. Indeed, if the graphs are to be used for AR purposes beyond localisation as previously pointed out, we should maximize the number of correct matches.

Both the adjacency and walk histogram descriptors are identical at $d = 1$ ("node label" descriptor) but the adjacency descriptor quickly improves registration as d increases. At $d = 2$ both methods provide similar results but the adjacency descriptor is much lighter. At $s(l, d, m) = 1600$, the walk histogram explores only a neighbourhood of $d = 2$ while the adjacency descriptor explores $d = 40$, with significant RMSD reduction in all configurations except that of ScanNet with SNs (see Figure 5.10 right column).

As seen in the previous section, this is likely due to the small size of these graphs, since increasing the depth improves results for larger scenes (ChangeSim) or bigger graphs (LNs). Moreover, for ScanNet the results are still better using SNs at $d = 2$ rather than LNs at any depth (see Figure 5.10 left column).

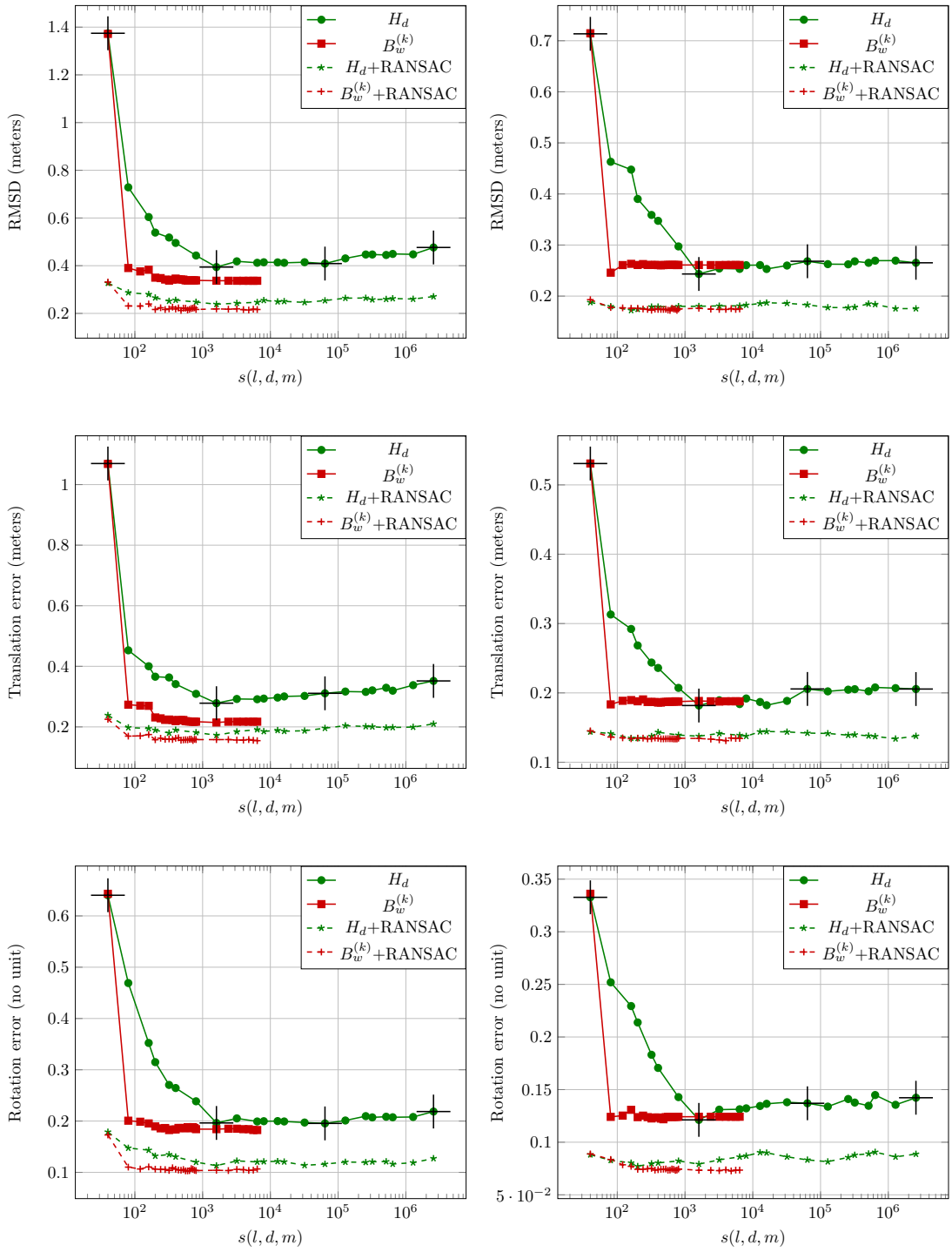


FIGURE 5.10: Median errors metrics on ScanNet [37] vs. $s(l, d, m)$ for the walk histogram descriptor and the best adjacency descriptor variant, with optional RANSAC filtering. For the walk histogram $d = 1, 2, 3, 4$ with $m = 1$ are shown as crosses. Left: using LNs, right: using SNs.

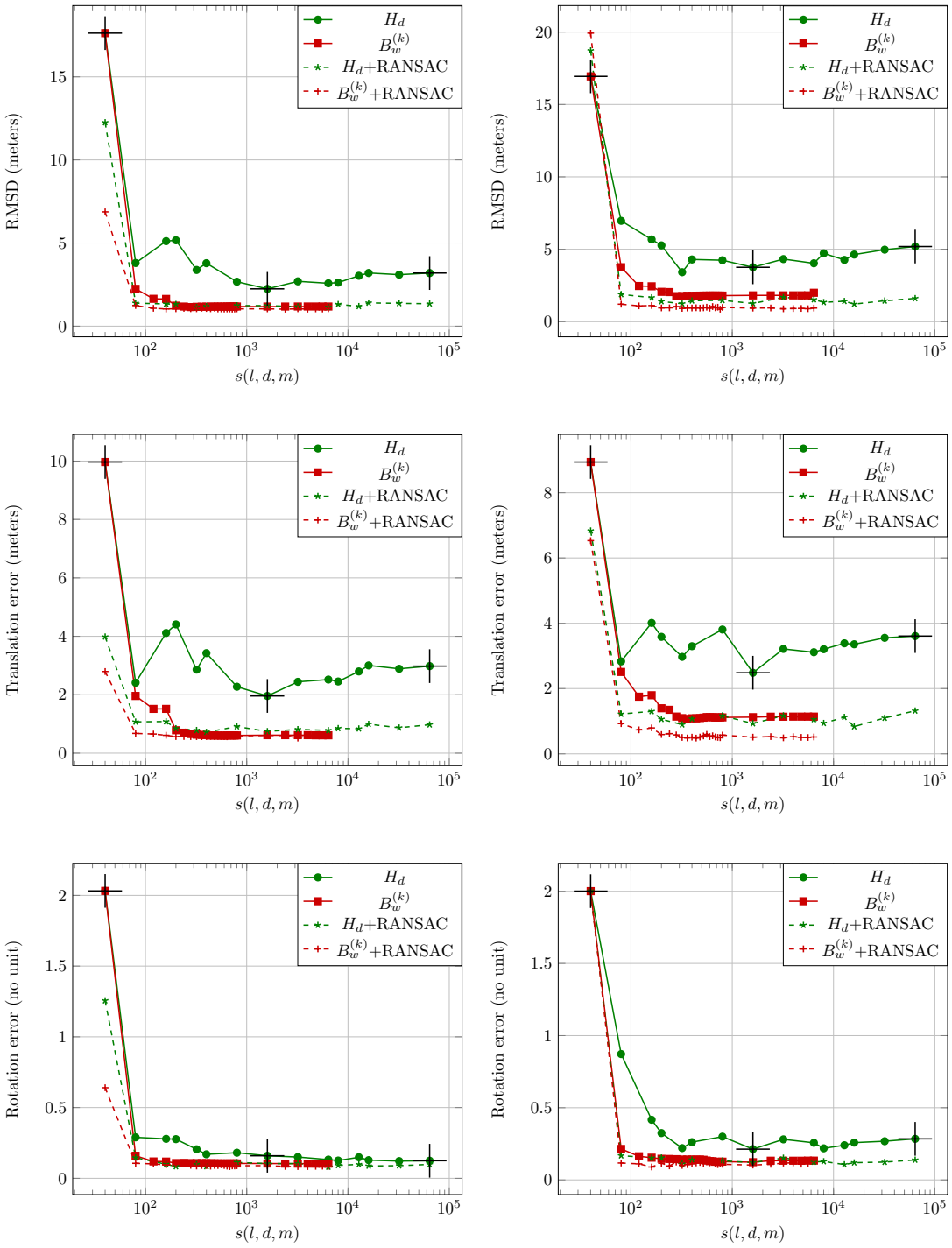


FIGURE 5.11: Median errors metrics on ChangeSim [139] vs. $s(l, d, m)$ for the walk histogram descriptor and the best adjacency descriptor variant, with optional RANSAC filtering. For the walk histogram $d = 1, 2, 3, 4$ with $m = 1$ are shown as crosses. Left: using LNs, right: using SNs.

The performance of H_2 and its corresponding ($B_w^{(k)}$) in depth ($d = 2$) and size ($d = 40$) over the whole dataset are detailed in Figure 5.12 for ScanNet and in Figure 5.13 for ChangeSim. We also provide results for H_3 as it was the optimal descriptor in [72] in both figures, and the node label descriptor ($d = 1$) for ScanNet.

In Figure 5.12, we plot on the y-axis the percentage of successfully registered scene pairs according an error metric threshold on the x-axis. The 2 or 3 vertical dotted lines show on the x-axis when 90% of successful registrations is reached for the plain curves (not dashed) of the matching color. For ScanNet, 90% success is always reached first with ($B_w^{(k)}$) at $d = 40$, then with H_2, then with the node label. Figure 5.12 also shows that ($B_w^{(k)}$) at $d = 2$ and H_2 give the same results despite the much smaller size of the former. Figure 5.12 and Figure 5.14 (left side) also show that after a fast growth, the SN curves (right column in Figure 5.12) do not near 100% by the time the LN curves do (left column in Figure 5.12): one is not consistently producing better results than the other on all scenes, but their performances are close on 90% of them.

In Figure 5.13, we plot the error metrics over the whole ChangeSim dataset, with each grouping of histogram columns representing the performance on one of the 16 instance pairs. It shows that ($B_w^{(k)}$) at $d = 40$ consistently outperforms H_2, but once again the ranking of performances between the SN (right) and LN (left) representations depends on the scene (as seen also in Figure 5.14).

Overall, for the walk histogram descriptor [72], increasing the depth eventually degrades results despite the larger descriptor size. Since it should not have the same issue of homogenization with increased depth as our non-weighted variant, it is likely that the scarcity of unique walks in the graphs compared to the size of the histogram makes the calculation of the normalized dot products less reliable.

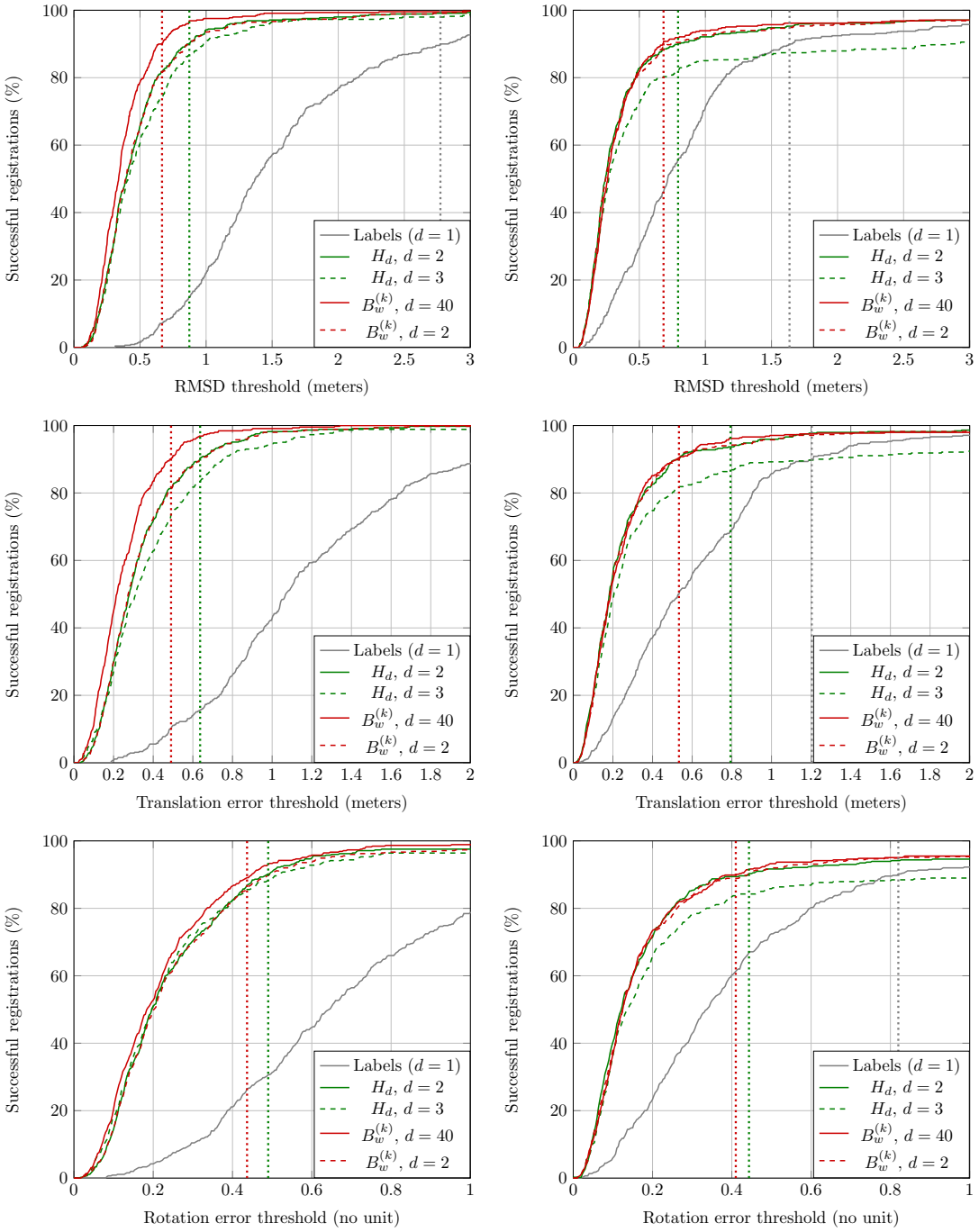


FIGURE 5.12: Percentage of successful registrations on ScanNet [37] vs. an error metric threshold for success, using the node's label, the walk histogram descriptor and the best adjacency descriptor variant. The adjacency descriptor's size at depth 40 matches the walk histogram's size at depth 2. Left: using LNs, right: using SNs.

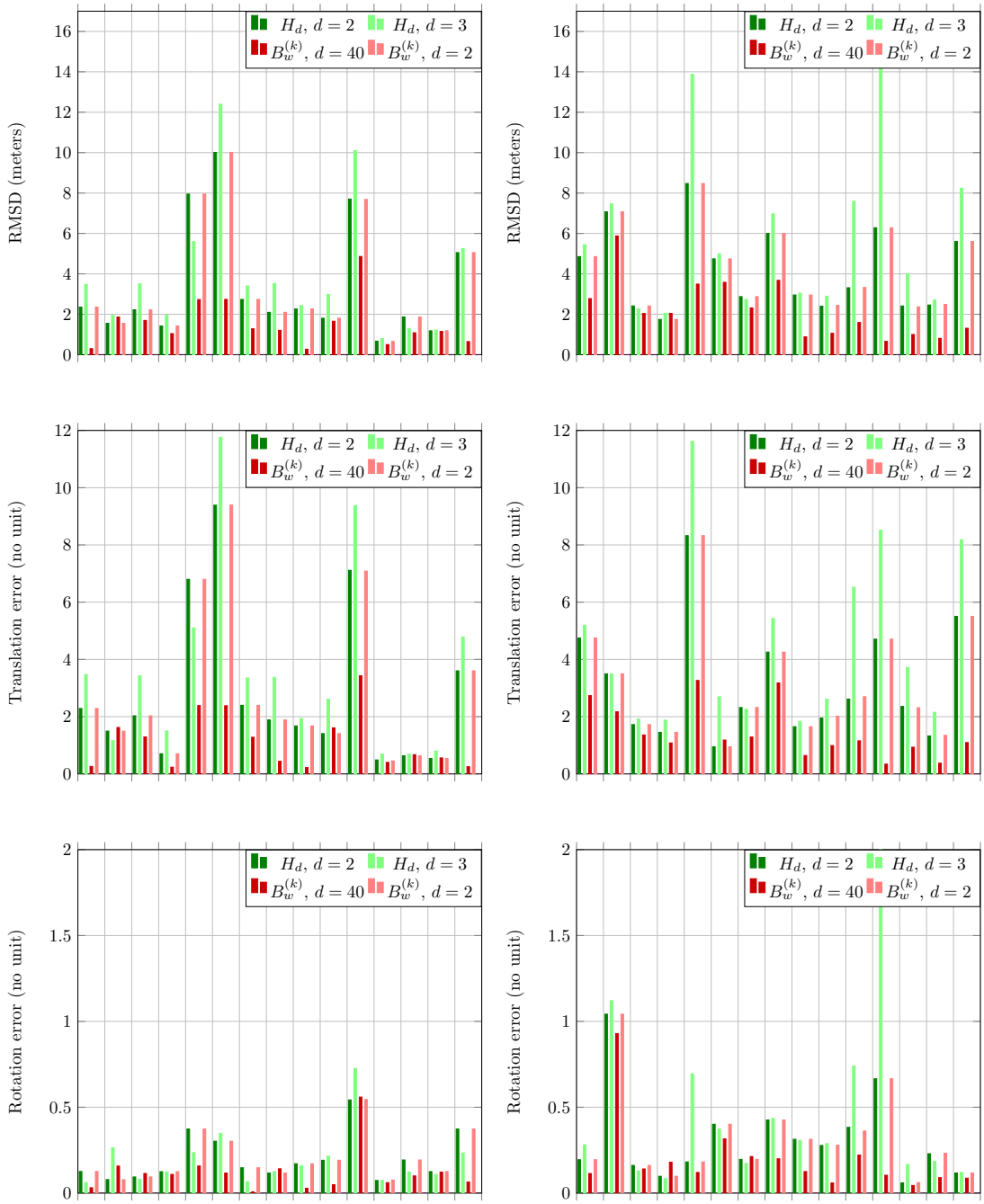


FIGURE 5.13: Error metrics for each ChangeSim [139] instance, using the node's label, the walk histogram descriptor and the best adjacency descriptor variant. The adjacency descriptor's size at depth 40 matches the walk histogram's size at depth 2. Left: using LNs, right: using SNs.

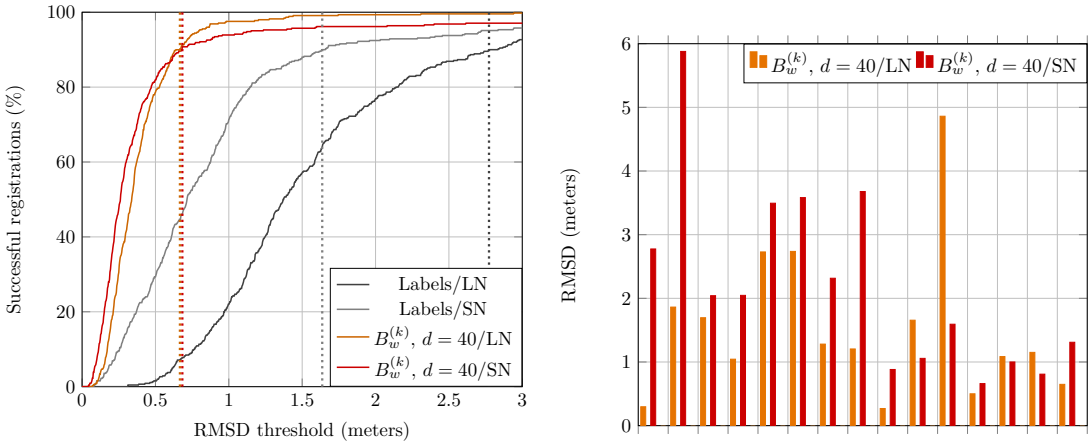


FIGURE 5.14: Left: Percentage of successful registrations on ScanNet [37] vs. an RMSD threshold for success, using the node's label and the best adjacency descriptor variant with LNs and SNs. Right: RMSD for each ChangeSim [139] instance using the best adjacency descriptor variant with LNs and SNs.

5.4.3.4 Generation and matching times

The descriptor generation and matching time is also measured for the two descriptors and displayed in Figure 5.15. ($B_w^{(k)}$) consistently shows lower RMSD than H_d at equivalent times. The different is particularly noticeable for $d = 2$, as both descriptors contain the same information but said information is more quickly encoded and compared using the adjacency histograms. Figure 5.15 also shows that the speed gain from using the "plain" adjacency descriptor ($A^{(k)}$) are not significant enough to justify its use. Conversely, the non-approximated variant ($C_w^{(k)}$) is much slower for no practical quality improvement.

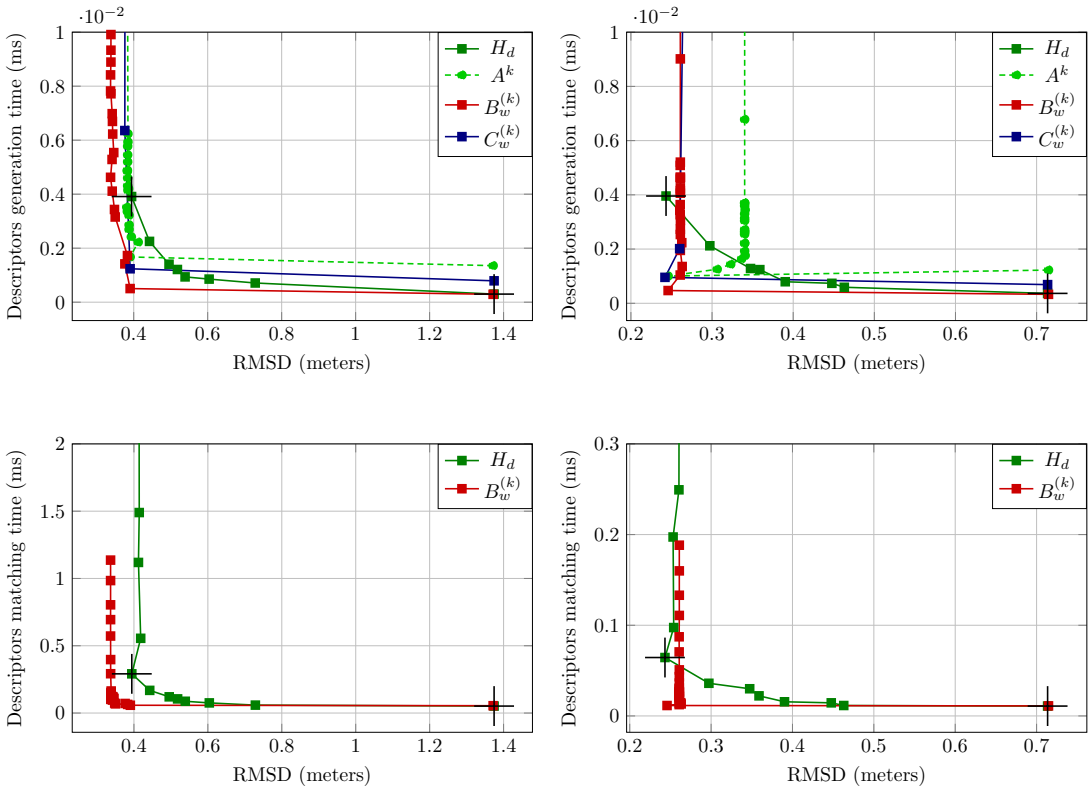


FIGURE 5.15: Average times on ScanNet [37] for the descriptors generation (per node, top) and matching (per graph, bottom) vs. the RMSD for the walk histogram descriptors and variants of the adjacency descriptor at different depths d. For the walk histogram $d = 1, 2$ with $m = 1$ are shown as crosses. Left: using LNs, right: using SNs.

As a conclusion, we note that descriptor generation lasts less than a millisecond for graphs containing hundreds of nodes using $(B_w^{(k)})$, on a virtual machine with 16GB of RAM and four 2.60GHz processors. The matching step is also done in less than a millisecond on average on ScanNet, at most depths.

5.4.3.5 Impact of the depth

In the previous sections we have shown that at a given depth d or for the same descriptor size, our new descriptor provides similar or better results than H_d. In addition, this compressed representation allows for a deeper exploration when constructing the descriptor. We can check the benefits of the deeper search by plotting how many instance pairs in the dataset are best registered at a given depth.

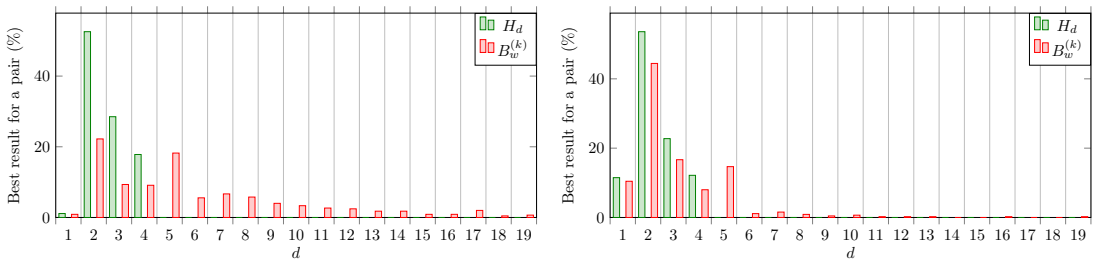


FIGURE 5.16: Distribution over the depth values in percent for when an instance pair from ScanNet [37] reaches its lowest RMSD. Left: using LNs, right: using SNs.

As previously seen for ScanNet, increasing the depth over $d = 2$ benefits the registrations when using LNs but not SNs. This is shown in the depth histograms of Figure 5.16, where the dataset was processed using H_d from depth 1 to 4 (the maximum possible depth on our hardware), and $(B_w^{(k)})$ from depth 1 to 20. Once again using the node's label $(d = 1)$ for the graph matching does not generally yield the best results, but it is more effective using SNs (right) than LNs (left). However, the optimal depth using $(B_w^{(k)})$ is shown to be 2 for SN graphs, with much less positive results after $d = 5$ when compared to LN graphs. This confirms that a deep exploration of the graphs is not required when using the smaller SN graphs produced for ScanNet scenes.

In contrast, Figure 5.17 shows that there are advantages to deeper graph exploration when confronted with bigger scenes with more duplicate objects as with the warehouses of ChangeSim. A significant number of scenes reach their best result for $d = 20$ for

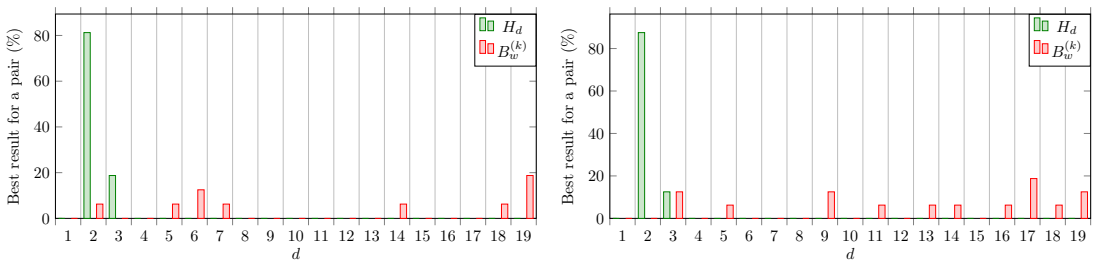


FIGURE 5.17: Distribution over the depth values in percent for when an instance pair from ChangeSim [139] reaches its lowest RMSD. Left: using LNs, right: using SNs.

($B_w^{(k)}$), and could benefit from further exploration, which is not possible using the walk histograms.

To conclude, the proposed adjacency descriptors offer more flexibility in allowing deeper exploration of large scenes when necessary. This can be done without prohibitively large execution time, memory cost, or without degrading the results as with the walk histograms, as the use of depth-wise weighting leads the error to stabilize rather than increase for large depth values.

5.4.3.6 Impact of graph quality

As mentioned in Subsection 5.4.3.1, to assess the quality of the graph extraction process, we also generated Ground-Truth (GT) graphs, that are based on the instance segmentation maps of ScanNet [37] (as shown in Figure 5.18) rather than our object delimitation process (see Subsection 5.3.2). In Figure 5.19 and Figure 5.20, we show how the descriptors perform under those "ideal" conditions for graph extraction.

The RMSD curves (left of Figure 5.19) show a slight improvement of the registration over using the SN graphs (about 3% at the optimal depth), but notably, using GT graphs fixes the issue of quality degradation with increasing depths. Indeed, with the GT graphs, $d = 2$ does not account for more than 40% of the best registration results when using ($B_w^{(k)}$) (right of Figure 5.19) as it did for SN graphs.

Using GT graphs also narrows the performance gap between H_2 and ($B_w^{(k)}$) at $d = 40$, but the latter still outperforms the former at all success thresholds in Figure 5.20 (left side). The performance is also improved for ($B_w^{(k)}$) when using the instance segmentation ground truth (right of Figure 5.20) as success rates of 90% and 100% are reached earlier.

FIGURE 5.18: Frame from a ScanNet [37] sequence (left), semantic segmentation (middle) and instance segmentation (right).
The "walls" –semantically represented by a uniform (pale blue) region– are broken up into two distinct (pale orange and purple) instances.

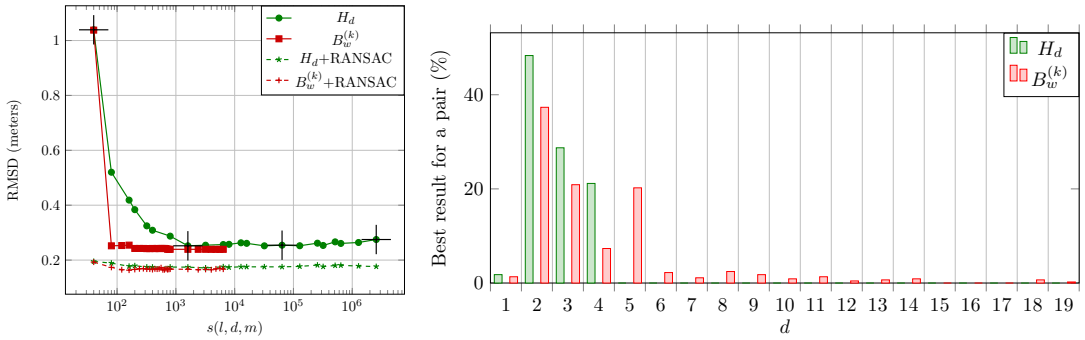


FIGURE 5.19: On ScanNet [37]: Median RMSD vs. $s(l,d,m)$ for H_d and $(B_w^{(k)})$, with optional RANSAC filtering (left); Distribution over the depth values in percent for when an instance pair reaches its lowest RMSD (right).

However, the performance is still the best using SN graphs when only the node label is used for matching (right of Figure 5.20). The most likely explanation is the reduced number of nodes in the graphs. In Subsection 5.4.3.1, we showed that GT graphs contained 31 nodes on average, while SN graphs tend to merge objects, resulting in a 19 nodes average. Reducing the graph's size reduces ambiguity when matching nodes solely on their label, seeing as only nodes of the same label are incorrectly merged using the SN extractor.

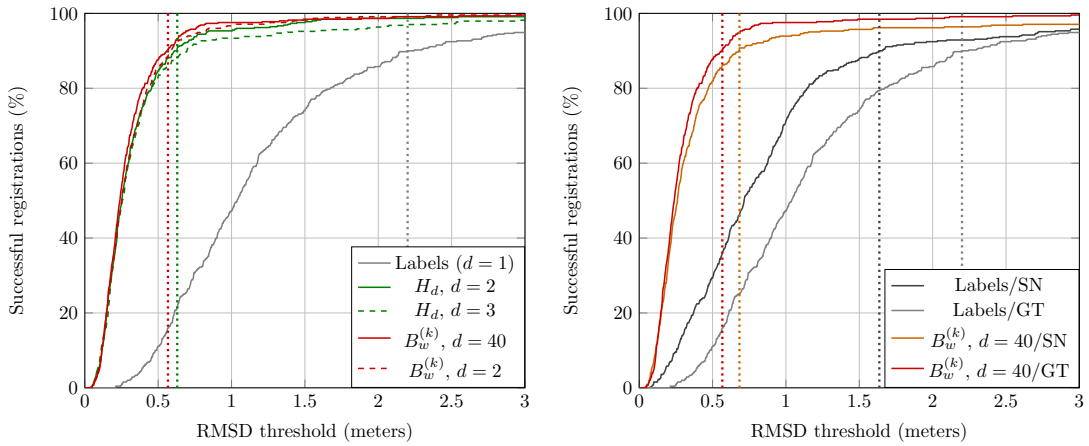


FIGURE 5.20: Percentage of successful registrations on ScanNet [37] vs. an RMSD threshold for success. Left: using the node's label, H_d and $(B_w^{(k)})$ with GT graphs. Right: using the node's label and $(B_w^{(k)})$ with SNs and GT graphs.

To test the robustness of our registration method to a lowering of graph quality, we use an off-the-shelf image segmentation tool that has been trained on the 40 NYU labels [173]. We decided to use 2021's ESANet [169], which we ran on the RGB images sequences provided in ScanNet [37] to produce new label maps, replacing the accurate maps given by ScanNet we otherwise use (see Subsection 5.3.1.1). This approach makes the labelled region-based graph creation less reliable in 2D (as shown in Figure 5.21), but also makes the temporal segmentation of the scene inconsistent, leading to ambiguous 3D segmentation.

In practice, the images are segmented into more regions than the ground truth, which causes the average graph size to increase from 19 to 32 for SNs and from 50 to 124 for LNs (see Table 5.3). It also greatly affects the quality of the object instance segmentation, as the ARI drops to 0.133 for SNs and 0.179 for LNs. As for the registration task, the results using the ESANet label maps are presented in Figure 5.22 and Figure 5.23.

In Figure 5.22, we only plot the walk histogram descriptor's performance for $m = 1$ (no merging). Once again, it is outperformed by $(B_w^{(k)})$ for most depths, and the performance of the adjacency descriptor improves with increased depth whether SNs or LNs are used, as the graph size is larger using the ESANet segmentation.

However, in Figure 5.22 and Figure 5.23, SNs are outperformed by LNs for the localisation task when higher depth values are used for the descriptors. In Figure 5.23, it is especially visible that using SNs benefits the node label descriptor ($d = 1$) –as LNs

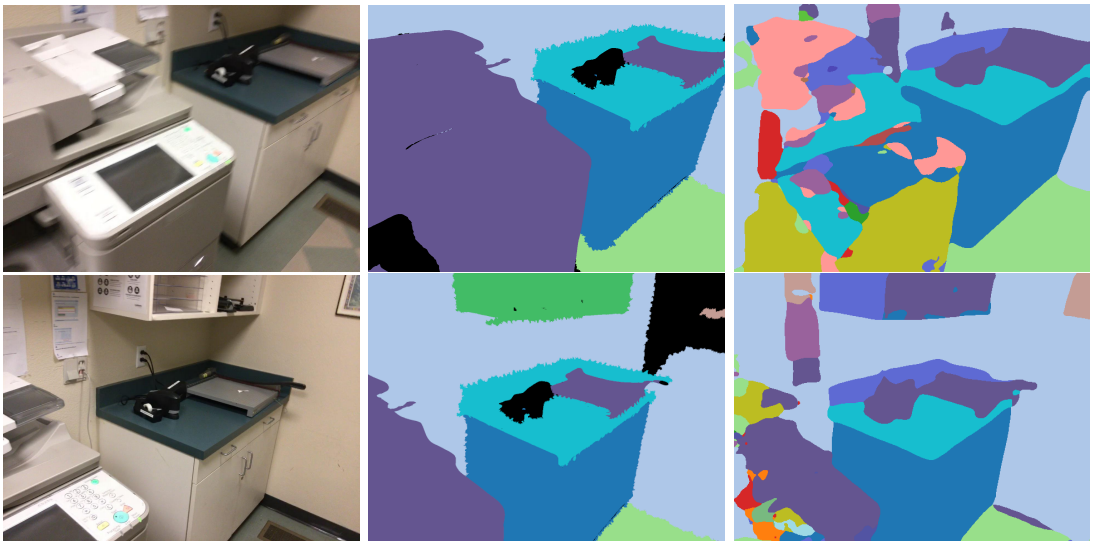


FIGURE 5.21: Frames from a ScanNet [37] sequence (left column), ground truth semantic segmentation (middle column) and semantic segmentation using ESANet [169] (right column). The labelling of 3D regions visible in both frames is inconsistent when using ESANet.

graphs are much larger– but they give a poorer performance for $d = 40$ with $(B_w^{(k)})$.

In conclusion, the quality of the localisation is correlated to the closeness of the graph representation with the actual semantic contents of the scene. Using the instance segmentation ground truth improves the results when descriptors encode a neighbourhood of depth 2 or more around each node. Conversely, not using the 2D label ground truth degrades the results but 90% of instance pairs are still registered within a RMSD of 0.80 meters if $(B_w^{(k)})$ descriptors are used in conjunction with LNs at depth 40 (see left of Figure 5.23). We therefore note that LNs seem more robust to a degradation of the input labels, which makes their use over SNs in the localisation task still relevant when the scene contents are difficult to identify.

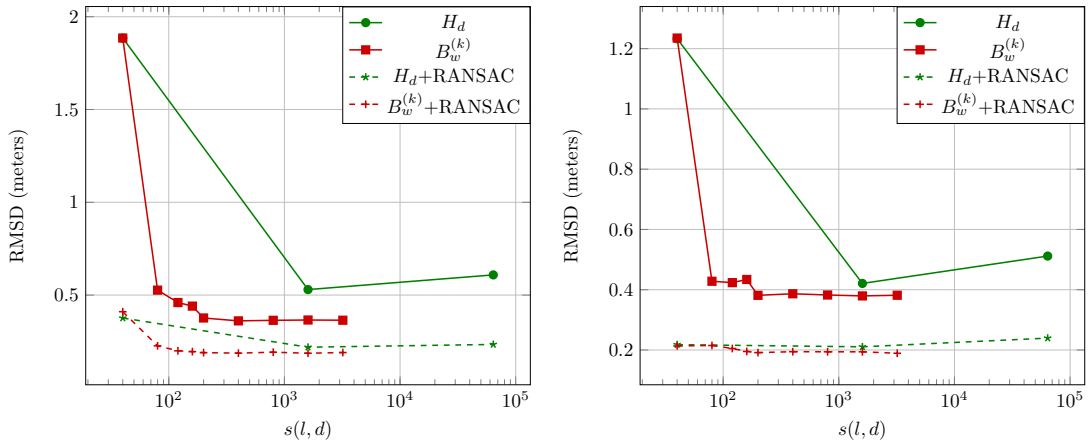


FIGURE 5.22: Median on ScanNet [37] of the RMSD with ESANet [169] labels vs. $s(l, d, m)$ for H_d and $(B_w^{(k)})$, with optional RANSAC filtering. Left: using LNs, right: using SNs.

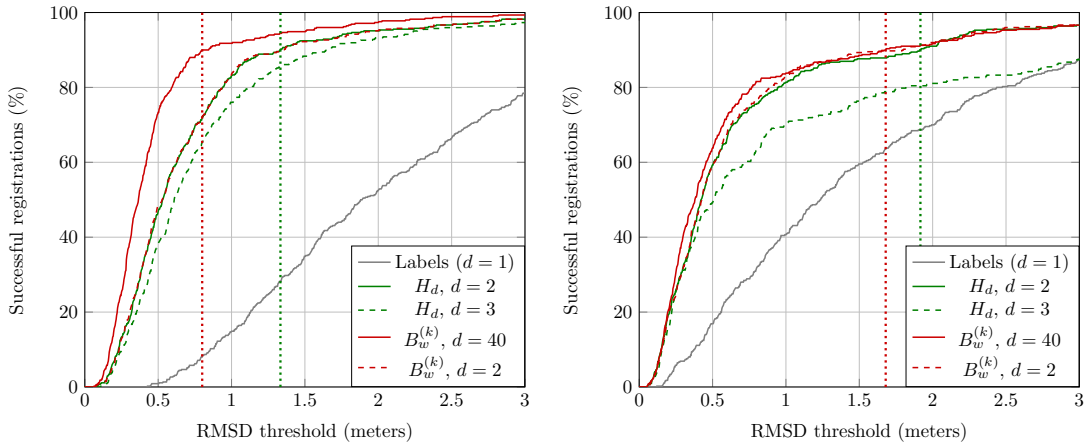


FIGURE 5.23: Percentage of successful registrations with ESANet [169] labels on ScanNet [37] vs. an RMSD threshold for success, using the node's label, H_d and $(B_w^{(k)})$. Left: using LNs, right: using SNs.

5.5 Conclusion

In this chapter, we presented a new graph node descriptor based on the adjacency matrix of a scene graph for indoor scenes. We evaluated its descriptive powers in the graph matching process as part of a pipeline for indoor scene registration. Compared to a similar state-of-the-art method [72], we found that it produces a more precise alignment of the scene models. We also demonstrated the computing efficiency of our proposed descriptor in semantically rich scenes. Indeed, the size of the descriptor necessary to encode the many different object labels for large neighbourhoods is greatly reduced from existing works [72, 61], while improving its computing time. The ScanNet [37] and ChangeSim [139] datasets were processed in order to be usable in our scene registration task: we generated the registration ground truth for ScanNet, and the camera poses ground truth for ChangeSim. The registration quality was evaluated using a metric for transformation error that we adapted from an earlier RMSD metric [90] to fit our data's profile.

We also experimented with an alternative graph representation and extraction method, which is much more representative of the object-level understanding of the scene required for tasks beyond localisation. This Super Node-based representation performed equivalently or better than the previous on this task, while the evaluation of its scene representation quality was quantitatively evaluated using a relevant clustering metric: the Adjusted Rand Index [86].

Given a pair of scene graphs, preliminary registration can be performed rapidly and reliably using the proposed graph descriptor and graph representation. These methods also offer more flexibility to different types of scenes thanks to their slower scaling with the scene's dimensions, semantic density, or observation radius required to reliably identify objects with to their surroundings.

Chapter 6

Conclusion and Future Work

In this thesis, we introduced novel methods to tackle the problem of geometric Change Detection for Mixed Reality experiences. Said CD was performed between two states of a real-world scene, be it through photometric and geometric observations or semantic analysis of the scene's contents. We defined MR as photo-realistic Augmented Reality, and found that "behavioural realism" would also need to be achieved to preserve its immersive quality. Behavioural realism involves the recreation of believable physical interactions of the virtual contents with the real world, which requires semantic information about the scene beyond its geometry (the real objects' types, materials, relationships etc.). This work was undertaken in order to provide a means of maintaining a model of the real environment, which would include the necessary information to achieve the desired level of realism for MR applications.The use of CD to construct said model was motivated by the need to provide a fast and lightweight solution to the maintenance problem: gradually applying updates to the model to mirror reality. Indeed, the widely available AR devices are currently not equipped with the necessary scanning or computing resources to recreate such a model without a prior with a high enough fidelity, or within a time acceptable to the AR user. Additionally, devices such as AR glasses and smartphones can only partially observe the scene –due to sensor limitations or occlusions– making the use of a reference scene model mandatory for the experience to be able to begin as the application is launched.

We learned in our study of the state-of-the-art (Chapter 2) in the field of CD, that most existing methods are too demanding for the hardware of mainstream AR devices, or are not able to provide 3D geometric results in real-time as required by most applications in this field. Nevertheless, some of the works for robotics –in autonomous navigation and interaction for instance– do provide approaches that aim to solve the problem of scene understanding at interactive time, which we have been able to exploit to fit our purposes [136, 151, 188].

In Chapter 3, we acknowledged the difference in nature between the data required

for an MR application, and that which can be acquired using an AR device, by implementing a CD method that can compare scene states represented by said heterogeneous data. We made the hypothesis of a "semi-static" environment: an environment where changes occur between AR sessions but not while it is observed during said sessions. Consequently, changes could be studied by comparing two static models of this environment: one for its past state –the "prior" or "reference"– and one for its current state –the "observation" or "query". We chose to represent the past state as an untextured 3D mesh –required for photo-realistic rendering but difficult to capture with the AR device– and the current state as a sequence of images –readily available using a simple camera. In this context, the 3D representation can be updated solely by detecting geometry that was either added or removed between the two scene states. We used our own image reprojection-based method, which corrects the existing bias [151] of similar methods [136, 188] toward the detection of geometry addition, to the detriment of removal detection.

Following the work in Chapter 3, we noticed that updating the geometry of the scene model would necessitate more precise results than our photo-geometric analysis could provide from an outdated mesh and up-to-date images. Therefore, we concluded in Chapter 4 that an object-level understanding of the scene, and subsequent segmentations of the 3D and 2D data from the scene model, would allow us to better preserve the model's fidelity to the real environment after maintenance. We settled on a semantic Scene Graph representation, where nodes would represent objects in the scene and edges their (spatial) relationships.In such a SG, the attributes of nodes –namely the location and type of object they represent– link semantic labels to the relevant scene geometry and textures. This SG is updated through semantic analysis of the scene, which allows us to better define the "relevancy" of changes –finding which changes affect the MR experience– as well as achieving behavioural realism.

Finally, we introduced a method to match our semantic SGs through the identification of similar nodes in Chapter 5. The purpose of this work was to show how this scene representation could be used for various tasks arising from general AR applications (i.e., not only MR), such as user localisation and virtual content integration based on semantic contents. With the assumption that modern AR devices are now equipped with a LiDAR sensor or stereo cameras –in contrast to Chapter 3– we presented a novel graph generation method from RGB-D sequences. In the same chapter, we also introduced a novel node descriptor that can be used to measure similarity between the objects (SG nodes) extracted from two different RGB-D sequences. Using these proposed methods, two scene states can now be compared and aligned through semantic matching of their graphs, and persistent virtual content can be reliably reintegrated into the scene between AR sessions.

In conclusion, using the new hardware and the reference data hypothesis introduced in Chapter 4, the photo-geometric CD approach from Chapter 3 could be used on an object-segmented mesh, opening up the possibility of fidelity-preserving updates to be made to said mesh. Additionally, with said objects being represented as nodes in a graph of the scene, the registration of the image sequence required for the CD framework from Chapter 3 could be handled by the SG-based method from Chapter 5. As outlined in Chapter 4, when changes are detected in the scene, the removed objects can simply be deleted from the segmented mesh, while displaced objects can have the corresponding mesh vertices match their trajectories. By using an AR device able to capture the RGB-D frames for the SG generation in Chapter 5, we now have access to 3D geometry relevant to the addition of novel objects to the mesh. In effect, a semantically semantic mesh of the scene can be updated using RGB-D observations of its current state thanks to our photo-geometric CD and semantic registration approaches. As such, only semantic CD remains to be solved in order to exhaustively describe the changes with the precision required to preserve the realism of MR experience in changing scenes.

We also found that our work on SGs could potentially be used in conjunction with higher precision scanning utilities as a scene authoring tool from real data, that would record and organise geometric, photometric and semantic information about the scene in our proposed lean and versatile SG representation. Considering this and the previously mentioned registration application, our SG generation method is relevant to both ends of an AR framework: the production and organisation of virtual assets and the analysis of the real environment. As graphs have been used as the translation medium between human instructions and robots' operations in robotics, they can also be used as a standardised interface between the real and virtual components of an AR application. Indeed, the ability to communicate a set of instructions relating to a real (or possibly virtual) scene will improve the behavioural realism of AR agents (useful for MR applications), and generally give them more ways to interact with the scene.

6.1 Future work

As previously mentioned, semantic CD would greatly benefit our scene model update framework for MR, yet remains to be addressed. In Chapter 4, starting from the semantic SG model, we outlined an approach to tackling CD for MR applications through the study of 4 problems:

- CD between two global SGs,
- CD between a partial query SG and a global reference SG,
- creation of the query SG guided by the prior knowledge contained in the reference,

- update of the scene geometry from semantic CD.

The first problem was partially addressed in Chapter 5, as being able to match the nodes of the two graphs is the first step towards tallying their differences. As we tackled the issue of registration through graph matching, we introduced a similarity metric between nodes (or objects) in the scene, which could be used to semantically match nodes whose other attributes (geometric or photometric properties) may differ. In our framework, the final registration step identifies the most geometrically coherent node matches when aligning the scene instances –not every match is required– which makes our approach robust to changes. CD will require the matching algorithm itself to identify "unmatchable" nodes due to changes in the scene impacting the graphs' topologies. This will be evaluated using a graph matching ground truth, rather than a registration one.

Following this, the context of AR would require an approach able to match a user's current limited view of the scene, against the global reference. This perspective can be emulated in our data used in Chapter 5 by simply shortening the image sequences used to simulate the observation of the scene. As outlined in Chapter 4, while SG-based localisation is addressed in the literature, CD will require an understanding of the difference between out-of-view and removed objects. Similar attention should be given to large objects that can only be partially observed at any time, as they can also help to define the boundaries of the AR device field of view. All these added concerns would impose new requirements on the scene model used, as a more thorough understanding of the objects' properties would be required to address them. This raises the question of how these new properties might be updated or checked against the current scene, given our assumed hardware limitations. Mirroring the method described in Chapter 3, semantic change detection and graph updates would be performed using heterogeneous data. Semantic and geometric analysis of the scene would be conducted with the help of the prior, so as to define regions of interest in the current observations and allow the maintenance of all that is stored in the scene model.

Finally, so far we have assumed that our reference was a perfect representation of the scene as far as its "tracked" properties were concerned (the scene, the nodes' attributes, the mesh...) but can this hypothesis hold after multiple updates to the model? If the scene only veered slightly from a stable reference state between AR sessions, updates to the model would not need to be permanent, and the updated version would only exist for that session. However, this hypothesis cannot be made in most real environments, and over time the reference state will differ too greatly from the current observations to be updated from change detection in real-time. Therefore, the scene representation and CD technique must be chosen to preserve the quality of the former over multiple iterations. This is especially true of geometric properties, which are the most difficult to capture with dedicated hardware.

Once the above-mentioned 4 problems have been solved, the resulting scene model will be able to provide the necessary data to MR applications to achieve the desired level of realism. It would then be possible to explore possible extensions to the scene model, which allow for better interactivity between the virtual and the real environments, or with the user. As our scene model was inspired by advancements in the field of robotics, it would likely be an effective translation medium between human and machine perceptions of the environment. More generally, the model is a means to communicate scene information, and while the raw 3D and 2D data may be sizeable, the semantic contents and layout represented by a graph are lightweight. This scene model tailored to monitoring scene changes may be as efficient as transmitting scene updates: the vocabulary of changes can be used to describe modification instructions. Once a scene is understood on a semantic level, the many geometric and photometric changes required to describe its dynamic parts can be effectively summarised as a set of semantic changes.

Bibliography

[1] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 208–216, USA, 2003. IEEE Computer Society. ISBN 0769520065. doi: 10.1109/ISMAR.2003.1240704.

[2] J. Ahlberg. Active contours in three dimensions, 1996.

[3] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi. Street-view change detection with deconvolutional networks. In D. Hsu, N. M. Amato, S. Berman, and S. A. Jacobs, editors, *Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016*, 2016. doi: 10.15607/RSS.2016.XII.044. URL http://www.roboticsproceedings.org/rss12/p44.html.

[4] R. Ambrus, J. Folkesson, and P. Jensfelt. Unsupervised object segmentation through change detection in a long term autonomy scenario. In *16th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2016, Cancun, Mexico, November 15-17, 2016*, pages 1181–1187. IEEE, 2016. doi: 10.1109/HUMANOIDS.2016.7803420. URL https://doi.org/10.1109/HUMANOIDS.2016.7803420.

[5] Y. Aoki, H. Goforth, R. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7156–7165. IEEE Computer Society, 2019. doi: 10.1109/CVPR.2019.00733. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00733.

[6] I. Armeni, Z.-Y. He, A. Zamir, J. Gwak, J. Malik, M. Fischer, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5663–5672. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/ICCV.2019.00576.

[7] G. Baruch, Z. Chen, A. Dehghan, Y. Feigin, P. Fu, T. Gebauer, D. Kurz, T. Dimry, B. Joffe, A. Schwartz, and E. Shulman. ARKitscenes: A diverse real-world dataset

for 3D indoor scene understanding using mobile RGB-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=tjZjv_qh_CE.

[8] J. Bauer, A. Klaus, K. Karner, C. Zach, and K. Schindler. MetropoGIS: a feature based city modeling system. In *Proc. Photogrammetric Computer Vision, ISPRS Commission III Symposium*, pages 187–192. International Society for Photogrammetry and Remote Sensing (ISPRS), 2002.

[9] J. Behley. Glow (opengl object wrapper). https://github.com/jbehley/glow, 2018. Accessed: 2021-01-05.

[10] A. Belz, A. Muscat, and B. Birmingham. Exploring different preposition sets, models and feature sets in automatic generation of spatial image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 65–69. Association for Computational Linguistics, 2016. doi: 10.18653/v1/W16-3209. URL https://aclanthology.org/W16-3209.

[11] A. Belz, A. Muscat, P. Anguill, M. Sow, G. Vincent, and Y. Zinessabah. SpatialVOC2K: A multilingual dataset of images with annotations and features for spatial relations between objects. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 140–145. Association for Computational Linguistics, 2018. doi: 10.18653/v1/W18-6516. URL https://aclanthology.org/W18-6516.

[12] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

[13] N. Bilton. Why google glass broke. *The New York Times*, 2015. URL https://www.nytimes.com/2015/02/05/style/why-google-glass-broke.html.

[14] B. Birmingham and A. Muscat. Clustering-based model for predicting multi-spatial relations in images. In O. Gusikhin, K. Madani, and J. Zaytoon, editors, *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2019, July 29-31*, volume 2, pages 147–156. SciTePress, 2019. doi: 10.5220/0008123601470156. URL https://doi.org/10.5220/0008123601470156.

[15] B. Birmingham, A. Muscat, and A. Belz. Adding the third dimension to spatial relation detection in 2d images. In E. Krahmer, A. Gatt, and M. Goudbeek, editors,

Proceedings of the 11th International Conference on Natural Language Generation, November 5-8, pages 146–151. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-6517. URL https://doi.org/10.18653/v1/w18-6517.

[16] B. M. Blakeslee. Lambdanet: A novel architecture for unstructured change detection. Master's thesis, Rochester Institute of Technology, 2019. URL https://scholarworks.rit.edu/theses/10260.

[17] G. Bradski. The OpenCV Library, 2000.

[18] R. C. Brost, W. C. McLendon, O. D. Parekh, M. D. Rintoul, D. Woodbridge, and D. R. Strip. Temporal analysis and change detection via geospatial-temporal semantic graphs, 2014. URL https://www.osti.gov/biblio/1141170.

[19] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1209–1218. IEEE, 2018. doi: 10.1109/CVPR.2018.00132.

[20] S. Chacon and B. Straub. *Pro Git*. The Expert's Voice. Apress, 2014. ISBN 9781484200766. URL https://git-scm.com/book/en/v2.

[21] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85. IEEE, 2017. doi: 10.1109/CVPR.2017.16.

[22] J. Chen, Z. Yuan, J. Peng, L. Chen, H. Huang, J. Zhu, T. Lin, and H. Li. Dasnet: Dual attentive fully convolutional siamese networks for change detection of high resolution satellite images. *CoRR*, abs/2003.03608, 2020. URL https://arxiv.org/abs/2003.03608.

[23] J. Chen, Z. Kira, and Y. K. Cho. Lrgnet: Learnable region growing for class-agnostic point cloud segmentation. *IEEE Robotics Autom. Lett.*, 6(2):2799–2806, 2021. doi: 10.1109/LRA.2021.3062607. URL https://doi.org/10.1109/LRA.2021.3062607.

[24] L. Chen, K. Francis, and W. Tang. Semantic augmented reality environment with material-aware physical interactions. In *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2017 Adjunct, Nantes, France, October 9-13, 2017*, pages 135–136. IEEE Computer Society, 2017. doi: 10.1109/ISMAR-Adjunct.2017.49. URL https://doi.org/10.1109/ISMAR-Adjunct.2017.49.

[25] L. Chen, W. Tang, N. W. John, T. R. Wan, and J. J. Zhang. Context-aware mixed reality: A learning-based framework for semantic-level interaction. *Computer Graphics Forum*, 39(1):484–496, 2020. doi: 10.1111/cgf.13887.

[26] M. Chen, A. Monroy-Hernández, and M. Sra. Scenear: Scene-based micro narratives for sharing and remixing in augmented reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 294–303, 2021. doi: 10.1109/ISMAR52148.2021.00045.

[27] S. Chen, Z. Li, and Z. Tang. Relation r-cnn: A graph based relation-aware network for object detection. *IEEE Signal Processing Letters*, 27:1680–1684, 2020. doi: 10.1109/LSP.2020.3025128.

[28] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534. IEEE, 2017. doi: 10.1109/CVPR.2017.691.

[29] Y. Chen, S. Liu, X. Shen, and J. Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12533–12542. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01255.

[30] W. Cheng, Y. Zhang, X. Lei, W. Yang, and G. Xia. Semantic change pattern analysis. *CoRR*, abs/2003.03492, 2020. URL https://arxiv.org/abs/2003.03492.

[31] Y. Cheng, Y. Yan, X. Yi, Y. Shi, and D. Lindlbauer. Semanticadapt: Optimization-based adaptation of mixed reality layouts leveraging virtual-physical semantic connections. In J. Nichols, R. Kumar, and M. Nebeling, editors, *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 10-14, 2021*, pages 282–297. ACM, 2021. doi: 10.1145/3472749.3474750.

[32] P. Chevaillier, T. Trinh, M. Barange, P. D. Loor, F. Devillers, J. Soler, and R. Querrec. Semantic modeling of virtual environments using MASCARET. In *5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2012, Costa Mesa, CA, USA, March 5, 2012*, pages 1–8. IEEE Computer Society, 2012. doi: 10.1109/SEARIS.2012.6231174. URL https://doi.org/10.1109/SEARIS.2012.6231174.

[33] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 1994. URL http://www.blender.org.

[34] A. I. Comport, É. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph.*, 12(4):615–628, 2006. doi: 10.1109/TVCG.2006.78. URL https://doi.org/10.1109/TVCG.2006.78.

[35] J. Cook, S. Gibson, T. Howard, and R. Hubbold. Real-time photo-realistic augmented reality for interior design. In *ACM SIGGRAPH 2003 Sketches & Applications*, SIGGRAPH '03, page 1, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 9781450374668. doi: 10.1145/965400.965427. URL https://doi.org/10.1145/965400.965427.

[36] S. Cunningham and M. J. Bailey. Lessons from scene graphs: using scene graphs to teach hierarchical modeling. *Comput. Graph.*, 25(4):703–711, 2001. doi: 10.1016/S0097-8493(01)00099-1. URL https://doi.org/10.1016/S0097-8493(01)00099-1.

[37] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443. IEEE, 2017. doi: 10.1109/CVPR.2017.261.

[38] R. C. Daudt, B. Le Saux, and A. Boulch. Fully convolutional siamese networks for change detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 4063–4067. IEEE, 2018. doi: 10.1109/ICIP.2018.8451652.

[39] R. C. Daudt, B. L. Saux, A. Boulch, and Y. Gousseau. Multitask learning for large-scale semantic change detection. *Computer Vision and Image Understanding*, 187: 102783, 2019. ISSN 1077-3142. doi: 10.1016/j.cviu.2019.07.003. URL https://doi.org/10.1016/j.cviu.2019.07.003.

[40] P. Dave. Microsoft's answer to google glass: Hololens. *Los Angeles Times*, 2015. URL https://www.latimes.com/business/technology/la-fi-tn-microsoft-hololens-20150121-story.html.

[41] J. A. de Jesus Osuna-Coutiño, J. Martinez-Carranza, M. Arias-Estrada, and W. Mayol-Cuevas. Dominant plane recognition in interior scenes from a single image. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1923–1928, 2016. doi: 10.1109/ICPR.2016.7899917.

[42] K. L. de Jong and A. S. Bosman. Unsupervised change detection in satellite images using convolutional neural networks. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pages 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8851762. URL https://doi.org/10.1109/IJCNN.2019.8851762.

[43] J. Deng and K. Czarnecki. Mlod: A multi-view 3d object detection based on robust feature fusion method. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 279–284, 2019. doi: 10.1109/ITSC.2019.8917126.

[44] E. Denis and C. Baillard. Refining existing 3D building models with terrestrial laser points acquired from a mobile mapping vehicule. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII: 195–200, June 2010.

[45] E. Derner, C. Gómez, A. C. Hernández, R. Barber, and R. Babuska. Towards life-long autonomy of mobile robots through feature-based change detection. In L. Preucil, S. Behnke, and M. Kulich, editors, *2019 European Conference on Mobile Robots, ECMR 2019, Prague, Czech Republic, September 4-6, 2019*, pages 1–6. IEEE, 2019. doi: 10.1109/ECMR.2019.8870940. URL https://doi.org/10.1109/ECMR.2019.8870940.

[46] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, and C. Rupprecht. Semantic image manipulation using scene graphs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 5212–5221. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00526.

[47] J. Döllner and H. Buchholz. Continuous level-of-detail modeling of buildings in 3D city models. In *13th ACM International Workshop on Geographic Information Systems*, pages 173–181. Association for Computing Machinery, Jan. 2005. doi: 10.1145/1097064.1097089.

[48] M. Dubská, I. Szentandrási, M. Zachariás, and A. Herout. Poor man's simulcam: Real-time and effortless matchmoving. In *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013, Adelaide, Australia, October 1-4, 2013*, pages 249–250. IEEE Computer Society, 2013. doi: 10.1109/ISMAR.2013.6671789. URL https://doi.org/10.1109/ISMAR.2013.6671789.

[49] A. Elu, G. Azkune, O. L. de Lacalle, I. Arganda-Carreras, A. Soroa, and E. Agirre. Inferring spatial relations from textual descriptions of images. *Pattern Recognit.*,

113:107847, 2021. doi: 10.1016/j.patcog.2021.107847. URL https://doi.org/10.1016/j.patcog.2021.107847.

[50] Euronews and AFP. What's remote work like in the metaverse? these companies are building the tools to make it possible. *euronews.next*, 2022. URL https://www.euronews.com/next/2022/02/16/what-s-remote-work-like-in-the-metaverse-these-companies-are-building-the-tools-to-mal

[51] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 5237–5244. IEEE, 2017. doi: 10.1109/ICRA.2017.7989614. URL https://doi.org/10.1109/ICRA.2017.7989614.

[52] C. Feng, Y. Taguchi, and V. R. Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6225, 2014. doi: 10.1109/ICRA.2014.6907776.

[53] S. Fontana, D. Cattaneo, A. L. Ballardini, M. Vaghi, and D. G. Sorrenti. A benchmark for point clouds registration algorithms. *Robotics and Autonomous Systems*, 140:103734, 2021. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2021.103734. URL https://www.sciencedirect.com/science/article/pii/S0921889021000191.

[54] W. Förstner. 3d-city models: Automatic and semiautomatic acquisition methods. *D. Fritsch & R. Spiller, eds., Photogrammetric Week '99*, pages 291–303, 1999.

[55] R. Freeman, A. Steed, and B. Zhou. Rapid scene modelling, registration and specification for mixed reality systems. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '05, pages 147–150, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930981. doi: 10.1145/1101616.1101647. URL https://doi.org/10.1145/1101616.1101647.

[56] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011. IEEE, 2018. doi: 10.1109/CVPR.2018.00214.

[57] A. Fuhrmann, G. Hesina, F. Faure, and M. Gervautz. Occlusion in collaborative augmented environments. In M. Gervautz, D. Schmalstieg, and A. Hildebrand,

editors, *Virtual Environments '99*, pages 179–190. Springer Vienna, 1999. ISBN 978-3-7091-6805-9.

[58] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. I. Nieto. Incremental object database: Building 3d models from multiple partial observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 6835–6842. IEEE, 2018. doi: 10.1109/IROS.2018.8594391. URL https://doi.org/10.1109/IROS.2018.8594391.

[59] T. Fuse and N. Yokozawa. Development of a change detection method with low-performance point cloud data for updating three-dimensional road maps. *ISPRS International Journal of Geo-Information*, 6(12):398, 2017. doi: 10.3390/ijgi6120398. URL https://doi.org/10.3390/ijgi6120398.

[60] E. Ganea and M. Brezovan. Graph object oriented database for semantic image retrieval. In B. Catania, M. Ivanović, and B. Thalheim, editors, *Advances in Databases and Information Systems*, pages 563–566. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15576-5.

[61] A. Gawel, C. D. Don, R. Siegwart, J. Nieto, and C. Cadena. X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3): 1687–1694, 2018. doi: 10.1109/LRA.2018.2801879.

[62] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. doi: 10.1109/CVPR.2012.6248074.

[63] T. Gevers, A. Gijsenij, J. van de Weijer, and J.-M. Geusebroek. *Color in Computer Vision: Fundamentals and Applications*. Wiley Publishing, first edition, 2012. ISBN 0470890843. doi: 10.5555/2432392.

[64] S. Gibson and A. Chalmers. Photorealistic augmented reality. In R. J. Hubbold, C. Ureña, and À. Vinacua, editors, *24th Annual Conference of the European Association for Computer Graphics, Eurographics 2003 - Tutorials, Granada, Spain, September 1-5, 2003*. Eurographics Association, 2003. doi: 10.2312/egt.20031098. URL https://doi.org/10.2312/egt.20031098.

[65] S. Gibson and T. Howard. Interactive reconstruction of virtual environments from photographs, with application to scene-of-crime analysis. In H. J. Kim and K. Yun,

editors, *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2000, Seoul, South Korea, October 22-25, 2000*, pages 41–48. ACM, 2000. doi: 10.1145/502390.502399. URL https://doi.org/10.1145/502390.502399.

[66] J. Gimeno, P. Morillo, S. Casas, and M. Fernández. An augmented reality (ar) cad system at construction sites. *Augmented Reality: Some Emerging Application Areas*, pages 15–32, 2011.

[67] A. Gressin, N. Vincent, C. Mallet, and N. Paparoditis. Semantic approach in image change detection. In J. Blanc-Talon, A. J. Kasinski, W. Philips, D. C. Popescu, and P. Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems - 15th International Conference, ACIVS 2013, Poznań, Poland, October 28-31, 2013. Proceedings*, volume 8192 of *Lecture Notes in Computer Science*, pages 450–459. Springer, 2013. doi: 10.1007/978-3-319-02895-8_40. URL https://doi.org/10.1007/978-3-319-02895-8_40.

[68] L. Gruber, T. Langlotz, P. Sen, T. Hoherer, and D. Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality, VR 2014, Minneapolis, MN, USA, March 29 - April 2, 2014*, pages 15–20. IEEE Computer Society, 2014. doi: 10.1109/VR.2014.6802044. URL https://doi.org/10.1109/VR.2014.6802044.

[69] F. Gu, S. Valaee, K. Khoshelham, J. Shang, and R. Zhang. Landmark graph-based indoor localization. *IEEE Internet of Things Journal*, 7(9):8343–8355, 2020. doi: 10.1109/JIOT.2020.2989501.

[70] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[71] E. Guo, X. Fu, J. Zhu, M. Deng, Y. Liu, Q. Zhu, and H. Li. Learning to measure change: Fully convolutional siamese metric networks for scene change detection. *CoRR*, abs/1810.09111, 2018. URL http://arxiv.org/abs/1810.09111.

[72] X. Guo, J. Hu, J. Chen, F. Deng, and T. L. Lam. Semantic histogram based graph matching for real-time multi-robot global localization in large scale environment. *IEEE Robotics and Automation Letters*, 6(4):8349–8356, 2021. doi: 10.1109/LRA.2021.3058935.

[73] A. Gupta, P. Dollár, and R. B. Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *IEEE Conference on Computer Vision and Pattern*

Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 5356–5364. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00550. URL https://doi.org/10.1109/CVPR.2019.00550.

[74] C. Gutierrez, C. Hurtado, and A. Vaisman. Temporal rdf. In *Proceedings of the Second European Conference on The Semantic Web: Research and Applications*, ESWC'05, pages 93–107, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3540261249. doi: 10.1007/11431053_7. URL https://doi.org/10.1007/11431053_7.

[75] J. K. Haas. A history of the unity game engine. Technical report, Worcester Polytechnic Institute, 100 Institute Road, Worcester MA 01609-2280 USA, March 2014.

[76] O. Haines and A. Calway. Recognising planes in a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1849–1861, 2015. doi: 10.1109/TPAMI.2014.2382097. URL https://doi.org/10.1109/TPAMI.2014.2382097.

[77] M. Haller. Photorealism or/and non-photorealism in augmented reality. In J. R. Brown and Y. Cai, editors, *Proceedings VRCAI 2004, ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry, Nanyang Technological University, Singapore, June 16-18, 2004*, VRCAI '04, pages 189–196, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138849. doi: 10.1145/1044588.1044627. URL https://doi.org/10.1145/1044588.1044627.

[78] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2003, Osaka, Japan, October 1-3, 2003*, pages 56–65. ACM, 2003. doi: 10.1145/1008653.1008665. URL https://doi.org/10.1145/1008653.1008665.

[79] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu. Advanced deep-learning techniques for salient and category-specific object detection: A survey. *IEEE Signal Process. Mag.*, 35(1):84–100, 2018. doi: 10.1109/MSP.2017.2749125. URL https://doi.org/10.1109/MSP.2017.2749125.

[80] A. Haugstvedt and J. Krogstie. Mobile augmented reality for cultural heritage: A technology acceptance study. In *11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, GA, USA, November 5-8, 2012*,

pages 247–255. IEEE Computer Society, 2012. doi: 10.1109/ISMAR.2012.6402563. URL https://doi.org/10.1109/ISMAR.2012.6402563.

[81] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016. doi: 10.1109/CVPR.2016.90.

[82] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[83] A. K. Hebborn. *Towards Believable Augmented Reality: Combining the Real and Virtual Worlds*. doctoralthesis, Universität Koblenz-Landau, Universitätsbibliothek, 2018.

[84] J. Hu, S. You, and U. Neumann. Approaches to large-scale urban modeling. *IEEE Computer Graphics and Applications*, 23(6):62–69, 2003.

[85] M.-K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theory*, 8:179–187, 1962.

[86] L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1): 193–218, dec 1985.

[87] V. Indelman, E. Nelson, N. Michael, and F. Dellaert. Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 593–600. IEEE, 2014. doi: 10.1109/ICRA.2014.6906915. URL https://doi.org/10.1109/ICRA.2014.6906915.

[88] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, Oct. 2011. ISBN 978-1-4503-0716-1. URL https://www.microsoft.com/en-us/research/publication/kinectfusion-real-time-3d-reconstruction-and-interaction-using-a-moving-depth-camera/.

[89] J. Jachnik, R. A. Newcombe, and A. J. Davison. Real-time surface light-field capture for augmentation of planar specular surfaces. In *11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, GA, USA,*

November 5-8, 2012, pages 91–97. IEEE Computer Society, 2012. doi: 10.1109/ISMAR.2012.6402544. URL https://doi.org/10.1109/ISMAR.2012.6402544.

[90] M. Jenkinson. Measuring transformation error by rms deviation. *Studholme, C., Hill, DLG, Hawkes, DJ*, 1999.

[91] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678. IEEE, 2015. doi: 10.1109/CVPR.2015.7298990.

[92] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1219–1228. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00133. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00133.

[93] P. Kán and H. Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, GA, USA, November 5-8, 2012*, pages 99–108. IEEE Computer Society, 2012. doi: 10.1109/ISMAR.2012.6402546. URL https://doi.org/10.1109/ISMAR.2012.6402546.

[94] M. Kanbara and N. Yokoya. Geometric and photometric registration for real-time augmented reality. In *2002 IEEE / ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002), 30 September - 1 October 2002, Darmstadt, Germany*, pages 279–280. IEEE Computer Society, 2002. doi: 10.1109/ISMAR.2002.1115112. URL https://doi.org/10.1109/ISMAR.2002.1115112.

[95] H. Kataoka, S. Shirakabe, Y. Miyashita, A. Nakamura, K. Iwata, and Y. Satoh. Semantic change detection with hypermaps. *CoRR*, abs/1604.07513, 2016. URL http://arxiv.org/abs/1604.07513.

[96] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top ar environment. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 111–119. IEEE, 2000. doi: 10.1109/ISAR.2000.880934.

[97] U. Katsura, K. Matsumoto, A. Kawamura, T. Ishigami, T. Okada, and R. Kurazume. Spatial change detection using voxel classification by normal distributions transform. In *International Conference on Robotics and Automation, ICRA*

2019, Montreal, QC, Canada, May 20-24, 2019, pages 2953–2959. IEEE, 2019. doi: 10.1109/ICRA.2019.8794173. URL https://doi.org/10.1109/ICRA.2019.8794173.

[98] Khronos. gltf. https://www.khronos.org/gltf/, 2015.

[99] S. Kim, K. Joo, and C. Youn. Graph neural network based scene change detection using scene graph embedding with hybrid classification loss. In *International Conference on Information and Communication Technology Convergence, ICTC 2021, Jeju Island, Korea, Republic of, October 20-22, 2021*, pages 190–195. IEEE, 2021. doi: 10.1109/ICTC52510.2021.9621009. URL https://doi.org/10.1109/ICTC52510.2021.9621009.

[100] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):4921–4933, 2020. ISSN 2168-2275. doi: 10.1109/TCYB.2019.2931042.

[101] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[102] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9396–9405. IEEE, 2019. doi: 10.1109/CVPR.2019.00963.

[103] T. Kiyokawa, K. Tomochika, J. Takamatsu, and T. Ogasawara. Fully automated annotation with noise-masked visual markers for deep-learning-based object detection. *IEEE Robotics and Automation Letters*, 4(2):1972–1977, 2019. doi: 10.1109/LRA.2019.2899153.

[104] M. Knecht, A. Dünser, C. Traxler, M. Wimmer, and R. Grasset. A framework for perceptual studies in photorealistic augmented reality. In F. Steinicke and P. Willemsen, editors, *Proceedings of IEEE Workshop on Perceptual Illusions in Virtual Environments (PIVE)*, page 6. University of Canterbury. Human Interface Technology Laboratory, 2011. URL http://pive.uni-muenster.de/index.php?f=program.

[105] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017. ISSN 0920-5691. doi: 10.1007/s11263-016-0981-7. URL https://doi.org/10.1007/s11263-016-0981-7.

[106] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012*, volume 25, pages 1106–1114. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[107] J. Kronander, F. Banterle, A. Gardner, E. Miandji, and J. Unger. Photorealistic rendering of mixed reality scenes. *Comput. Graph. Forum*, 34(2):643–665, 2015. doi: 10.1111/cgf.12591. URL https://doi.org/10.1111/cgf.12591.

[108] M. Krueger, P. Delmas, and G. L. Gimel'farb. Active contour based segmentation of 3d surfaces. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II*, volume 5303 of *Lecture Notes in Computer Science*, pages 350–363. Springer, 2008. doi: 10.1007/978-3-540-88688-4_26. URL https://doi.org/10.1007/978-3-540-88688-4_26.

[109] T. Ku, S. Galanakis, B. Boom, R. C. Veltkamp, D. Bangera, S. Gangisetty, N. Stagakis, G. Arvanitis, and K. Moustakas. SHREC 2021: 3d point cloud change detection for street scenes. *Comput. Graph.*, 99:192–200, 2021. doi: 10.1016/j.cag.2021.07.004. URL https://doi.org/10.1016/j.cag.2021.07.004.

[110] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:199–218, 2000. ISSN 1573-1405. doi: 10.1023/A:1008191222954.

[111] Y. Lang, W. Liang, and L.-F. Yu. Virtual agent positioning driven by scene semantics in mixed reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 767–775. IEEE, 2019. doi: 10.1109/VR.2019.8798018.

[112] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua. Fully automated and stable registration for augmented reality applications. In *2003 IEEE / ACM International*

Symposium on Mixed and Augmented Reality (ISMAR 2003), 7-10 October 2003, Tokyo, Japan, pages 93–102. IEEE Computer Society, 2003. doi: 10.1109/ISMAR.2003.1240692. URL https://doi.org/10.1109/ISMAR.2003.1240692.

[113] C. Li, W. Li, H. Huang, and L.-F. Yu. Interactive augmented reality storytelling guided by scene semantics. *ACM Transactions on Graphics*, 41(4), 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530061.

[114] W. Li, J. Wang, M. Liu, and S. Zhao. Real-time occlusion handling for augmented reality assistance assembly systems with monocular images. *Journal of Manufacturing Systems*, 62:561–574, 2022. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2022.01.012. URL https://www.sciencedirect.com/science/article/pii/S0278612522000139.

[115] X. Li, J. K. Pontes, and S. Lucey. Pointnetlk revisited. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12758–12767. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01257.

[116] Y. Li, W. Ouyang, B. Zhou, J. Shi, C. Zhang, and X. Wang. Factorizable net: An efficient subgraph-based framework for scene graph generation. In *The European Conference on Computer Vision (ECCV)*. Springer, 2018.

[117] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3835–3845. PMLR, 2019. URL http://proceedings.mlr.press/v97/li19d.html.

[118] Z. Li, T. Yu, S. Sang, S. Wang, M. Song, Y. Liu, Y. Yeh, R. Zhu, N. B. Gundavarapu, J. Shi, S. Bi, H. Yu, Z. Xu, K. Sunkavalli, M. Hasan, R. Ramamoorthi, and M. Chandraker. Openrooms: An open framework for photorealistic indoor scene datasets. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 7190–7199. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.00711. URL https://doi.org/10.1109/CVPR46437.2021.00711.

[119] W. Liang, X. Yu, R. Alghofaili, Y. Lang, and L. Yu. Scene-aware behavior synthesis for virtual pets in mixed reality. In Y. Kitamura, A. Quigley, K. Isbister, T. Igarashi, P. Bjørn, and S. M. Drucker, editors, *CHI '21: CHI Conference on*

Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021, page 472. ACM, 2021. doi: 10.1145/3411764.3445532.

[120] Z. Liao, Q. Huang, Y. Liang, M. Fu, Y. Cai, and Q. Li. Scene graph with 3d information for change captioning. In H. T. Shen, Y. Zhuang, J. R. Smith, Y. Yang, P. César, F. Metze, and B. Prabhakaran, editors, *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, pages 5074–5082. ACM, 2021. doi: 10.1145/3474085.3475712. URL https://doi.org/10.1145/3474085.3475712.

[121] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48. URL https://doi.org/10.1007/978-3-319-10602-1_48.

[122] Z. Lin, J. Yu, L. Zhou, X. Zhang, J. Wang, and Y. Wang. Point cloud change detection with stereo V-SLAM: dataset, metrics and baseline. *IEEE Robotics Autom. Lett.*, 7(4):12443–12450, 2022. doi: 10.1109/LRA.2022.3219018. URL https://doi.org/10.1109/LRA.2022.3219018.

[123] B. Liu, L. Ding, and L. Meng. Spatial knowledge acquisition with virtual semantic landmarks in mixed reality-based indoor navigation. *Cartography and Geographic Information Science*, 48(4):305–319, 2021. doi: 10.1080/15230406.2021.1908171.

[124] Y. Liu, Y. Petillot, D. Lane, and S. Wang. Global localization with object-level semantics and topology. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4909–4915. IEEE, 2019. doi: 10.1109/ICRA.2019.8794475.

[125] S. Looper, J. R. Puigvert, R. Siegwart, C. Cadena, and L. M. Schmid. 3d VSG: long-term semantic scene change prediction through 3d variable scene graphs. *CoRR*, abs/2209.07896, 2022. doi: 10.48550/arXiv.2209.07896. URL https://doi.org/10.48550/arXiv.2209.07896.

[126] C. Lu, R. Krishna, M. S. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, October 11-14, 2016, Proceedings, Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 852–869. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46448-0_51. URL https://doi.org/10.1007/978-3-319-46448-0_51.

[127] S. G. Lukosch, M. Billinghurst, L. Alem, and K. Kiyokawa. Collaboration in augmented reality. *Comput. Support. Cooperative Work.*, 24(6):515–525, 2015. doi: 10.1007/s10606-015-9239-0. URL https://doi.org/10.1007/s10606-015-9239-0.

[128] A. Masrur, J. Zhao, J. O. Wallgrün, P. LaFemina, and A. Klippel. Immersive applications for informal and interactive learning for earth science. In *Proceedings of the Workshop on Immersive Analytics, Exploring Future Interaction and Visualization Technologies for Data Analytics*, pages 1–5, 2017.

[129] C. Mazzotti, N. Sancisi, and V. Parenti-Castelli. A measure of the distance between two rigid-body poses based on the use of platonic solids. In V. Parenti-Castelli and W. Schiehlen, editors, *ROMANSY 21 - Robot Design, Dynamics and Control*, pages 81–89. Springer International Publishing, 2016.

[130] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. ISSN 0079-7421. doi: 10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[131] T. Murase, K. Tanaka, and A. Takayama. Change detection with global viewpoint localization. In *4th IAPR Asian Conference on Pattern Recognition, ACPR 2017, Nanjing, China, November 26-29, 2017*, pages 31–36. IEEE Computer Society, 2017. doi: 10.1109/ACPR.2017.21. URL https://doi.org/10.1109/ACPR.2017.21.

[132] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010. doi: 10.1109/TPAMI.2010.46.

[133] Y. Nakashima, T. Sato, Y. Uno, N. Yokoya, and N. Kawai. Augmented reality image generation with virtualized real objects using view-dependent texture and geometry. In *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013, Adelaide, Australia, October 1-4, 2013*, pages 1–6. IEEE Computer Society, 2013. doi: 10.1109/ISMAR.2013.6671827. URL https://doi.org/10.1109/ISMAR.2013.6671827.

[134] B. Okumura, M. Kanbara, and N. Yokoya. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *Fifth IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2006, October 22-25, 2006, Santa Barbara, CA, USA*, pages 219–225. IEEE Computer Society, 2006. doi: 10.1109/ISMAR.2006.297817. URL https://doi.org/10.1109/ISMAR.2006.297817.

[135] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011. doi: 10.1109/ICRA.2011.5979561.

[136] E. Palazzolo and C. Stachniss. Fast image-based geometric change detection given a 3d model. In *Proceedings of the IEEE Int. Conf. on Robotics & Automation (ICRA)*. IEEE, 2018. URL https://github.com/PRBonn/fast_change_detection.

[137] G. Palma, P. Cignoni, T. Boubekeur, and R. Scopigno. Detection of geometric temporal changes in point clouds. *Computer Graphics Forum*, 35(6):33–45, 2016. doi: 10.1111/cgf.12730. URL https://doi.org/10.1111/cgf.12730.

[138] D. H. Park, T. Darrell, and A. Rohrbach. Robust change captioning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4623–4632. IEEE, 2019. doi: 10.1109/ICCV.2019.00472. URL https://doi.org/10.1109/ICCV.2019.00472.

[139] J. Park, J. Jang, S. Yoo, S. Lee, U. Kim, and J. Kim. Changesim: Towards end-to-end online scene change detection in industrial indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 8578–8585. IEEE, 2021. doi: 10.1109/IROS51168.2021.9636350. URL https://doi.org/10.1109/IROS51168.2021.9636350.

[140] L. Pereira, J. Souza, and R. Silva. Blending real and virtual objects in augmented reality environments. *Journal of Mobile Multimedia*, 2022. doi: 10.13052/jmm1550-4646.1868.

[141] S. A. Pessoa, G. de S. Moura, J. P. S. do Monte Lima, V. Teichrieb, and J. Kelner. Photorealistic rendering for augmented reality: A global illumination and BRDF solution. In B. Lok, G. Klinker, and R. Nakatsu, editors, *IEEE Virtual Reality Conference, VR 2010, Waltham, Massachusetts, USA, March 20-24, 2010*, pages 3–10. IEEE Computer Society, 2010. doi: 10.1109/VR.2010.5444836. URL https://doi.org/10.1109/VR.2010.5444836.

[142] E. Petre. The role of ar and vr in corporate communications. *PwC Australia*, 2017. URL https://www.pwc.com.au/digitalpulse/augmented-virtual-reality-corporate-communications.html.

[143] N. Pion, M. Humenberger, G. Csurka, Y. Cabon, and T. Sattler. Benchmarking image retrieval for visual localization. In *2020 International Conference on 3D Vision (3DV)*, pages 483–494. IEEE, 2020. doi: 10.1109/3DV50981.2020.00058.

[144] A. D. Pon, J. Ku, C. Li, and S. L. Waslander. Object-centric stereo matching for 3d object detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8383–8389, 2020. doi: 10.1109/ICRA40945.2020.9196660.

[145] G. Postica, A. Romanoni, and M. Matteucci. Robust moving objects detection in lidar data exploiting visual cues. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 1093–1098. IEEE, 2016. doi: 10.1109/IROS.2016.7759185. URL https://doi.org/10.1109/IROS.2016.7759185.

[146] S. Prakash, A. Bahremand, L. D. Nguyen, and R. LiKamWa. Gleam: An illumination estimation framework for real-time photorealistic augmented reality on mobile devices. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, pages 142–154, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366618. doi: 10.1145/3307334.3326098. URL https://doi.org/10.1145/3307334.3326098.

[147] M. J. Pratt, P. A. Skemer, and R. E. Arvidson. Developing an augmented reality environment for earth science education. In *AGU Fall Meeting Abstracts*, volume 2017, pages ED11C–0133, 2017.

[148] J. Pustejovsky and Z. Yocum. Image annotation with iso-space: Distinguishing content from structure. In N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 426–431. European Language Resources Association (ELRA), 2014. URL http://www.lrec-conf.org/proceedings/lrec2014/summaries/1026.html.

[149] J. Qian, V. Chatrath, J. Yang, J. Servos, A. P. Schoellig, and S. L. Waslander. POCD: probabilistic object-level change detection and volumetric mapping in semi-static scenes. *CoRR*, abs/2205.01202, 2022. doi: 10.48550/arXiv.2205.01202. URL https://doi.org/10.48550/arXiv.2205.01202.

[150] C. Qin, Y. Zhang, Y. Liu, and G. Lv. Semantic loop closure detection based on graph matching in multi-objects scenes. *Journal of Visual Communication and Image Representation*, 76:103072, 2021. ISSN 1047-3203. doi: https:

//doi.org/10.1016/j.jvcir.2021.103072. URL https://www.sciencedirect.com/science/article/pii/S1047320321000389.

[151] R. Qin and A. Gruen. 3d change detection at street level using mobile laser scanning point clouds and terrestrial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:23–35, Apr. 2014. doi: 10.1016/j.isprsjprs.2014.01.006.

[152] G. Qiu and J. Sun. Plane detection in stereo images based on homography. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, volume 5, pages 141–144, 2010. doi: 10.1109/ICACTE.2010.5579124.

[153] S. Ramnath, A. Saha, S. Chakrabarti, and M. M. Khapra. Scene graph based image retrieval - a case study on the clevr dataset. *ArXiv*, abs/1911.00850, 2019.

[154] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[155] W. Ribarsky, T. Wasilewski, and N. Faust. From urban terrain models to visible cities. *IEEE Computer Graphics and Applications*, 22:10–15, Aug. 2002. doi: 10.1109/MCG.2002.1016692.

[156] B. Robinson. Remote work is here to stay and will increase into 2023, experts say. *Forbes*, 2022. URL https://www.forbes.com/sites/bryanrobinson/2022/02/01/remote-work-is-here-to-stay-and-will-increase-into-2023-experts-say/.

[157] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. Interactive near-field illumination for photorealistic augmented reality on mobile devices. In *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2014, Munich, Germany, September 10-12, 2014*, pages 29–38. IEEE Computer Society, 2014. doi: 10.1109/ISMAR.2014.6948406. URL https://doi.org/10.1109/ISMAR.2014.6948406.

[158] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243. IEEE, 2016. doi: 10.1109/CVPR.2016.352.

[159] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May*

31 - August 31, 2020, pages 1689–1696. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196885. URL https://doi.org/10.1109/ICRA40945.2020.9196885.

[160] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Proceedings of Robotics: Science and Systems*. Robotics: Science and Systems, 2020. doi: 10.15607/RSS.2020.XVI.079.

[161] O. Roupin, M. Fradet, C. Baillard, and G. Moreau. Detection of removed objects in 3D meshes using up-to-date images for mixed-reality applications. *Electronics*, 10(4), 2021. ISSN 2079-9292. doi: 10.3390/electronics10040377. URL https://www.mdpi.com/2079-9292/10/4/377.

[162] B. Roy. Transitivité et connexité. *Comptes Rendus Hebdomadaires Des Seances De L Academie Des Sciences*, 249(2):216–218, 1959.

[163] J. Royan and J. Lacoche. Towards an interoperable world representation for xr. https://www.brighttalk.com/webcast/12761/562141, 2022.

[164] F. Sadeghi, H. Arefi, A. Fallah, and M. Hahn. 3D building façade reconstruction using handheld laser scanning data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-1/W5:625–630, 2015. doi: 10.5194/isprsarchives-XL-1-W5-625-2015. URL https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1-W5/625/2015/.

[165] K. Sakurada, M. Shibuya, and W. Wang. Weakly supervised silhouette-based semantic scene change detection. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 6861–6867. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196985. URL https://doi.org/10.1109/ICRA40945.2020.9196985.

[166] H. Samih, S. Rady, M. A. Ismail, and T. F. Gharib. Semantic graph representation and evaluation for generated image annotations. In A.-E. Hassanien, K.-C. Chang, and T. Mincong, editors, *Advanced Machine Learning Technologies and Applications*, pages 369–384. Springer International Publishing, 2021. ISBN 978-3-030-69717-4.

[167] J. Schauer and A. Nüchter. Removing non-static objects from 3d laser scan data. *Isprs Journal of Photogrammetry and Remote Sensing*, 143:15–38, 2018. doi: 10.1016/j.isprsjprs.2018.05.019.

[168] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2538–2547. IEEE, 2017. doi: 10.1109/CVPR.2017.272.

[169] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H. Gross. Efficient RGB-D semantic segmentation for indoor scene analysis. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 13525–13531. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561675. URL https://doi.org/10.1109/ICRA48506.2021.9561675.

[170] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, 2012. doi: 10.1145/2366145.2366155. URL https://doi.org/10.1145/2366145.2366155.

[171] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011. ISSN 1532-4435.

[172] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 245—254, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0897911660. doi: 10.1145/325334.325242. URL https://doi.org/10.1145/325334.325242.

[173] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 746–760. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33715-4.

[174] G. Simon. Automatic online walls detection for immediate use in ar tasks. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 39–42. IEEE, 2006.

[175] M. Sonogashira, M. Iiyama, and Y. Kawanishi. Towards open-set scene graph generation with unknown objects. *IEEE Access*, 10:11574–11583, 2022. doi: 10.1109/ACCESS.2022.3145465. URL https://doi.org/10.1109/ACCESS.2022.3145465.

[176] E. V. Sousa, D. Lucio, L. A. F. Fernandes, and L. Velho. Hough transform for real-time plane detection in depth images. *Pattern Recognition Letters*, 103:8–15,

2018. ISSN 0167-8655. doi: 10.1016/j.patrec.2017.12.027. URL https://doi.org/10.1016/j.patrec.2017.12.027.

[177] M. Speicher, B. D. Hall, and M. Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–15, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359702. doi: 10.1145/3290605.3300767. URL https://doi.org/10.1145/3290605.3300767.

[178] D. Stoyanov, M. A. ElHelw, B. P. L. Lo, A. J. Chung, F. Bello, and G. Yang. Current issues of photorealistic rendering for virtual and augmented reality in minimally invasive surgery. In E. Banissi, K. Börner, C. Chen, G. Clapworthy, C. Maple, A. Lobben, C. J. Moore, J. C. Roberts, A. Ursyn, and J. J. Zhang, editors, *Seventh International Conference on Information Visualization, IV 2003, 16-18 July 2003, London, UK*, pages 350–359. IEEE Computer Society, 2003. doi: 10.1109/IV.2003.1218010. URL https://doi.org/10.1109/IV.2003.1218010.

[179] P. A. Studios. Introduction to usd. https://graphics.pixar.com/usd/release/intro.html, 2016.

[180] E. Stumm, C. Mei, S. Lacroix, J. Nieto, M. Hutter, and R. Siegwart. Robust visual place recognition with graph kernels. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4535–4544. IEEE, 2016. doi: 10.1109/CVPR.2016.491.

[181] S. Su, L. Gao, J. Zhu, J. Shao, and J. Song. *Fully Functional Image Manipulation Using Scene Graphs in A Bounding-Box Free Way*, pages 1784–1792. Association for Computing Machinery, 2021. ISBN 9781450386517. URL https://doi.org/10.1145/3474085.3475326.

[182] N. Sünderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In L. E. Kavraki, D. Hsu, and J. Buchli, editors, *Robotics: Science and Systems XI, July 13-17, 2015*. Robotics: Science and Systems, 2015. doi: 10.15607/RSS.2015.XI.022. URL http://www.roboticsproceedings.org/rss11/p22.html.

[183] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1): 32–46, 1985. doi: 10.1016/0734-189X(85)90016-7. URL https://doi.org/10.1016/0734-189X(85)90016-7.

[184] R. Szeliski and R. Szeliski. Structure from motion. *Computer Vision: Algorithms and Applications*, pages 303–334, 2011.

[185] T. Tahara, T. Seno, G. Narita, and T. Ishikawa. Retargetable ar: Context-aware augmented reality in indoor scenes based on 3d scene graph. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 249–255. IEEE Computer Society, 2020. doi: 10.1109/ISMAR-Adjunct51615.2020.00072. URL https://doi.ieeecomputersociety.org/10.1109/ISMAR-Adjunct51615.2020.00072.

[186] M. Tajrobehkar, K. Tang, H. Zhang, and J. H. Lim. Align r-cnn: A pairwise head network for visual relationship detection. *IEEE Transactions on Multimedia*, 2021. doi: 10.1109/TMM.2021.3062543.

[187] A. Taneja, L. Ballan, and M. Pollefeys. Modeling dynamic scenes recorded with freely moving cameras. In R. Kimmel, R. Klette, and A. Sugimoto, editors, *Computer Vision - ACCV 2010 - 10th Asian Conference on Computer Vision, Queenstown, New Zealand, November 8-12, 2010, Revised Selected Papers, Part III*, volume 6494 of *Lecture Notes in Computer Science*, pages 613–626. Springer, 2010. doi: 10.1007/978-3-642-19318-7_48. URL https://doi.org/10.1007/978-3-642-19318-7_48.

[188] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. V. Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2336–2343. IEEE Computer Society, 2011. doi: 10.1109/ICCV.2011.6126515. URL https://doi.org/10.1109/ICCV.2011.6126515.

[189] A. Taneja, L. Ballan, and M. Pollefeys. Geometric change detection in urban environments using images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(11): 2193–2206, 2015. doi: 10.1109/TPAMI.2015.2404834. URL https://doi.org/10.1109/TPAMI.2015.2404834.

[190] C. Tang, Y. Ling, X. Yang, W. Jin, and C. Zheng. Multi-view object detection based on deep learning. *Applied Sciences*, 8(9), 2018. ISSN 2076-3417. doi: 10.3390/app8091423. URL https://www.mdpi.com/2076-3417/8/9/1423.

[191] L. A. Thomaz, E. Jardim, A. F. da Silva, E. A. B. da Silva, S. L. Netto, and H. Krim. Anomaly detection in moving-camera video sequences using principal subspace analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(3):1003–1015, 2018. doi: 10.1109/TCSI.2017.2758379.

[192] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *Int. J. Robotics Res.*, 32(14):1662–1678, 2013. doi: 10.1177/0278364913502830. URL https://doi.org/10.1177/0278364913502830.

[193] Turbosquid. Turbosquid. https://web.archive.org/web/20201214222713/https://www.turbosquid.com/, 2000. Accessed: 2020-12-14.

[194] A. Ückermann, C. Elbrechter, R. Haschke, and H. Ritter. 3d scene segmentation for autonomous robot grasping. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1734–1740, 2012. doi: 10.1109/IROS.2012.6385692.

[195] A. Ulusoy and J. Mundy. Image-based 4-d reconstruction using 3-d change detection. In *Lecture Notes in Computer Science (LNCS)*, volume 8691. Springer, 2014. doi: 10.1007/978-3-319-10578-9_3.

[196] A. O. Ulusoy, O. Biris, and J. L. Mundy. Dynamic probabilistic volumetric models. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 505–512. IEEE Computer Society, 2013. doi: 10.1109/ICCV.2013.68. URL https://doi.org/10.1109/ICCV.2013.68.

[197] K. Vaiapury, B. Purushothaman, A. Pal, and S. Agarwal. Can we speed up 3d scanning? A cognitive and geometric analysis. In *2017 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2017, Venice, Italy, October 22-29, 2017*, pages 2690–2696. IEEE Computer Society, 2017. doi: 10.1109/ICCVW.2017.317. URL https://doi.org/10.1109/ICCVW.2017.317.

[198] A. Varghese, J. Gubbi, A. Ramaswamy, and P. Balamuralidhar. Changenet: A deep learning architecture for visual change detection. In L. Leal-Taixé and S. Roth, editors, *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part II*, volume 11130 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2018. doi: 10.1007/978-3-030-11012-3_10. URL https://doi.org/10.1007/978-3-030-11012-3_10.

[199] C. Vazquez, N. Tan, and S. Sadalgi. Fused photo: Augmented reality staging for photorealistic visualization in online home retail. In Y. Kitamura, A. Quigley, K. Isbister, and T. Igarashi, editors, *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama Japan, May 8-13, 2021, Extended Abstracts*, pages 296–296. ACM, 2021. doi: 10.1145/3411763.3451723. URL https://doi.org/10.1145/3411763.3451723.

[200] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Trans. Vis. Comput. Graph.*, 16(3):355–368, 2010. doi: 10.1109/TVCG.2009.99. URL https://doi.org/10.1109/TVCG.2009.99.

[201] J. Wald, H. Dhamo, N. Navab, and F. Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3960–3969. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00402.

[202] C. Wang, H. Chen, and L. Fu. Vpfnet: Voxel-pixel fusion network for multi-class 3d object detection. *CoRR*, abs/2111.00966, 2021. URL https://arxiv.org/abs/2111.00966.

[203] F. Wang, C. Zhang, F. Tang, H. Jiang, Y. Wu, and Y. Liu. Lightweight object-level topological semantic mapping and long-term global localization based on graph matching. *CoRR*, abs/2201.05977, 2022. URL https://arxiv.org/abs/2201.05977.

[204] G. Wang, Z. Xiong, D. Liu, and C. Luo. Cascade mask generation framework for fast small object detection. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2018. doi: 10.1109/ICME.2018.8486561.

[205] P. Wells. Apple doubles down on hardware with new ipad pro. *The Sydney Morning Herald*, 2020. URL https://www.smh.com.au/technology/apple-doubles-down-on-hardware-with-new-ipad-pro-20200325-p54doj.html.

[206] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, I3D '95, pages 5–12, New York, NY, USA, 1995. Association for Computing Machinery. ISBN 0897917367. doi: 10.1145/199404.199405. URL https://doi.org/10.1145/199404.199405.

[207] D. Wood, M. Lanthaler, and R. Cyganiak. RDF 1.1 concepts and abstract syntax, 2014. URL http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[208] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[209] F. Yang, Q. Sun, H. Jin, and Z. Zhou. Superpixel segmentation with fully convolutional networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern*

Recognition (CVPR), pages 13961–13970. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01398.

[210] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph R-CNN for scene graph generation. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, September 8-14, 2018, Proceedings, Part I*, volume 11205 of *Lecture Notes in Computer Science*, pages 690–706. Springer, 2018. doi: 10.1007/978-3-030-01246-5_41. URL https://doi.org/10.1007/978-3-030-01246-5_41.

[211] X. Yang, K. Tang, H. Zhang, and J. Cai. Auto-encoding scene graphs for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10685–10694. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.01094. URL https://doi.org/10.1109/CVPR.2019.01094.

[212] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11821–11830. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01184.

[213] B. Yu, C. Chen, F. Zhou, F. Wan, W. Zhuang, and Y. Zhao. A bottom-up framework for construction of structured semantic 3d scene graph. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8224–8230. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341228.

[214] G. W. Zack, W. E. Rogers, and S. A. Latt. Automatic measurement of sister chromatid exchange frequency. *Journal of Histochemistry and Cytochemistry*, 25: 741–753, 1977. doi: 10.1177/25.7.70454.

[215] Y. Zhan, J. Yu, T. Yu, and D. Tao. On exploring undetermined relationships for visual relationship detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5123–5132. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00527.

[216] J. Zhang, Y. Yao, and B. Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3450–3466, 2022. doi: 10.1109/TPAMI.2021.3054619.

[217] H. Zhu, J. Deng, Y. Zhang, J. Ji, Q. Mao, H. Li, and Y. Zhang. Vpfnet: Improving 3d object detection with virtual point based lidar and stereo data fusion. *CoRR*, abs/2111.14382, 2021. URL https://arxiv.org/abs/2111.14382.

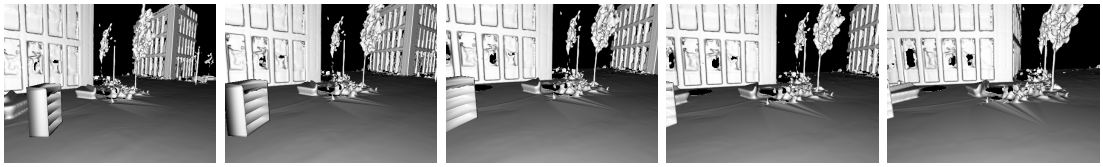
Appendix A

Photo-geometric change detection dataset

This appendix showcases the datasets of scenes on which photo-geometric change detection was performed for our work in Chapter 3. Each scene's past and present states are respectively described by an outdated 3D mesh and a sequence of registered current images. All the images of a sequence are taken within a narrow time frame to reduce structural or lighting changes. For clarity, the virtual cameras chosen to render the meshes (first row of the figures) align with the estimated camera poses of the image sequences (second row), and a 2D ground truth is provided for each such pose (third row). The cameras' intrinsic and extrinsic matrices are taken from the original datasets [37, 136] and are sometimes slightly inaccurate in their estimation of the camera's pose or image distortion from its optics.

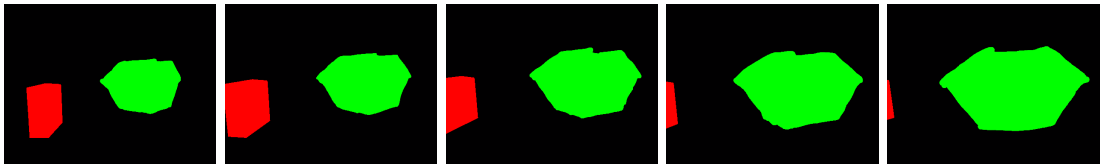
The last row presents the results of the 3D detection of removed objects from Chapter 3, projected in 2D for comparison with the ground truth. To simulate object removal (from a reference mesh to current images), 3D models of objects were downloaded from Turbosquid [193] and added to the meshes, and the download links are provided for each scene. More quantitative results are provided in Chapter 3, notably the ROC curves (*true positive rate* against *false positive rate*) produced by our removal detection algorithm.

A.1 Scenes from Palazzolo et al. dataset

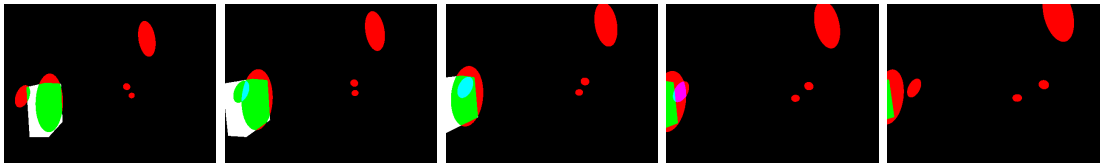


(A) Outdated 3D mesh showing removed object (shelf): www.turbosquid.com/3d-models/1548060

(B) Up-to-date images showing inserted object (container)

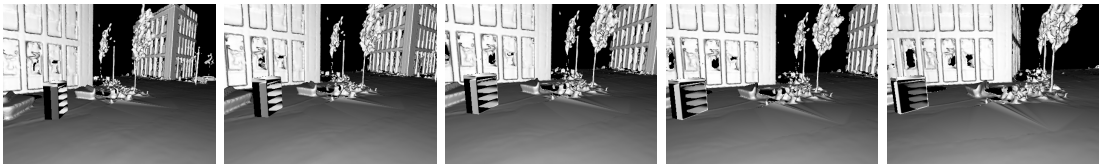


(C) Ground truth: insertions (green) and removals (red).



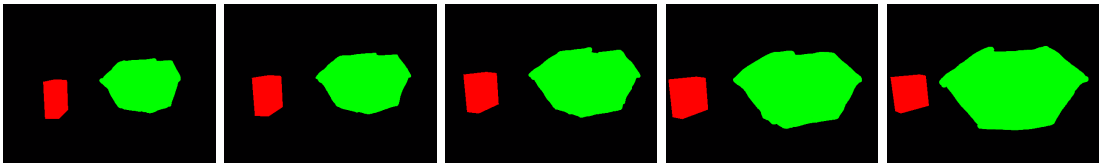
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.1: container-shelf.

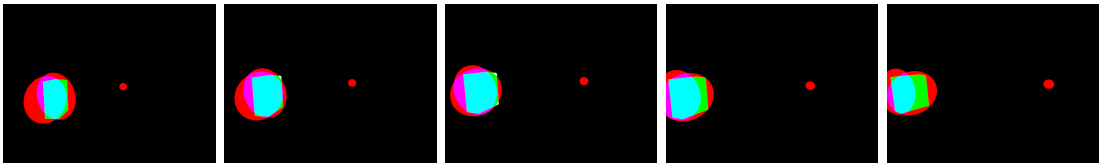


(A) Outdated 3D mesh showing removed object (shelf): www.turbosquid.com/3d-models/1548060

(B) Up-to-date images showing inserted object (container)



(C) Ground truth: insertions (green) and removals (red).

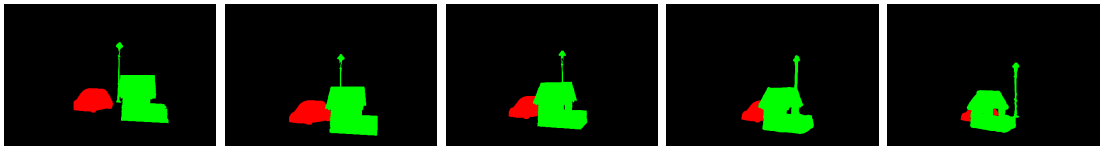


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

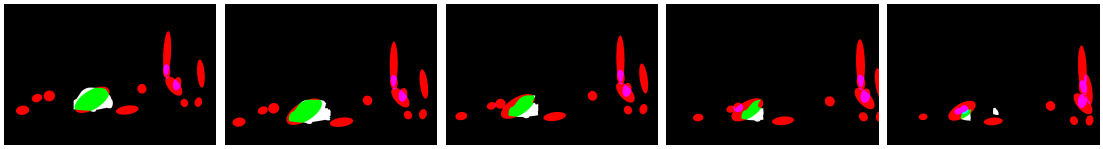
FIGURE A.2: container-shelf2.

(A) Outdated 3D mesh showing removed object (car): www.turbosquid.com/3d-models/1330846

(B) Up-to-date images showing inserted object (play house)



(C) Ground truth: insertions (green) and removals (red).

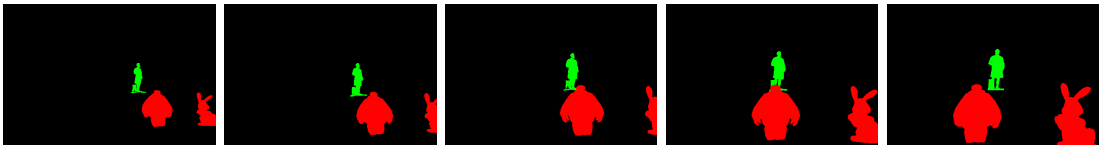


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

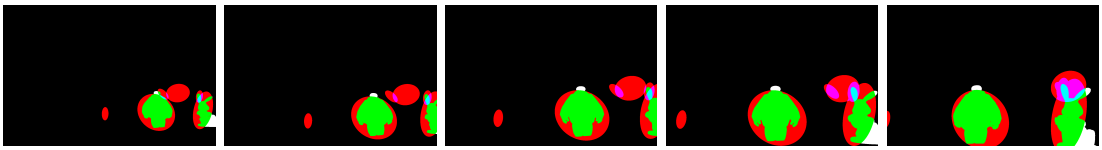
FIGURE A.3: playground-car.

(A) Outdated 3D mesh showing removed object (robot, rabbit): www.turbosquid.com/3d-models/?

(B) Up-to-date images showing inserted object (statue)



(C) Ground truth: insertions (green) and removals (red).

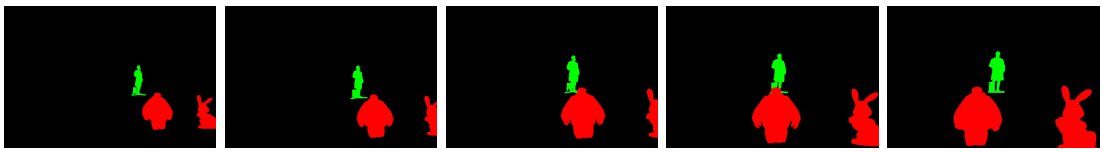


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

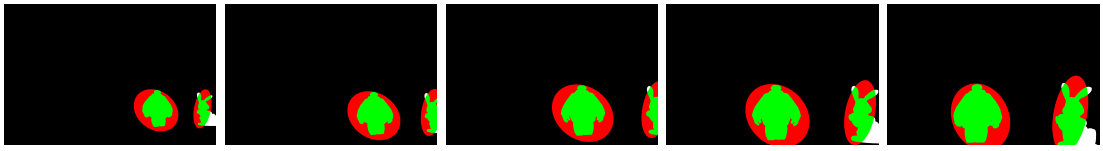
FIGURE A.4: statue-robot.

(A) Outdated 3D mesh showing removed object (robot, rabbit): www.turbosquid.com/3d-models/?

(B) Up-to-date images showing inserted object (statue)



(C) Ground truth: insertions (green) and removals (red).

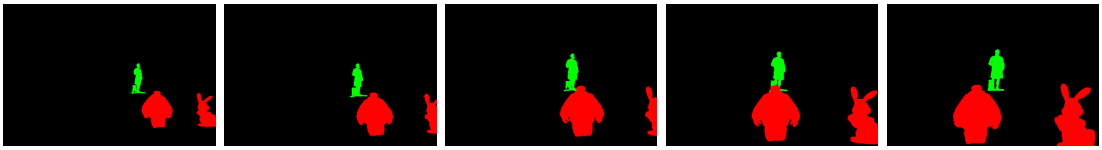


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

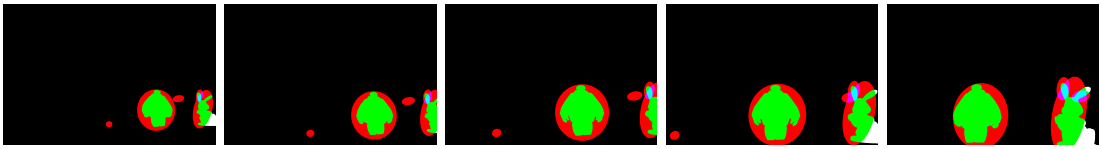
FIGURE A.5: statue-robot-bad-exp.

(A) Outdated 3D mesh showing removed object (robot, rabbit): www.turbosquid.com/3d-models/?

(B) Up-to-date images showing inserted object (statue)



(C) Ground truth: insertions (green) and removals (red).

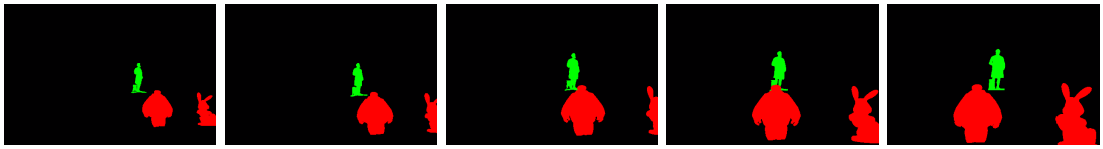


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

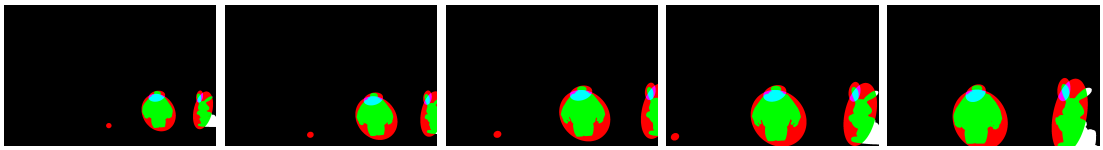
FIGURE A.6: statue-robot-bad-temp.

(A) Outdated 3D mesh showing removed object (robot, rabbit): www.turbosquid.com/3d-models/?

(B) Up-to-date images showing inserted object (statue)

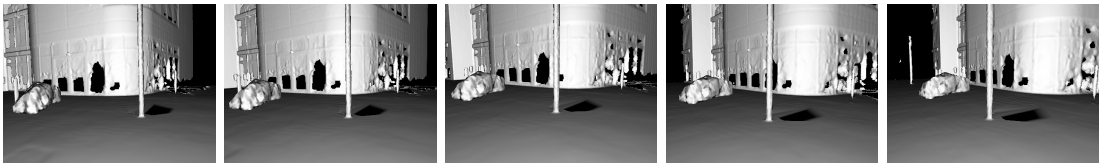


(C) Ground truth: insertions (green) and removals (red).



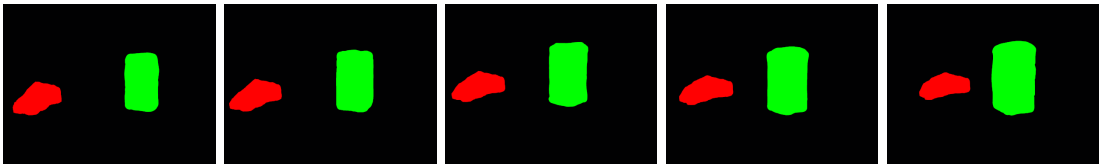
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.7: statue-robot-temp.

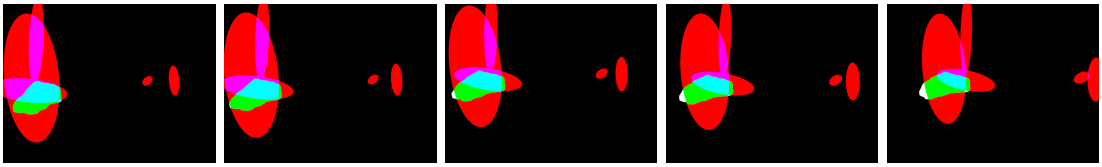


(A) Outdated 3D mesh showing removed object (stone): www.turbosquid.com/3d-models/1300107

(B) Up-to-date images showing inserted object (portable toilet)



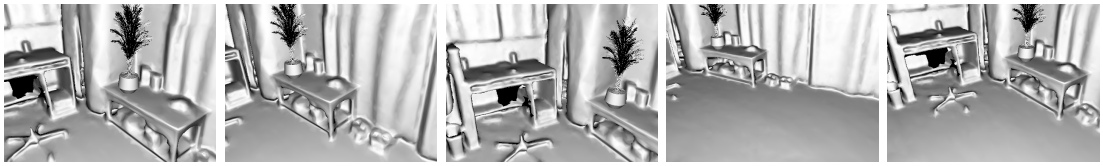
(C) Ground truth: insertions (green) and removals (red).



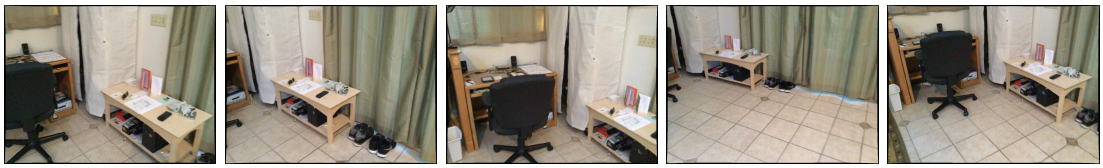
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.8: toilet-stone.

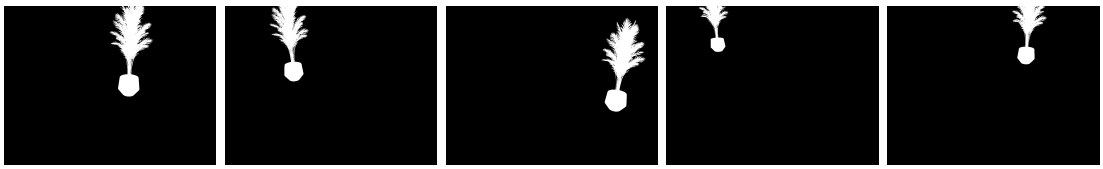
A.2 Scenes from ScanNet dataset



(A) Outdated 3D mesh showing removed object (plant): www.turbosquid.com/3d-models/1528072



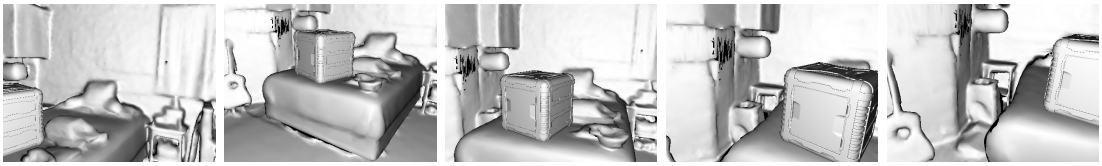
(B) Up-to-date images.



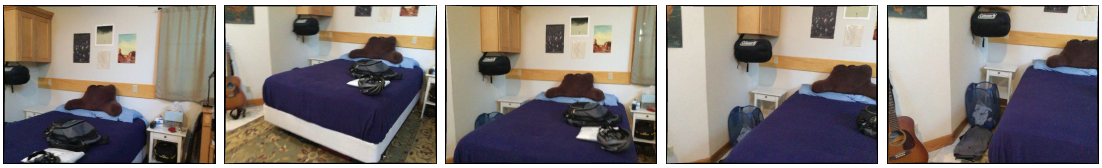
(C) Ground truth: removals (white).

(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

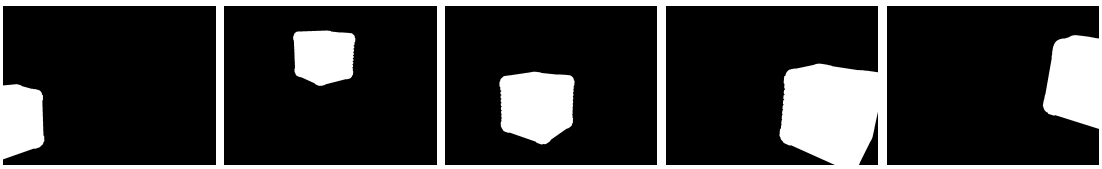
FIGURE A.9: 0000_00+plant.



(A) Outdated 3D mesh showing removed object (box): www.turbosquid.com/3d-models/1369994



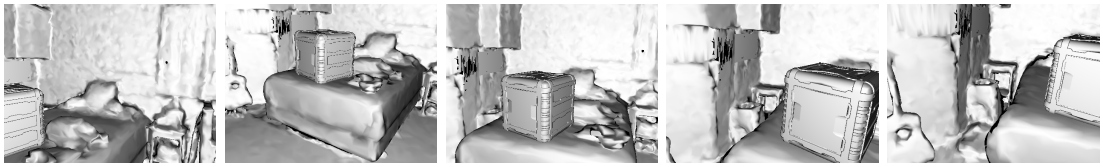
(B) Up-to-date images.



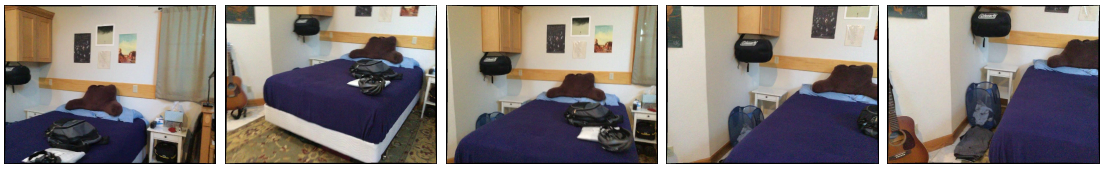
(C) Ground truth: removals (white).

(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

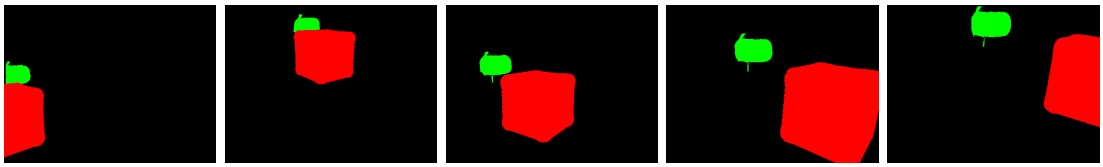
FIGURE A.10: 0000_01+box.



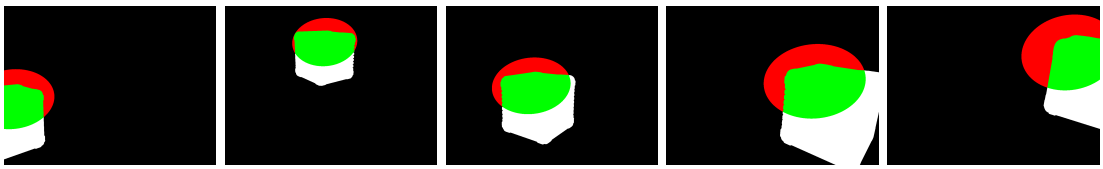
(A) Outdated 3D mesh showing removed object (box): www.turbosquid.com/3d-models/1369994



(B) Up-to-date images showing inserted object (gym bag)

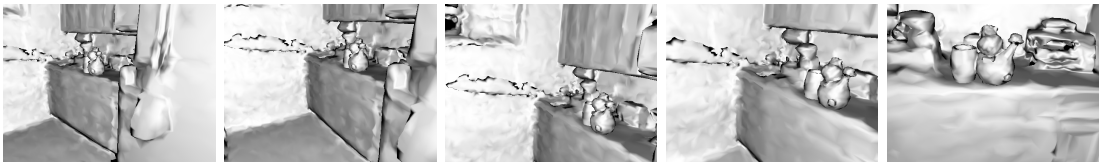


(C) Ground truth: insertions (green) and removals (red).

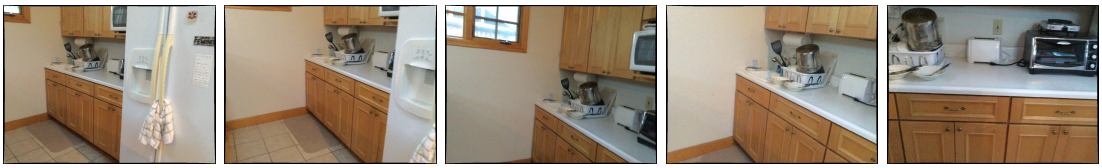


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.11: 0000_01-insert+box.

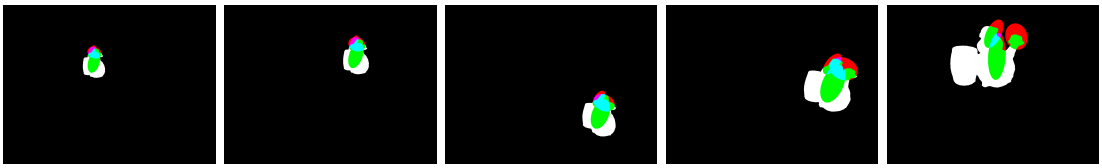


(A) Outdated 3D mesh showing removed object (statue): www.turbosquid.com/3d-models/1335035



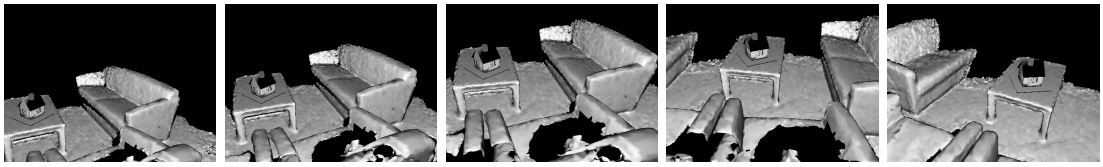
(B) Up-to-date images.

(C) Ground truth: removals (white).

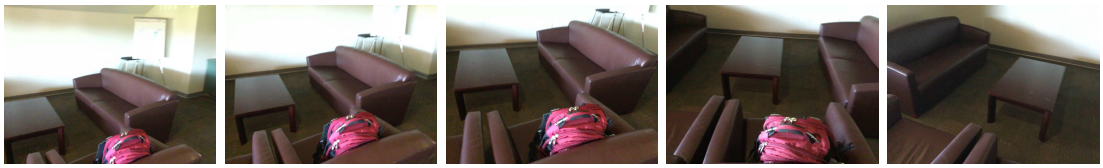


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.12: 0000_02+statue.

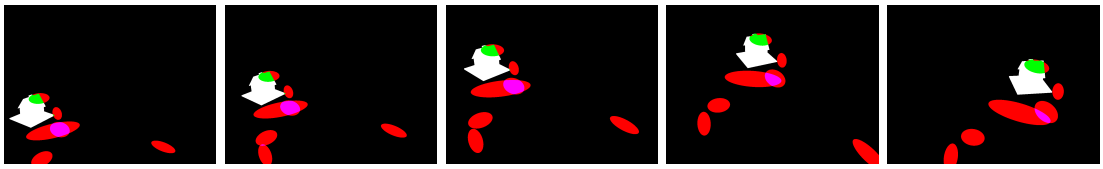


(A) Outdated 3D mesh showing removed object (doll house): www.turbosquid.com/3d-models/1576949



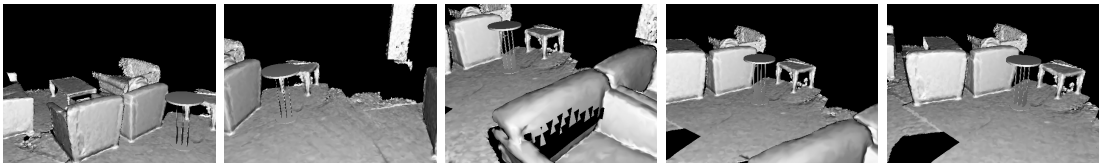
(B) Up-to-date images.

(C) Ground truth: removals (white).



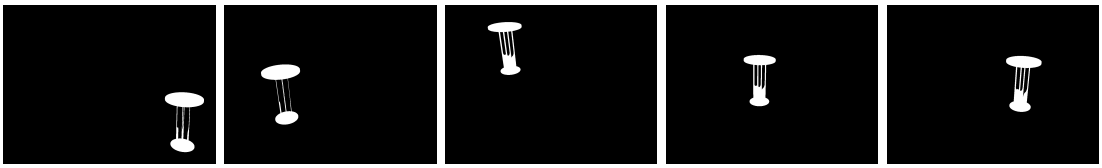
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.13: 0001_00+dollhouse.

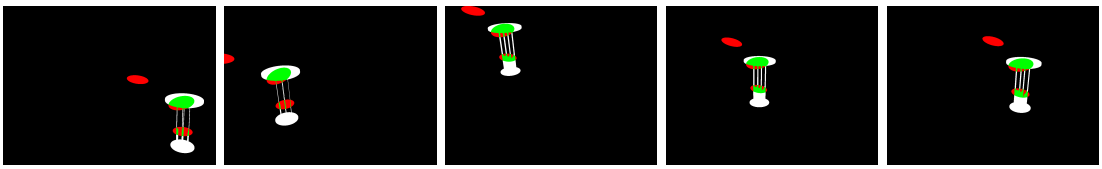


(A) Outdated 3D mesh showing removed object (table): www.turbosquid.com/3d-models/1578760

(B) Up-to-date images.



(C) Ground truth: removals (white).

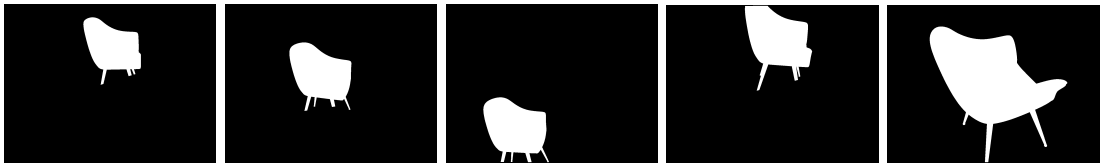


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.14: 0001_01+table.

(A) Outdated 3D mesh showing removed object (chair): www.turbosquid.com/3d-models/1551213

(B) Up-to-date images.



(C) Ground truth: removals (white).

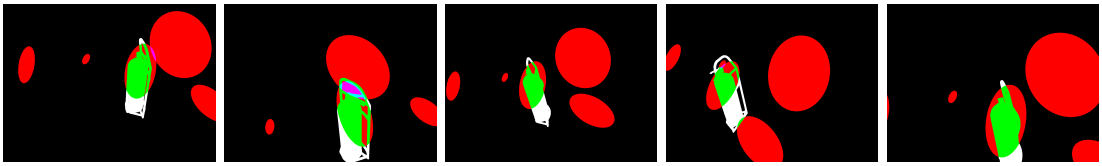
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.15: 0002_00+chair.

(A) Outdated 3D mesh showing removed object (fire extinguisher): www.turbosquid.com/3d-models/1447524

(B) Up-to-date images.

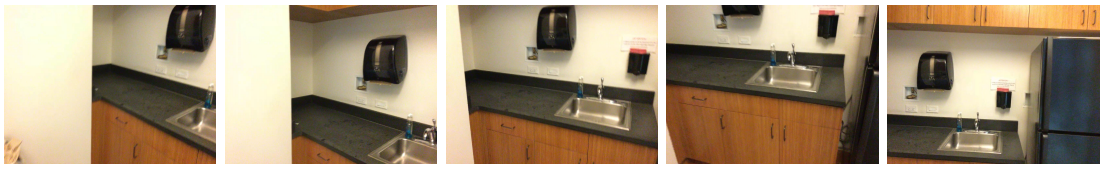
(C) Ground truth: removals (white).



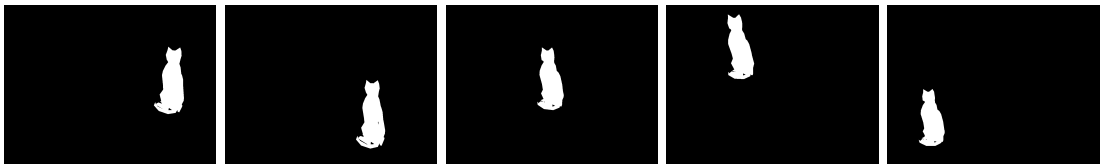
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.16: 0002_01+extinguisher.

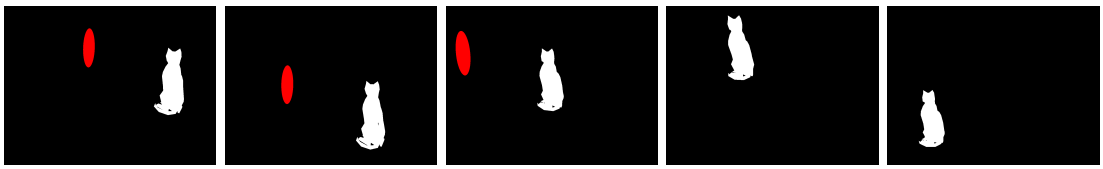
(A) Outdated 3D mesh showing removed object (cat): www.turbosquid.com/3d-models/1340490



(B) Up-to-date images.



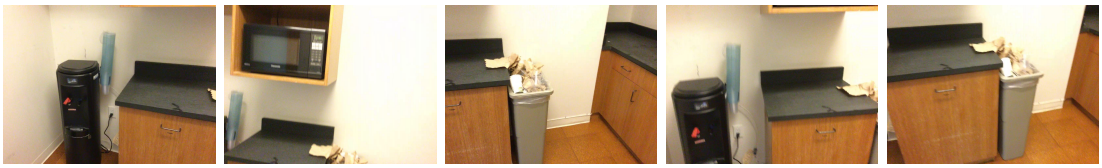
(C) Ground truth: removals (white).



(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.17: 0003_00+cat.

(A) Outdated 3D mesh showing removed object (desk lamp): www.turbosquid.com/3d-models/1522080

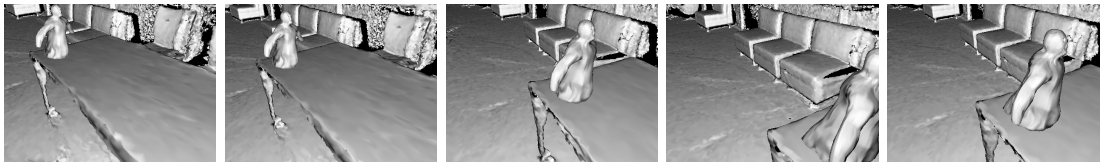


(B) Up-to-date images.

(C) Ground truth: removals (white).

(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

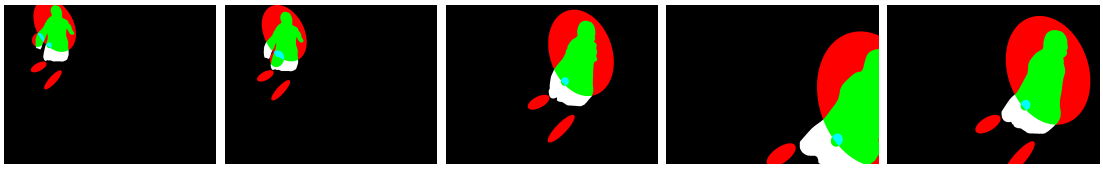
FIGURE A.18: 0003_01+desklamp.



(A) Outdated 3D mesh showing removed object (ghost): www.turbosquid.com/3d-models/1419900

(B) Up-to-date images.

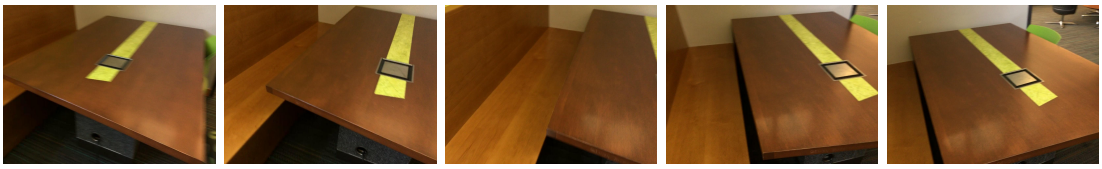
(C) Ground truth: removals (white).



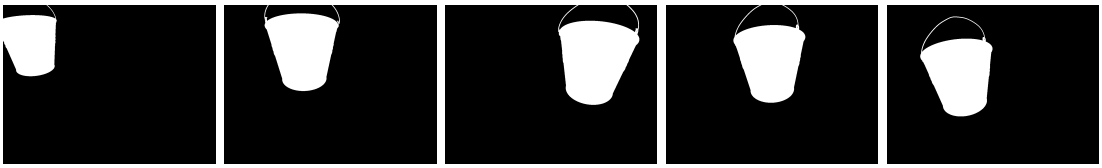
(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.19: 0004_00+ghost.

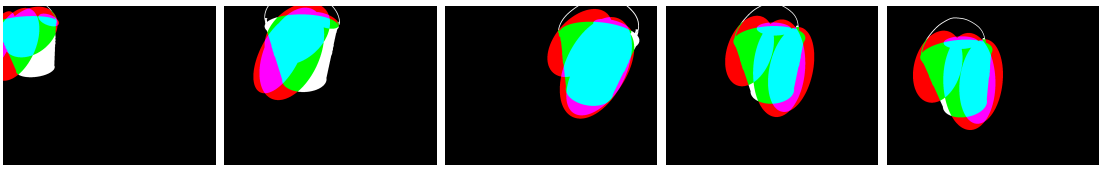
(A) Outdated 3D mesh showing removed object (bucket): www.turbosquid.com/3d-models/1288741



(B) Up-to-date images.



(C) Ground truth: removals (white).

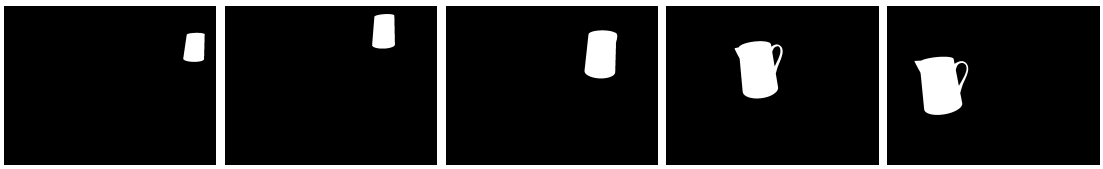


(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

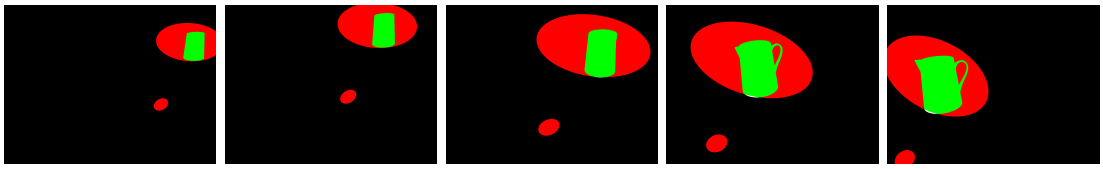
FIGURE A.20: 0005_00+bucket.

(A) Outdated 3D mesh showing removed object (pitcher): www.turbosquid.com/3d-models/1368224

(B) Up-to-date images.



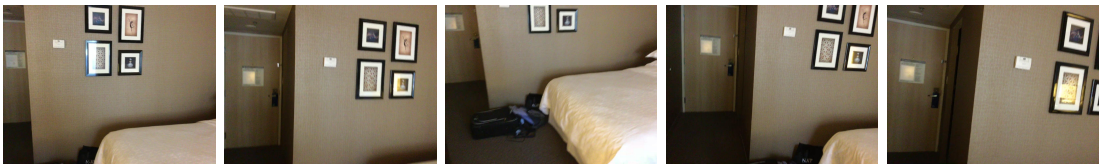
(C) Ground truth: removals (white).



(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

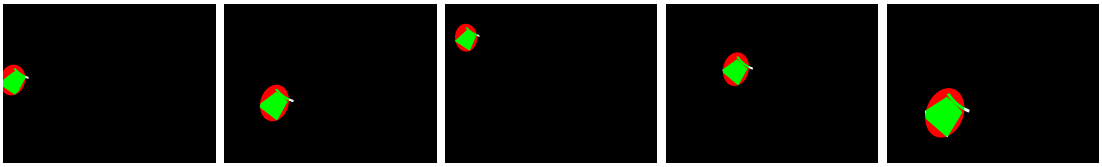
FIGURE A.21: 0005_01+pitcher.

(A) Outdated 3D mesh showing removed object (lamp): www.turbosquid.com/3d-models/1428616



(B) Up-to-date images.

(C) Ground truth: removals (white).



(D) Projected 3D removal detection (white: false negative, green/cyan: true positive, red/magenta: false positive).

FIGURE A.22: 0006_00+lamp.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Vers une détection des changements géométriques et sémantiques pour des expériences de réalité mixte

Mots clés : Réalité Mixte, détection de changements, analyse sémantique, reconstruction 3D

Résumé : La Réalité Augmentée (RA) photoréaliste (« Réalité Mixte ») est utile pour diverses applications immersives, du divertissement visuel à la formation par simulation, et pour la prévisualisation d'aspect. L'intégration réaliste d'éléments virtuels dans une scène nécessite un modèle à jour de sa géométrie et son contenu sémantique. La structure 3D d'un environnement impacte le rendu visuel ainsi que l'organisation spatiale du contenu virtuel et ses interactions avec le réel. En raison des limitations matérielles des appareils RA grand public, faire un scan géométrique exhaustif de la scène à chaque utilisation est irréaliste. Néanmoins, une partie de l'information structurelle dans la scène reste inchangée et ne requiert que des mises à jour locales pour rester fidèle au réel.

Cette thèse a pour objectif de donner les moyens d'identifier et corriger les régions changeantes d'une scène en détectant les différences entre une réprésentation antérieure et des observations courantes.
Nous présenterons un nouveau système de détection de changements géométrique léger, utilisant une méthode de reprojection d'images.
Nous présenterons également l'élaboration d'un modèle sémantique de scène pour la Réalité Mixte à base de graphes, ainsi qu'une méthode de génération de ce modèle par analyse sémantique. Ce même modèle sera alors exploité dans un système d'alignement de scènes 3D, pour évaluer son applicabilité à la localisation en RA.

Title : Towards geometric and semantic change detection for mixed reality experiences

Keywords : Mixed Reality, change detection, semantic analysis, 3D reconstruction

Abstract : Photorealistic Augmented Reality (AR) –or "Mixed Reality"– is applicable to various immersive experiences, from entertainment to simulation-based training, and for previsualization tasks. The realistic integration of virtual elements requires an accurate and up-to-date model of the real scene's geometry and semantic contents. The 3D structure of the environment impacts the visual rendering as well as the spatial layout of the virtual content and its interactions with real objects. Given the hardware constraints of widely available AR devices, it is impractical to thoroughly scan the geometry of the scene with each use. Still, some the scene's structural information will remain constant over time only requiring local updates to accurately represent the scene's current state.

The aim of this doctoral study has been to provide the means to identify and correct areas of change in a scene through detection of inconsistencies between a prior representation and an current observations.
In this thesis, we present the completion of a lightweight reprojection-based geometric change detection framework. We also present the elaboration of a graph-based semantic scene model for Mixed Reality, and a method for its generation from semantic analysis. This model will then be used in a 3D scene registration system, in order to test its applicability to the localization task in AR.