Improving 3D Reconstruction using Adaptative RanSaC and Benchmarking with Semi-Artificial Data Generation

Clément Riu

Université Gustave Eiffel

Thèse présentée pour obtenir le grade de docteur

Université Gustave Eiffel

❧

Laboratoire d'Informatique Gaspard Monge
École doctorale MSTIC

Discipline : Informatique

Improving 3D Reconstruction using Adaptative RanSaC and Benchmarking with Semi-Artificial Data Generation

---

Par : **Riu Clément**

Membres du jury:

Reviewer : Moisan Lionel , Professor at Université Paris Cité

Reviewer : Pajdla Tomas , Professor at Czech Technical University in Prague

Examiner : Desolneux Agnès , Research Director at CNRS, Centre Borelli

President : Gouet-Brunet Valérie, Research Director at IGN-ENSG, Université Gustave Eiffel

Supervisor : Vincent Nozick, Maître de Conférence, HDR at Université Gustave Eiffel

Supervisor : Pascal Monasse, Professor at Laboratoire d'Informatique Gaspard-Monge

Date de soutenance : 12 Décembre 2023

École des Ponts ParisTech
LIGM-IMAGINE
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2
France
Université Paris-Est Marne-la-Vallée
École Doctorale Paris-Est MSTIC
Département Études Doctorales
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77454 Marne-la-Vallée cedex 2
France

Abstract - Résumé

Short Abstract - English Version

Research around Robust Estimator in the context of 3D Stereo Reconstruction is complicated by the difficulty to obtain reliable Ground Truth data. The literature around this subject proposes a number of RanSaC like algorithms that are compared to each other using either fully artificial data or estimated real data which leads to unreliable conclusions. Recent work focuses on methods that do not use user-defined thresholds which require strong hypothesis on the data distribution. To efficiently compare this new methods we propose a new benchmarking solution that will help leveraging recent work to improve the State of the Art. Our first contribution is a novel data generation method that relies on real data to obtain a realistic model and distribution of data points to create reliable benchmarks with precise metrics. We then used this method to compare most recent adaptative methods on a variety of Multi-View Stereo (MVS) and Structure-from-Motion (SfM) problems and obtain insight in the capabilities of each algorithm. Using these conclusions, we tried to improve ColMap using user-set-threshold-free methods. Finally, we provide the complete code base used to generate the benchmarks, with all tested methods in a unified framework. This includes an implementation of an algorithm making use of Likelihood Ratio Tests which had not been made available by the original authors.

Keywords: RanSaC, Semi-artificial dataset, Multi-View Stereo (MVS), Structure-from-Motion, Perspective-from-n-Points (PnP), Benchmark, Image processing, 3D Reconstruction, ColMap

Résumé court - Version Française

La recherche sur les estimateurs robustes pour la reconstruction stéréo 3D est compliquée par la difficulté d'obtention de données précises pour évaluer les méthodes. La littérature sur ce sujet propose un certain nombre d'algorithmes de type RanSaC qui sont comparés les uns aux autres en utilisant soit des données entièrement artificielles, soit des données réelles estimées ce qui conduit à des conclusions peu fiables. Les travaux récents se concentrent sur les méthodes qui n'utilisent pas de seuils définis par l'utilisateur. Pour comparer efficacement ces nouveaux algorithmes, nous proposons une nouvelle méthode d'analyse. Notre première contribution est une méthode de génération de données qui s'appuie sur des données réelles pour obtenir un modèle et une distribution réalistes des données afin de créer des benchmark fiables. Nous avons ensuite utilisé cette méthode pour comparer les algorithmes adaptatifs les plus récentes sur une variété de problèmes de reconstruction stéréo et obtenir un aperçu des capacités de chaque algorithme. Sur la base de cette analyse, nous avons essayé d'améliorer ColMap en utilisant des méthodes adaptatives. Enfin, nous fournissons la base de code complète utilisée pour générer les benchmarks, avec toutes les méthodes testées dans un framework unifié. Cela inclut une implémentation d'un algorithme utilisant le Test de Rapport de Vraisemblance qui n'avait pas été mise à disposition par les auteurs originaux.

Keywords: Perspective à partir de n Points (PnP), Stéréoscopie Multi-Vues (MVS), Structure à partir d'un Mouvement, RanSaC, Jeux de données semi-artificielles, Benchmark, Traitement d'images, Reconstruction 3D, ColMap

Résumé substantiel - Version Française

La reconstruction 3D à partir d'images, telle que la stéréoscopie multi-vues (MVS) et la structure à partir du mouvement (SfM), consiste à récupérer des données de profondeur à partir de données 2D, généralement en utilisant l'analyse géométrique d'une scène et en reliant les images entre elles. En utilisant différents points de vue d'une même scène ou d'un même objet, la profondeur de chaque point peut théoriquement être récupérée car les rayons lumineux qui se réfléchissent sur le point suivent une trajectoire droite depuis le point jusqu'à chaque position de la caméra et le point peut donc être trouvé à l'intersection de ces rayons. C'est le cas d'un modèle de sténopé parfait, où chaque point est capturé par un seul rayon lumineux qui aboutit à une position unique sur le capteur de la caméra. Bien sûr, dans la réalité, le trajet de la lumière sera déformé par les lentilles situées devant le capteur, des effets de quantification s'ajouteront aux pixels des images et un flou causé par le mouvement ou une mauvaise mise au point apparaîtra. Des solutions ont été mises au point pour tenir compte de ces différents problèmes et, aujourd'hui, la reconstruction 3D a de nombreuses applications industrielles, de la conservation du patrimoine à la robotique et aux véhicules autopilotés. La reconstruction 3D est également à la base d'autres tâches, comme la reconstruction de surfaces, la modélisation d'objets en 3D, le suivi d'objets et la relocalisation.

Aussi intuitif que cela puisse paraître, la reconstruction nécessite l'exécution de plusieurs tâches difficiles, tout d'abord la recherche de la position des caméras autour de la scène afin de pouvoir tracer la position des points. Cela implique généralement de trouver des correspondances entre les images et d'analyser la déformation entre elles pour trouver les positions relatives les unes par rapport aux autres et enfin évaluer la profondeur de la scène jusqu'à une certaine échelle. Si plus de deux images sont utilisées pour la reconstruction, l'ajout d'images ajoutera des erreurs et entraînera une dérive qui doit également être prise en compte. Ces défis ont été en grande partie relevés et des logiciels tels que ColMap [79], Bundler [81] ou VisualSFM [96] offrent des solutions prêtes à l'emploi pour effectuer la reconstruction 3D.

La plupart des travaux récents dans le domaine du traitement des images 3D utilisent des réseaux neuronaux et l'apprentissage profond au lieu de méthodes plus traditionnelles. Ils peuvent être utilisés pour remplacer des parties du pipeline, comme les caractéristiques apprises ou l'estimation de la pose, ou pour effectuer une estimation de la profondeur de bout en bout. Les méthodes d'apprentissage automatique s'avèrent efficaces dans une grande variété de tâches, en particulier les plus complexes, comme la reconstruction d'une vue unique, où la profondeur et les éléments non vus doivent être reconstruits à partir d'une seule vue d'un objet, ou la génération de la vue suivante, où une nouvelle image 2D est générée à partir de la scène et d'une nouvelle position de la caméra. Ces méthodes ont également un impact considérable sur les problèmes 3D dérivés, comme la modélisation

et la reconstruction d'objets en 3D.

Cependant, lorsque l'on utilise des méthodes d'apprentissage, la question des données d'entraînement se pose et il est difficile d'obtenir des données 3D fiables. Tous les ensembles de données dépendent d'une certaine forme d'estimation pour obtenir des données de vérité terrain précises, soit en utilisant un capteur actif comme un LIDAR ou une caméra de profondeur et en calibrant le résultat de leurs mesures sur l'ensemble de l'image, soit en utilisant une estimation de pipeline traditionnelle comme vérité terrain. Cela rend les algorithmes d'évaluation comparative complexes, car tous les résultats sont comparés à des données partiellement incorrectes et introduisent un biais quant à la manière dont les données ont été obtenues. Les méthodes d'apprentissage profond sont bien plus performantes que les méthodes traditionnelles pour les tâches les plus complexes, mais en ce qui concerne la reconstruction 3D de base, les méthodes traditionnelles comme ColMap sont toujours très précises et plus robustes [21]. L'amélioration des méthodes traditionnelles permettra d'obtenir des données d'entraînement plus précises qui pourraient contribuer à l'apprentissage de meilleurs modèles. Ces méthodes permettent de reconstruire une image à partir d'une seule vue d'un objet, de générer une nouvelle image en 2D à partir de la scène et d'une nouvelle position de la caméra. Ces méthodes ont également un impact considérable sur les problèmes 3D dérivés, comme la modélisation et la reconstruction d'objets en 3D.

À la lumière de ces observations, nous examinons comment améliorer la méthode traditionnelle. Le pipeline de reconstruction habituel se compose principalement de trois éléments : la détection et la mise en correspondance de points caractéristique, l'estimation robuste du modèle et les optimisations globales. Des travaux récents ont été réalisés sur tous ces éléments mais, comme nous l'avons déjà dit, il est difficile de comparer les nouvelles méthodes entre elles et la plupart des publications récentes se contredisent sur les algorithmes les plus performants dans tel ou tel scénario, principalement parce qu'elles utilisent des méthodes différentes pour générer leurs vérités de base et qu'elles utilisent des mesures empiriques différentes. Cela est particulièrement vrai dans le domaine des estimations robustes, où de nombreuses solutions nouvelles sont encore produites et sont censées battre l'état de l'art précédent, mais ne sont pas intégrées dans des solutions logicielles telles que ColMap.

Les améliorations les plus récentes autour des estimateurs robustes se concentrent sur les méthodes adaptatives. Un estimateur robuste évalue un modèle en présence de données aberrantes qui empêchent l'utilisation d'un estimateur global. Ces estimateurs fournissent généralement en même temps un modèle estimé uniquement sur les données valides et une classification des points de données comme valides ou aberrants en fonction de leur distance par rapport au modèle et d'un seuil d'acceptation fixé par l'utilisateur. La qualité d'un modèle dépend de la taille de son ensemble consensus : l'ensemble des valeurs validant le modèle selon le seuil. La définition d'une valeur appropriée pour ce

seuil nécessite des connaissances sur les données qui ne sont pas disponibles a priori et implique soit une recherche itérative des paramètres, soit l'utilisation d'estimations prudentes qui peuvent conduire à des performances médiocres. Pour éviter ces problèmes, des méthodes adaptatives qui estiment simultanément la valeur du seuil et le modèle ont été développées. Ces méthodes nécessitent une fonction objective différente qui permet d'équilibrer le nombre de valeurs valides et la qualité du modèle, car la taille classique de l'ensemble consensus ne peut pas être utilisée lorsque le seuil varie. Nous concentrons notre étude sur ces méthodes et sur la manière de les utiliser pour améliorer le pipeline classique.

Pour analyser efficacement les différentes méthodes disponibles dans la littérature, nous avons créé une nouvelle méthode de génération de données. Il s'agit d'une méthode semi-artificielle qui utilise des données réelles pour sélectionner un modèle et guider la génération de valeurs aberrantes contrôlées afin d'éviter les lacunes des méthodes d'évaluation habituelles. Les méthodes entièrement artificielles nécessitent un choix sur la distribution du modèle, des points de données valides et des points de données aberrantes, ce qui peut avoir un impact considérable sur les performances des algorithmes, car la plupart des méthodes adaptatives doivent faire des hypothèses sur la distribution des données pour calculer une fonction objective différente de la taille de l'ensemble consensus. Les données artificielles soulèvent également la question de la possibilité d'application des résultats dans le monde réel. D'autre part, comme indiqué précédemment, tous les ensembles de données du monde réel présentent une certaine imprécision, qu'ils utilisent des capteurs actifs ou non, car certains algorithmes doivent être exécutés et des estimations doivent être faites pour créer l'ensemble de données, ce qui introduit une incertitude dans le résultat. La plupart des comparaisons effectuées à l'aide de ces données reposent sur une analyse qualitative et les résultats quantitatifs se concentrent sur l'obtention d'une qualité de base. L'utilisation de données réelles empêche également la catégorisation des scènes en fonction de leur difficulté, car il n'y a pas de mesure objective du défi présenté, qui ne peut être analysé qu'a posteriori en regardant quelle configuration s'est avérée plus difficile à traiter pour les algorithmes testés.

C'est pourquoi notre solution guide la méthode de génération à l'aide d'estimations réelles, tout en créant une vérité de terrain artificielle sans incertitude quant aux valeurs valides et aberrantes. Nous pouvons ensuite analyser les performances des méthodes adaptatives en faisant varier l'ensemble de données de base, le bruit des valeurs valides et le ratio des valeurs aberrantes pour créer des scènes de difficulté variable et révéler le comportement intrinsèque des algorithmes si des tendances se dégagent dans toutes les situations testées. Nous commençons par effectuer une analyse comparative des méthodes adaptatives sur des tâches à deux vues telles que l'estimation de l'homographie, l'estimation de la matrice fondamentale et l'estimation de la matrice essentielle, ainsi que le problème PnP. Cette analyse nous permet de mieux comprendre les méthodes adap-

tatives et leurs performances afin d'exploiter les meilleures améliorations proposées dans un pipeline de reconstruction complet. Nous avons choisi les algorithmes adaptatifs les plus performants pour la tâche PnP et analysé leur impact sur l'ensemble du pipeline en remplaçant le RanSaC classique de la tâche PnP dans le logiciel ColMap. Notre objectif est d'améliorer la robustesse de ColMap en supprimant un hyper-paramètre et donc en supprimant la nécessité d'adapter sa valeur aux tâches à accomplir, et potentiellement, grâce à de meilleurs algorithmes RanSaC, de traiter des scènes plus difficiles que ColMap ne parvient toujours pas à traiter correctement.

La première contribution que nous apportons est une nouvelle méthode d'évaluation comparative qui s'appuie sur des données semi-artificielles. Afin d'obtenir des mesures fiables des performances des algorithmes d'estimation robuste tout en disposant de données réalistes pouvant être étendues à des configurations réelles, nous avons développé une nouvelle méthode de génération de données. Cette méthode repose sur l'utilisation d'une estimation corrigée d'une scène réelle pour créer des valeurs valides et des valeurs aberrantes fiables, mais toujours basées sur des éléments réels. Elle propose également une nouvelle approche pour générer des valeurs aberrantes qui posent davantage de problèmes à un estimateur robuste.

Afin de réaliser un jeu de données de test, n'importe quelle paire d'images ou ensemble de correspondances 2D-3D obtenu d'une scène réelle peut être utilisé comme entrée. Un modèle est alors estimé avec un algorithme de RanSaC quelconque (nous avons choisi AC-RanSaC) et ce modèle, bien qu'imparfait, devient la Vérité Terrain (*Ground Truth*) semi-artificielle de notre générateur. Les correspondances sont alors corrigées pour s'aligner parfaitement avec ce model vérité terrain, il s'agit des *ground truth inliers*. Ainsi, les algorithmes testés seront bel et bien évalués sur leur capacité à reproduire la vérité terrain et non leur capacité à reproduire le résultat de l'estimation initiale. À partir de ce modèle et des correspondances associées, il est possible de générer du bruit pour les valeurs valides, *inliers* et des valeurs aberrantes, *outliers* pour étudier la robustesse des différentes méthodes.

Le bruit associé aux *inliers* est dans la plupart des cas relativement proche d'un bruit gaussien, néanmoins nos expériences n'ont pas révélé de différences notables de comportement en utilisant un bruit uniforme. Utiliser un bruit uniforme permet d'avoir plus de contrôle sur l'erreur maximale des *inliers* et leur variance. Du bruit est donc généré pour les données *inliers* en choisissant un «côté» de la donnée d'entrée, soit une des images pour des cas de géométrie à deux vues, soit le plan 2D pour le problème PnP, et en s'éloignant du modèle vérité terrain, en choisissant une distance et direction aléatoire pour des problèmes point à point et en choisissant une distance aléatoire et une direction perpendiculaire à la ligne épipolaire pour les problèmes points à ligne. Le bruit maximal généré devient alors le seuil entre données valides et données aberrantes qui sera utilisé pour générer les données aberrantes.

Pour pouvoir mesurer la capacité des algorithmes de RanSaC à discriminer les *inliers* des *outliers* nous générons de vrais *outliers*, c'est à dire des points dont la distance au modèle est supérieure aux niveaux de bruit de tous les *inliers*. En effet, un point qui tomberait par hasard dans la zone d'acceptation du modèle, même si sémantiquement aberrant, n'aurait pas de différence géométrique avec des données valides. Il faut donc s'assurer d'une distance minimale des points aberrants. Lorsque les *outliers* sont générés uniformément dans la région outlier du modèle, ceux-ci ne présentent que peu de défis aux algorithmes d'estimation robuste. Pour palier ce problème nous utilisons une distribution uniforme non pas dans l'espace des correspondances mais dans l'espace des erreurs. La génération d'un point aberrant passe alors par la génération d'une erreur d'une valeur tirée aléatoirement. Pour ce faire, un point est tiré de l'autre côté de la donnée par rapport aux *inliers*, l'autre image ou l'espace 3D, et sa correspondance exacte *via* le modèle est calculée. Ensuite, une perturbation est générée uniformément parmi les perturbations possibles pour perturber la correspondance et assurer qu'elle soit aberrante. En contrôlant le niveau de bruit des *inliers* et le pourcentage d'*outliers* on peut alors tester les performances des algorithmes, notamment en calculant la précision et le rappel, en s'assurant de la fiabilité des métriques.

En utilisant cette méthode de génération de données, nous évaluons les différentes solutions de seuils adaptatifs qui éliminent la nécessité d'un seuil de valeurs aberrantes / valides défini par l'utilisateur. Ce *benchmark* permet de révéler le comportement et la durée d'exécution des algorithmes en fonction des niveaux de bruit et de valeurs aberrantes avec une plus grande confiance qu'auparavant. Cela permet de déterminer quels algorithmes fonctionnent le mieux dans quels scénarios et quels sont les seuils de performance où un algorithme commencera à être moins performant que le RanSaC de base. Les différents algorithmes testés sont MUSE [58], StaRSaC [13], A-Contrario RanSaC (AC-RanSaC) [61], Likelihood Ratio Test (LRT) [17], Marginalizing Sample Consensus (MAGSAC) [6] et deux variations Fast-AC-RanSaC[65] et MAGSAC++ [7]. Le benchmark a été lancé sur différentes tâches : l'estimation d'homographie, de matrice fondamentale, de matrice essentielle et le problème PnP. Afin de simultanément vérifier la validité de la méthode de génération de données, de multiples scènes issues de différents types de jeux de données ont été utilisées, en faisant à chaque fois varier le niveau de bruit des *inliers* de 0 à 3 pixels par incrément de 0.1 et le pourcentage d'*outliers* de 0% à 90% par incrément de 10% pour créer des tâches de complexité variée. Nous avons mesuré à la fois la précision et le rappel des différents algorithmes ainsi que leur temps d'exécution. La première observation que nous faisons est que StaRSaC est simplement trop lent par rapport aux autres méthodes, et ne présente pas de suffisamment bonne performance pour être utilisé de façon raisonnable. Ensuite, nous identifions que LRT, RanSaC et MUSE sont les algorithmes les plus rapides tandis que MAGSAC, MAGSAC++ et Fast-AC-RanSaC ont un temps de calcul intermédiaire et AC-RanSaC est le plus lent. Néanmoins,

lorsque la complexité de la tâche augmente trop, LRT, RANSAC et MAGSAC voient leur temps de calcul rapidement augmenter et dépasser parfois celui de AC-RanSaC. En termes de performance, les algorithmes présentent systématiquement une meilleure précision que leur rappel. Les algorithmes de MAGSAC, MAGSAC++ et AC-RanSaC présentent des résultats très robustes, quelque soit le niveau de complexité de la tâche, tandis que LRT et RanSaC voient leur performance diminuer significativement pour les tâches complexes. MUSE présente des résultats stables mais inférieurs à la plupart des autres algorithmes dans la plupart des situations. Ces observations se font quelque soit le problème d'estimation ou le *dataset* utilisé pour générer les données. Il y a bien sûr des variations dans les valeurs exactes de précision, rappel, et temps d'exécution mais les tendances des différents algorithmes et les comparaisons entre algorithme restent globalement inchangées et ne varient qu'en fonction des paramètres de génération, le niveau de bruit des *inliers* et le pourcentage d'*outliers*. Cela montre que la méthode de génération de données semi-artificielles permet bel et bien de révéler les qualités intrinsèques des algorithmes utilisés. La méthode de génération de données et le *benchmark* des algorithmes de RanSaC adaptatifs ont été publiés dans [77], qui a ensuite été étendu dans un journal [76].

La troisième contribution de cette thèse a été d'analyser l'impact de ces méthodes RanSaC adaptatives sur le logiciel ColMap et d'analyser comment améliorer ses performances en supprimant un hyper-paramètre et en gérant potentiellement des scènes plus difficiles. Actuellement, ColMap utilise LO-RanSaC [16] et RanSaC [26] comme estimateurs robustes, tous deux dépendant d'un seuil de valeurs aberrantes / valides défini par l'utilisateur qui est, par défaut, fixé à une valeur élevée. Au vu des performances sur le problème PnP des algorithmes LRT, AC-RanSaC et Fast-AC-RanSaC, nous avons testé leur impact sur la qualité, la rapidité et la robustesse de la reconstruction 3D avec le *pipeline* complet. Le *pipeline* de génération des données est adapté à ColMap, le modèle vérité terrain ainsi que les *inliers* bruités sont calculés de la même façon que pour le problème PnP pour chaque image, et pour générer les *outliers* les chaînes de correspondances entre les images sont modifiées. Pour les modifier, chaque paire d'images qui constitue la chaîne est traitée indépendamment, à l'exception des deux images qui servent à initialiser la reconstruction, dont les correspondances sont laissées telles quelles. Pour les autres paires, une des deux images est choisie et une portion de ses correspondances 2D-2D est corrompue par des correspondances 2D-3D erronées, afin de créer des *outliers*. Le bruit des *inliers* et les correspondances 2D-3D erronés sont créés de la même façon que pour le problème PnP. Après avoir évalué les algorithmes choisis sur plusieurs scènes contenant différents nombres d'image, différentes densités de prise de vue et avec une variation du niveau de bruit des *inliers* de 0 à 6 pixels par incréments de 0.5 et le pourcentage d'*outliers* de 0% à 90% par incrément de 10%, on observe que les méthodes adaptatives présentent des performances similaires à LO-RanSaC utilisé par ColMap sauf

dans les cas les plus complexes. Lorsque la qualité de la reconstruction diminue fortement, et que toutes les images n'arrivent pas à être recalées, les algorithmes adaptatifs voient leur performance diminuer moins rapidement que LO-RanSaC. En termes de temps de calcul, aucune amélioration du temps passé à effectuer les tâches autres que RanSaC n'est notée et donc le temps d'exécution global se retrouve légèrement augmenté vu que les algorithmes adaptatifs sont plus longs que LO-RanSaC. Nous avons donc réussi à retirer un hyper-paramètre de ColMap, le seuil d'acceptation de LO-RanSaC, ainsi qu'à augmenter la robustesse dans les cas les plus complexes. Néanmoins, ces résultats ont peu d'impact sur la qualité globale de la reconstruction dans la plupart des cas et il est nécessaire de faire d'autres améliorations.

La dernière contribution est une analyse détaillée des améliorations proposées par [17] pour réduire le temps d'exécution lors de l'utilisation du test du rapport de vraisemblance comme objectif pour l'estimation robuste ainsi que leur impact sur la performance de la classification, et une implémentation de LRT, qui n'était pas encore disponible publiquement, sur IPOL: [75]. L'algorithme LRT propose différentes solutions pour réduire le temps de calcul, la première et principale étant l'interruption du calcul des erreurs des points lorsque la probabilité de trouver suffisamment d'*inliers* diminue trop. Les différentes solutions sont testées en utilisant la même méthodologie que précédemment et on observe bien l'effet voulu, dans la majorité des cas le temps de calcul diminue bien et les performances sont similaires à celles sans les optimisations. Cela nous permet donc de valider précisément l'impact des améliorations proposées.

Le premier chapitre de cette thèse présente la littérature autour du pipeline générique de reconstruction 3D, ainsi que les problèmes de stéréoscopie. Il inclut une revue des travaux relatifs aux estimateurs robustes, en particulier les algorithmes inspirés de RanSaC qui sont décrits en détail car au centre de notre étude. Ce chapitre présente également de manière succincte des éléments du pipeline tels que les méthodes pour obtenir des correspondances 2D à partir d'images, ou les différents estimateurs de modèles pour les problèmes inclus dans notre benchmark.

Le deuxième chapitre présente en détail notre nouvelle méthode de génération d'ensembles de données semi-artificielles. Après avoir présenté les jeux de données existants utilisés dans la littérature, nous présentons notre méthode et la manière de l'appliquer aux différents problèmes à deux vues ainsi qu'au problème PnP. L'application de cette méthode au pipeline ColMap est laissée au chapitre 4, après une présentation détaillée du pipeline.

Le troisième chapitre présente l'analyse comparative des méthodes RanSaC adaptatives pour l'estimation d'homographie, l'estimation des matrices fondamentales et essentielles et pour le problème de la perspective à partir de n points (PnP). Il comprend une variété de méthodes RanSaC adaptatives appliquées à diverses configurations générées à partir de divers ensembles de données et d'un large éventail de paramètres de génération.

Il permet de révéler les différences de performance et d'efficacité entre les algorithmes plus clairement que les comparaisons précédentes disponibles.

Le quatrième et dernier chapitre présente notre travail d'utilisation des méthodes adaptatives dans ColMap [79]. Il détaille le pipeline générique de reconstruction 3D et les améliorations proposées par le cadre ColMap. Ensuite, l'adaptation de la méthode de génération de jeux de données au cas multi-vues est présentée et les performances des meilleurs algorithmes trouvés dans le chapitre 3 sont analysées.

Table of Contents

Abstract **1**
Short Abstract - English Version . . . 1
Résumé court - Version Française . . . 2
Résumé substantiel - Version Française . . . 3

Introduction **13**

1 Traditional Reconstruction Pipeline **19**
1.1 Notations . . . 20
1.2 Model estimators . . . 22
1.3 Feature detection and matching . . . 29
1.4 Robust fitting methods . . . 32
1.5 Bundle Adjustment . . . 57

2 Semi-Artificial Data Generator **61**
2.1 Reasoning and motivation . . . 61
2.2 Metrics . . . 65
2.3 Model and inlier generation . . . 67
2.4 Outlier generation . . . 73

3 Adaptative RanSaC Benchmark **79**
3.1 Tested algorithms . . . 79
3.2 Benchmark parameters and datasets . . . 83
3.3 Time . . . 85
3.4 Classification performance . . . 91
3.5 Validity of method and algorithm choice . . . 95

4 Application to ColMap **97**
4.1 The ColMap pipeline . . . 97
4.2 Experimental setup . . . 100
4.3 Observation . . . 105
4.4 Conclusion . . . 109

Conclusion **111**

Bibliography **121**

Introduction

3D reconstruction from images, such as Multi-view Stereo (MVS) and Structure-from-Motion (SfM), consists in recovering depth data from 2D data, usually using geometric analysis of a scene and relating images together. Using different view points a same scene, or object, the depth of each point could theoretically be retrieved as light rays reflecting on it follow a straight path from the point to each camera position and thus the point can be found at the intersection of those. This is in the case of a perfect Pinhole Model, where each point is captured by only one light ray that ends to a unique position on the camera sensor. Of course, in reality, the light path will be deformed by the lenses in front of the sensor, quantification effects will append on the pixels of the images and blur caused by motion or out of focus will appear. Solutions have been developed to take those different problems into account and nowadays 3D reconstruction has many industrial applications, from heritage conservation to robotics and self-driving vehicles. 3D reconstruction is also the basis of other tasks, like surface reconstruction, 3D object modelling, object tracking and relocalisation.

As intuitive as it seems, performing the reconstruction requires to perform multiple difficult tasks, first and foremost finding the position of the cameras around the scene to be able to trace the position of the points. This usually implies finding correspondances between images and analysing the deformation between them to find the relative positions to each other and finally evaluate the depth of the scene up to a scale. If more than two images are used for the reconstruction, adding images will add errors and will lead to drift which also needs to be taken into account. Those challenges have been mostly answered and softwares like ColMap, Bundler or VisualSFM offer off-the-shelf solutions to perform 3D reconstruction.

Most of the recent work in 3D image processing uses Neural Networks and Deep Learning instead of more traditional methods. They can be used to replace parts of the pipeline, like with learned features or pose estimation, or can be used to perform end-to-end depth estimation. Machine Learning method prove efficient in a huge variety of tasks, especially the more complex ones, like single view reconstruction where depth and unseen elements have to be reconstructed from a single view of an object, next view generation, where a novel 2D image is generated from the scene and a new camera position. These methods also have a huge impact on derived 3D problems, like 3D object modelling and reconstruction.

However, when using learning methods, the question of training data rises and reliable 3D data is hard to obtain. All datasets relie on some sort of estimation to get accurate

ground truth data, either using an active sensor like a LIDAR or a depth camera and calibrating the result of their measurements to the image set, or using a traditional pipeline estimation as ground truth. This makes benchmarking algorithms complex, as all results are compared to partially incorrect data and will introduce bias regarding how the data was obtained. Deep Learning methods perform way better than traditional methods on the most complex of tasks but regarding basic 3D reconstruction, traditional methods like ColMap are still very accurate and more robust, [21]. Improving the traditional methods will lead to more accurate training data which could help learning better models.

In light of these observations, we look at how to improve the traditional method. The usual reconstruction pipeline is made mainly of three elements, the feature detection and matching, the robust model estimation and the global optimisations. Recent work has been made on all these elements but, as said earlier, comparing new methods together is difficult and most recent publications contradict themselves on which algorithms perform better on which scenarios, mostly because they use different methods to generate their ground truths and use different empirical metrics. This is especially true in the domain of robust estimations, where plenty of novel solutions are still produced and supposedly beat the previous state-of-the-art but are not integrated in software solutions like ColMap.

Most recent improvements around Robust Estimators focus on adaptative methods. A Robust Estimator evaluates a model in the presence of outlier data that prevents the use of a global estimator. Those estimators usually provide at the same time a model estimated only on inlier data and a classification of data point as inlier or outliers depending on their distance to the model and a user set acceptance threshold. The quality of a model will depend on the size of its consensus set, the set of inliers according to the threshold. Defining an appropriate value for this threshold requires knowledge about the data that is not available *a priori* and will either imply iterative research of parameter or using conservative estimates that can lead to poor performance. To avoid these problems, adaptative methods that estimate the inlier/outlier threshold and the model simultaneously have been developed. These methods require a different objective function that will balance the number of inliers with the quality of the model as the classic size of consensus set cannot be used when the threshold varies. We focus our study on those methods and how to use them to improve the classical pipeline.

To efficiently analyse the different methods available in the literature we create a novel data generation method. This is a semi-artificial method that uses real data to select a model and guide the generation of controlled inliers to avoid the shortcomings of usual benchmarking methods. Fully artificial methods require a choice on the distribution of model, inlier points and outliers points, which can have a huge impact on the performance of algorithms as most adaptative methods have to make assumptions about the distribution of data to compute an objective function different from the size of the consensus set. Artificial data also raises the question of portability of the results to real world set-

tings. On the other hand, as outlined earlier, all real world dataset introduce some sort of inaccuracy, whether they use active sensors or not, as some algorithms have to be run and estimations have to be made to create the dataset and thus uncertainty is introduced in the result. Most comparison made using such data use a lot of qualitative analysis and quantitative results focus on reaching a baseline quality. Using in-the-wild data also prevents categorisation of scenes by difficulty as there is no objective measurement of the challenge presented, it can only be analysed *a posteriori* by looking at which setup proved more difficult to handle for the tested algorithms. This is why our solution guides the generation method using real life estimation, while still creating an artificial ground truth with no uncertainty about inliers and outliers. We can then analyse performance of adaptative methods by varying the base dataset, the inlier noise and outlier ratio to create settings of various difficulty and reveal the intrinsic behaviour of algorithms if tendencies emerge in all tested situations. We first perform a benchmark of adaptative methods on two-view tasks such as homography estimation, fundamental matrix estimation and essential matrix estimation as well as the PnP problem. Using this analysis we provide insight on adaptative methods and their performance to leverage the best proposed improvements in a complete reconstruction pipeline. We chose the best performing adaptative algorithms on the PnP task and analyse their impact on the whole pipeline by replacing the classical RanSaC of the PnP task in the ColMap software. Our goal is to improve the robustness of ColMap by removing an hyperparameter and thus removing the need to tailor its value to the tasks at hand, and potentially, thanks to better RanSaC algorithms, handle more challenging scenes that ColMap still fails to handle properly.

Contributions

The first contribution we make is a new benchmarking method that relies on semi-artificial data. In order to have reliable measures of robust estimation algorithms performance while still having realistic data which can be extended to real world setups we developed a novel data generation method. This method relies on the use of a corrected estimation of an in-the-wild scene to create inliers and outliers that can be trusted but still based on real elements. It also proposes a new approach to generate outliers that are more challenging to a Robust Estimator.

Using this benchmarking method, we evaluate the different adaptative threshold solutions that remove the need for a hard user-set inlier/outlier threshold. This benchmark allows to reveal behaviour and runtime of algorithms depending on noise, outlier levels with greater confidence than earlier. This allows to determine which algorithms works best in which scenarios and what are the performances cut-offs, where an algorithm will start to perform worst than baseline RanSaC. Both those contributions have been published in [77] which was then extended in a journal has yet to be released.

The third contribution of this thesis was to analyse the impact of those adaptative

RanSaC methods to the ColMap software and analyse how to improve its performance by removing an hyperparameter and potentially handle more challenging scenes. Currently, ColMap uses LO-RanSaC [16] and RanSaC [26] as its Robust Estimators, both depending on a user set inlier/outlier threshold that is, by default, set to a high value.

The final contribution is a detailed analysis of improvement proposed by [17] to reduce runtime when using the likelihood ratio test as an objective for robust estimation and their impact on classification performance, as well as an implementation for LRT, which was not yet available publicly, published on IPOL: [75].

Manuscript Outline

The first chapter of this thesis presents literature around the generic 3D reconstruction pipeline, as well as two-view stereo problems. It includes a review of related work in Robust Estimator, especially RanSaC-inspired algorithms which are described in detail as they are the main focus of our study. This chapter also presents succinctly over elements of the pipeline such as methods to obtain 2D features from images, that is detectors, descriptors and matchers, or the various model estimators for the problems included in our benchmark.

The second chapter presents our novel semi-artificial dataset generation method in details. After presenting the existing datasets used in the literature, we present our method and how to apply it to the various two-view problem as well as the PnP problem. The application of this method to the ColMap pipeline is left to chapter 4, after a detailed presentation of the pipeline.

The third chapter presents the benchmark and analysis of the Adaptive RanSaC methods for homography estimation, fundamental and essential matrix estimation and for the Perspective-from-n-Points (PnP) problem. It includes a variety of adaptative RanSaC methods applied to various setups generated from various datasets and a wide range of generation parameters. It helps to reveal performance and efficiency differences between algorithms clearer than previous available comparisons.

The fourth and final chapter presents our work using Adaptative methods in ColMap [79]. It details the generic 3D-reconstruction pipeline and the improvements proposed by the ColMap framework. Then the adaptation of the dataset generation method to the multi-view case is presented and the performances of the best algorithms found in chapter 3 are analysed.

Associated publications

This thesis lead to the following publications. First a publication in the IPOL journal: [75] which proposes a detailed analysis of the method proposed by [17] with a closer look on the impact of early bailout and other optimisation strategies. This publication

includes an implementation of the method, in C++, as it was not yet available with a demo made available at `https://www.ipol.im/pub/art/2022/357/`. The second publication, in the VISAPP conference, [77] presents our novel data generation method and uses it to benchmark various adaptative RanSaC algorithms in a unified framework. It focuses on two-view stereo tasks, homography estimation, fundamental matrix estimation and essential matrix estimation, using RanSaC [26], AC-RanSaC [62, 61, 60] and Fast-AC-RanSaC, LRT [17], MUSE [58] and MAGSAC [6]. This publication was extended for a special issue journal with increased details about the generation method as well as an extended benchmark, adding MAGSAC++ [7] and a better version of Fast-AC-RanSaC from [66]. The extension also includes the PnP task in the benchmark which offers different behaviours than two-view geometry problems. This paper was published in [76]. Finally, we make available the C++ code used to generate data and evaluate robust estimators, including all adaptative RanSaC implementations that were used in the experiments in a unified pipeline for ease of comparison and reuse, at `https://github.com/ClementRiu/AdaptativeRanSaCBenchmark.git`.

Chapter 1

Related Work and Presentation of the Traditional Reconstruction Pipeline

A Multi-View Stereo (MVS) or Structure-from-Motion (SfM) pipeline requires multiple steps which are all studied extensively, independently or jointly. The usual pipeline uses the following steps:

1. Get features from the input data, like an image pair, using a feature detector, descriptor, and matcher, see Section 1.3.
2. Use a Robust Estimator to estimate models from the features, see Section 1.4.
3. To estimate such a model, use an *ad-hoc* estimator, see Section 1.2.
4. Perform those tasks multiple times to add the multiple views and perform a global adjustment of parameters, see Section 1.5.

How all these elements are linked together will be described in more details in Chapter 4. Most of the algorithms and methods described here can have other applications than image 3D-reconstruction but we focus on this aspect specifically here. Feature detector and matches, model estimators and bundle adjustment methods are presented succinctly here in order to give a good understanding of the data and questions handled in the rest of the document. Robust estimators and most importantly Random Sample Consensus (RanSaC) algorithms are the focus of most of the work presented here and so are detailed the most extensively.

A lot of the recent efforts around 3D reconstruction are using machine learning methods and specifically neural networks, to perform different 3D tasks, like multi-view stereo reconstruction [22, 86], 3D surface generation [11, 86] novel point of view generation [93] and many more. In most fields state-of-the-art solutions are based on neural networks. However, these methods require training data which usually comes from more traditional 3D reconstruction solutions but not from an accurate 3D measure. For example, one of the most used dataset is MegaDepth [46], whose 3D ground truth is generated using ColMap [79] a state-of-the-art software for multi-view stereo which still gives competitive

Table 1.1: Notations used throughout this thesis.

	Definition	Description
$\|x\|$	$x \in 2^{\mathcal{S}}$	cardinal of set x
$\|x\|$	$x \in \mathbb{R}^{m \times m}$	determinant of matrix x
$\|x\|$	$x \in \mathbb{R}^m$	norm of vector x
$[n]$	$n \in \mathbb{N}$	interval of integers from 1 to n included
k	$\in \mathbb{N}^*$	dimension of data points
$\mathcal{S}$	$= \mathbb{R}^k$	space of data points
$\mathcal{P}$	$\subset \mathcal{S}$	set of input data points
p	$\in \mathcal{P}$	a data point
ϵ	$\in [0, 1]$	ratio of inliers
d	$\in \mathbb{N}^*$	degrees of freedom of a model
Θ	$\subset \mathbb{R}^d$	space of model parameters
θ	$\in \Theta$	parameter vector of a model
s	$\in \mathbb{N}^*$	data sample size
Sa	$: \Omega \to \mathcal{S}^s$	random sampling function
$\mathcal{F}$	$: \mathcal{S}^s \to 2^\Theta$	fitting function
U_s	$: [0, 1] \to [0, 1]$	proba. of sampling inliers only
D	$: \mathcal{S} \times \Theta \to \mathbb{R}$	point-model residual function
σ	$\in \mathbb{R}$	inlier/outlier threshold or noise level
I	$: \Theta \times \mathbb{R} \to 2^\mathcal{P}$	inlier selector function
$\mathcal{I}$	$= I(\theta, \sigma) \subset \mathcal{P}$	set of model inliers
Q	$: 2^\mathcal{P} \times \Theta \to \mathbb{R}$	model quality function
X_{gt}		variable X is relative to ground truth data
X^*		variable X is relative to the best model estimated

result in 3D reconstruction [21]. Other famous datasets for training data like Phototourism and Building Rome in a Day [81, 2] use classic RanSaC procedures to generate their data, or HPatches [5] which uses classical Difference of Gaussian, Hessian and Harris detectors — see section 1.3 — to detect its patches. Others will use ColMap as a reference to beat or in their training process directly [22, 86, 93]. Despite all recent efforts in machine learning solutions we believe it is still relevant to revisit classical methods and how to improve them and offer both a better solution for industry and as a training tool, so that neural networks can learn with better data.

1.1 Notations

The notations we use are summarised in Table 1.1. The notation $|x|$ can be the cardinal of a set, the determinant of a matrix or the norm of a vector, depending on the type of x. $[n]$ if n is an integer, is the interval of non negative integers less or equal to n.

The entry data of RanSaC algorithms are data points of dimension k. It can be $k = 3$ when fitting a plane in 3D space for example but the problems we consider in SfM and

MVS require matches between two images, with $k = 4$[1] or for the PnP problem matches between putative 3D points and an image, with $k = 5$[2]. The set of data points p is $\mathcal{P}$ a subset of $\mathcal{S} = \mathbb{R}^k$ the ambiant space of the inputs.

The main interest of RanSaC algorithms is their capability to handle corrupted data. Thus, usually $\mathcal{P}$ includes both inliers and outliers. Inliers are points that are actually originating from the true model but that might not perfectly fit due to noise. The true model being the actual geometric relation between the images or data points. Outliers are points that have no relation to the model and can be the product from issues with data acquisition or issues with the algorithms generating the matches. The true ratio of inliers in the dataset is noted $\epsilon_{gt} \in [0, 1]$. Usually, this ratio is unknown, and ϵ will generally denote the inlier ratio of a given estimated model.

The underlying model is also unknown and models estimated through the pipeline are parametrised in the parameter space $\Theta \subset \mathbb{R}^d$ where d is the degree of freedom. $d = 8$ for homographies, 7 for fundamental matrices because of rank 2 constraint, 5 for essential matrices, and 6 for PnP. Models are estimated using an *ad hoc* fitting function, or estimator, $\mathcal{F}$. This function can yield more than one model depending on the algorithm, see Section 1.2. Most RanSaC algorithm involve two estimators: one using the minimal number of data points necessary to estimate a model and one using all available data — usually a least square estimation. This second algorithm will be noted $\mathcal{F}_{full}$.

To estimate a model, usually a sample $\mathcal{P}_s$ is drawn from $\mathcal{P}^s$, where s is the number of points used to estimate the model. This sample is obtained with a sampler Sa. Most algorithms use a uniform sampler, which gives probability $U_s(\epsilon) = \epsilon^s$ to draw an uncontaminated sample[3]; however some algorithms presented in this chapter use a non-uniform sampler and try to improve the sampling strategy. s is usually set to the lowest possible number of samples required to estimate a model in order to increase the aforementioned probability of drawing an outlier-free sample.

Once a model θ is computed, it is possible to evaluate its quality using the function Q. This function usually depends on the selected inliers of the model $\mathcal{I} = I(\theta)$, where I is the selector function. The classic selector of RanSaC $I(\theta, \sigma) = \{p \in \mathcal{P} : D(p, \theta) \leq \sigma\}$ depends on an inlier/outlier threshold σ and the computation of all residuals of data points $p \in \mathcal{P}$ with error function $D(p, \theta)$. This error function is usually a reprojection error, that is the distance between point and the projection of its matching point. Other errors computation are possible like the Sampson error which approximates the Gold Standard error between the noisy inliers and the supposed ground truth inlier. For standard RanSaC, the quality function $Q(\mathcal{I}, \theta) = |\mathcal{I}|$ is the number of inliers. Symbol σ might also be used to represent

[1] Two 2D matches.

[2] One 2D match and one 3D match.

[3] A sample is said "uncontaminated" when it is composed exclusively of inliers. This probability is not exact as we do sampling without replacement but the number of point is usually big enough to use this definition.

the noise level of inliers, which ideally would be the same as the inlier/outlier threshold. The noise existing on inliers can have multiple sources, and be modelled through different distributions. The most common source is acquisition or a previous algorithm imprecision. Not all RanSaC methods need a hypothesis on the distribution but the most common ones are Gaussian and uniform.

Among those different quantities, D, k, Θ and d vary with the model while $\mathcal{F}$, Sa, Q, I vary with the chosen algorithm. For example algorithms that would estimate σ and θ simultaneously cannot use the same quality function Q as standard RanSaC where the number of inliers increases with σ. Another usual parameter of the presented algorithms are the confidence they have regarding some errors, like the type II error—the risk of missing a valid solution because the algorithm ends to early, called premature termination—for classic RanSaC.

We denote with subscripts gt a quantity relative to ground truth data, for example the ground truth model θ_{gt} and with superscript $*$ quantities relative to the best model estimated by an algorithm, like θ^* for best model parameters.

Most functions and variables depend on a lot of elements, for example, the quality function Q of an algorithm might usually depend on the dimension of the data k, the dataset $\mathcal{P}$, the computed threshold σ^*, the resulting model θ^*, the inlier set $\mathcal{I}(\theta, \sigma)$ and so on. For the sake of simplicity of notations, throughout this thesis, dependence of a quantity to other parameters will only be marked when it is most relevant.

1.2 Model estimators

RanSaC algorithms can be used for many different problems and most different variations of the RanSaC algorithm detailed in Section 1.4 just require two estimators, one with as small a sample as possible and one with all available data points. The scope of this thesis is Multi-View Stereo (MVS) and Structure-from-Motion (SfM) and specifically homography estimation, fundamental and essential matrix estimation and the Perspective-from-n-Points (PnP) problem. Only these problems will be described in this section. Detailed explanation about multi-view problems can be found in [33].

The input data of the model estimators described in this section are 2D points matching between two images for two view geometry problems and 2D points from an image to 3D points correspondences for the PnP problem. The 2D points have coordinates in the image which are usually extracted using some algorithm described in Section 1.3. They are usually extracted after some processing of the image to remove distorsion and other effects from the camera to get closer to the ideal pinhole model. These processes are not described here but they are extensively documented in different software frameworks like OpenCV [9] and ColMap [79]. The pinhole model is an ideal model of a camera that represents the camera aperture as a single point through which light will make a perfect,

non blurry image on the image plane. This model implies that each world point visible to the camera results in one precise point on the image which can then be used to compute information about the camera setup. Most of the following computations are based around this model which enables us to use decimal value for the position of points in pixel space while still making sense in a geometric point of view.

All cameras have intrinsic parameters which are summed up in a calibration matrix K. Those include the focal length of the camera, the principal point of the camera and a potential skew s even though this last parameter is almost always zero. The focal length f can either be the same for both direction or different along x and y axis, noted then f_x and f_y. The principal point coordinates are written, in pixel coordinates, c_x and c_y. They are the entries in K such that:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.1}$$

The intrinsic parameters of a camera can be found using a calibration technique. Many different methods exist for this [33] but are not described here.

Points can be either expressed in image coordinates, in number of pixel with origin (c_x, c_y) or be expressed in camera coordinates, expressed in external unit. When it matters it would be differentiated by the subscript im or cam and for a point m: $m_{im} = Km_{cam}$ in homogenous coordinates. In the following, unless specified, all pairs of images need not be from cameras with the same intrinsic parameters.

1.2.1 Homography estimation

An homography results when two pictures are taken of a planar scene from different viewpoints or from a non-planar scene with only a rotation around the optical center of the camera. This transformation H conserves alignments and elements of a match $p = (m, m')$ are related by: $m' = Hm$. For a planar scene we have:

$$H = K \begin{pmatrix} R_1 & R_2 & T \end{pmatrix}, \tag{1.2}$$

and for a pure rotation and potential change of intrinsic parameters:

$$H = K' R K^{-1}, \tag{1.3}$$

where $R = \begin{pmatrix} R_1 & R_2 & R_3 \end{pmatrix}$ is the rotation of the camera—where R_1, R_2 and R_3 are the columns of R—and T is the translation between optical centres. This matrix H is invertible. Homography can be used to stitch images together and create mosaics for example, see Figure 1.1.

Figure 1.1: Two images acquired from the same viewpoint are related by an homography (left and right) and the stitched image resulting from overlapping the transformed left image on the right one (center). Images from [60].

To be precise, in pixel coordinates we have the relation:

$$\lambda m' = Hm, \tag{1.4}$$

where λ is a unknown scale parameter. This yields:

$$m' \times (Hm) = 0, \tag{1.5}$$

where $\times$ is the vector product. If we write $H = \begin{pmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{pmatrix}$ and $m = (x, y, 1)^T$, $m' = (x', y', 1)^T$ we have:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \times \begin{pmatrix} H_{1,1}x + H_{1,2}y + H_{1,3} \\ H_{2,1}x + H_{2,2}y + H_{2,3} \\ H_{3,1}x + H_{3,2}y + H_{3,3} \end{pmatrix} = 0 \tag{1.6}$$

$$\begin{pmatrix} y'(H_{3,1}x + H_{3,2}y + H_{3,3}) - (H_{2,1}x + H_{2,2}y + H_{2,3}) \\ (H_{1,1}x + H_{1,2}y + H_{1,3}) - x'(H_{3,1}x + H_{3,2}y + H_{3,3}) \\ x'(H_{2,1}x + H_{2,2}y + H_{2,3}) - y'(H_{1,1}x + H_{1,2}y + H_{1,3}) \end{pmatrix} = 0, \tag{1.7}$$

which means, if we write this as $A_p h$ with h the vector of the elements of H:

$$A_p = \begin{pmatrix} 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \\ x & y & 1 & 0 & 0 & 0 & -x'x & -x'y - x' & \\ -y'x & -y'y & -y' & x'x & x'y & x' & 0 & 0 & 0 \end{pmatrix}. \tag{1.8}$$

This means each match p yields two independent equations, as the last line of A_p is a linear combination of the two first ones. We can either suppose $H_{3,3} = 1$ of fix $|h| = 1$, with h the vector of elements of H, and this leaves 8 values to find so with 4 matches p_1, p_2, p_3, p_4

we get 8 equations and can find H as solution of $Ah = 0$ where $A = (A_{p_1}, A_{p_2}, A_{p_3}, A_{p_4})^T$. This is the 4-points algorithm as only 4 data points are required to find an homography. The associated non-minimal solver still uses equation (1.8) and tries to minimise algebraic or Sampson error by finding H as close as possible to the kernel of A where A has twice as many lines as there are data points.

1.2.2 Epipolar Geometry

When the movement of the camera between two pictures of a non-specific scene, that is not planar, includes at least a translation, the resulting images are not linked by an homography. Instead, the images are linked by a rotation R and a translation T.

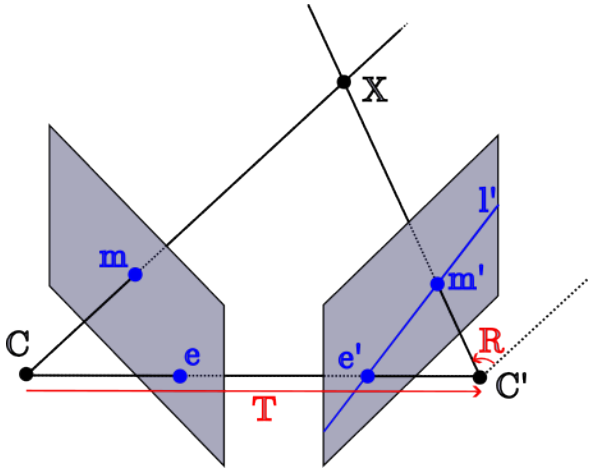


Figure 1.2: Two images defined by optical center C and C', differentiated by translation T and rotation R, observing a same point X. The light rays from this point create the match $p = (m, m')$. The epipole e and e' are, respectively, the image of C' and C in the opposite image. The epipolar line l' associated to m is the line of all possible positions of X in the right image.

Given the two optical centres C and C' the light rays originating from an element and creating a match $p = (m, m')$ leads to the three vectors Cm, $C'm'$ and T being coplanar, see figure 1.2, and thus the determinant:

$$|m \quad T \quad Rm'| = 0, \tag{1.9}$$

which can be re-written as $m^T E m' = 0$ with $E = T \times R$, with $\times$ the cross product of T with each column of R, the vector product of the translation and the rotation which we call the Essential Matrix, it was introduced in [51]. A matrix is an essential matrix if, and only if, its singular values are 0 and 2 equal positive ones. The later condition can be expressed as: $2EE^TE - tr(EE^T)E = 0$ provided the first condition is satisfied. The essential matrix can be computed if we have the points expressed in camera coordinate

frame:

$$m_{cam}^T E m'_{cam} = 0, \tag{1.10}$$

which requires knowing the calibration matrix of the two images. If they are unknown, we compute the Fundamental matrix, introduced in [24] defined a $F = K^{-T} E K'^{-1}$:

$$m_{im}^T F m'_{im} = 0. \tag{1.11}$$

A matrix is a fundamental matrix if, and only if, it is rank 2.

The projection of the left optical centre in the right image is the left epipole which satisfies $e^T F = 0$ and $e = KT$ and the projection of the right optical centre in the left image is the right epipole: $Fe' = 0$ and $e' = K'R^{-1}T$. Each point m on the left image has an associated line $F^T m$ which is the epipolar line of m in the right image, and reciprocally for a point m' in the right image, the line Fm' is the epipolar line associated to m' in the left image. For a given point, all points along its associated epipolar line can thus be matches.

Given equations (1.9) and (1.11) if we write $F = \begin{pmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \\ F_{3,1} & F_{3,2} & F_{3,3} \end{pmatrix}$ and $m = (x, y, 1)^T$, $m' = (x', y', 1)^T$, using F as an example, we have:

$$m_{im}^T F m'_{im} = F_{1,1}xx' + F_{1,2}xy' + F_{1,3}x + F_{2,1}yx' + F_{2,2}yy' + F_{2,3}y + F_{3,1}x + F_{3,2}y + F_{3,3}, \tag{1.12}$$

which yields one equation using all 9 parameters of F. However, as F is defined up to a scale, only 8 parameters are necessary to know F and thus, with 8 matches or more F can be computed. This is the 8-point algorithm. Equation (1.12) defines the lines of a matrix A so that $A_m^T f = 0$ with f the line vector of elements of F. To include the indifferent scale, f has to be of norm 1 and we get f an eigenvector of $A^T A$. Then, to ensure the rank 2 constraint, the fundamental matrix is decomposed using an SVD and the smallest eigenvalue is set to 0 before recomposing the fundamental matrix. However, equation (1.12) clearly shows difference of scale for each parameter in A, as $F_{1,1}$, $F_{1,2}$, $F_{2,1}$, $F_{2,2}$ are multiplied by two terms in pixel coordinates so in the 10^0 to 10^3 range, while $F_{1,3}$, $F_{2,3}$, $F_{3,1}$, $F_{3,2}$ are multiplied by one term in this range and $F_{3,3}$ is not multiplied. This creates a badly conditioned matrix A and leads to instability. However, by transforming the input, as proposed in [34], with a matrix $N = \begin{pmatrix} 10^{-3} & 0 & 0 \\ 0 & 10^{-3} & 0 \\ 0 & 0 & 1 \end{pmatrix}$ as $\tilde{m} = Nm$ and $\tilde{m'} = Nm'$ we find $\tilde{F}$ which leads to $F = N^{-1}\tilde{F}N$ and reduce instability. To ensure the constraints on the essential matrix, we decompose it, set the first two eigenvalues to 1 and the last one to 0, and then recompose it. As the scale is not important setting the non-zero eigenvalues to 1 is not a problem.

As presented in Section 1.4, if it is possible to use less points to compute a model, it is preferable for the efficiency of the RanSaC procedure. A 7-point procedure can be devised by using the same equation (1.12) to construct a 7×9 A matrix and find solutions of $Af = 0$. Using an SVD, two independent vectors $\tilde{F}_1$ and $\tilde{F}_2$ of the kernel of A are used to form a polynomial of degree 3 $|F_1 + xF_2|$ where F_1 and F_2 are the 3×3 matrix of the elements of $\tilde{F}_1$ and $\tilde{F}_2$ respectively. Then, by using the fact that the determinant of F is null, we solve:

$$|F_1 + xF_2| = 0 \tag{1.13}$$

for x which gives 1 or 3 solutions and all can be tested to find the best one, see Section 1.4.

For essential matrix it is possible to derive a method using only 5 points, as it only depends on 5 parameters — 3 for rotations, 3 for translations and 1 less for scale. The 5-point method was introduced in [69]. Given 5 matches with equation (1.9), a 5×9 matrix A is constructed and four vectors $\tilde{X}, \tilde{Y}, \tilde{Z}, \tilde{W}$ of its null space can be found and written as 3×3 matrices X, Y, Z, W. Then we search the essential matrix as $E = xX + yY + zZ + wW$ with x, y, z, w scalars. As E can be estimated up to as scale we define $w = 1$. Then, using the fact that $|E| = 0$ and $2EE^TE - tr(EE^T)E = 0$ we derive a polynomial system with all monomials of degree at most 3: $x^3, x^2, x, y^3, y^2, y, z^3, z^2, z, x^2y, xy^2, x^2z, y^2z, xz^2, yz^2, xyz, xy, xz, yz$. By combining the lines of the system a tenth degree single variable polynomial can be obtained and its roots will yield up to 10 solutions. See the original paper for details of the methods and efficiency considerations.

1.2.3 Perspective-from-n-Points

Contrary to previous presented problems, the Perspective-from-n-Points (PnP) problem is not a two-view geometry problem. Its goal is to determine the position and orientation of a calibrated camera given a set of n 3D-2D correspondences. It can be used in many applications, particularly in multi-view stereo reconstruction when adding new images to an existing reconstruction. In order to do so, the 3D points in world coordinates must be expressed in camera coordinates which will lead to finding the rotation and translation associated with the camera. This final step can be done using an analysis of the covariance matrix between the two coordinates systems [92].

There exists a quantity of options to solve this problem, including iterative [54, 1] and non-iterative ones. Some only work for a specific value of n like P3P [23, 27, 28, 71, 25] or more — P4P, P5P [36, 71, 27, 90]. Others will work for any value of n like [71, 43, 23, 4]. In the following, we only describe succinctly two solutions to this problem, the P3P case and the EPNP [43] algorithm as they are used respectively as minimal and non-minimal ones in RanSaC implementations, see 1.4.

The most studied element is the minimal case $n = 3$, called P3P, dating back to 1841

according to [30] and improved many times over in multiple papers [23, 27, 28, 71, 25]. With only three points, up to four solutions are possible and how to choose between them is described in the robust estimator Section 1.4. The basic method to solve this problem is based around geometric conditions. Given c the center of projection of the camera and the three data points $\{p_i = (m_i, m_i^p), i \in [3]\}$, with m_i a 3D world point and m_i^p its projection in the image, we can write the distance between two world points, for any two points i and j:

$$\mathcal{D}(m_i, m_j)^2 = \mathcal{D}(c, m_i)^2 + \mathcal{D}(c, m_j)^2 - 2\mathcal{D}(c, m_i)\mathcal{D}(c, m_j)\cos(\mathcal{A}(m_i c m_j)) \tag{1.14}$$

where $\mathcal{D}$ is the distance between two world points and $\mathcal{A}$ is the angle between three world points, which is the same as the one using m_i^p. Stacking equation (1.14) for all three possible pair of points, yields the following system:

$$\begin{cases} \mathcal{D}(c, m_1)^2 + \mathcal{D}(c, m_2)^2 - 2\mathcal{D}(c, m_1)\mathcal{D}(c, m_2)\cos(\mathcal{A}(m_1 c m_2)) - \mathcal{D}(m_1, m_2)^2 &= 0 \\ \mathcal{D}(c, m_1)^2 + \mathcal{D}(c, m_3)^2 - 2\mathcal{D}(c, m_1)\mathcal{D}(c, m_3)\cos(\mathcal{A}(m_1 c m_3)) - \mathcal{D}(m_1, m_3)^2 &= 0 \\ \mathcal{D}(c, m_2)^2 + \mathcal{D}(c, m_3)^2 - 2\mathcal{D}(c, m_2)\mathcal{D}(c, m_3)\cos(\mathcal{A}(m_2 c m_3)) - \mathcal{D}(m_2, m_3)^2 &= 0 \end{cases}, \tag{1.15}$$

which can be solved by substituting $\mathcal{D}(c, m_2)$ and $\mathcal{D}(c, m_3)$ judiciously and finding a fourth degree polynomial in $\mathcal{D}(c, m_1)^2$ in [71] or in $\frac{\mathcal{D}(c,m_2)}{\mathcal{D}(c,m_1)}$ in [27]. With some additional constrains it is ensured that at most four solutions for $\mathcal{D}(c, m_1), \mathcal{D}(c, m_2), \mathcal{D}(c, m_3)$ are possible from the root of this polynomial.

EPnP [43] and its variants are the most used solutions currently. Its approach is based around rewriting the 3D reference points in a new coordinate system based around four — or three for planar setups — virtual control points. It then solves for the coordinates of those control points in camera coordinate system instead of n depths, which will lead to a solution in the kernel of a matrix $\mathbf{M}$ of size $2n \times 12$ — or $2n \times 9$ in the planar case. Given the four control points $\{c_j, j \in [4]\}$, each point $\{p_i = (m_i, m_i^p), i \in [n]\}$ can have its 3D part m_i be expressed as —in either camera or world coordinates, which we will denote with respectively ^{c} and ^{w}:

$$m_i = \sum_{j=1}^{4} \alpha_{ij} c_j, \tag{1.16}$$

where α_{ij} are the homogenous barycentric coordinates. The authors propose to choose the control point as the centre of the n points and using the principal directions of the data to choose the 3 remaining points and form a basis. For the 2D projections m_i^p we have:

$$w_i \begin{pmatrix} m_i^p \\ 1 \end{pmatrix} = K m_i^c = K \sum_{j=1}^{4} \alpha_{ij} c_j^c, \tag{1.17}$$

with K the calibration matrix and w_i the projective parameter. Writing $c_j^c = (x_j^c, y_j^c, z_j^c)^T$ and $m_i^p = (x_i^p, y_i^p)^T$ we have a system:

$$w_i \begin{pmatrix} x_i^p \\ y_i^p \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix}, \tag{1.18}$$

with the unknown being the projective parameters and the camera coordinates of the control points. The last row yields $w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$ and, when substituted in the first two equations:

$$\sum_{j=1}^{4} \alpha_{ij} f_x x_j^c + \alpha_{ij} s y_j^c + \alpha_{ij}(c_x - x_i^p) z_j^c = 0 \tag{1.19}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_y y_j^c + \alpha_{ij}(c_y - y_i^p) z_j^c = 0. \tag{1.20}$$

Stacking equations (1.19) and (1.20) for all n points we get a system $\mathbf{M}X = 0$ of size $2n \times 12$ in $X = (c_j^T, j \in [4])^T$the 12 camera coordinates of the control points, where the projection parameters do not appear anymore (and will be computed later). The solution then lies within the kernel of M as a linear combination of the singular vectors associated with the null singular values. At least 6 points are needed and with a perfect camera model the null space should only have dimension one. However, in practice, and with more points or with more realistic camera models, the dimension of the null space can be bigger, up to 4 for the four control points or even bigger because of the noise on data points. The computation of the parameters of the linear combination are based around the distance between control points in camera versus world coordinates and small quadratic equations can be derived to find the best parameters.

1.3 Feature detection and matching

In most two view geometry pipelines, the data points are matched between two images. In order to obtain these matches, two problems need to be answered: finding a way to detect and describe points of interest and then matching them together. An element of an image that can matched is called a feature. The two kinds of algorithms, detectors and matchers, can be independent or a single one that will perform both tasks. The matches are usually represented simply by the two 2D coordinates of the points in the two images but it can also have attached information about the points, like their pixel greyscale or RGB value or the description of the points, depending on the usage. Indeed besides two-view geometry that mostly relies on geometry — but not only, see PROSAC in subsection 1.4.2 — other applications need to have information about the matches like

image retrieval or object detection which use additional information to create some sort of identification of the element to retrieve. A good detector and descriptor should be robust to a wide variety of transformations in the pictures, both local and global. For example, scale can be modified and a feature can span across a few pixels in an image and across tens in the other. Orientation, illumination, and intrinsic parameters changes can also have an impact. Issues that can impact mostly the matcher are partial occlusion, when an object of an image is hidden in the other one, non-rigid scene, when something in the scene moved in-between views and repetition, when an object can be present multiple times, like the windows of a building or bricks. A few classical detectors and matchers are quickly presented here to give a better understanding on what features will be used in most of the rest of this thesis.

A natural idea when designing a detector is to use edges and corners, which is what the Harris corner detector does [31]. This descriptor works in greyscale, where an edge is defined as a rapid change in brightness, and a corner the intersection of two edges. To detect those, the spatial derivative of the image is computed along both axis, I_x and I_y, which gives a good approximation of the brightness changes. Then, the auto-correlation matrix $M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$ is computed and smoothed with a Gaussian convolution. The response of the resulting matrix $|M| - ktr(M)$, with k an empirical constant, will be selected if above a certain threshold. If the selected response is a local maxima in each direction it is a corner, if only in one direction, an edge.

To remove the issues of scale, blob detection like the Laplacian of Gaussian (LoG) [50, 47, 49] works in the scale space representation. The image is convolved by a Gaussian kernel $g(x, y, t) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2t}}$ and the Laplacian of the resulting scale space representation $L(x, y; t) = g(x, y, t) * I(x, y)$ can be used to detect blobs, which correspond to extrema of radius $\sqrt{2t}$. To remove the dependency in scale, the Laplacian is computed as: $t(L_{xx}+L_{yy})$ and extrema are detected. Other methods exists, like the Determinant of the Hessian (DoH) [49] or the Difference of Gaussian (DoG) [48] which use similar solutions with different operators.

Detecting objects of interest (blobs, points, lines, ...) in an image is not enough as they need to be matched between images. In order to do so, features have to be described, using a descriptor. A variety of algorithms exist for this, that can tackle local or generic features. Histogram of Oriented Gradient (HOG) [19] for example works by comparing gradient value and orientation between interest points. The gradient of the image is decomposed in magnitude and angle and it can be compared. Binary Robust Independent Elementary Features (BRIEF) [10] is based around comparing the intensity of points in a patch. The descriptor is built as a bitstring of dimension 128, 256 or 512, where each component's value depends on a pair of locations (p_i, p_j) and whether intensity in p_i is lower than in p_j. To match patches together, it is also required to have some sort

of similarity measure. For example the Sum of Squared Differences, which for a patch of size n in both images will compute the difference between corresponding pixel positions: $SSD(I_1(x_1, y_1), I_2(x_2, y_2), n) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} (I_1(x_1 + i, y_1 + j) - I_2(x_2 + i, y_2 + j))^2$. The Zero-mean Normalised Cross-Correlation works similarly but with normalised images[4] and replacing the squared difference by a product.

These are historic methods but for Multi-View Stereo and Structure-from-Motion the most commonly used algorithms do detection, description and matching altogether. The most used one is Scale-Invariant Feature Transform (SIFT) [52, 53]. This algorithm proposes both a powerful detector and descriptor as well as a matching procedure. To remove the impact of scale, this algorithm works in scale space. The image I is convolved successively by a Gaussian kernel with standard deviation σ, $G(x, y, \sigma)$, $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ and for different resolutions. To get features of scale σ, the difference of Gaussian is used as $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$ with k a preset parameter. Getting local extrema of D along both space and scale will yield points of interest at different scales. To increase the quality of the features their coordinates are interpolated using Taylor expansion to the quadratic terms at perturbation δ: $D(p + \delta) = D(p) + \frac{\partial D^T}{\partial \delta}(p)\delta + \frac{1}{2}\delta^T \frac{\partial^2 D}{\partial \delta^2}(p)\delta$ which, when looking for extrema, by setting its derivative to zero, will give $\delta = -\frac{\partial^2 D}{\partial \delta^2}^{-1}(p)\frac{\partial D}{\partial \delta}(p)$. This δ is an offset that, if superior to 0.5 in at least one of the dimensions means the feature point is one of the neighbours and this new feature is taken into account. If less than 0.5 the location of the feature is shifted by this δ to get a subpixel location. The value of $|D(p + \delta)|$ is then thresholded to remove low contrast features: all features such that $|D(p+\delta)| < 0.03$ are discarded. To avoid instability along edges, where a feature could easily shift position alongside an edge of the image, where all points have similar behaviour, points are filtered using a process similar to Harris Edge and Corner detectors [31]. The Hessian matrix $H = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$ is computed and the ratio of its eigenvalues is examined, as $r = \frac{tr(H)^2}{|H|}$ which needs to be lower than a given threshold. To remove sensibility to rotation, an orientation is attributed to each feature. The magnitude $m(x, y)$ and angle $\theta(x, y)$ of $L(x, y, \sigma)$ is computed in a neighbourhood of an interest point and the histogram of all pixel angles is computed by using magnitude and a Gaussian-weighted window as weights for each angle. All angles with weight 80% or more of the maximum weight will be selected so that all major orientations are taken into account. All these steps can create duplicates, as a feature is now described as x, y, σ, θ where two features can be identical except for scale or orientation. This provides robustness to the detected points. To now attribute features to the points, a square pixel zone is defined around each point, divided in a 4×4 grid where the histogram of orientation is computed similarly as before, but with only 8 bins, compared to 36 for the

[4]For each patch, the mean value of the patch is subtracted, and then intensities are divided by the standard deviation of the patch.

orientation. The 16 8-bin histograms are concatenated in a 128-row vector that will form the descriptor of the feature. To further strengthen robustness to brightness changes, the vector is normalised, its coefficient thresholded at 0.2 and normalised again. At this point, for each image we have a number of features, usually in the thousands, which need to be matched. The proposed method is to compute Euclidean distance between all pairs.For each feature on the left image, the feature on the right image with lowest distance is selected if this distance is sufficiently low compared to the distance to the second best feature. The comparison relies on a user set threshold. This avoids features that are ambiguous. However, if this threshold is too high, it might remove a lot of features, especially when there are repetitive patterns in an image. Other methods like Speeded Up Robust Features (SURF) [8] exist, inspired by SIFT but using different solutions for each step, however, SIFT remains one of the most used methods of feature detection, description and matching today.

1.4 Robust fitting methods

Robust fitting methods try to answer the problem of finding the underlying model of a set of data points corrupted both by inlier noise and outliers. Inlier noise represents the fact that measurements are not exact and that the data points originating from the model do not perfectly fit it. Outliers are data points that do not originate from the model. They can be due to error in measurements or to the way data is gathered. Contrary to simple fitting methods that just deal with inlier noise and the need to interpolate information from available data points, robust fitting methods also need to be able to discriminate inliers and outliers, like Random Sample Consensus (RanSaC) [26] or to find an estimator that is not impacted by the presence of outliers, like Least Median of Squares (LMS) [44].

Traditional fitting methods like Least Square fitting can be arbitrarily disturbed by one single outlier and so some have proposed methods like LMS to circumvent that. Such methods use all the data but try to estimate a model without giving any individual data point too much impact on the final result. Least Median of Square (LMS) [78] tries to solve the impact of outliers by replacing the mean by the median in the least mean square method. This creates non-linearity which is solved by drawing multiple samples and finding the best one. Minimise the Probability of Randomness (MINPRAN) [84] uses a similar method but tests the randomness of the fits to select the best one. However, such algorithms typically fail if more than 50% of the data points are outliers and are not very robust to noise.

RanSaC algorithms have a different approach to handle outliers. To avoid the risk of computing a model using an outlier, they select the minimum number of data points to generate an hypotetic model and then evaluate the quality of this model.

1.4.1 The original RanSaC algorithm

The vanilla RanSaC algorithm, published in [26] and described in algorithm 1, proposes an iterative approach to find the best fitting model. Instead of taking into account all the data and optimising a model fitting them, like Least Square Means, a small random subset $\mathcal{P}_s$ of s_Θ data points is extracted from $\mathcal{P}$ and a model is computed from this sample using a chosen fitting function $\mathcal{F}_\Theta$. Then, the number of data points that lie within a user-set threshold σ of the model is computed and if this number is greater than the previous so-far-the-best model's number of inliers, this model becomes the new best model. Depending on the algorithm used to compute a model, a sample does not necessary yield a unique set of parameters. Algorithms like the 7-point algorithm can yield up to three different models for a given sample of seven data points and, occasionally, some samples can yield no valid models when the data points are in singular positions and produce a degenerate system of equations. Thus, for each candidate model θ, the set of inliers is determined using the error to model function D, and all data points whose error is less than σ are deemed inliers. The quality function Q of RanSaC is thus

$$Q(\theta) = |\mathcal{I}(\theta, \sigma)| = |\{p \in \mathcal{P} : D(p, \theta) \leq \sigma\}|. \tag{1.21}$$

At the end of the procedure, a refined model is usually computed taking into account all inliers $\mathcal{I}^*$. Typically, a least squares minimisation is performed

$$\mathcal{F}_{full}(\mathcal{I}^*) = \tilde{\theta} = \arg\min_\theta \sum_{p \in \mathcal{I}^*} D(p, \theta)^2. \tag{1.22}$$

This procedure can be iterated as many times as needed but a termination criterion depending on a confidence p_{II} that an outlier-free sample was drawn during computation is proposed to end computation early. This optional feature of the algorithm consists in a dynamic update of its maximum number of iterations T when a better model is found. Initially, T is proportional to the time budget allocated to the algorithm. It is then lowered during the procedure: the iterations stop when there is sufficient confidence that at least one uncontaminated sample has been drawn. With ϵ the ratio of inliers, the probability of drawing a sample contaminated by outliers is $1 - \epsilon^s$. This event happening for T samples has a probability $(1 - \epsilon^s)^T$. During T iterations we can get a confidence level p_{II} (usually 95% or 99%) that at least one uncontaminated sample was drawn:

$$p_{II} = 1 - (1 - U_s(\epsilon))^T = 1 - (1 - \epsilon^s)^T, \tag{1.23}$$

where ϵ is computed from number of estimated inliers, which means we need at least:

$$T(\epsilon) = \left\lceil \frac{\log(1 - p_{II})}{\log(1 - \epsilon^s)} \right\rceil \tag{1.24}$$

Table 1.2: Evolution of minimum number of iteration T to get confidence level $p_{II} = 0.95$ that an outlier-free sample is drawn, depending on the number of points used to estimate a model s and inlier ratio ϵ. In bold, values used in our experiments.

$k \setminus \epsilon$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
2	2	3	5	7	11	18	32	74	299
3	3	5	8	13	23	46	110	373	2995
4	3	6	11	22	47	116	369	1871	$3.0 \cdot 10^4$
5	4	8	17	38	95	292	1232	9361	$3.0 \cdot 10^5$
6	4	10	24	63	191	730	4108	$4.7 \cdot 10^4$	$3.0 \cdot 10^6$
7	5	13	35	106	382	1827	$1.4 \cdot 10^4$	$2.3 \cdot 10^5$	$3.0 \cdot 10^7$
8	6	17	51	177	766	4570	$4.6 \cdot 10^4$	$1.2 \cdot 10^6$	$3.0 \cdot 10^8$
9	7	21	73	296	1533	$1.1 \cdot 10^4$	$1.5 \cdot 10^5$	$5.9 \cdot 10^6$	$3.0 \cdot 10^9$
10	7	27	105	494	3067	$2.9 \cdot 10^4$	$5.1 \cdot 10^5$	$2.9 \cdot 10^7$	$3.0 \cdot 10^{10}$
15	13	84	630	6370	$9.8 \cdot 10^4$	$2.8 \cdot 10^6$	$2.1 \cdot 10^8$	$9.1 \cdot 10^{10}$	$3.0 \cdot 10^{15}$

iterations to reach the desired confidence. The probability $1 - p_{II}$ represents a type II error: missing a good model because the number of tested samples is not sufficient. The function $T(\epsilon)$ is decreasing, hence a higher inlier ratio means fewer iterations are needed. The function $T(s)_{\epsilon,p_{II}}$ is increasing which is why model estimators $\mathcal{F}_\Theta$ which require the smallest sample size s_Θ are preferred. The value $p_{II} = 1$ (100% confidence) requires an infinite number of iterations in (1.24), in which case dynamic adaptation of T is disabled and all possible samples of s points generate hypotheses. The evolution of T for a given confidence level p_{II} (here 0.95) is presented in table 1.2.

Algorithm 1: Regular RanSaC procedure [26]. Blue lines are optional. This algorithm originates from paper [75].

parameter : Initial maximum number of iterations T, error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^D$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$ `// Draw random subset of` s_Θ `data points`
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\theta^* \leftarrow \theta$, $\mathcal{I}^* \leftarrow \mathcal{I}$ `// Keep maximum consensus model`
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon)$ `// Updated number of iterations` (1.24)
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
$\theta^* \leftarrow \mathcal{F}_{full}(\mathcal{I}^*)$
return $(\theta^*, \mathcal{I}^*)$

A simple example of RanSaC can be seen in line estimation in a 2D plane. The true model is the line $y = -0.608\,x - 0.138$ and $\mathcal{P}$ consists of 15 000 points in $\mathcal{S} = [-1, 1]^2$, 10

000 of which are generated from the model, plus some centred Gaussian noise of standard deviation 0.1 and the remaining 5000 are uniformly drawn in $\mathcal{S}$, see Figure 1.3.

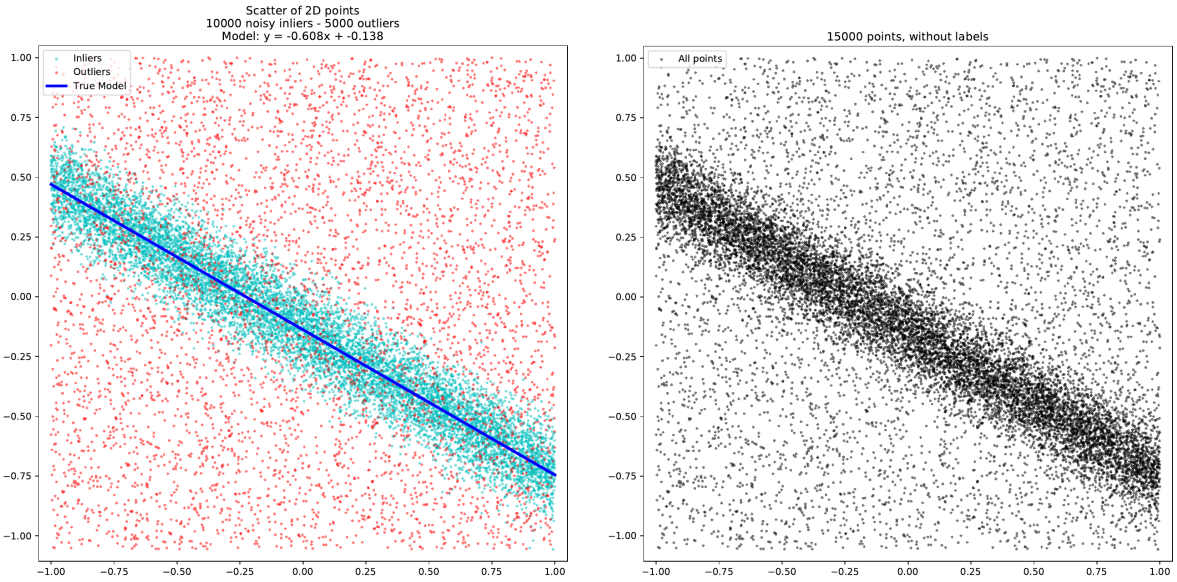


Figure 1.3: 2D data to illustrate RanSaC algorithm. On the left, labelled data: the true model is the dark blue line, inlier points are marked in light blue and outliers in red. On the right: unlabelled data as input of the RanSaC algorithm.

The data is fed unlabelled to the RanSaC algorithm which will iteratively choose $s = 2$ points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, the minimum number to estimate a line, compute the associated model using the estimator $\mathcal{F}(p_1, p_2) = \left(\frac{y_2 - y_1}{x_2 - x_1}, y_1 - x_1 \times \frac{y_2 - y_1}{x_2 - x_1}\right)$. This procedure is iterated until a satisfactory number of inliers is found using a threshold $\sigma = 0.1$. Figure 1.4 and 1.5 shows different iterations of the algorithm.

Here, after 28 iterations the model has confidence $p_{II} = 0.99$ that an outlier free sample was drawn, and stops. The best model estimated was at iteration 28. The number of inliers is 7,599 out of 10,000, with model $y = -0.656\,x - 0.091$ instead of $y = -0.608\,x - 0.138$. This ratio of inliers yields a stopping iteration of $T(\frac{7599}{15000}) = 16$. We can see on Figure 1.6 that the estimated model is slightly skewed as the inlier/outlier threshold $\sigma = 0.1$ is too small to encompass all inliers — which are distributed around the model following a Gaussian distribution of standard deviation σ and thus can end further than 0.1 from the model with 36% probability. We also see that points that were generated as outliers have fallen by chance inside the inlier region and thus cannot be distinguished from actual inliers.

In this example the data provided is quite easy to handle and the value of the chosen inlier/outlier threshold is adapted to the noise level of the data. Yet, the chosen sample, even though outlier free, gives a slightly skewed model as an inlier sample can still not represent perfectly the data due to the noise. Moreover, with data including more outliers than inliers, or different distribution of inliers and outliers it can be challenging for

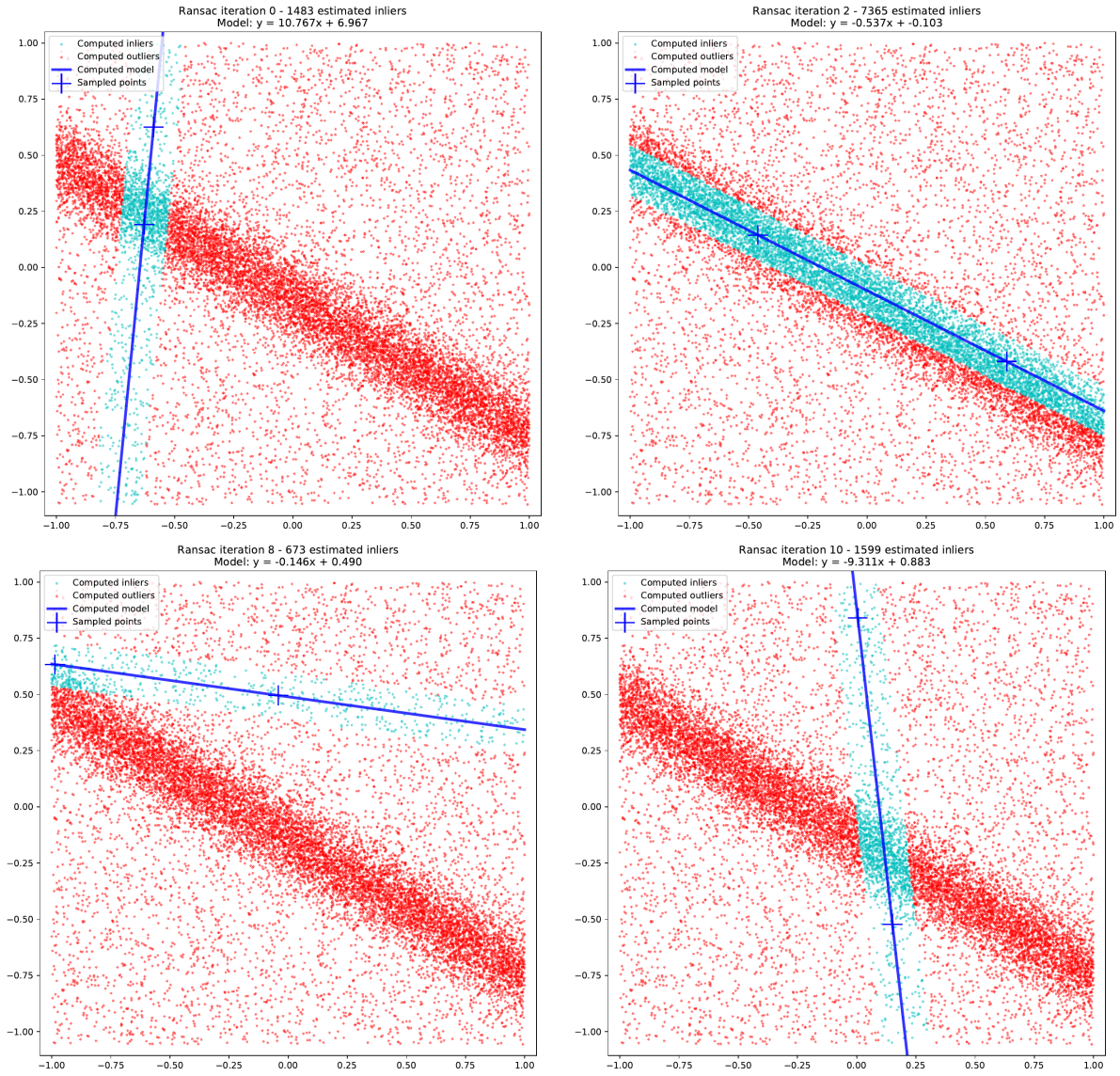


Figure 1.4: Iterations of RanSaC algorithm. The two selected data points are marked by dark blue crosses, the computed model by the dark blue line, the points lying within 0.1 of the model as putative inliers in light blue and putative outliers in red. Iteration number, number of inliers and model parameters are above each graph.

the basic RanSaC to find a valid model, especially if no information on the appropriate threshold is available.

Since its introduction, many improvements of RanSaC have been proposed to tackle its shortcomings. Some change the sampler Sa based on some hypothesis about inlier distribution, others add optimisation or verification steps to improve the quality of the estimated models. Several more recent methods try to remove the user set threshold in order to tackle unknown data more easily. This requires a change in quality function Q as the inlier number depends on this threshold. The latter methods are presented in the next subsections. Some methods are tailored to answer two-view geometry problems while other remain general in their approach.

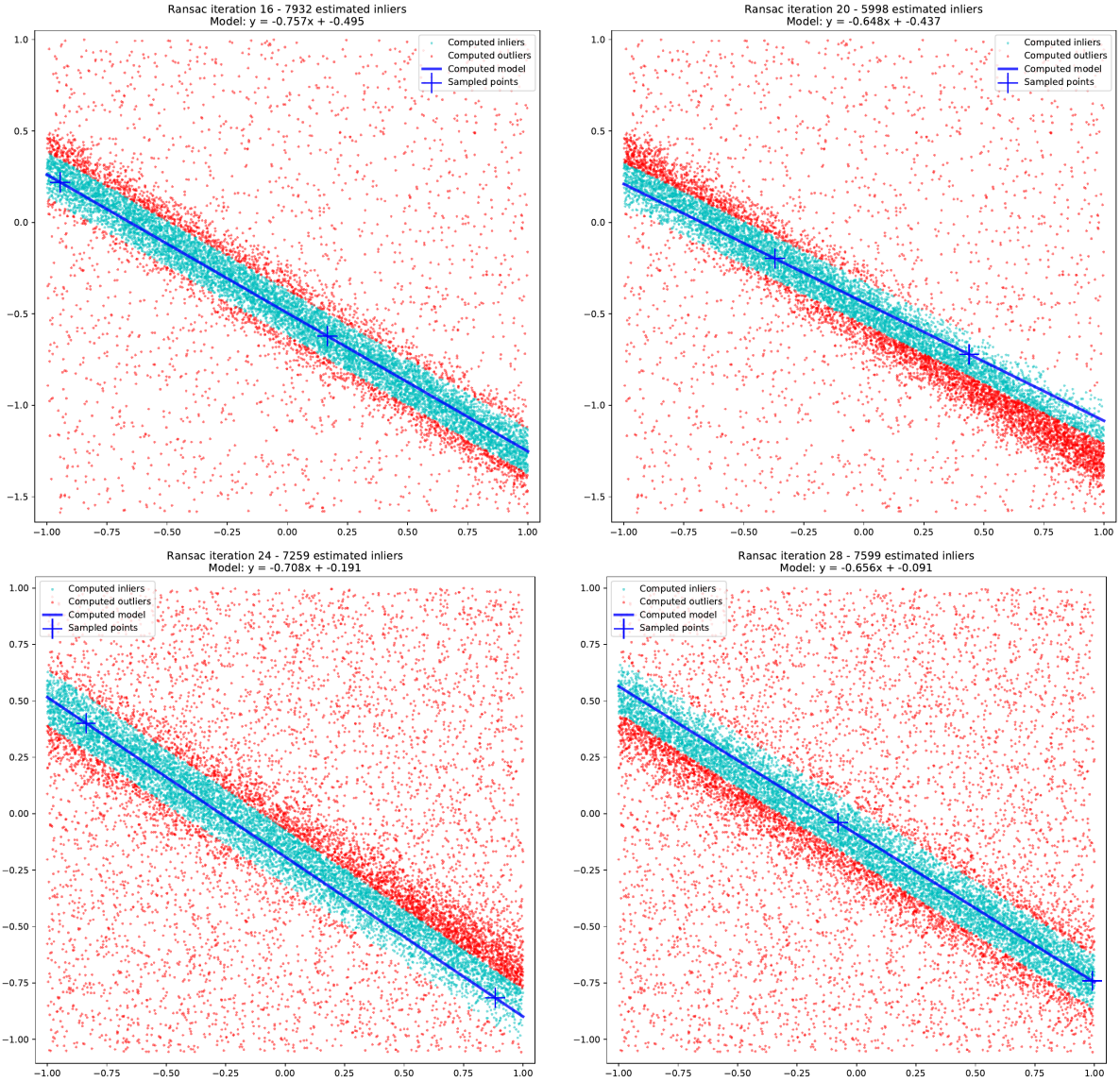


Figure 1.5: Iterations of RanSaC algorithm. The two selected data points are marked by dark blue crosses, the computed model by the dark blue line, the points lying within 0.1 of the model as putative inliers in light blue and putative outliers in red. Iteration number, number of inliers and model parameters are above each graph.

1.4.2 Early improvements of RanSaC

Improvements on the RanSaC algorithm that still use a threshold propose different solutions depending on the identified shortcomings. Some will propose improvements of the model estimation to increase the final model quality. Locally Optimized RanSaC (LO-RanSaC) [16] and its improved version in [42] is based on the observation that a sample containing only inliers might still produce a subpar model: as RanSaC uses minimal samples to estimate a model it is very sensitive to inlier noise. It is presented in algorithm 2. The authors claim that this leads to an increase in the required number of iterations needed to obtain a satisfactory model and a diminution in the quality of the model. To solve this, they propose different local optimisations steps when a new best

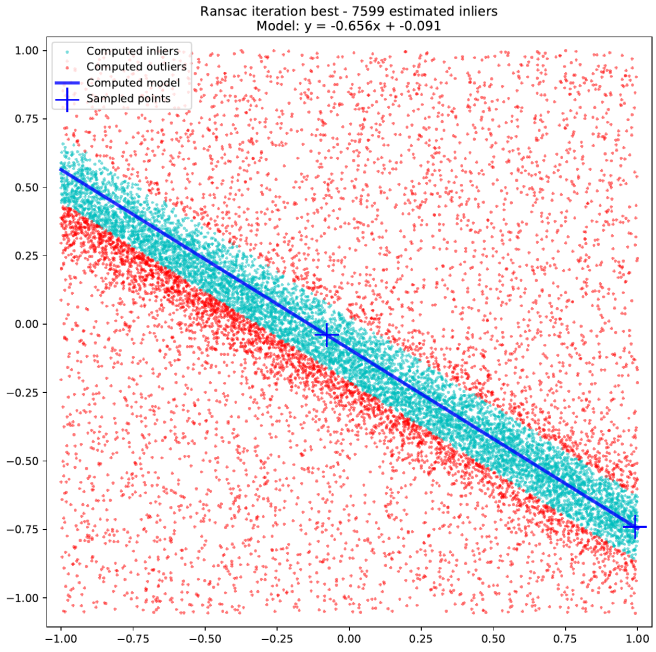


Figure 1.6: Best model estimated by the RanSaC algorithm. The two selected data points are marked by dark blue crosses, the computed model by the dark blue line, the points lying within 0.1 of the model as putative inliers in light blue and putative outliers in red.

model is found.

1. The first proposed option is a simple estimation using all putative inliers within a threshold σ with the global estimator $\mathcal{F}_{full}$.

2. The second option is an iterative version of the first one, by increasing the threshold by a factor K, $K\sigma$, and iteratively estimating a model and reducing the threshold until it reaches σ.

3. The third option is to use a RanSaC estimation with a non-minimal estimator where the size of a sample is adapted to the problem — for epipolar geometry $s = \min(\frac{|\mathcal{I}|}{2}, 14)$ and for homography estimation $s = \min(\frac{|\mathcal{I}|}{2}, 12)$.

4. The last method is a combination of the second and third.

In their experiments the fourth method gives the best results and improves runtime in most cases.

Other methods focus on the sampling strategy based on some assumption about the data. N-Adjacent Points Sample Consensus (NAPSAC) [89] is based around the assumption that inliers are more likely to be found closer to each other. The assumptions are that outlier data is uniformly distributed in the space while inlier data exist in cluster with a certain range. In order to do so, the uniform sampler of RanSaC is replaced by the following procedure:

Algorithm 2: LO-RANSAC procedure [16]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Initial maximum number of iterations T, error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^D$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$ `// Draw random subset of` s_Θ `data points`
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      Compute $\hat{\theta}$ and $\hat{\mathcal{I}}$ from θ `// LO-RANSAC optimisation, Algorithm 3, 4, 5 or 6`
      $\theta^* \leftarrow \hat{\theta}, \mathcal{I}^* \leftarrow \hat{\mathcal{I}}$ `// Keep maximum consensus model`
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon)$ `// Updated number of iterations` (1.24)
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

Algorithm 3: LO-RANSAC first optimisation method [16].

parameter : Error tolerance $\sigma \geq 0$
input : Model class $\Theta \subset \mathbb{R}^d$, best-so-far inlier set $\mathcal{I} \subset \mathbb{R}^D$
output : Model parameters $\hat{\theta} \in \Theta$, inliers $\hat{\mathcal{I}} \subset \mathcal{P}$

$\hat{\theta} \leftarrow \mathcal{F}_{full}(\mathcal{I})$
$\hat{\mathcal{I}} \leftarrow I(\hat{\theta}, \sigma)$
return $(\hat{\theta}, \hat{\mathcal{I}})$

Algorithm 4: LO-RANSAC second optimisation method [16].

parameter : Error tolerance $\sigma \geq 0$, increase parameter K
input : Model class $\Theta \subset \mathbb{R}^d$, best-so-far model θ
output : Model parameters $\hat{\theta} \in \Theta$, inliers $\hat{\mathcal{I}} \subset \mathcal{P}$

$\hat{\theta} \leftarrow \theta$
for $i = K \dots 1$ **do**
  $\hat{\mathcal{I}} \leftarrow I(\hat{\theta}, i \cdot \sigma)$
  $\hat{\theta} \leftarrow \mathcal{F}_{full}(\hat{\mathcal{I}})$
$\hat{\mathcal{I}} \leftarrow I(\hat{\theta}, \sigma)$
return $(\hat{\theta}, \hat{\mathcal{I}})$

Algorithm 5: LO-RANSAC third optimisation method [16].

parameter : Error tolerance $\sigma \geq 0$, increase parameter K
input : Model class $\Theta \subset \mathbb{R}^d$, best-so-far model θ
output : Model parameters $\hat{\theta} \in \Theta$, inliers $\hat{\mathcal{I}} \subset \mathcal{P}$

$(\hat{\theta}, \hat{\mathcal{I}}) \leftarrow$ RanSaC on $\mathcal{I}$ with non-minimal sample and threshold σ
`// Algorithm 1 with different` s
return $(\hat{\theta}, \hat{\mathcal{I}})$

Algorithm 6: LO-RANSAC fourth optimisation method [16].

parameter : Error tolerance $\sigma \geq 0$, increase parameter K
input : Model class $\Theta \subset \mathbb{R}^d$, best-so-far model θ
output : Model parameters $\hat{\theta} \in \Theta$, inliers $\hat{\mathcal{I}} \subset \mathcal{P}$

$\hat{\mathcal{I}} \leftarrow I(\hat{\theta}, K \cdot \sigma)$
for $i = K \dots 1$ **do**
  $(\hat{\theta}, \hat{\mathcal{I}}) \leftarrow$ RanSaC on $\hat{\mathcal{I}}$ with non-minimal sample and threshold $i \cdot \sigma$
  `// Algorithm 1 with different` s
$\hat{\mathcal{I}} \leftarrow I(\hat{\theta}, \sigma)$
return $(\hat{\theta}, \hat{\mathcal{I}})$

1. Choosing a random point p_0 uniformly among all possible ones.
2. Finding all points within a hypersphere S_{x_0} of radius r of point p_0.
3. Reject point p_0 if there is less points in S_{x_0} than s.
4. Uniformly sample $s - 1$ points from S_{x_0}.

The rest of the algorithm is kept the same as normal RanSaC. It is presented in algorithm 7. The value of the radius r depends on the dataset and is user set.

Progressive Sample Consensus (PROSAC) [14] leverages the similarity measure (see Section 1.3) of the matches to sample data points in a more efficient order. This means that PROSAC can only be used when dealing with data points with such a measure, which is the case for two-view geometry problems. It is presented in algorithm 8. In order to reorder data samples without relying too much on the link between similarity of a match p and probability that this match is indeed an inlier, the method assumes that the ordering of data points in similarity measure is correlated to their ordering in probability of being inlier. Given all points $p_{(i)} \in \mathcal{P}$ ordered using this similarity measure, increasing subsets of potential samples $\mathcal{U}_n$ of size $n \in [s, T]$ are created where T is the termination criterion. A counter t is setup so that, at each iteration, either the sample is drawn from $\mathcal{U}_n$ at random if $t > T'_n$ or contains $s - 1$ points from $\mathcal{U}_{n-1}$ and $p_{(n)}$. $t = T'_n$ also dictates when to add p_{n+1} to increase $\mathcal{U}_n$. T'_n is computed as $T'_s = 1$; $T'_{n+1} = T'_n + \lceil T_{n+1} - T_n \rceil$ where $T_n = T \frac{\binom{n}{s}}{\binom{|\mathcal{P}|}{s}}$ is the expected number of samples among all samples drawn by normal

Algorithm 7: NAPSAC procedure [89]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Initial maximum number of iterations T, error tolerance $\sigma \geq 0$, radius r of hypersphere, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^D$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
for $i = 1 \dots T$ **do**
  $p_0 \leftarrow \text{Rand}(\mathcal{P}, 1)$ `// Draw random initial data point`
  $\mathcal{P}_{s-1} \leftarrow \text{Rand}(p_0, s-1, r)$ `// Draw` $s-1$ `point uniformly in a sphere of radius` r `around` p_0
  $\mathcal{P}_s \leftarrow \mathcal{P}_{s-1} \cup \{p_0\}$
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\theta^* \leftarrow \theta, \mathcal{I}^* \leftarrow \mathcal{I}$ `// Keep maximum consensus model`
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon)$ `// Updated number of iterations` (1.24)
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

RanSaC sampling that contain only points from $\mathcal{U}_n$. This ensures that each subset $\mathcal{U}_n$ is explored enough while converging towards normal RanSaC sampling when no good sample is found early. PROSAC also adds a new termination criterion, in addition to the classic RanSaC one, see equation (1.24) that adds a non-randomness condition to the possible max number of iteration T. This condition requires the number of inliers of $\mathcal{U}_T$ to be greater than the expected number of inliers given a binomial distribution of probability of a point being randomly inlier given an incorrect model.

GroupSAC [68] authors make the assumption that there exist a clustering of data points that will classify them as either high inlier clusters or low inlier clusters. A sample will then be drawn among a certain subset of the total number of groups n_g, called a configuration, ensuring that every group contributes at least one data point. It is presented in algorithm 9. They model the repartition as the number of inliers in a group G as following a mixture of binomial distributions respectively of parameters $(|G|, \epsilon)$ and $(|G|, 0)$. This leads to the conclusion that the fewer groups used to draw a sample, the higher the probability of this sample to be containing only inliers (see corollary 2.1 and equation (8) of [68]) and that the bigger the groups, the higher this probability as well (see equation (17) of [68]). This allows to sort configurations according to how many groups they contain and how many data points their groups contain which should reflect the likeliness of the configuration to yield an inlier only sample. Following this ordering and using the same logic as PROSAC, multiple samples are drawn from each configurations

Algorithm 8: PROSAC procedure [14]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0,1)$, confidence w.r.t. type I error p_I
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\tilde{\mathcal{P}} \subset \mathbb{R}^D$ sorted by similarity
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$t \leftarrow 0$, $n \leftarrow s$, $T \leftarrow |\mathcal{P}|$
$T'_n \leftarrow 1$
$\mathcal{U}_n \leftarrow \{p_i, i \in [s-1], p_i \in \tilde{\mathcal{P}}\}$ `// Initialise sample set with` $s-1$ `best matches`
for $i = 1 \dots T$ **do**
  $t \leftarrow t + 1$
  if $t = T'_n$ *and* $n < T$ **then**
    $n \leftarrow n + 1$
    $\mathcal{U}_n \leftarrow \mathcal{U}_n \cup \{p_n\}$ `// If enough draws in` $\mathcal{U}_n$`, add next point.`
  if $T'_n < t$ **then** `// Sample` p_n `and other points from` $\mathcal{U}_{n-1}$ `to get enough samples containing` p_n`.`
    $\mathcal{P}_{s-1} \leftarrow \text{Rand}(\mathcal{U}_n, s-1)$ `// Draw` $s-1$ `point uniformly in` $\mathcal{U}_{n-1}$
    $\mathcal{P}_s \leftarrow \mathcal{P}_{s-1} \cup \{p_n\}$
  else `// Enough samples contain` p_n `to draw randomly.`
    $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{U}_n, s)$ `// Draw` s `point uniformly in` $\mathcal{U}_s$
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\theta^* \leftarrow \theta$, $\mathcal{I}^* \leftarrow \mathcal{I}$ `// Keep maximum consensus model`
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon)$ `// Updated number of iterations with inequation (9) and (12) of [14].`
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

until a certain confidence is reached that the next configuration can be explored without missing a valid sample. The number of samples drawn from a configuration $C = \{G_i\}$, T_C is proportional to the number of samples it is possible to create by drawing at least one point from each group G_i divided by the number of possible samples to draw from all points, $\binom{|\mathcal{P}|}{s}$. The global termination criterion is adapted to include a non-randomness confidence that a random outlier sample does not yield by chance a high number of inliers and to compute the number of iterations using only the inlier ratio compared to the current configuration and not all data points. Different clustering solutions can be used, the authors tested the performance of optical flow and image segmentation for example.

Algorithm 9: GroupSAC procedure [68]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0, 1)$, confidence w.r.t. type I error p_I
input : Model class $\Theta \subset \mathbb{R}^d$, sorted configurations $\{C_n = \{G_u, |G_u| = n\}, n \in [1, \min(s, n_g)]\}$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$n \leftarrow 1$
for $i = 1 \dots T$ **do**
  if $i = T_{C_n}$ **then** $n \leftarrow n + 1$
    `// Compute` T_{C_n} `with equation (11) and (13) of [68]`
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{C}_n, s)$ `// Sample` s `data points from` C_n `so that each group of` C_n `contributes at least one point.`
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\theta^* \leftarrow \theta, \mathcal{I}^* \leftarrow \mathcal{I}$ `// Keep maximum consensus model`
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon_{G_n})$ `// Updated number of iterations with inequation (15) of [68].`
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

Finally some methods include some tests of validity about the estimated model to increase efficiency or the quality of the resulting model. It might be a simple geometric verification that all data points verify some real world constraint like lying in front of the camera for computer imaging or a chirality constraint. Such a test can happen before the estimation of the quality of the model, to discard early incorrect models and avoid computing errors or after to increase the quality of the final model. DegenSac [15] is made to avoid epipolar geometries that are based on homographies on a dominant plane of the scene. It is presented in algorithm 10. The authors propose a test to check whether the seven correspondences used to compute a fundamental matrix, see Subsection 1.2.2, are actually consistent with a homography. A homography H is consistent with a fundamental

matrix F if all points satisfying the homography constraint $x' = Hx$, see equation (1.4), also satisfy the epipolar constraint $x'^T Fx = 0$. The authors prove that if five, six or seven correspondences used to estimate a fundamental matrix are homography consistent then the fundamental matrix is *H-degenerate*. To test this, three points $\{p_i = (x_i, x'_i), i \in [1, 3]\}$ are used to compute the homography associated to F, see equation (4) of [15], and test 2 other points of the sample. Only five homographies need to be tested out of all the combinations possible from the seven points as some will be duplicates, so the author only test those using points of the sample with index $P_H = \{P_s[1], P_s[2], P_s[3]\}$, $\{P_s[4], P_s[5], P_s[6]\}$, $\{P_s[1], P_s[2], P_s[7]\}$, $\{P_s[4], P_s[5], P_s[7]\}$ or $\{P_s[3], P_s[6], P_s[7]\}$. This test is performed after estimating whether a model is the new so-far-the-best model and if a homography consistent with the current model is found, a specific estimator, called the plane-and-parallax algorithm [37] is used to compute a new fundamental matrix. If this new fundamental matrix is still better than the so-far-the-best model, then it is kept. It is also possible to compute the best homography to find the dominant plane of the image by computing the inlier set of each homography found and storing its support.

The previously presented algorithms focus on one identified issue and address this specific problem. However, Universal RanSaC (USAC) [73] proposes a unified algorithm that combines the best of breed to get an improved algorithm. It is presented in algorithm 11. For a long time, USAC was considered state-of-the-art and is still used as baseline for some recent papers like [17, 39]. Many combinations were explored to develop USAC but the final algorithm uses the following methods:

1. PROSAC [14] sampling.
2. Model dependent sample check, like chirality.
3. Estimation of the model.
4. Model check with the SPRT test [57].
5. DegenSac [15] test when applicable.
6. LO-RanSaC [16] optimisation with both inner RanSaC and iterative least-square.

The SPRT test is based on checking during computation whether the points are validating a good or a bad model. The termination criterion is adapted to take into account the non-randomness and confidence like PROSAC but also risk of rejecting good models during post evaluation checks.

[6, 7] authors proposed another improvement, called VSAC [39]. It is meant as an aggregate of other recent algorithms to get the best of each of them, hence the name derived from USAC [73] and is supposed to be better than [7]. They introduce the concept of independent random inliers to guide the estimation process and get a better estimate

Algorithm 10: DegenSac procedure [15]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Initial maximum number of iterations T, error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class Θ of fundamental matrices, input matches $\mathcal{P} \subset \mathbb{R}^D$
output : Fundamental matrix $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$, Output homography H^* of dominant plane.

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$H^* \leftarrow$ `Null`, $\mathcal{I}_H^* \leftarrow \emptyset$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$ `// Draw random subset of` s_Θ `data points`
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** `// Estimate models from sample`
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ `// Inliers for model` θ
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\theta^* \leftarrow \hat{\theta}, \mathcal{I}^* \leftarrow \hat{\mathcal{I}}$ `// Keep maximum consensus model`
      for $P_3 \in P_H$ **do** `// Test if sample is` H`-degenerate.`
        $H \leftarrow \mathcal{F}(P_3)$
        for $p = \{x, x'\} \in P_s \backslash P_3$ **do** `// Test consistency of other matches with` H
          if $x' \propto Hx$ *AND* $x'Fx = 0$ **then** p is consistent
        if *5 or more points are consistent with* H **then**
          $\mathcal{I}_H \leftarrow \{p_i \in \mathcal{P} : D(p_i, H) \leq \sigma\}$ `// Inliers for homography` H
          if $|\mathcal{I}_H| > |\mathcal{I}_H^*|$ **then** $\mathcal{I}_H^* \leftarrow |\mathcal{I}_H|$
          Compute θ_H with [37]
          $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta_H) \leq \sigma\}$
          if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
            $\theta^* \leftarrow \theta_H, \mathcal{I}^* \leftarrow \mathcal{I}$
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ `// Ratio of inliers`
      Compute $T(\epsilon)$ `// Updated number of iterations` (1.24)
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

Algorithm 11: USAC procedure [73]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : Initial maximum number of iterations T, error tolerance $\sigma \geq 0$, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^D$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$t \leftarrow 0$, $n \leftarrow s$, $T \leftarrow |\mathcal{P}|$ // PROSAC (algo 8)
$T'_n \leftarrow 1$ // PROSAC (algo 8)
$\mathcal{U}_n \leftarrow \{p_i, i \in [s-1], p_i \in \tilde{\mathcal{P}}\}$ // PROSAC (algo 8)
for $i = 1 \dots T$ **do**
  Sample $\mathcal{P}_s$ using PROSAC // Algorithm 8 line 6 to 14.
  Check sample $\mathcal{P}_s$ // For example chirality constraint.
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do** // Estimate models from sample
    Check model θ // For example orientation constraint.
    $\mathcal{I} \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \sigma\}$ // Inliers for model θ
    Verify model θ with SPRT test [57]
    if $|\mathcal{I}| > |\mathcal{I}^*|$ **then**
      $\hat{\theta} \leftarrow \theta$, $\hat{\mathcal{I}} \leftarrow \mathcal{I}$ // Keep maximum consensus model
      Check degenerency using DegenSac // Algorithm 10 line 9 to 19
      if $|\hat{\mathcal{I}} \cap \mathcal{I}^*| \leq 0.95|\mathcal{I}^*|$ **then** // Only apply LO-RANSAC if enough new inliers
        Compute θ^* and $\mathcal{I}^*$ from $\hat{theta}$ with inner RanSaC // LO-RANSAC (algo 5).
        Compute θ^* and $\mathcal{I}^*$ from $\hat{theta}$ with iterative least-square // LO-RANSAC (algo 4).
      $\theta^* \leftarrow \hat{\theta}$, $\mathcal{I}^* \leftarrow \hat{\mathcal{I}}$
      $\epsilon \leftarrow |\mathcal{I}|/|\mathcal{P}|$ // Ratio of inliers
      Compute $T(\epsilon)$ // Updated number of iterations with equation (15) of [73]
      if $T > T(\epsilon)$ **then** $T \leftarrow T(\epsilon)$
return $(\theta^*, \mathcal{I}^*)$

of the distribution of inliers, hypothesised to be a Poisson distribution with parameter λ once all dependent inliers are removed. The parameter λ is estimated for each image pair as the mean number of independent inliers consistent with a bad model, estimated on the first few n models, see equation (2) of the original paper. A data point is said to be a dependent inlier if it is below the inlier/outlier threshold and follows one of the conditions below:

- it is in the sample used to estimate the model,
- it is close to an independent inlier,

and in the case of epipolar geometry, a match can also satisfy:

- one of its element is close to the epipole,
- it does not pass the chirality check,
- it is closer than the inlier/outlier threshold to the epipolar lines of a independent inlier.

This way of defining dependent and independent inliers ensures that local structures are ignored. To do so, a DEGENSAC+ procedure is introduced, inspired by DegenSac [15]. The homography estimated in DegenSac is decomposed in rotation R and translation T — which has two solutions for both R and T, each is tested — and the associated fundamental matrix is directly estimated using calibration matrices K and K' as: $\hat{F} = K'^{-T}TRK^{-1}$. The support of this new fundamental matrix is then compared to the independent inlier support of the best-so-far one and it can decide whether the model is degenerate. If the calibration matrices are unknown they are estimated, using the center of the image as principal point and testing different focal lengths. To speed up the procedure in computing the support of a model the Adaptative-SPRT (A-SPRT) test is used, adapted from the SPRT test [57] which can end the computation if the chances of the tested model being a good one is too low, with a user defined definition of good models. To improve the original design and remove user defined parameters, A-SPRT evaluates during the first n runs the time to estimate a model during run, instead of it being user defined, as well as the number of valid models per sample. Using the parameter λ computed before the probability of finding good or bad models is also better estimated. Finally, A-SPRT includes a check to avoid refusing good models when the time gain is too low compared to the probability of rejecting a good model. Finally, VSAC proposes to use both a fast local optimisation and a slower final one instead of using least-squares in each step as in LO-RANSAC [16]. The fast local one is only applied when enough independent inliers are found and when the intersection over union of the inlier sets of the model and previous best-so-far model is above 0.95. An iterative version of the third proposed method of LO-RANSAC is used but with 40 and 42 points used for homography and fundamental matrix

estimation respectively, and 5 and 12 iterations respectively. The final slow optimisation is the σ-consensus++ algorithm as proposed in [7].

The algorithms presented above all require a user set threshold that should be close to the actual inlier noise level of the dataset. This value can only be estimated and requires knowledge of the data or trial and error to find a good value. This is why methods that do not require such knowledge have been designed. They are presented in the next section. Note that such method might still have user set parameters, even user-set inlier/outlier threshold but usually those values are not hard thresholds nor the threshold actually used to compute inlier/outliers and might be used as upper bounds or computation tools.

1.4.3 Threshold-free algorithms

The need for a user-set threshold between inlier and outlier requires some knowledge about the data and the scale of the inlier noise level. In general, such information is not available and thus, the performance of the algorithm is highly dependent on the validity of the selected threshold value. Some authors try to tackle this issue by estimating the threshold as well as the model and inlier/outlier classification or entirely remove the classification. However, a new measure of model quality is required as quality function of RanSaC and algorithms presented in the previous Section 1.4.2 use the number of inliers of a model, see equation (1.21). This function increases with the inlier/outlier threshold σ and thus would yield models with highest thresholds if used directly.

Stable Random Sample Consensus (STARSAC) [13] aims at solving this issue by testing different thresholds and selecting the best one. It is presented in algorithm 12. The proposed method is based around running RanSaC multiple times, $n_{starsac}$ times, for each tested threshold and selecting the threshold which yields the most stable results. This is based around the hypothesis that poorly chosen thresholds will yield models too sensitive to inlier noise or outliers and thus have high variance of final model parameters whereas the appropriate threshold will consistently lead to similar variance in model parameters. The quality function is then the Variance-over-Parameter (VoP):

$$Q(\sigma) = -\frac{1}{n_{starsac}} \left(\sum_{i=0}^{n_{starsac}} \left(\bar{\theta}_i(\sigma) - \theta_i(\sigma) \right)^2 \right), \tag{1.25}$$

where $\theta_i(\sigma)$ is the model parameter at threshold σ for $i \in [n_{starsac}]$ and $\bar{\theta}_i(\sigma) = \frac{1}{n_{starsac}} \sum_{i=0}^{n_{starsac}} \theta_i(\sigma)$ is the mean of estimated parameters.

Minimum Unbiased Scale Estimator (MUSE) [58] is designed to handle high outlier ratios and multiple surfaces but its scale estimate can be used to perform a robust estimation when faced with only one model. It is an adaptation of Least Median of Squares [44] but replaces the median by a scale estimate computed for each possible residuals and still uses the iterative procedure of RanSaC. For a given model all absolute residuals

Algorithm 12: STARSAC procedure [13]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

hyperparameter: Σ, set of possible values for threshold σ, K number of RanSaC per run.
parameter : Maximum number of iterations T of RanSaC, confidence w.r.t type II error $p_{II} \in (0, 1)$
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^D$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$, threshold σ^*

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$, $\sigma^* \leftarrow \infty$
for $\sigma \in \Sigma$ **do**
  for $i = 1 \dots K$ **do**
    Compute $\theta_{i,\sigma}$ with RanSaC (algo 1)
  $Q_\sigma \leftarrow VoP(\theta_{i,\sigma})$ `// Compute with equation` (1.25).
$\hat{\sigma} \leftarrow \arg\min_\sigma(Q_\sigma)$
$\theta^* \leftarrow \frac{1}{K}\sum_{i=1}^{K} \theta_{i,\hat{\sigma}}$
$\sigma^* \leftarrow$ best $\sigma > \hat{\sigma}$ consistent with θ^*
$\mathcal{I}^* \leftarrow I(\theta^*, \sigma^*)$
return $(\theta^*, \mathcal{I}^*)$

$\{r_j, j \in [s, |\mathcal{P}|]\}$ are sorted and the j^{th} residual yields a scale estimate $s_j = \frac{r_j}{E[u_j]}$ where $E[u_j]$ is the expected value of a random variable u following a standardised distribution such that the residual distribution r have standard deviation σ_{dev} and $r = \sigma_{dev} u$. The computation of $E[u_j]$ depends on the hypothesis made on the residual distribution. The quality function Q of a model is the unbiased scale estimate $\hat{\sigma}$ which is the minimum scale estimate $s^*_\theta = s_{j'}$ divided by the expected value of the scale estimate of a standardised normal distribution v'_j:

$$Q(\sigma, \theta) = -\hat{\sigma}_\theta = \frac{s^*_\theta}{E[\min v_{j'}]}. \tag{1.26}$$

Authors of [17] propose to use the Likelihood-Ratio Test to estimate the quality of models for different thresholds. It is presented in algorithm 14. The authors make the assumption that data points follow a mixture of two uniform distributions. One, with weight ρ is the inlier distribution and draws uniformly in the inlier region $\mathcal{I}(\theta, \sigma)$ and the other, with weight $1 - \rho$ is the outlier distribution which draws uniformly from the entire space $\mathcal{S}$. Given this hypothesis on data points distribution, the likelihood that the data is not random — meaning $\rho > 0$ — is estimated. The quality function is thus the log-likelihood:

$$Q(\sigma) = L(\epsilon, \sigma) = 2|\mathcal{P}| \left(\epsilon \log \frac{\epsilon}{p_\sigma} + (1 - \epsilon) \log \frac{1 - \epsilon}{1 - p_\sigma} \right), \tag{1.27}$$

where $p_\sigma = |\mathcal{I}(\theta, \sigma)|$ is the area of the inlier region. Thanks to this quality function, the inlier/outlier threshold σ and model parameters θ can be estimated simultaneously during a RanSaC like iterative procedure. In order to do so, a range of potential thresholds

Algorithm 13: MUSE procedure [58]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

parameter : T, allocated number of iterations.
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^k$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$, scale estimate $\hat{\sigma}^*$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$\hat{\sigma}^* \leftarrow \infty$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do**
    $\mathcal{D}_\theta \leftarrow \{D(p, \theta),\ p \in \mathcal{P}\}$
    Sort $\mathcal{D}_\theta$
    for $p_k \in \mathcal{P}$ **do**
      Compute $s_k = \frac{D(p_k,\theta)}{E[u_k]}$
      Compute s^*_θ and $\hat{\sigma}_\theta$ `// Use equation` (1.26).
    if $\hat{\sigma}_\theta < \hat{\sigma}^*$ **then**
      $\hat{\sigma}^* \leftarrow \hat{\sigma}_\theta$
      $\mathcal{I}^* \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq \hat{\sigma}^*\}$
      $\theta^* \leftarrow \theta$
return $(\theta^*, \mathcal{I}^*, \hat{\sigma}^*)$

$\{\sigma_{min}, \dots, \sigma_{max}\}$ is used at each iteration to get different inlier ratios and the ratio that yields the best log-likelihood is selected if it beats the best-so-far model's log-likelihood L^*. To ensure control over the type I error, the likelihood of a selected model must be below a certain threshold c computed from a confidence p_I and a lookup table for the χ^2 distribution with $d + 2$ degrees of freedom which the likelihood statistic approaches asymptotically:

$$\int_0^c \chi^2_{d_\Theta+2}(t)\, dt = p_I, \tag{1.28}$$

and yields a lower bound for the log-likelihood:

$$L_{\min} = \frac{c}{2|\mathcal{P}|}. \tag{1.29}$$

The type II error is controlled with the same stopping criterion approach as RanSaC. The authors also add an early bailout strategy that stops the residual computation with confidence p'_{II} if the model cannot beat a minimal inlier ratio $\epsilon_{\min}$:

$$L(\epsilon_{\min}(\sigma), \sigma) = L^*, \tag{1.30}$$

which leads to a bound on the probability of premature termination:

$$\tau_m = \sqrt{-\frac{\log(1 - p'_{II}) - \log\left\lfloor \frac{|\mathcal{P}|}{B} \right\rfloor}{2s}}. \tag{1.31}$$

This test adds a computational cost, which is mitigated by applying it only every B data points, where B is set empirically to balance the expected gains and costs. This requires an adaptation of the termination criterion of RanSaC (1.24):

$$T(\epsilon) = \left\lceil \frac{\log(1 - p_{II})}{\log(1 - \epsilon_{min}^{s} \times p_{II'})} \right\rceil, \tag{1.32}$$

We name this method LRT.

Algorithm 14: LRT procedure [17]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

hyperparameter: Σ, set of possible values for threshold σ
parameter : T, allocated number of iterations, confidence probability p_I of avoiding type I error
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^k$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$, threshold σ^*

$L^* \leftarrow 0$, $\theta^* \leftarrow$ `Null`, $\sigma^* \leftarrow 0$
Compute $\epsilon_{\min}(\sigma)$, $\forall \sigma \in \Sigma$ from $L_{\min}$ `// Algorithm 15,` $L_{\min}$ `computed from` (1.28) `and` (1.29)
while *number of iterations is below* T AND Σ *is not empty* **do**
  $X_s \leftarrow \text{Rand}(\mathbf{X}, s_\Theta)$ `// Get random minimal sample`
  for $\theta \in \mathcal{F}_\Theta(X_s)$ **do**
    Compute ratio of inliers $\hat{\epsilon}(\sigma)$, $\forall \sigma \in \Sigma$ with early bailout `// Algorithm 16`
    $\hat{\sigma} \leftarrow \arg\max_\sigma L(\hat{\epsilon}(\sigma), \sigma)$, $\hat{L} \leftarrow L(\hat{\epsilon}(\hat{\sigma}), \hat{\sigma})$
    if $L^* < \hat{L}$ **then**
      $L^* \leftarrow \hat{L}$, $\theta^* \leftarrow \theta$, $\sigma^* \leftarrow \hat{\sigma}$
      Compute $\epsilon_{\min}(\sigma)$, $\forall \sigma \in \Sigma$ from L^* `// Algorithm 15`
      Update T from $\epsilon_{\min}(\sigma)$ `// Algorithm 17`
return $(\theta^*, \mathcal{I}^*, \sigma^*)$ `// Model parameters, inlier set and threshold`

Algorithm 15: Compute equivalent inlier ratios and reduce Σ procedure for LRT [17].

input : L^*, target likelihood, Σ, set of possible values for threshold σ
output : Minimal equivalent inlier ratios $\epsilon_{\min}(\sigma)$, $\forall \sigma \in \Sigma$, reduced set Σ

for $\sigma \in \Sigma$ **do**
  if $L(1, \sigma) < L^*$ **then**
    Discard σ from Σ
  else
    Compute $\epsilon_{\min}(\sigma)$, from a bisection of (1.30)
return $(\{\epsilon_{\min}(\sigma),\ \sigma \in \Sigma\}, \Sigma)$

A-Contrario RanSaC (AC-RanSaC), also named Optimized RanSaC (ORSA), [62, 61, 60] tries to determine whether a model occurred by chance or not. It is presented in algorithm 18. The only hypothesis made is that background points are uniformly distributed

Algorithm 16: Compute inlier ratio $\hat{\epsilon}(\sigma),\ \forall\sigma \in \Sigma$ with early bailout procedure for LRT [17].

hyperparameter: Set of possible thresholds Σ, bailout test periodicity B
parameter : Probability of avoiding premature bailout p'_{II}
input : data $\mathbf{X}$, model to evaluate θ, $\{\epsilon_{\min}(\sigma),\ \sigma \in \Sigma\}$, minimal required inlier ratios
output : $\{\hat{\epsilon}(\sigma),\ \sigma \in \Sigma\}$

1 $\hat{\epsilon}(\sigma) \leftarrow 0,\ \forall\sigma \in \Sigma$
2 **for** $X_i \in \mathbf{X}$ **do**
3 Compute $e(X_i, \theta)$
4 **for** $\sigma \in \Sigma$ **do**
5 **if** $e(X_i, \theta) \leq \sigma$ **then**
6 $\hat{\epsilon}(\sigma) \leftarrow \hat{\epsilon}(\sigma) + \frac{1}{n}$
7 **if** $i \equiv 0 \pmod{B}$ **then** // Early bailout
8 Compute τ_i // Apply (1.31)
9 **for** $\sigma \in \Sigma$ **do**
10 **if** $\hat{\epsilon}(\sigma) \times n/i \geq \epsilon_{\min}(\sigma) - \tau_i$ **then**
11 Go to next point // Back to line 2
12 Bailout // Back to Algorithm 14, line 5
13 **return** $\{\hat{\epsilon}(\sigma),\ \sigma \in \Sigma\}$

Algorithm 17: Update number of iterations T while controlling type II error procedure for LRT [17].

parameters: Confidence probabilities against type II error: p_{II}, confidence that no valid model was missed because of lack of iterations, p'_{II}, probability of a valid model avoiding premature bailout
input : Σ, set of possible values for threshold σ
output : T, number of iterations

1 **return** $T(\sigma_{\min})$ // Apply (1.32)

in the whole space $\mathcal{S}$. From this hypothesis, and a threshold σ, the Number of False Alarms (NFA) of a model θ can be computed, that is an indication of the expected number of occurrences by chance. It is computed as the number of tests times the probability of n inlier validating a model by chance.

$$NFA(\theta, \sigma) = N_{estimator}(|\mathcal{P}| - s)\binom{|\mathcal{P}|}{n}\binom{n}{s}(\sigma^d\alpha_0)^{n-s}, \tag{1.33}$$

where $N_{estimator}$ is the number of potential models and α_0 the probability of a random data point having residual at most 1. It is used as a quality function by ordering all residuals after estimating a model and computing the NFA for each residual, testing all sorted $D(p_n, \theta), n \in [|\mathcal{P}|]$ as potential thresholds σ:

$$Q(\theta, \sigma) = -NFA(\theta, D(p_n, \theta)) = N_{estimator}(|\mathcal{P}| - s)\binom{|\mathcal{P}|}{n}\binom{n}{s}(D(p_n, \theta)^d\alpha_0)^{n-s}. \tag{1.34}$$

As the residuals are sorted n data points are considered inliers using $D(p_n, \theta)$ as threshold. The model and threshold that yields lowest NFA is then selected. 10% of the total number of possible iterations so that once a model with a NFA below a threshold $NFA_{\max}$ is found, it is improved by iterating only over its inliers to refine the model. Normally, the NFA is computed for all residuals but to improve performance an upper bound $\sigma_{\max}$ can be defined to ignore data points that would most likely be outliers of the considered model.

Algorithm 18: AC-RANSAC procedure [62, 61, 60]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

hyperparameter: $NFA_{\max}$ the threshold of the NFA
parameter : T, allocated number of iterations
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^k$
output : Model parameters $\theta^* \in \Theta$, inliers $\mathcal{I}^* \subset \mathcal{P}$, threshold σ^*

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$T_{reserve} \leftarrow T/10$, $T \leftarrow T - T_{reserve}$
$\mathcal{P}_{in} \leftarrow \mathcal{P}$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do**
    $\mathcal{D}_\theta \leftarrow \{D(p_i, \theta),\ p_i \in \mathcal{P}\}$
    Sort $\mathcal{D}_\theta$
    Find best NFA and index of best residual: $\widehat{NFA}_\theta$ and $\hat{i}$ `// Use algorithm 19.`
    if $\widehat{NFA}_\theta < NFA^*$ **then**
      $NFA^* \leftarrow \widehat{NFA}_\theta$
      $\mathcal{I}^* \leftarrow \{p_i \in \mathcal{P} : D(p_i, \theta) \leq D(p_{\hat{i}}, \theta)\}$
      $\theta^* \leftarrow \theta$
      New best model was found
  if *(A new best model was found* AND $NFA^* < NFA_{\max}$*)* OR *(*$i = T$ AND $T_{reserve} > 0$*)* **then**
    if $\mathcal{I}^* = \emptyset$ **then** $T \leftarrow T + 1$, $T_{reserve} \leftarrow T_{reserve} - 1$
    else
      $\mathcal{P}_{in} \leftarrow \mathcal{I}^*$
      $T \leftarrow T_{reserve}$
      $T_{reserve} \leftarrow 0$
return $(\theta^*, \mathcal{I}^*, D(p_{\hat{i}}, \theta^*))$

Marginalised Sample Consensus (MAGSAC) [6] uses a novel method to remove the need for an inlier/outlier threshold by giving weights to pseudo-inliers and estimating models using weighted estimation. It is presented in algorithm 20. At each RanSaC iteration models are estimated using a minimal sample and then a procedure called σ-consensus is used to compute the weights. Each data point with residual lower than a threshold σ_{max} gets sorted in $n_{partition}$ partitions $\mathcal{P}_i$ whose maximum residual is $\sigma_{max,i}$, those are the pseudo-inliers. Models θ_i are then estimated using increasing sets of pseudo-

Algorithm 19: Compute and find NFA for AC-RANSAC [62, 61, 60].

parameter : $\sigma_{\max}$ maximum residual to compute NFA for
input : Sorted residuals of all data points $\{D(p_i, \theta),\ p_i \in \mathcal{P}\}$
output : Best NFA $\widehat{NFA}_\theta$, index of best residual $\hat{i}$

$\widehat{NFA}_\theta \leftarrow \infty$
$\hat{i} \leftarrow s$
for $i = s \dots |\mathcal{P}|$ *so that* $D(p_i, \theta) \leq \sigma_{\max}$ **do**
  Compute NFA_i with equation (1.34)
  if $NFA_i < \widehat{NFA}_\theta$ **then**
    $\hat{i} \leftarrow i$
    $\widehat{NFA}_\theta \leftarrow NFA_i$
return $(\widehat{NFA}_\theta, \hat{i})$

inliers: $\theta_i = \mathcal{F}_{full}(\bigcup_{j=0}^{i} \mathcal{P}_j)$ and each point $p \in \mathcal{P}$ gets a weight $W(p)$ depending on its residual to the estimated model:

$$W(p) = \frac{2C(\rho)}{\sigma_{max}} \sum_{i=1}^{n_{partition}} (\sigma_{max,i} - \sigma_{max,i-1}) \sigma_{max,i}^{-\rho} D(p, \theta_i)^{\rho-1} \exp\left(\frac{-D(p, \theta_i)^2}{2\sigma_{max,i}^2}\right), \quad (1.35)$$

where ρ is the dimension of the error space, $C(\rho) = \frac{1}{2^{\rho/2}\Gamma(\rho/2)}$ with Γ the gamma function. This quantity originates from assuming the squared residuals to be chi-squared distributed with standard deviation σ and inliers uniformly distributed with standard deviation σ and outliers uniformly distributed with standard deviation l. This skews the residual thresholds by a multiplicative factor of 3.64. In practice, this means a point gets a higher weight depending on how many models it belongs to and thus will be more important if it validates many models. A final model is estimated from those weights and its quality is computed as the marginalised over σ log-likelihood of model θ given σ:

$$Q(\theta) = -|\mathcal{P}| \ln(l) + \frac{1}{\sigma_{max}} \sum_{i=1}^{n_{partition}} \Bigg(i(\ln(2C(\rho)l) - \rho \ln(\sigma_{max,i})) - \frac{R_i}{\sigma_{max,i}^2} + (\rho - 1) Lr_i \Bigg) (\sigma_{max,i} - \sigma_{max,i-1}), \quad (1.36)$$

where $R_i = \frac{1}{2} \sum_{j=1}^{i} D(p_j, \theta)^2$ is, $Lr_i = \sum_{j=1}^{i} \ln(D(p_j, \theta))$. Finally the number of iteration is also marginalized over σ as:

$$T(\theta) = \frac{1}{\sigma_{max}} \sum_{i=1}^{n_{partition}} \frac{(\sigma_{max,i} - \sigma_{max,i-1}) \ln(1 - p_{II})}{\ln\left(1 - \left(\frac{|I(\theta,\sigma)|}{|\mathcal{P}|}\right)^s\right)}. \quad (1.37)$$

MAGSAC's authors proposed an improvement named MAGSAC++ [7]. It is presented in algorithm 22. They propose both to change the RanSaC procedure used by a mixture of NAPSAC [89] and PROSAC [14] and a new σ-consensus method named σ-consensus++.

Algorithm 20: MAGSAC procedure [6]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

hyperparameter: $NFA_{\max}$ the threshold of the NFA
parameter : T, allocated number of iterations
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^k$
output : Model parameters $\theta^* \in \Theta$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$q^* \leftarrow 0$
for $i = 1 \dots T$ **do**
  $\mathcal{P}_s \leftarrow \text{Rand}(\mathcal{P}, s_\Theta)$
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do**
    if θ *is not valid* **then** Reject θ
    Compute $\hat{\theta}$ from σ-consensus `// See algorithm 21`
    Compute $Q(\hat{\theta})$ `// Use equation (1.36)`
    if $Q(\hat{\theta}) > q^*$ **then**
      $q^* \leftarrow Q(\hat{\theta})$
      $\theta^* \leftarrow \hat{\theta}$
      Update T `// Use equation (1.37)`
return θ^*

Algorithm 21: σ-consensus algorithm from MAGSAC [6].

parameter : Number of partition $n_{partition}$, max residual of points to weight σ_{max}.
input : Model $\theta \subset \Theta$, input data $\mathcal{P} \subset \mathbb{R}^k$.
output : $\hat{\theta}$

$\mathcal{I} \leftarrow I(\theta, \sigma_{max})$ `// Only consider points with residual less than` σ_{max}
Sort $\mathcal{I}$ by residual $D(p, \theta)$
$\sigma_{max} \leftarrow \max_{p \in \mathcal{P}} D(p, 3.64\theta)$
$\{w(p) \leftarrow 0,\ p \in \mathcal{I}\}$
$\delta_\sigma \leftarrow \frac{\sigma_{max}}{n_{partition}}, \sigma_{max,n} \leftarrow \delta_\sigma$
$\mathcal{I}_{tmp} \leftarrow \emptyset$
for $p \in \mathcal{I}$ **do**
  if $D(p, \theta) \leq 3.64\sigma_{max,n}$ **then** `// Check if` p `belongs to partition` n.
    $\mathcal{I}_{tmp} \leftarrow \mathcal{I}_{tmp} \cup \{p\}$
    Go to next p
  $\theta_n \leftarrow \mathcal{F}_{full}(\mathcal{I}_{tmp})$ `// Once all points of partition` n `are found, estimate model.`
  for $p' \in \mathcal{I}$ **do** `// Update weight of all points according to local model.`
    $w(p) \leftarrow w(p) + W(p, \theta_n, \sigma_{max,n}, \mathcal{I}_{tmp})/\sigma_{max}$ `// Use equation (1.35)`
  $\mathcal{I}_{tmp} \leftarrow \mathcal{I}_{tmp} \cup \{p\}$ `// Go to next partition.`
$\hat{\theta} \leftarrow \mathcal{F}_{full}(\mathcal{I}, \{w(p),\ p \in \mathcal{I}\})$
return $\hat{\theta}$

The assumptions made are that inlier noise σ is uniformly distributed in a range $[0, \sigma_{max}]$ where σ_{max} is a user set threshold and given a noise level σ residuals are distributed following the square root of the χ^2-distribution with n degrees of freedom, the dimension of the residual space. This yields a quality function as the inverse of a quantity that depends on the weights of σ-consensus++:

$$Q(\theta) = \frac{1}{\sum_{p \in \mathcal{P}} \rho(D(p, \theta))}, \tag{1.38}$$

where $\rho(D(p,\theta)) = \int_0^{D(p,\theta)} pw(p)dp$ with $W(p)$ the weights. ρ can be computed, the exact formula can be found below equation (3) of the paper [7]. To improve the RanSaC procedure with NAPSAC and PROSAC, first an initial point is chosen using PROSAC sampling, it is the seed of the NAPSAC-like procedure. However, instead of drawing uniformly in a sphere around this seed, points are drawn using a local PROSAC procedure where the metric used is the distance to the seed. This method is a way to leverage the power of localised sampler while still being able to tackle more global models if no local ones are found early. The other proposed improvement is to replace σ-consensus by σ-consensus++, an iterative reweighted least squares instead of the normal weighted least square. At each step, weights and models are iteratively estimated based on the putative inliers of the estimated model, given a user-set maximum threshold for residual σ_{max}. The weight of a point is computed as the marginal density of the inlier residual:

$$w(p) = \frac{1}{\sigma_{max}} C(n) 2^{\frac{n-1}{2}} \left(\Gamma\left(\frac{n-1}{2}, \frac{D(p,\theta_i)^2}{2\sigma_{max}^2}) \right) - \Gamma\left(\frac{n-1}{2}, \frac{k^2}{2} \right) \right), \tag{1.39}$$

for points with residual less than $k\sigma_{max}$ where k is determined by a chosen quantile of the χ-distribution, $w(p) = 0$ otherwise. Γ is the upper incomplete gamma function and $C(n) = \left(2^{\frac{n}{2}} \Gamma(\frac{n}{2}, 0) \right)^{-1}$. θ_i is the model estimated at the i^{th} step of the iterative reweighted least square and θ_0 is the model estimated with the minimal sampler in the RanSaC step.

1.4.4 Multi model methods

Some algorithms have been designed specifically to tackle multi-model problems. Those are situations where there are multiple underlying models describing the inliers and thus, finding one can be harder. An example would be a scene with multiple flat surfaces. Finding the homography defining each surface requires different processing. Most of the time, the processing is based around finding the best model, removing points considered inliers for it and then looking for the next best models. The number of underlying models might be required before hand to find all the models.

MUSE [58] described earlier is tailored to tackle such situations and will simply use the solution of removing points belonging to a model before computing the next one.

Algorithm 22: MAGSAC++ procedure [7]. Blue lines are optional. Red lines are changes from RanSaC (algorithm 1).

hyperparameter: $NFA_{\max}$ the threshold of the NFA
parameter : T, allocated number of iterations
input : Model class $\Theta \subset \mathbb{R}^d$, input data $\mathcal{P} \subset \mathbb{R}^k$
output : Model parameters $\theta^* \in \Theta$

$\theta^* \leftarrow$ `Null`, $\mathcal{I}^* \leftarrow \emptyset$
$q^* \leftarrow 0$
for $i = 1 \dots T$ **do**
  Select random seed $p_{init} \in \mathcal{P}$
  $\mathcal{U}_{init} \leftarrow s^{\text{th}}$ nearest neighbours
  Follow PROSAC procedure to increase $\mathcal{U}_{init}$ `// Algoritmh 8 from line 6 to line 14 but line 14 is modified to draw in` $\mathcal{P}$.
  for $\theta \in \mathcal{F}_\Theta(\mathcal{P}_s)$ **do**
    if θ *is not valid* **then** Reject θ
    Compute $\hat{\theta}$ from σ-consensus++ `// See algorithm 23`
    Compute $Q(\hat{\theta})$ `// Use equation (1.38)`
    if $Q(\hat{\theta}) > q^*$ **then**
      $q^* \leftarrow Q(\hat{\theta})$
      $\theta^* \leftarrow \hat{\theta}$
      Update T `// Use adapted version of equation (1.37) to include the impact of the new sampling strategy.`
return θ^*

Multiple AC-RanSaC (MAC-RANSAC) [72] goes a little further than simply launching multiple times the AC-RanSaC algorithm [62, 61, 60]. Indeed, to avoid classical issues of simply launching multiple runs, like splitting a single surface or merging different ones, or detecting multiple surfaces in a repeating pattern, after each run a fusion detection and filtering of repeating patterns is used. Other algorithms like J-linkage or T-linkage [55, 87, 88] compute multiple models and find which consensus set each point will belong to. By then clustering points according to the Jaccard distance using this characteristic, models and their consensus set can be selected. The different versions add a real time speed-up to work on-line instead of computing everything at once [88] or using a continuous version and the Tanimoto distance [55].

These solutions are not developed further as multiple model fitting is not the focus of this thesis.

1.5 Bundle Adjustment

Bundle adjustment refers to a global optimisation of all the estimated parameters of a pipeline. It can adjust the 3D coordinates of estimated points, the position of the cameras and their estimated intrinsic parameters. The main principle of bundle adjustment is to

Algorithm 23: σ-consensus++ algorithm from MAGSAC++ [7].

parameter : Max residual of points to weight σ_{max}, reference threshold σ_{ref}, number of iteration of the iterative reweighted least square T_{irwls}.
input : Model $\theta \subset \Theta$, input data $\mathcal{P} \subset \mathbb{R}^k$, so-far-the-best model θ^*.
output : $\hat{\theta}$

$\mathcal{I} \leftarrow \emptyset$
if θ^* IS *NULL* **then** `// Select all points with residual below` σ_{max}
  for $p \in \mathcal{P}$ **do**
    if $D(p, \theta) < \sigma_{max}$ **then** $\mathcal{I} \leftarrow \mathcal{I} \cup \{p\}$
else `// If there is already a so-far-the-best model, check if it is possible to beat it.`
  $r \leftarrow |I(\theta^*, \sigma_{ref})|$ `// Threshold to check if enough points are left to beat the model.`
  for $p \in \mathcal{P}$, $i \in [|\mathcal{P}|]$ **do**
    if $D(p, \theta) < \sigma_{max}$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{p\}$
      if $D(p, \theta) < \sigma_{ref}$ **then** $r \leftarrow r - 1$
    if $|\mathcal{P}| - i < r$ **then** Stop σ-consensus++
$\hat{\theta} \leftarrow \theta$
for $it \in [T_{irwls}]$ **do** `// Compute the weights.`
  $\mathcal{I} \leftarrow \emptyset$
  for $p \in \mathcal{P}$ **do**
    if $D(p, \hat{\theta}) < \sigma_{max}$ **then** $\mathcal{I} \leftarrow \mathcal{I} \cup \{p\}$
  for $p \in \mathcal{I}$ **do**
    Compute $w(p)$ using equation (1.39)
  $\hat{\theta} \leftarrow \mathcal{F}_{full}(\mathcal{I}, \{w(p),\ p \in \mathcal{I}\})$
return $\hat{\theta}$

minimise the reprojection error between the detected features and the projection of the observed 3D points. To do this, a large number of non-linear functions needs to be handled and it often requires sparse solutions as most points are often seen by only a few images.

A classic solution to the bundle adjustment problem is the Levenberg-Marquardt algorithm [45, 56]. It is an iterative procedure that uses a modified version of the Gauss-Newton method, where the equation $(\mathbf{J}^T\mathbf{J})\delta = \mathbf{J}^T(y - f(x, \beta))$ with f the fitting function, δ an increment and $\mathbf{J}$ the Jacobian of F with respect to the parameters β is replaced by:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\delta = \mathbf{J}^T(y - f(x, \beta)), \tag{1.40}$$

with $\mathbf{I}$ the identity and λ a damping factor that is adjusted dynamically. If λ is small, the method is closer to the Gauss-Newton algorithm, and if it is big it is closer to the gradient descent. Multiple solutions exist to adapt λ at each step, usually it increases or decreases depending on whether the current step improved or not the model and how

fast the convergence needs to be. As most optimisation methods it requires an initial estimation to be provided but it is far less sensitive to its quality than other methods like Gauss-Newton.

In recent years, the main focus of improvement of bundle adjustment has been efficiency as it is a time consuming step, see Chapter 4. [12] is focused on how to perform the bundle adjustment in a parallel setting to handle larger datasets as they are more common nowadays. Many other solutions have been developed for bundle adjustment. An extensive review of those can be found in [91], where the methods are categorised by type, with both generic and *ad-hoc* solutions analysed and recommendations on design and implementations. As the scope of this thesis is not bundle adjustment, we do not go further in details about those methods.

Chapter 2

Presentation of a Novel Semi-Artificial Data Generator

In this section we detail one of the main contribution of this work, semi-artificial data generation. This consists in a novel way of getting data to run benchmarks on with reliable metrics while keeping a realistic aspect to ensure that results can easily transcribe to real world scenarios. Section 2.1 presents existing datasets, how they were built and their shortcomings and the logic behind our solution. Section 2.2 presents the different metrics that we can study thanks to our solution and how they are defined for specific algorithms. Sections 2.3 and 2.4 present respectively how our method generates inliers and how it generates outliers to create a dataset. This chapter handles generation of data for the homography, fundamental matrix and essential matrix estimation problems. Element specific to the whole 3D reconstruction pipeline are presented in Chapter 4 alongside the pipeline.

2.1 Reasoning and motivation

Most of the previous work on RanSaC algorithms uses either real data processed through another algorithm or artificial data, or both, as their benchmark. When using real data, different kinds of solutions exist to create a dataset. For two-view geometry tasks, some will prefer to use an active sensor, like a LIDAR or a Microsoft Kinect and align the resulting 3D point cloud with the 2D images and detected matches. This is the case for some famous datasets like Tanks and Temple [40] (example in Figure 2.1) that uses scanners, TUM RGB-D [85] (Figure 2.2) or NYU Depth [80] (Figure 2.3) that use Kinect, Kitti [29] (Figure 2.4) that uses stereo cameras, GPS, and laserscan. The use of GPS or a mean to control the camera, like an articulated mechanical arm as in [82] (Figure 2.5) gives a reasonable estimation of the ground truth for the camera pose in the image that can be used as ground truth for the tested model. Others will use some reconstruction pipeline like ColMap or simply RanSaC with many iterations to find a satisfying solution to consider as ground truth. The famous USAC paper [73] (Figure 2.6) uses as a benchmark real images processed through 10^7 iterations of RanSaC to produce ground truth inliers. [64] uses data processed through Blender [18] without any further processing. The

Extreme View Dataset [59] (Figure 2.7) uses LORANSAC [16] to create ground truths as well as artificially distorted images. EPnP [43] uses known 3D models to compute the pose of real images to test its algorithm. A step of manual verification can be added to validate inliers and clean the dataset. This is the case for the PROSAC [14] paper that uses SIFT [52, 53] matches with simple manual annotations of inliers and outliers. The AdelaideRMF dataset [95] is also built with real images processed manually to obtain the ground truth. NAPSAC [89] even uses only manually created matches for its experiments. Both solutions present two main drawbacks. Firstly, there is an uncontrolled noise level on the inliers. Indeed, whether it comes from active sensor or a pipeline, the matches used as inliers for the benchmark are computed with another algorithm and there is no indication or certainty on the actual noise level between the two matching points. Secondly, the outlier to inlier ratio is not controlled any more than the distribution of the outliers. Thus it requires extensive computation and analysis to categorize dataset elements according to their difficulty and obtain a large, controlled set of data on which precise metrics can be computed. This work can be costly especially if using an active sensor and adding manual verification which has not really been done extensively. Some datasets are extensive and are known to offer different cases with varying difficulty but not precisely quantified. This makes it hard to compute precise and meaningful metrics when using real data and it is hard to conclude on the generalisation power of the results.

On the other hand, the use of purely synthetic data can solve this by offering perfect control over the inliers and outliers. Most papers that use synthetic data do not exclusively use those. It is often a way of confirming hypotheses and get quantitative results before doing some test on real data to get qualitative results. In LRT [17] and RECON [74] random 3D planes, homographies and fundamental matrices are drawn to generate data for experiments. MINPRAN [83] uses 3D planes to test different scenarios and analyse its behaviour. The EPnP paper [43] uses artificial scenes created through the software Blender[1] to get test cases. AMLESAC [41] uses 2D lines or random poses to measure precision and accuracy. When using such scenarios, usually the model is random and inliers and outliers are drawn uniformly in the relevant spaces. Some geometric constraints can be added to increase the realism like making sure the points are in front of the hypothetical camera.

All such methods lack likelihood relatively to real data as inliers distribution is not based on any analysis of real data and outliers can be poorly generated. Results on purely synthetic data are not enough to conclude on the quality of RanSaC algorithms and are always complemented by an analysis on real data. Tests on synthetic data are used as a proof of concept more than an actual benchmarking solution as it is not possible to conclude on the performance of algorithms with such artificial data.

In order to do meaningful improvements on the reconstruction pipeline, it is important

[1] `https://www.blender.org/`

Figure 2.1: Exemples of images and scan from the dataset Tank and Temples [40]. Scan are obtained with a LIDAR.

Figure 2.2: Exemples of images from the dataset TUM RGB-D [85]. Images and depth maps are acquired with a Kinect.

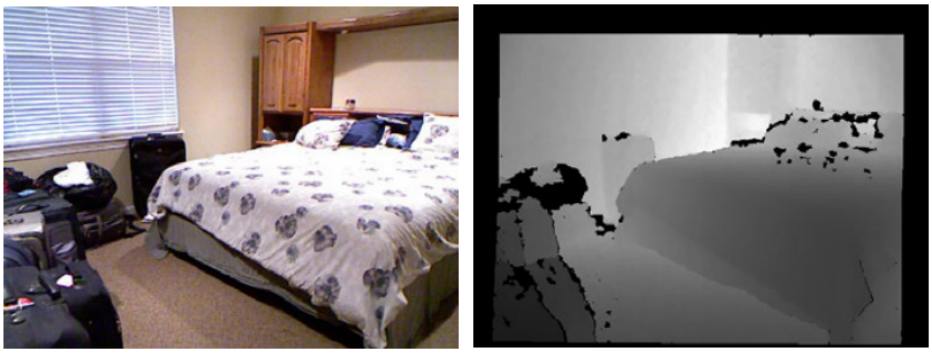


Figure 2.3: Exemples of image and scan from the dataset NYU Depth [80]. Images and depth maps are acquired with a Kinect.

Figure 2.4: Exemples of image and capture setup from the dataset Kitti [29]. The vehicule is equiped with GPS, laser and stereo cameras to capture an annotated video flux.

to have actual control over the impact of the changes implemented. To evaluate the quality of a RanSaC algorithm one can look at the retrieval capacity of the algorithm or at the

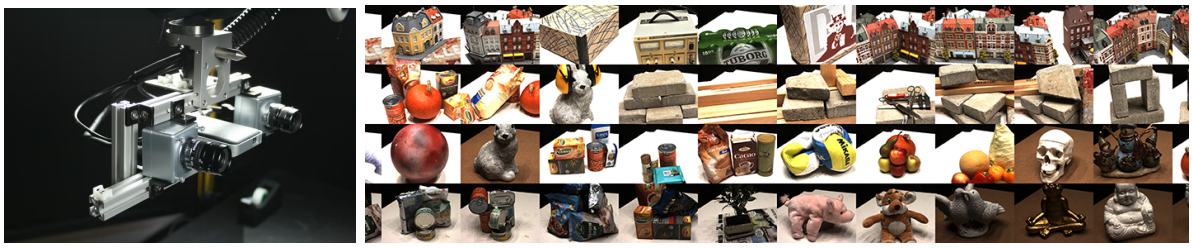


Figure 2.5: Exemples of capture setup and image from the dataset DTU Robot [82]. All objects captured are miniature installed in a photo box with lighting control and an articulated robot arm with stereo cameras.

Figure 2.6: Exemples of image pairs from the dataset of USAC [73]. Image pairs come with unlabelled matches and need to be processed to get a ground truth.

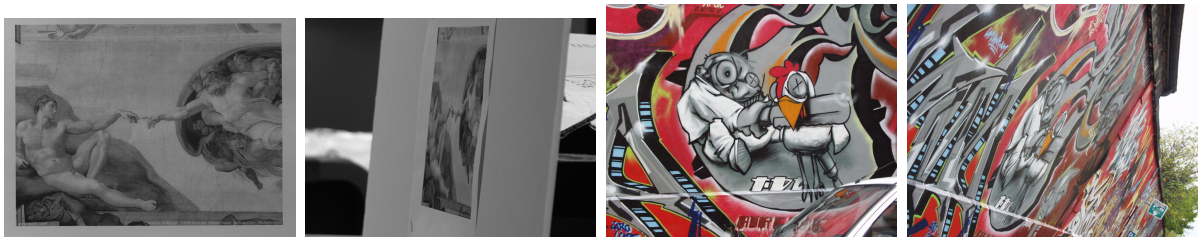


Figure 2.7: Exemples of image pairs from the dataset of EVD [59].

quality of the resulting model. However, as the final model is estimated using global optimisation on the final supposed inlier subset, a good retrieval performance should increase the quality of the computed model parameters. Moreover, the inlier subset has its own interest in a full MVS or SfM pipeline, as it is used for further computation and verification. This motivates the need for a method to reliably measure the capacity of RanSaC algorithms to discriminate between inliers and outliers. To do so, our solution uses data extracted from real datasets as a basis. This ensures that the resulting models are actually likely and the benchmark can be extended to real data with more confidence than simple toy problems. This also enables us to use true inlier distributions and make sure the results are not impacted by poor generation of inliers. Finally, we propose a way to control inlier noise and outlier/inlier ratio in order to compare solutions in different scenarios with accurate description of settings to be able to observe what impacts each algorithm's performance.

2.2 Metrics

RanSaC algorithms are meant to be used to perform robust model estimation; however, in an MVS or SfM pipeline, the inlier/outlier classification is useful as the inliers are used to create the 3D model and help computing future poses. This is why we can also consider RanSaC algorithms as classifiers and evaluate their capacity to separate inliers from outliers. Our studies focus mainly on this aspect by evaluating the precision and the recall of each algorithm for a given dataset.

Thanks to the data generation methodology we introduce below we have accurate labels for each data point and can thus compare predicted labels with ground truth labels. Precision is a metric that evaluates the quality of the prediction as the percentage of correctly inlier-labelled data points over the number of data points predicted as inlier. Lets denote $\mathcal{I}_{GT}$ the ground truth set of inliers generated by our method. The precision P of an algorithm is computed as:

$$P = \frac{|\mathcal{I}_{GT} \bigcap \mathcal{I}(\theta^*)|}{|\mathcal{I}(\theta^*)|} \tag{2.1}$$

where θ^* is the model returned by the algorithm. This shows whether an algorithm is accurate in its prediction and not just too broadly accepting data points to populate the model. A good precision will mean the resulting 3D points that are added to the model are real elements of the model and thus can be used to compute future views. On the other hand, recall is a metric that evaluates the retrieval power of the prediction as the percentage of correctly inlier-labelled data points over the true number of inliers. The recall R of an algorithm is computed as:

$$R = \frac{|\mathcal{I}_{GT} \bigcap \mathcal{I}(\theta^*)|}{|\mathcal{I}_{GT}|}. \tag{2.2}$$

This measures the capacity of an algorithm to find all good points and not just be too restrictive in its classification. Good recall means more valid points will be added to the 3D model, which will help further reconstruction.

As σ-consensus and σ-consensus++ introduced in [6, 7] remove entirely the need for inlier/outlier classification in their method by removing the threshold and using a weighted estimation of the best model, we cannot compute directly the precision and the recall. However, as the authors of those papers present interesting results and claim to reach state-of-the-art performance, we wanted to include those in our evaluation. Instead of using the arbitrary threshold used by those algorithms — see Section 1.4.3 — to compute inliers, as this would not exploit the novel weights computation of the algorithms, we introduce four metrics that can be analysed jointly in order to conclude on the necessity to adapt σ-consensus to a full 3D pipeline. The four metrics we analyse are a weighted

equivalent of precision P_w and recall R_w as well as the best precision obtainable to reach the recall of AC-RanSaC [60, 61] $P_{R(\text{AC-RanSaC})}$ and the best recall obtainable to reach the precision of AC-RanSaC $R_{P(\text{AC-RanSaC})}$.

The first two metrics give a simple insight of whether the data points with high weights are true inliers or not. We consider "proposed inlier" data points, points that have residual below the maximum allowed residual σ_{max} of MAGSAC and MAGSAC++. The weighted precision P_w is computed as follows:

$$P_w = \frac{\sum_{p \in \mathcal{I}(\theta^*)} w(p) * \mathbb{I}_{GT}(p)}{\sum_{p \in \mathcal{I}(\theta^*)} w(p)} \tag{2.3}$$

where, for a proposed inlier data point $p \in \mathcal{I}(\theta^*)$, $w(p)$ is the weight returned by σ-consensus or σ-consensus++ and $\mathbb{I}_{GT}$ is the indicator function of the ground truth inliers. By doing the ratio of the sum of true inliers weights over all weight, we want to reveal whether true inliers have enough importance in the final estimation. The weighted recall R_w is computed as follows:

$$R_w = \frac{\sum_{p \in \mathcal{I}(\theta^*)} w(p) * \mathbb{I}_{GT}(p)}{|\mathcal{I}_{GT}| \times \max_{p \in \mathcal{I}(\theta^*)}(w(p))}. \tag{2.4}$$

Here, we consider that true inliers should have the maximum weight possible for the estimation to be perfect, so we do the ratio of the sum of true inlier weights by the maximum weights across all ground truth inliers. As these metrics depend on points with residual under σ_{max} which is voluntarily higher than the true threshold, they will always be under evaluated. We keep this in mind when comparing them to other metrics.

The two other metrics, best precision at AC-RanSaC recall and best recall at AC-RanSaC precision, are computed by sorting data points by residual and computing the precision and recall at each possible residual. For the first metric, $P_{R(\text{AC-RanSaC})}$, the residual that gives the best precision when recall is at least equal to the mean recall of AC-RanSaC for this experiment setting[2]. For the second one, $R_{P(\text{AC-RanSaC})}$, it is the residual that yields the best recall for data points with at least the same precision as the mean precision of AC-RanSaC for the given experiment setting[2].

For all situations were precision and recall is used, another metric can be considered, the F1-Score F_1. It is the harmonic mean of the two metrics:

$$F_1 = 2\frac{P \times R}{P + R} \tag{2.5}$$

where precision P and recall R can be substituted by the different metrics computed specifically for MAGSAC and MAGSAC++ if need be. Usually, performance in precision

[2] An experiment setting is the inlier noise and outlier ratio. For each setting, multiple run are averaged to get clearer results. See Section 3.2

and performance in recall constitute a trade-off, with some hyperparameter threshold that will increase recall but reduce precision or the other way around. As, in our case, this inlier/outlier threshold is evaluated directly by the algorithms, the choice of trade-off should be inherent to design choices of the RanSaC algorithms. So, if algorithms do not change behaviour depending on the settings, F1-Score can be used to compare algorithms. We see in Section 3.4 that it is the case, no algorithms change behaviour relatively to this trade-off depending on test setting.

2.3 Model and inlier generation

2.3.1 Generic method

Our proposed solution to create realistic yet controlled data relies on removing unknown elements from real data and replacing them with artificial perturbation. From unlabelled features with unknown noise and underlying model, the classic RanSaC pipeline is run. It gives a putative model with inlier/outlier classification. This model is supposedly realist as, bare a failure of the pipeline, it is close to the true model. Thus, it can be considered as an artificial ground truth model. It is not the true ground truth but it will be the basis on which we build our artificial data generation. From the artificial ground truth model, ground truth inliers are generated by using the n_{in} inliers proposed by the RanSaC run and projecting them on the model to create artificial ground truth inliers. Again, they are not the true ground truth inliers, but they are noise-free inliers relatively to our artificial ground truth model. This method ensures the inlier spatial distribution is realistic as it is issued from a real feature extraction. It removes the impact of the choice of the inlier repartition for a random artificial model. It also ensures that the benchmarked algorithms will be tested on their ability to find a model as close as possible to our ground truth and not their ability to reproduce the estimations made to generate the test data.

At this point, we have a ground truth model and ground truth inliers which are noise free. To test the sensitivity of algorithms to inlier noise and the capacity to select a suitable threshold, we can add noise to these inliers. RanSaC algorithms usually make some sort of assumption on the inlier noise. The original paper [26] assumes normal distribution, as well as [89, 74, 41]. Some paper assume uniform distribution within the inlier/outlier threshold, such as [83, 17, 6, 7].

The selected noise needs to be controlled so that an unequivocal inlier/outlier cut-off exists. Given the sets of noisy inliers $\mathcal{I}_{noisy}$ and generated outliers $\mathcal{O}_{gen}$, this cut-off is defined so that:

$$\max_{p_i \in \mathcal{I}_{noisy}} D(p_i) \leq \min_{p_o \in \mathcal{O}_{gen}} D(p_o). \tag{2.6}$$

When launching an estimation with AC-RanSaC and arbitrary precision, we observe like in figures 2.8, 2.9, 2.10 and 2.11 an error distribution of proposed inliers that is closer

to a normal distribution with a slight offset than to a uniform one, though not perfectly Gaussian. However, to be able to respect the desired cut-off (2.6), the Gaussian would need to be truncated to a certain value, to ensure that no points are generated too far from ground truth model and create a dataset where outliers are very far from the ground truth model. During preliminary experiments we observed that the inlier distribution made little to no impact on the performance of algorithms. Thus, to get more control over the true inlier/outlier threshold, instead of using a truncated Gaussian distribution, we chose a uniform distribution for the inlier noise.

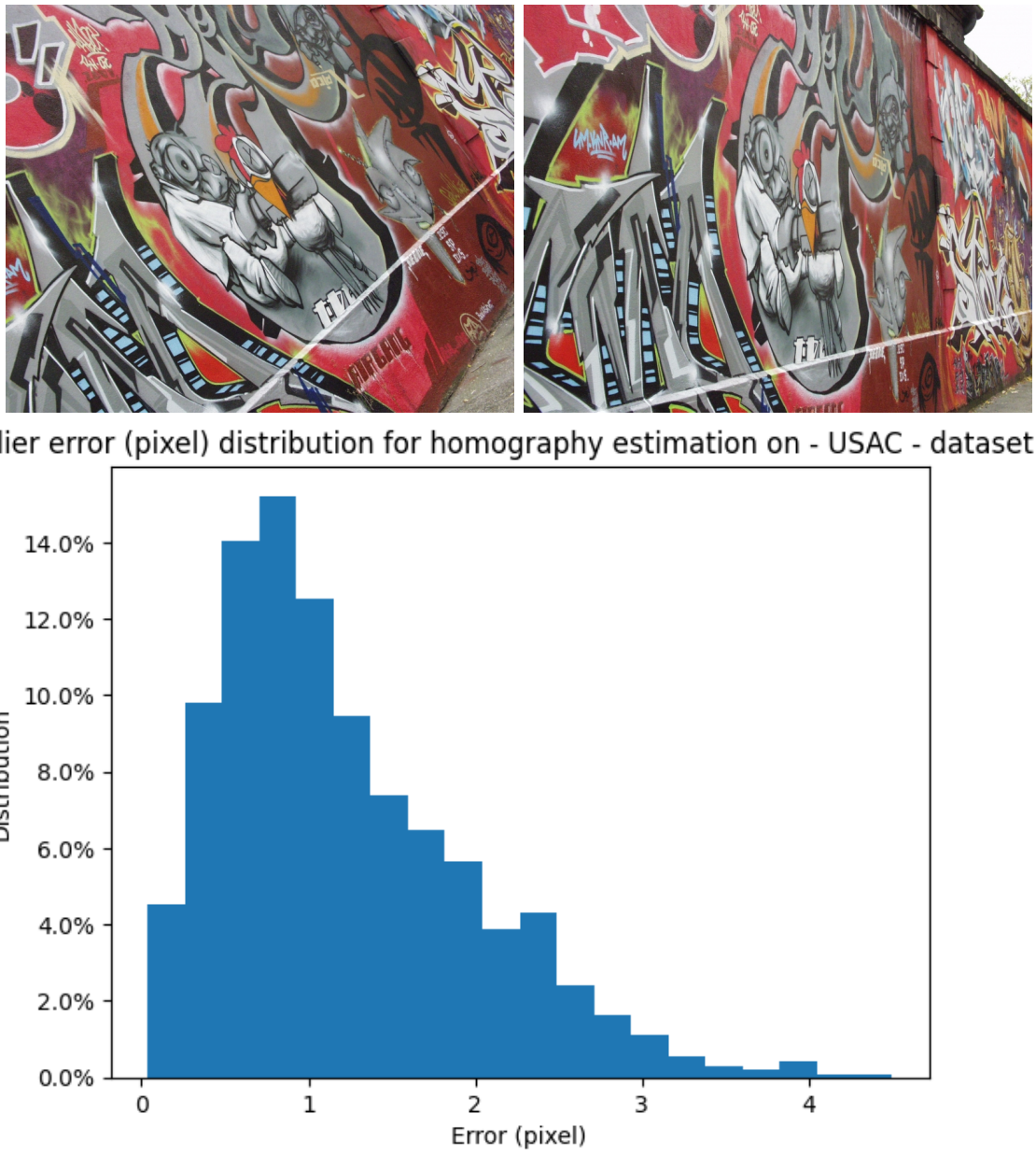
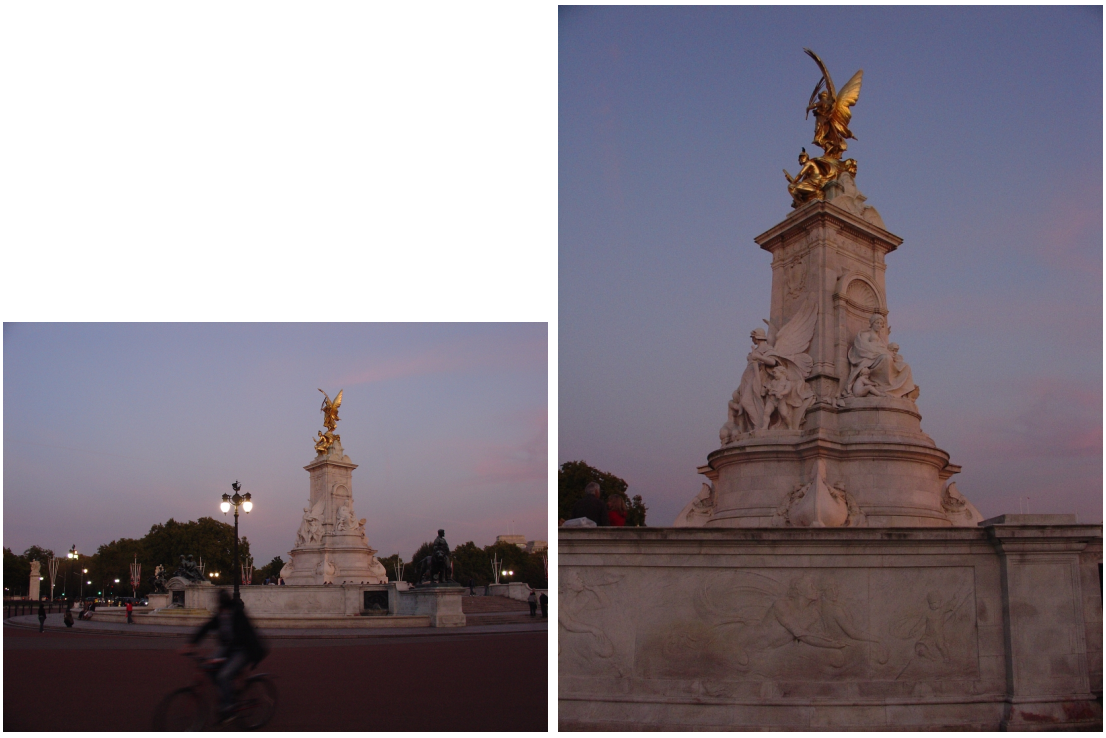


Figure 2.8: Top: Images on which AC-RanSaC was computed. Bottom: Distribution of residual (in pixel) of inliers in percentage of inlier points after estimation by AC-RanSaC with arbitrary precision on the first image of the homography USAC dataset [73].

[3]See Section 3.2 for more detail about our sub-dataset extracted from MegaDepth.



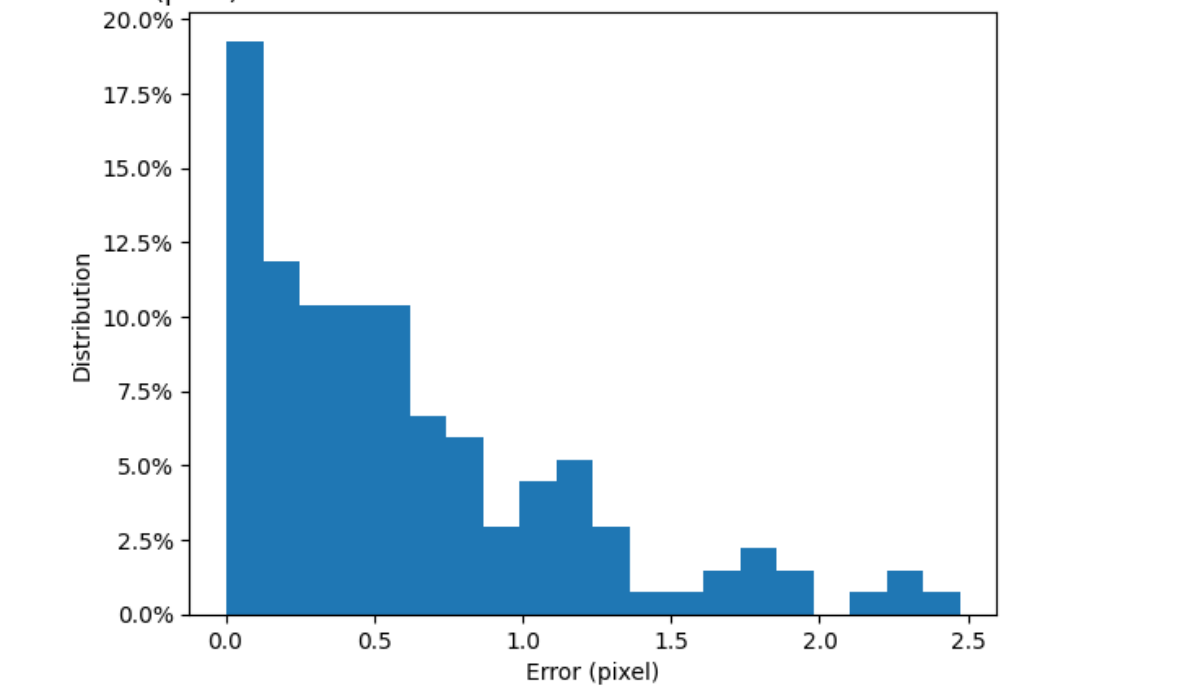


Figure 2.9: Top: Images on which AC-RanSaC was computed. Bottom: Distribution of residual (in pixel) of inliers in percentage of inlier points after estimation by AC-RanSaC with arbitrary precision on the fith image of the fundamental matrix USAC dataset [73].

This way, for a theoretical noise level σ_{noise} we are sure inliers are distributed around the ground truth model within a known bound so outliers can be ground truth outliers, *i.e.* a point further from the model than the maximum inlier error. The method to generate outliers properly depends on the application and thus is detailed in Section 2.4.

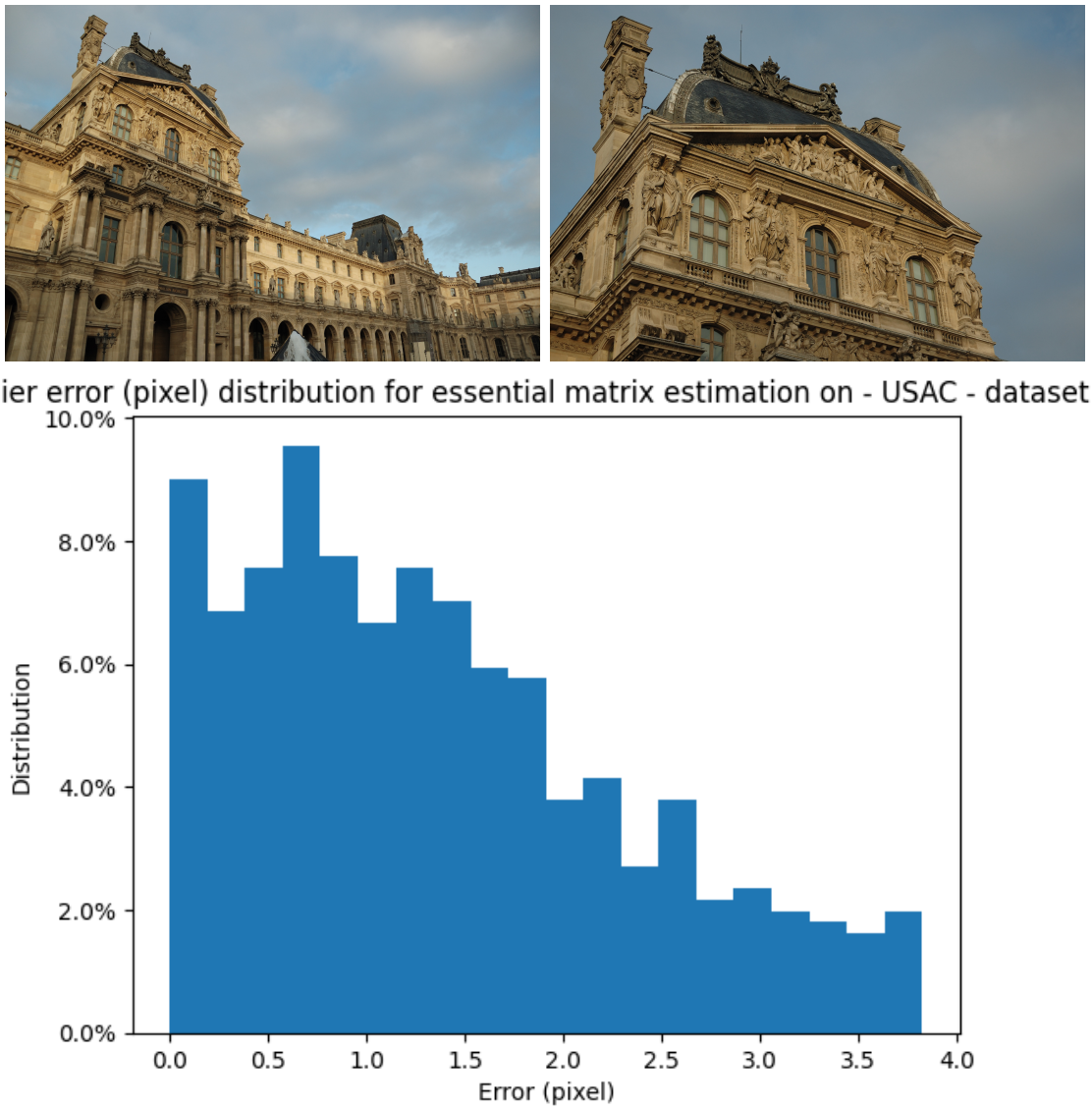


Figure 2.10: Top: Images on which AC-RanSaC was computed. Bottom: Distribution of residual (in pixel) of inliers in percentage of inlier points after estimation by AC-RanSaC with arbitrary precision on the fourth image of the essential matrix USAC dataset [73].

2.3.2 Generation for MVS and SfM

As presented in Chapter 1 we focus our study on two-view geometry and Perspective-from-n-Points problems. The presented method can easily be applied to each estimation problem with little adaptation.

For two-view geometry, the input data is a pair of images. From this image pair, matches are extracted using SIFT [52, 53]. Those matches are used as input data containing both inliers and outliers. SIFT is used as it is the most used in the literature and especially in [79]. For PnP, the input data is a 3D set of points and a 2D set of matching points on an image plane. Then AC-RanSaC [60, 61] is run with arbitrary precision and up to 10^6 iterations. Preliminary work used RanSaC to get the initial estimation, but AC-RanSaC proved more robust to tackle various datasets and no noticeable differences

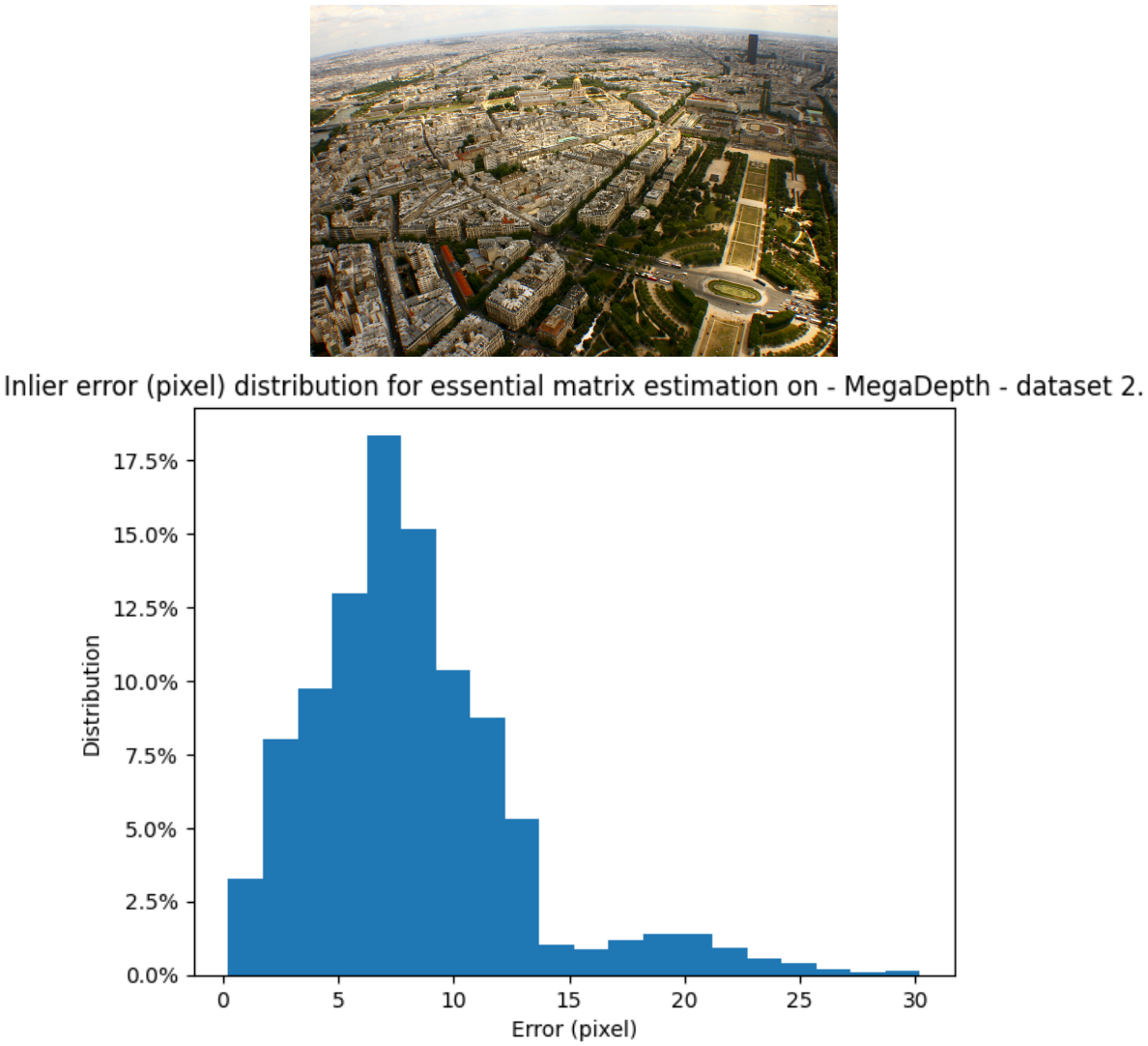


Figure 2.11: Top: Image on which AC-RanSaC was computed. Bottom: Distribution of residual (in pixel) of inliers in percentage of inlier points after estimation by AC-RanSaC with arbitrary precision on the second image of the PnP MegaDepth sub-dataset[3] [46].

were observed in results after switching algorithms. The main advantage of using AC-RanSaC is that it has a refinement step that iterates over the first good inlier set to get a better model even when the stopping criterion is met. This estimation gives a hopefully coherent model. That model can be quickly checked to ensure its coherence with the image pair or input data. The check consists in a simple observation of the reconstruction for homography and of the consistency of epipolar lines for Fundamental and Essential matrix estimation. For PnP visualization is more complex as it requires observing the pose in a 3D representation. This visual check is not necessary to run the pipeline as any model could be used as ground truth but it is useful to ensure that the model is close to the real one. If the model is valid, the proposed inliers of AC-RanSaC can be aligned to fit it. It gives us the artificial ground truth inliers on which to apply the uniform noise. Noise is only added on one side of the new matches as we want to control its value and

adding it to both side would risk cancelling it out or accumulating itself. A final check is usually added to ensure that the noisy inlier stays in the realm of possible values, for instance that it does not get out of the input image.

For homographies, one image side of the image pair is selected as reference and the features from this image of each inlier match are kept. The matching features in the other image are discarded and replaced by the projection of the point with the ground truth homography, see Figure 2.12. The inlier perturbations are then drawn uniformly in $[-\sigma_{noise}, \sigma_{noise}]^2$ and added to the new points to create the input noisy matches.

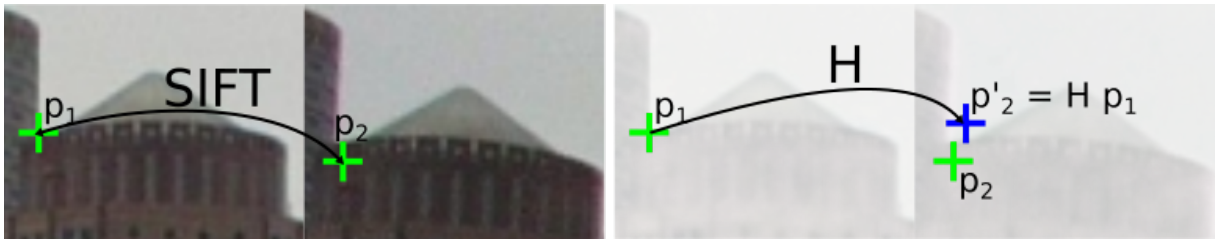


Figure 2.12: From an imperfect match (p_1, p_2) considered inlier by AC-RanSaC, the "perfect match" (p_1, p'_2) is constructed such that $p'_2 = Hp_1$ using a realistic homography H given by AC-RanSaC. Figure originates from [77].

For fundamental and essential matrices, one side from the image pair is selected as reference as well. However, the position of the feature in the other image is not discarded but orthogonally projected on the epipolar line associated with the feature in the reference image, see Figure 2.13. The noise is then drawn in $[-\sigma_{noise}, \sigma_{noise}]$ and added perpendicularly to the epipolar line to the new point to create the input noisy match.

Figure 2.13: From an imperfect match (p_1, p_2) considered inlier by AC-RanSaC, the "perfect match" (p_1, p'_2) is constructed using p'_2 the orthogonal projection of p_2 on the epipolar line $\mathcal{L}_1 = Fp_1$ where F is a realistic fundamental matrix given by AC-RanSaC. This does not guarantee that p'_2 represents the same physical point as p_1, but that some 3D point at possibly different depth projects exactly at p_1 and p'_2. Figure originates from [77].

For Perspective-from-n-Points, the 3D point is kept as reference and the matching 2D point is discarded. Then, the 3D point is projected on the image plane and noise is added in $[-\sigma_{noise}, \sigma_{noise}]^2$ in the image similarly to the homography setup.

2.4 Outlier generation

2.4.1 Choice of outlier distribution

Once noisy inliers are generated we can generate ground truth outliers. A "ground truth outlier" is an element lying further from the ground truth model than the worst noisy inlier. Thanks to this distinction, metrics can be computed accurately. When algorithms make assumption about the outliers distribution, it is usually uniform in the whole image as in [64, 41, 6, 7, 83, 17]. However, a data point inside the inlier region should be considered an inlier so we restrict outliers to have residual at least bigger than the maximum inlier residual, see equation (2.6). To do so we could simply choose a random match in $\mathcal{S}$ with uniform distribution. However, such a random distribution is not challenging enough for RanSaC algorithms. In figures 2.14, 2.15, 2.16 and 2.17 we compare the distribution of residuals of outliers estimated by AC-RanSaC to the estimated model and points generated by our method and points generated by uniform distribution in the match space. This is just a qualitative analysis of the distribution but our method is usually closer to the estimated distribution than uniform outliers.

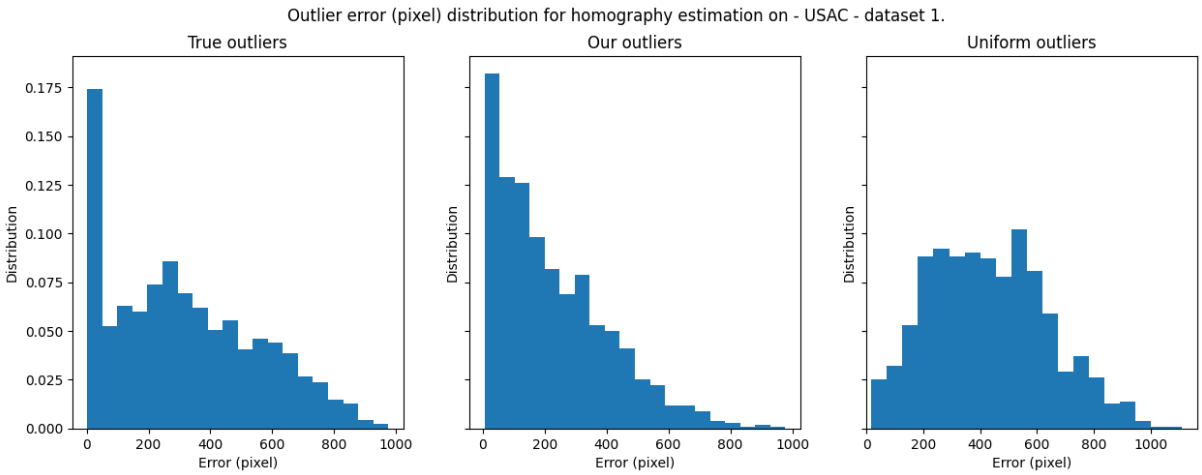


Figure 2.14: Distribution of residual (in pixel) of outliers in percentage of outliers points on the first image pair of the homography USAC dataset [73]. See Figure 2.8 for image pair. From left to right: Outliers estimated by AC-RanSaC with arbitrary precision, outliers generated with our methodd, uniformly distributed outliers.

That is why we propose a distribution of data points in error-space. It means that, for each outlier, its residual is drawn uniformly from the set of possible residuals. This method is more complex but brings more realistic challenges to the RanSaC algorithms as presented below.

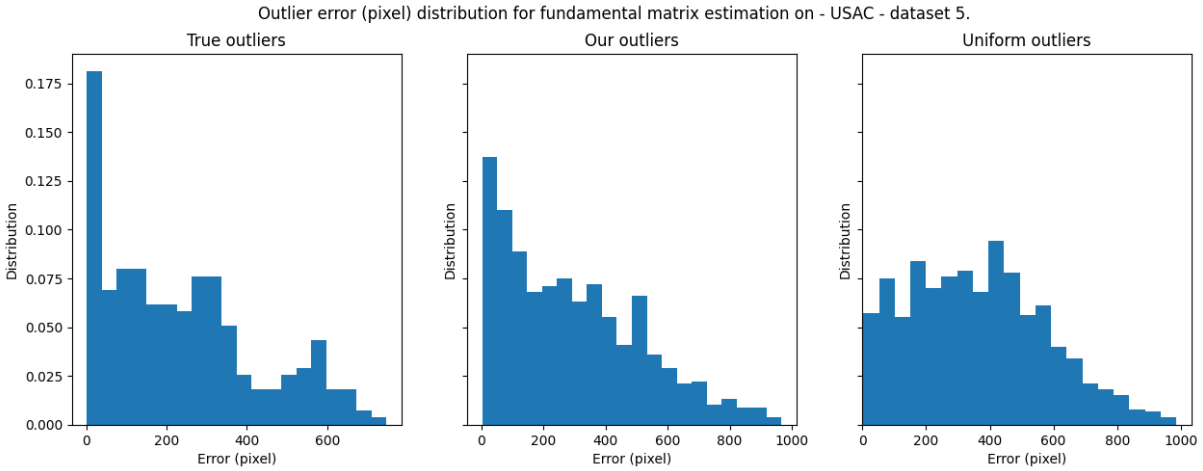


Figure 2.15: Distribution of residual (in pixel) of outliers in percentage of outliers points on the fifth image pair of the fundamental matrix USAC dataset [73]. See Figure 2.9 for image pair. From left to right: Outliers estimated by AC-RanSaC with arbitrary precision, outliers generated with our methodd, uniformly distributed outliers.

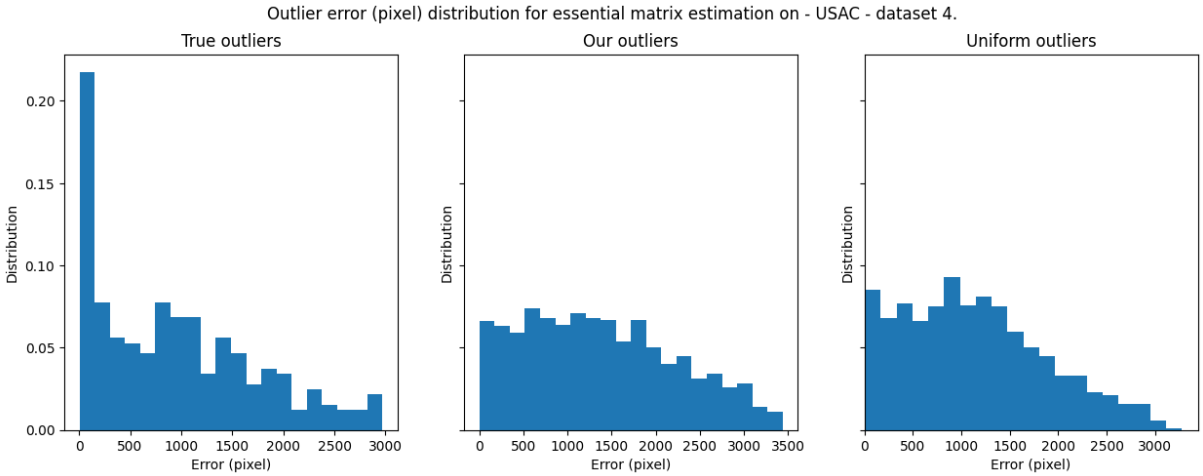


Figure 2.16: Distribution of residual (in pixel) of outliers in percentage of outliers points on the fourth image pair of the essential matrix USAC dataset [73]. See Figure 2.10 for image pair. From left to right: Outliers estimated by AC-RanSaC with arbitrary precision, outliers generated with our methodd, uniformly distributed outliers.

2.4.2 For MVS and SfM

In Two-View Geometry and PnP, we use the same logic to generate an outlier match $p_o = (q_1, q_2)$ according to this distribution. First a point q_1 is drawn uniformly in one side of the match. This point will form the first part of the match. Then, its perfect match and inlier region is computed. From this perfect match, a direction is selected in which to place the second point q_2. With the direction and inlier region, the range of possible residuals can be computed. By drawing the perturbation uniformly in this range, the outlier match $p_o = (q_1, q_2)$ is generated. This method does not generate a uniform distribution in error space but it ensures that enough matches are close enough to the

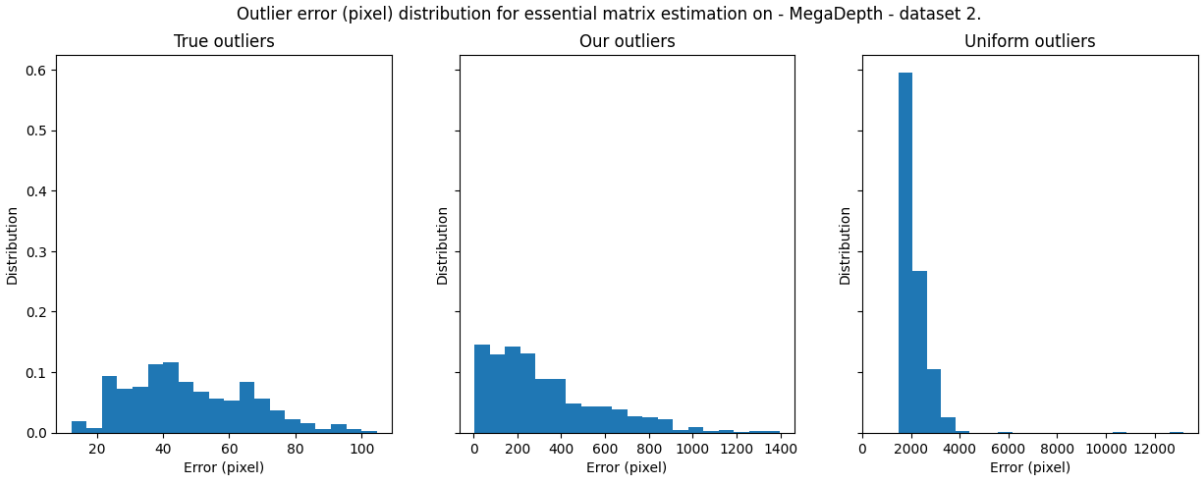


Figure 2.17: Distribution of residual (in pixel) of outliers in percentage of outliers points on the second image of the MegaDepth dataset [46]. See Figure 2.11 for image pair. From left to right: Outliers estimated by AC-RanSaC with arbitrary precision, outliers generated with our methodd, uniformly distributed outliers. Note that the right image starts around 2000 pixel error.

ground truth model to present an actual challenge to the RanSaC algorithm.

Specifically, for point-to-point residual models, like homography estimation and PnP, the principle is the same. In the first case, a 2D point is drawn in one of the two images and its projection on the second image is computed. In the second case, a 3D point is drawn in a bounding box bb around inlier points defined as:

$$\begin{aligned} bb = &[r_{aug} \min_{\forall i \in [n_{in}]}(x_i), r_{aug} \max_{\forall i \in [n_{in}]}(x_i)] \times \\ &[r_{aug} \min_{\forall i \in [n_{in}]}(y_i), r_{aug} \max_{\forall i \in [n_{in}]}(y_i)] \times \\ &[r_{aug} \min_{\forall i \in [n_{in}]}(z_i), r_{aug} \max_{\forall i \in [n_{in}]}(z_i)] \end{aligned} \tag{2.7}$$

where r_{aug} is a small factor — set to 1.1 in our experiments — to increase the range of outliers around the inliers and $\mathcal{I}_{noisy} = \{p_i = (x_i, y_i, z_i), \forall i \in [n_{in}]\}$. The 3D point is then projected onto the 2D image. The inlier region is then a circle of radius $\max_{p_i \in \mathcal{I}_{noisy}} D(p_i)$ around this projection. A random direction θ is uniformly selected and, from this direction, the maximum possible residual is computed to ensure the generated point does not exit the image. To do so, once the perfect match $q_2' = (u, v)$ is generated, depending of which zone of the image is selected, the distance to the border max_{offset} is computed as

such:

$$max_{offset} = \begin{cases} \frac{w-u}{\cos(\theta)} & \text{, if } \theta \in [0, \theta_1] \bigcup [\theta_4, 2\pi] \\ \frac{h-v}{\cos\left(\theta - \frac{\pi}{2}\right)} & \text{, if } \theta \in [\theta_1, \theta_2] \\ \frac{u}{\cos(\theta - \pi)} & \text{, if } \theta \in [\theta_2, \theta_3] \\ \frac{v}{\cos\left(\theta - \frac{3\pi}{2}\right)} & \text{, if } \theta \in [\theta_3, \theta_4] \end{cases} \tag{2.8}$$

where the image is of width and height (w, h) and the values of $\theta_i, \forall i \in [4]$ depend on the image zones, as presented in Figure 2.18:

$$\theta_1 = \arctan\left(\frac{h-v}{w-u}\right) \tag{2.9}$$

$$\theta_2 = \arctan\left(\frac{u}{h-v} + \frac{\pi}{2}\right) \tag{2.10}$$

$$\theta_3 = \arctan\left(\frac{v}{u} + \pi\right) \tag{2.11}$$

$$\theta_4 = \arctan\left(\frac{w-u}{v} + \frac{3\pi}{2}\right) \tag{2.12}$$

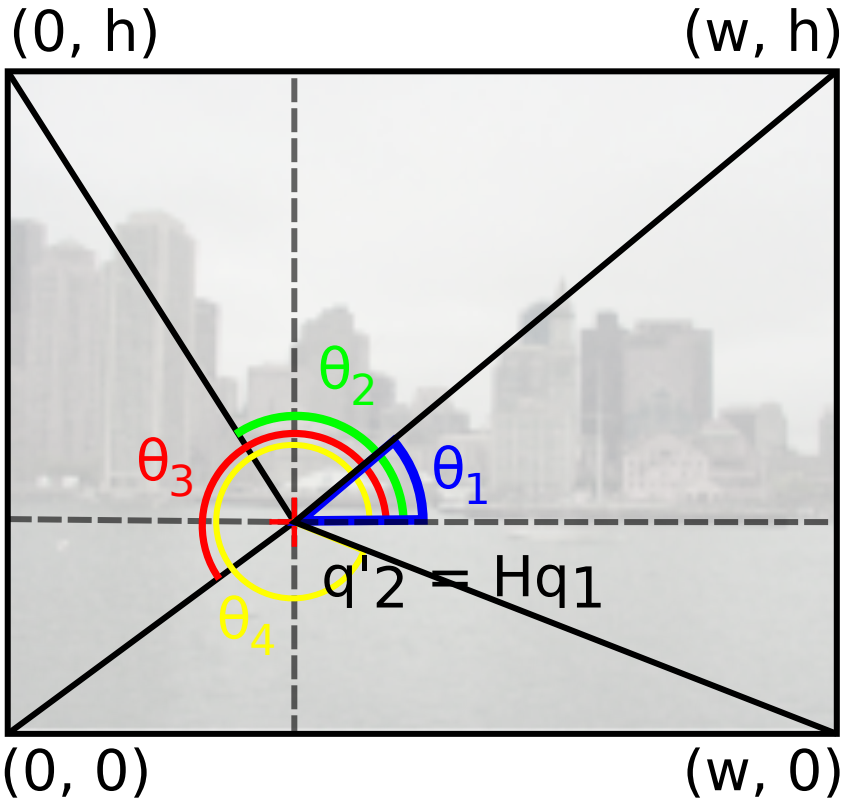


Figure 2.18: Illustration of the four angles θ_1, θ_2, θ_3 and θ_4 on the second image given the image q'_2 of a point q_1 in the first image.

The perturbation is then drawn uniformly in $[\max_{p_i \in \mathcal{I}_{noisy}} D(p_i), max_{offset}]$ and applied in the direction θ. This gives q_2 to create $p_o = (q_1, q_2)$, see Figure 2.19

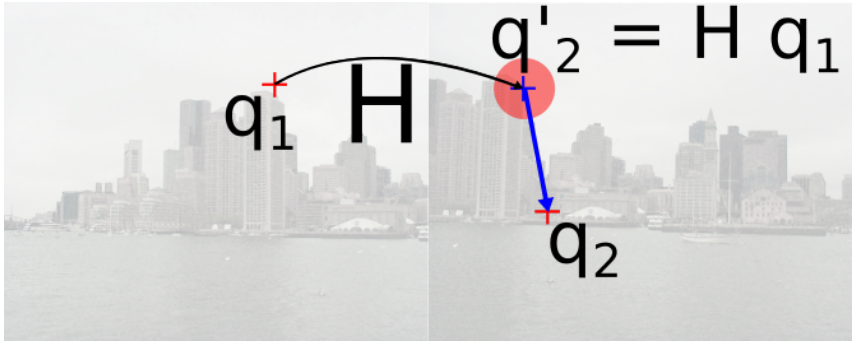


Figure 2.19: A random point q_1 is drawn from the left image. Using the ground truth model H, its perfect match $q'_2 = Hq_1$ is computed. Then a direction and a distance to q'_2 are drawn uniformly in order to create q_2 so that it remains in the image and out of the inlier zone (marked in red) defined by the inlier noise level. Figure originates from [77].

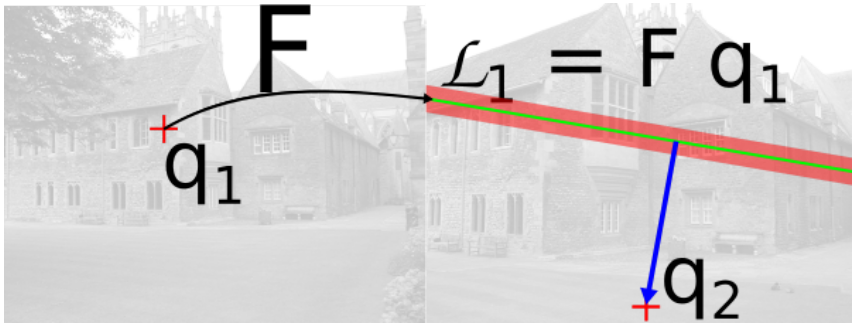


Figure 2.20: A random point q_1 is drawn from the left image. Using the ground truth model F, the epipolar line $\mathcal{L}_1 = Fq_1$ is computed. Then a position on this line and a distance to $\mathcal{L}_1$ are drawn uniformly in order to create q_2 so that it remains in the image and out of the inlier zone (marked in red) defined by the inlier noise level. Figure originates from [77].

Regarding point-to-line error models, like fundamental and essential matrix estimation, the same logic is applied with small adaptations. To create an outlier $p_o = (q_1, q_2)$, a 2D point q_1 is drawn in one of the images and the associated epipolar line is computed. A point is then randomly drawn on the epipolar line while making sure it stays in the image. To do so, three cases have to be considered. If the epipolar line is horizontal or vertical, it just requires to select the abscissa and ordinate and project on the epipolar line. In any other scenario, we compute the min min_{abs} and max abscissa max_{abs} to generate the point on the epipolar line. For an epipolar line $0 = ax + by + c$, those values depend on the sign of the $slope = \frac{-a}{b}$:

$$min_{abs} = \begin{cases} \max\left(\frac{-c}{a}, 0\right) & \text{, if } slope > 0 \\ \max\left(\frac{h}{slope} - \frac{c}{a}, 0\right) & \text{, if } slope < 0 \end{cases} \tag{2.13}$$

$$max_{abs} = \begin{cases} \min\left(\frac{h}{slope} - \frac{c}{a}, w\right) & \text{, if } slope > 0 \\ \min\left(\frac{-c}{a}, w\right) & \text{, if } slope < 0 \end{cases} \tag{2.14}$$

An abscissa is uniformly drawn in $[min_{abs}, max_{abs}]$ and its image is computed on the epipolar line. In all three cases, from the reference point on the epipolar line, the max_{offset} is computed similarly to the homography cases. Instead of choosing a random direction in $[0, 2\pi]$, the two possibilities are up and down from the epipolar line — or left and right if the line is vertical. Then, equation (2.8) is applied with $\theta = \arctan(slope) \pm \frac{\pi}{2}$ to get max_{offset} and the perturbation can be drawn and applied in perpendicularly to the epipolar line. This gives q_2 and thus the outlier match. See Figure 2.20.

Chapter 3

Analysis of Adaptative RanSaC Methods for MVS and SfM Tasks

In the previous chapter we presented how to create datasets designed to evaluate RanSaC algorithms as classification methods to increase the quality of the pipeline. As such we propose a benchmark using this method to compare automatic threshold RanSaC algorithms to find out their strengths and shortcomings depending on the situation.

We simultaneously try to observe whether our data generation methodology reveals different behaviour depending on the generation parameters. Of course, different estimation problems or input data lead to different values for each metrics but if the general behaviour of a particular algorithm and how it compares to the others is not impacted then the method is appropriate to analyse an algorithm accurately.

Finally, our goal is to establish when to use each algorithms and which to include in ColMap [79] to try and remove the user threshold in the whole pipeline.

3.1 Tested algorithms

For this benchmark we used only a subset of the algorithms presented in Section 1.4. Methods were chosen based on the following criteria: being a method that does not require a hard user set threshold — hyper-parameters for max range of thresholds are allowed but they will not be tuned — tailored for MVS and SfM, and availability of an implementation or feasibility of implementation. Classical RanSaC [26] is used as a baseline to validate performance of other methods to ensure they do better than a fixed threshold. It is tested with two thresholds to better assess its possible performances. The selected threshold-free method are MUSE [58], StaRSaC [13], A-Contrario RanSaC (AC-RanSaC) [61], Likelihood Ratio Test (LRT) [17], Marginalizing Sample Consensus (MAGSAC) [6] and two related variations Fast-AC-RanSaC[65] and MAGSAC++ [7]. Multi-model specific methods like [38, 87, 55] are excluded from the benchmark as we concentrate our benchmark around the classification performance of the algorithms when faced with a single model.

RanSaC, presented in Section 1.4.1, was used with two thresholds, σ_1 and σ_2, in pixel to offer a baseline of performance of non-adaptative methods with varying thresholds.

Algorithm RanSaC is described in algorithm 1 but we modified its stopping criterion so that we have confidence β that at least $n = 5$ good samples have been drawn. The choice of $n = 5$ is a balance to increase the confidence without slowing down the computations too much. Equation (1.24) is changed to:

$$it_{max} = \frac{\log(1-\beta)}{\log(1-\epsilon^s)} + \frac{-\log\left(\sum_{i=0}^{n-1}\left(\frac{\epsilon^s}{1-\epsilon^s}\right)^i\right)}{\log(1-\epsilon^s)}, \tag{3.1}$$

with the first term being the classic formula and the second one a positive value that increases the required number of iteration to be confident that $n = 5$ elements are drawn. The implementation was adapted from [63].

All other methods included in the benchmark remove the need for a user-set inlier/outlier threshold. They can present some hyperparameters that determine a maximum range of thresholds to consider. MUSE [58] is an adaptation of Least Median Squares [44]. It uses scale estimates as objective function to rank models using the standard iterative sampling of minimal samples of RanSaC. The authors claim that the new objective function is more robust to higher outlier ratios and can be used to detect multiple models by iteratively removing points using a well chosen scale estimate. This objective, the unbiased scale estimate $\hat{\sigma}_\theta$ of a model with parameters θ, is computed by sorting all N residuals D_k of all data points and evaluating the scale estimate s_k for the k^{th} residual:

$$s_k = \frac{D_k}{0.5\left(1 + \frac{k}{N+1}\right)}, \tag{3.2}$$

and finding minimal scale $s^*_\theta = s_{k'}$ which gives the unbiased scale estimate as:

$$Q(\sigma, \theta) = \hat{\sigma}_\theta = \frac{s^*_\theta}{E[\min v_{k'}]}, \tag{3.3}$$

where $\min v_{k'}$ is the k'^{th} scale estimate of a standardized normal distribution. This algorithm does not include a stopping criteria when confidence β is reached, thus we added the RanSaC one as it adapts seamlessly to the framework. The implementation was adapted from `https://github.com/vxl/vxl`.

Likelihood Ratio Test (LRT) [17] estimates the likelihood that the data presents non-random structure and tries to find the best model to describe the data if it is indeed non-random. It is based on points being drawn from two uniform distributions, one on the inlier space $I(\theta, \sigma)$ and the other on the whole space, the outlier background distribution. A theoretical ratio ρ of points are inliers and the null hypothesis is $\rho = 0$ while the alternative hypothesis requires to determine θ, σ and ρ. The likelihood ratio test of both hypotheses gives, after some rearrangements, the quality function of a model:

$$Q(\sigma) = L(\epsilon, \sigma) = 2|\mathcal{P}| \left(\epsilon \log \frac{\epsilon}{p_\sigma} + (1-\epsilon) \log \frac{1-\epsilon}{1-p_\sigma} \right), \tag{3.4}$$

with inlier ratio $\epsilon = k(\sigma)/|\mathcal{P}|$ and σ spanning a predefined list $\{\sigma_{min}, \ldots, \sigma_{max}\}$. The author did not provide the chosen range of σ so we chose $\sigma_i = 0.25 * 2^{\frac{i}{2}}, \quad \forall i \geq 0, \sigma_i \leq \sigma_{max}$. This range gives reasonable values and distribution for σ. This objective must be above a certain threshold c, computed from a confidence α and a lookup table for the χ^2 distribution with $d+2$ degrees of freedom which the likelihood statistic approaches, to ensure the control over the type I error. To find the best threshold, at each iteration, the inlier ratio is computed for a range of thresholds and the threshold with best log-likelihood is selected. The type II error is controlled by a confidence β that works similarly to the stopping criteria of RanSaC. Moreover, at each iteration, the empiric inlier ratio ϵ is compared to minimal inlier ratio $\underline{\epsilon}$ given the so-far-the-best model. This gives an early bailout strategy to stop the residual computation if the model should be discarded which requires a new parameter γ to control the increased risk. This early bailout strategy shifts the value of the stopping criteria:

$$it_{max} = \frac{\log(1-\beta)}{\log(1-\underline{\epsilon}^s \times \gamma)}, \tag{3.5}$$

where $\underline{\epsilon}$ is the minimal value of the possible future inlier ratios to find a better model. We reimplemented this algorithm in [75].

StaRSaC [13] simply proposes to test RanSaC with multiple thresholds and find which one offers the most stable results. To do so, the quality function Q is defined as the variance over parameters computed by evaluating RanSaC $n_{starsac}$ times for each threshold σ in the same range as LRT above. The authors expect that a good threshold should have small variance of model parameters as the same data points should be inliers for the right threshold.

$$Q(\sigma) = -\frac{1}{n_{starsac}} \left(\sum_{i=0}^{n_{starsac}} \left(\bar{\theta}_i(\sigma) - \theta_i(\sigma) \right)^2 \right), \tag{3.6}$$

where $\theta_i(\sigma)$ is the model parameter at threshold σ for $i \in [n_{starsac}]$ and $\bar{\theta}_i(\sigma) = \frac{1}{n_{starsac}} \sum_{i=0}^{n_{starsac}} \theta_i(\sigma)$ is the mean of estimated parameters. We reimplemented this algorithm ourselves.

Authors of *A-Contrario*-RanSaC (AC-RanSaC) [60, 61] use the Number of False Alarm (NFA) to find whether the model was selected by chance from random uniform outliers. Testing all data points residual ϵ_k, in increasing order, as potential inlier/outlier threshold the NFA is the number of tests times the probability of k random data points falling within ϵ_k of the model. This probability is computed as $(\epsilon_k^d \alpha_0)^{k-s}$ with α_0 the probability of a random data point having residual at most 1. The number of tests is computed as the number of possible model given the estimation algorithm $N_{estimator}$, see Section 1.2, times

all possible residual to test $|\mathcal{P}| - s$, times the number of possible sets of inliers of size k $\binom{|\mathcal{P}|}{k(\sigma)}$, times the number of possible sample in this inlier set $\binom{k(\sigma)}{s}$. This yields the quality function as:

$$Q(\theta) = -NFA(\theta, \sigma) = N_{estimator}(|\mathcal{P}| - s)\binom{|\mathcal{P}|}{k}\binom{k}{s}(\epsilon_k^d \alpha_0)^{k-s}. \tag{3.7}$$

This quality function controls the type I Error. An upper bound of the NFA, NFA_{max}, is set and, once an inlier set and threshold verify this bound, 10% of the maximum number of iterations are reserved to improve the model by iterating over this so-far-the-best inlier set. To reduce computation time, an upper bound to the tested residual σ_{max} can be set but it works as well on tests with all data points. The implementation was adapted from [63].

As presented in Section 3.3, AC-RanSaC is relatively slow compared to other algorithms. This is usually due to its sorting step which can take the same amount of time than the residual computation. In OpenMVG [65], the original paper author proposed an improvement quantizing the residuals and using a histogram to classify errors instead of sorting all the residuals. To compute the Number of False Alarms, the possible inlier/outlier threshold are set to the values separating the n_{bin}, B_i, $\forall i \in [n_{bin}]$ and the number of inliers of all bins $k(B_i)$ can be computed in $\mathcal{O}(n)$ instead of $\mathcal{O}(n \log n)$:

$$Q(\theta) = NFA(\theta, B_i) \sim \binom{|\mathcal{P}|}{k(B_i)}\binom{k(B_i)}{s}(B_i^d \alpha_0)^{k(B_i)-s}. \tag{3.8}$$

The implementation was adapted from OpenMVG [65].

MAGSAC [6] introduces σ-consensus to remove the need for inlier/outlier threshold. σ-consensus weighs data points to do local optimization using a weighted least square fitting. At each iteration, samples are drawn uniformly and then a model is estimated, or multiple models depending on the estimator. Then, each data point with residual lower than σ_{max} gets sorted and ordered in $n_{partition}$ partitions of max residual $\sigma_{max,i}$. A model θ_i is estimated from the increasing sets of pseudo-inliers of each partition and each point p gets a weight $W(p)$ depending on its residual to the estimated models:

$$W(p) = \sum_{i=1}^{n_{partition}} \exp \frac{-D(p, \theta_i)^2}{\sigma_{max,i}^2}. \tag{3.9}$$

This equation is not faithful to equation (6) in [6], but is the one used in their implementation. The σ-consensus optimization's goal is to improve the quality of the model estimated at each iteration. To determine the best model to select during the RanSaC procedure, MAGSAC assumes uniform inliers and outliers in different space and derives

the likelihood of a model from it.

$$Q(\theta) = \sum_{p, D(p,\theta)<\sigma_{max}} \sum_{i=1}^{n_{partition}} \left(1 - \frac{D(p,\theta)^2}{\sigma_{max,i}^2}\right), \tag{3.10}$$

which deviates again from the proposed marginalised quality function in equation (5) of [6]. The implementation was adapted from [6].

MAGSAC has been improved by the original authors as MAGSAC++ [7]. σ-consensus is replaced by σ-consensus++ that does reweighted least square fitting at each iteration. The hypotheses are also changed: inliers residuals are distributed according to a χ-distribution with a uniformly drawn noise level in $\mathcal{U}(0, \sigma_{max})$. The author also propose a new sampling inspired from [89] but it was not made available in their scripts so we did not include it. This leads to a modification of equations 3.9 and 3.10. The authors claim that this new version leads to similar or better result while being faster up to a factor ten than normal MAGSAC. This increase in performance is indeed observed in Section 3.3.1. The implementation was adapted from [7].

As specified, all algorithms were either implemented from scratch or adapted from an available repository. However, several publications presenting the algorithms do not disclose all required implementation details nor design choices. When elements were missing, hyperparameter values were either divined from available experiments or set after some initial tuning. When entire procedure or formulas were missing, we used our best judgement to design an *ad hoc* solution. For example, the procedure to compute the set of possible thresholds for LRT was not available and we had to design our own solution. When differences occurred between a paper presenting an algorithm and its available implementation appeared, like for MAGSAC and MAGSAC++, we used the proposed implementation without further analysis, as we aim to compare the results of the authors' algorithms and if changes were introduced, either to ease implementation or propose better results, those are the version we are interested in.

3.2 Benchmark parameters and datasets

This section regroups the different information used for the benchmark: the estimation problems included, the datasets from which we extracted the input data for the generator and the parameters of the generator and the choice of hyper-parameters for the tested algorithms.

Estimation problems:

The different estimation problem we included in the benchmark are homography, fundamental matrix and essential matrix estimation and the Perspective-from-n-Points prob-

lem. All those are classic MVS and SfM tasks that can be used in the global reconstruction pipeline but most RanSaC algorithms, like [17, 6, 60, 61, 73, 62, 64, 16], are only tested on homography, fundamental matrix and essential matrix problems whereas the PnP problem is a staple of the pipeline as it is used to incrementally add new snapshots to the reconstruction, see Chapter 4.

As minimal solvers F, we use the standard 4-point estimator for homography and 7-point estimator for fundamental matrix [32], the 5-point estimator for essential matrix [70] and the P3P algorithm for the PnP problem. For non-minimal solvers, least-square evaluation was used for two-view geometry problems and the EPNP[43] algorithm for PnP. For MAGSAC and MAGSAC++ weighted version of the non minimal solvers are required and we adapted the EPNP algorithm to work in a weighted case. In order to do so, equations (1.19) and (1.20) are multiplied by the point's weight.

Datasets:

Different datasets and different tasks are included in the benchmark in order to test whether our data generation methodology does indeed allow generalisation of results or if differences appear between algorithms depending on the test setting.

The first dataset comes from the USAC [73] paper. It is reused by [17, 6, 61, 60]. It contains 10 image pairs for homography estimation, 11 for fundamental matrix estimation and 6 for essential matrix estimation which will be called "*Problem* - USAC - Dataset i" where *Problem* can be Homography, Fundamental or Essential and i is the index of the dataset according to the ordering of USAC. Alongside the image pairs, unlabelled SIFT [52, 53] matches are provided as well as calibration matrices for the six essential matrix image pairs. The image pairs are in the wild and present a variety of setup most from outdoor urban setup but a few aerial or indoor images.

Multi-H [20] is a dataset that can be used for multiple homography detection or for fundamental matrix estimation.[1] In our case, we use it for the latter. In contains 24 image pairs for fundamental matrix estimation without any matches. We compute the matches using [52, 53] with SIFT ratio of 0.6—this ratio is the maximal feature distance between the closest matching and second closest matching point for a point to create a match. Its image pairs are in the wild with mostly outdoor images of academic building and a few indoor images.

`homogr` [97] contains 16 image pairs for homography estimation.[1] It contains mostly challenging images with huge occlusions and high change in point of view. Images are either from outdoor, in the wild, setting or controlled indoor setups. SIFT matches are computed similarly to the Multi-H dataset.

Kusvod2 is a fundamental matrix dataset containing 16 image pairs.[1] It is designed

[1]Multi-H, kusvod2 and homogr can be found at `http://cmp.felk.cvut.cz/data/geometry2view/`

to offer challenging setups with illumination differences and significant point of view variation using both indoor and outdoor images. Again, no SIFT matchs are available so we compute them using the same protocol as other datasets.

Finally, for PnP estimation, the megadepth [46] dataset was used. We extract 16 images from this dataset, with 2D-3D correspondance and calibration matrix. The images are from the wild, found on the Internet. The correspondences and calibration are computed using ColMap [79].

Data generation parameters:

To reveal performance of algorithms according to noise levels σ_{noise} and outlier ratio $1-\epsilon$, a range of values are used in the benchmark. Inlier noise ranges from no added noise up to 3 pixels with steps of 0.1. Outlier ratio varies from 0% to 90% by steps of 10%. A choice of both parameters $(\sigma_{noise}, 1-\epsilon)$ constitutes a test setting.

To minimise the impact of both the resulting generated dataset and the randomness of RanSaC algorithms, for each test setting $N_{gen} = 5$ different datasets are generated and for each dataset $N_{run} = 5$ estimations are computed for each algorithm. Results are then averaged on all success cases, meaning if an algorithm did not succeed in achieving at least 10% precision or recall it is not included. Early experiments included studies of standard deviation but no significant variation of standard deviation appeared across algorithm nor settings. It was thus removed from figures to make them more readable. This threshold ensures that results are not skewed downward because of a single failure case and, if need be, the number of failure cases can be analysed separately.

Algorithms hyperparameters:

Our goal is to compare user-threshold-free methods and establish which work best in which scenarios without any tuning. This is why, when possible we extracted the parameters of the tested algorithms from their original publications, otherwise we chose value after some initials tests but keep them fixed those during the benchmark. By doing so, the choice of algorithm will not be impacted by the quality of the tuning and thus the algorithm can be used off the shelf. Such values are summarized in table 3.1.

3.3 Time

Analysis of runtime answers two main objectives. Firstly, seeing whether an algorithm can be used in real-time application or should be used in offline reconstruction depending on the order of magnitude of its runtime. Secondly, keeping in perspective classification performance improvements, for example if an algorithm performs slightly better than another for a huge increase in runtime, its interest is reduced.

Table 3.1: Hyperparameters of the tested algorithms, see Section 3.1 for definitions.

Parameter name	Value	Algorithms using it
Success confidence $\beta = 1 - p_{II}$	0.99	RanSaC, StarSac, MUSE, LRT, AC-RanSaC, Fast-AC-RanSaC
Inlier search cut-off σ_{max}	16 pixels	AC-RanSaC, Fast-AC-RanSaC, StarSaC and LRT
NFA maximum value NFA_{max}	1	AC-RanSaC, Fast-AC-RanSaC
Expected type I error $\alpha = 1 - p_I$	0.01	LRT
Increase in type II error from early bailout $\gamma = 1 - p'_{II}$	0.05	LRT
Number of data partitions p	10	MAGSAC, MAGSAC++
Pseudo-inlier threshold σ_{max}	10 pixels	MAGSAC, MAGSAC++
Reference threshold σ_{ref}	1 pixel	MAGSAC, MAGSAC++

To measure runtime, all algorithms are run on the same computer with exactly the same dataset, sampler and estimator to make the RanSaC algorithm the only algorithm that changes. The runtime does include the time spent in sampling and running the estimator, as both can be run a different number of times because of different numbers of iterations or different optimization strategies.

After some preliminary tests, StaRSaC [13] proved extremely slow for baseline or slightly above baseline performance. A usual run will take between 3 and 5 minutes to compute a result which is two orders of magnitude slower than the next slowest algorithm. For this reason, we removed StaRSaC from the benchmark.

3.3.1 Runtime analysis

The first element a time analysis reveals is whether the noise level, at fixed outlier ratio, impacts the runtime of an algorithm. MUSE, Fast-AC-RanSaC, AC-RanSaC, and MAGSAC++ are not really impacted by the noise level but mostly by the number of points of the datasets. On the other hand, LRT, RanSaC and MAGSAC are hugely impacted by the noise level and can have runtime multiplied by up to 5 times.

The fastest algorithms are RanSaC, LRT and MUSE. These three can have runtimes around a hundredth of a second for small noise level, below 1 pixel. However, in hard settings, with high noise levels, runtime for LRT and RanSaC can increase drastically, up to the point of being the slowest algorithms, taking multiple seconds to stop on a satisfactory solution. MUSE always keeps its low runtime.

MAGSAC and MAGSAC++ usually show average time performance, with small range of variations. For the most difficult settings with high outlier ratio and high inlier noise MAGSAC can show high increase in runtime while MAGSAC++ is more stable and, usually faster. Both algorithm sometimes fail to terminate even when a good model is found so a time limit of 2 s was set. However, this time limit does not impact classification

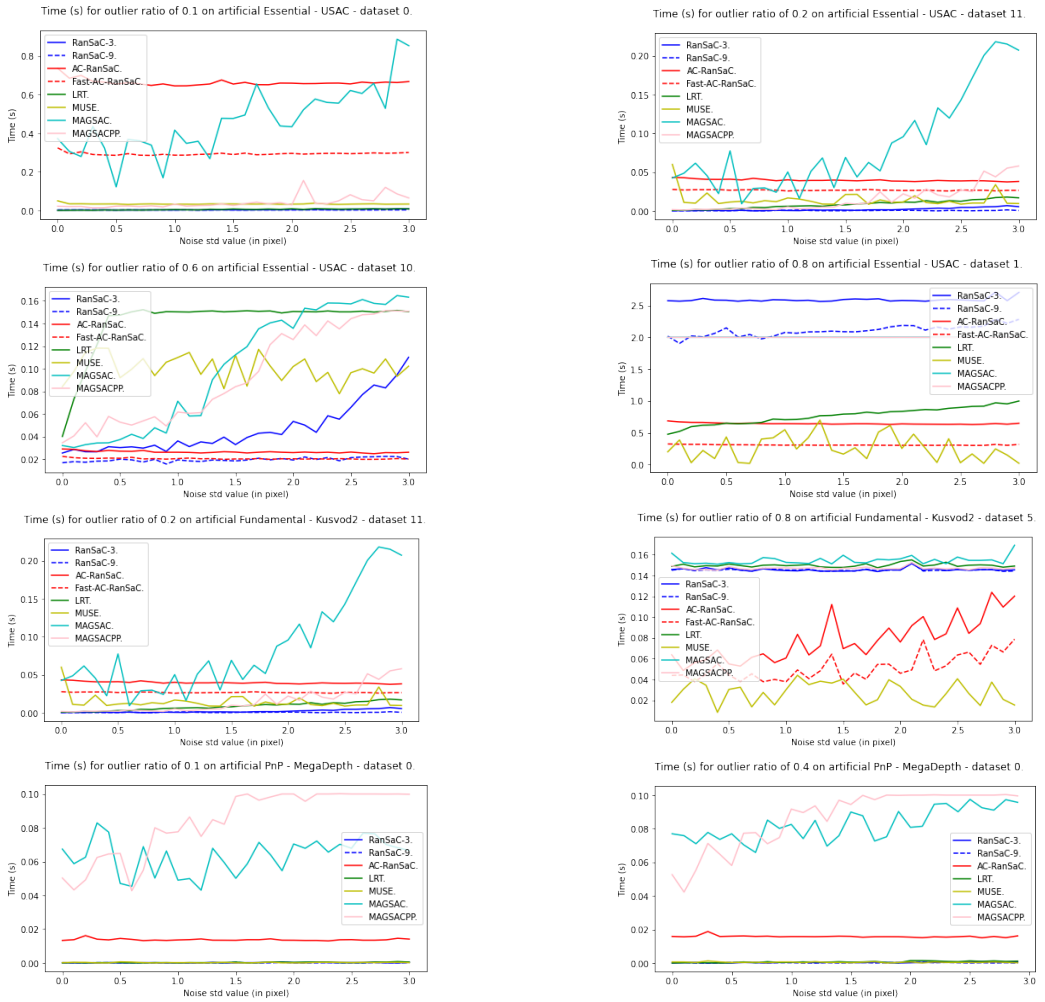


Figure 3.1: Runtimes over different inlier noise levels and outlier ratios. The fitting problem, dataset name, and image pair number are in each graph title. `Ransac-`σ corresponds to RanSaC with user threshold σ.

performance as usually the algorithm finds a good model before it is reached and just fails to recognise it and ends iterations.

For easy to medium settings, AC-RanSaC shows higher runtime than most over algorithms. It becomes competitive on difficult test settings as its runtime is not impacted by inlier noise but by dataset size, so when other algorithm see their runtime increase drastically, it can be faster. Fast-AC-RanSaC is most of the time twice as fast as AC-RanSaC and not impacted by noise as well.

3.3.2 Using early bailout to improve runtime

LRT appears faster than AC-RanSaC and Fast-AC-RanSaC but, as presented in the next section, with worst performance in most difficult scenarios. In order to try and improve the runtime of Fast-AC-RanSaC we analyse the early bailout strategy of LRT, see Section 1.4.3 and algorithm 14 and see if it is worth adapting to other algorithms.

This work was done in [75]. All runtime analysis are done on real data and averaged over 100 runs.

We first observe the runtime impact of this strategy for all two-view geometry tasks on USAC dataset, see Figure 3.2. The left scale presents the ratio of runtime with bailout over runtime without bailout, and should be as low as possible. The right scale presents the Verification Per Model (VPM), the number of points whose residual has been computed, as a proportion of the total number of points. It presents the sensitivity of the early bailout on the model estimation. For most cases the runtime ratio is below one, and can drop to 20% while increase in runtime is never above 150% but such increase is due to premature rejection of models as the VPM is extremely low in such cases.

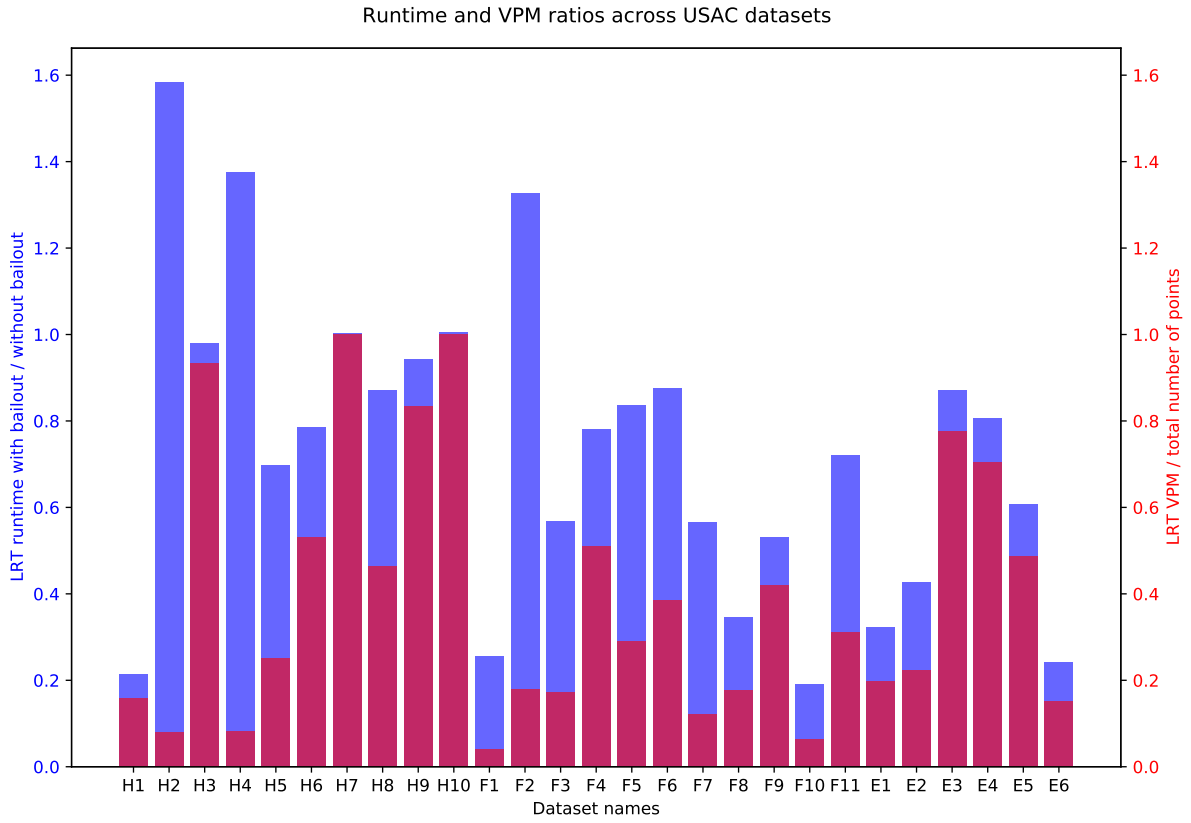


Figure 3.2: Ratios of LRT Runtime and Verification Per Model (VPM) with the early bailout strategy and without the bailout. Results are presented across all available USAC [73] datasets where `H` stands for homography estimation, `F` for fundamental matrix estimation and `E` for essential matrix estimation with same numbering as USAC.

To ensure the decrease in runtime is due to the early bailout strategy, we test LRT with and without all possible optimization strategy and compare the results together. The different options are the early bailout, the update of the maximum number of iterations T and the iterative diminution of σ_{max} once the so-far-the-best likelihood cannot be reached anymore for σ_{max}. Figures 3.3, 3.4 and 3.5 present the runtime ratio of different combinations of options over the runtime without any options. It appears that both

the early bailout strategy and update of T produce significant runtime decrease in most situations and even just using the bailout strategy (orange bars) can decrease the runtime by a factor of 10.

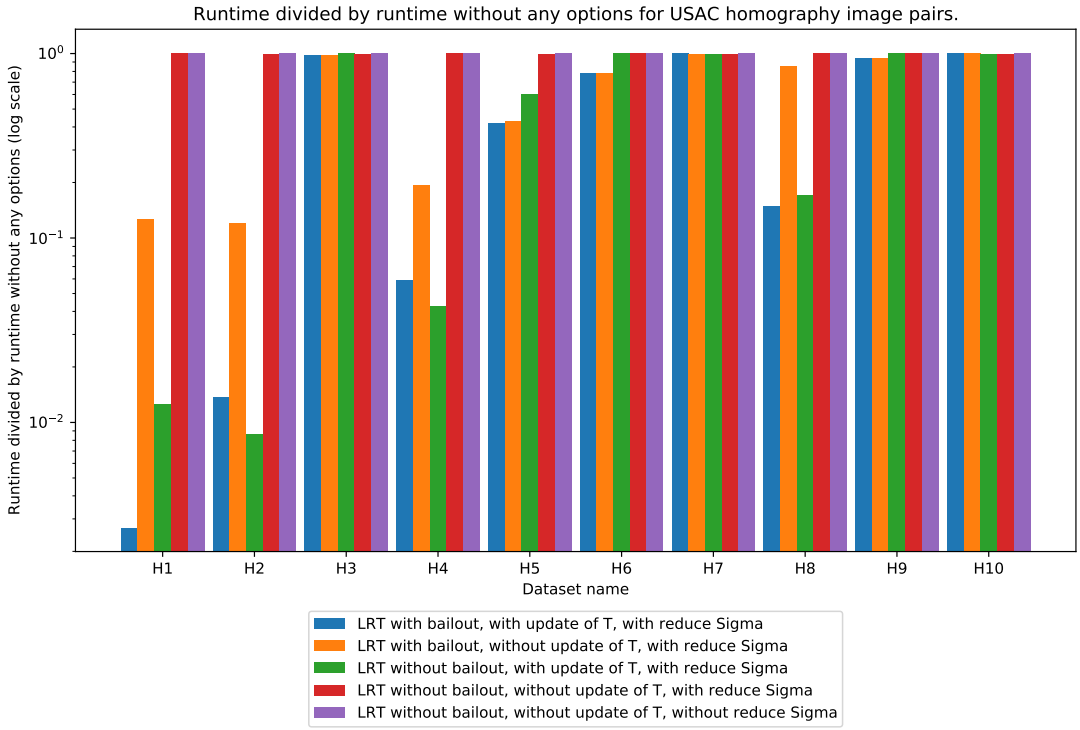


Figure 3.3: Ratios of LRT Runtime with different options enabled over LRT runtime without any options. Results are presented across all available USAC [73] datasets for homography estimation. The ratio is presented in logscale.

Finally, to be able to conclude on the validity of the early bailout strategy we observe how it impacts the classification performance of LRT. Figure 3.6 presents the ratio of the precision of LRT with early bailout and of LRT without bailout. Figure 3.7 presents the ratio of the recall of LRT with early bailout and of LRT without bailout. Both graphs are presented as boxplots with first and third quartiles defining the box and the median for the middle line. The whiskers are set at 1.5 times the interquartile range. The results shown are averaged over different noise levels σ_{noise}—from 0 to 3 pixels by 0.1 pixel increments, different outlier ratios—from 0 to 90% by increments of 10%, and 25 different runs.

Both metrics present a median slightly lower than 1 with a spread on a very small area around this value, sometimes above 1. This means the early bailout strategy is not consistently worsening the quality of the result, and not by a lot when it does, thus it is a valid improvement of the algorithm.

In view of those results, we tried to adapt the early bailout strategy to Fast-AC-RanSaC. However, as presented in [77] this reduced drastically the performance of the algorithm and it was not pursued.

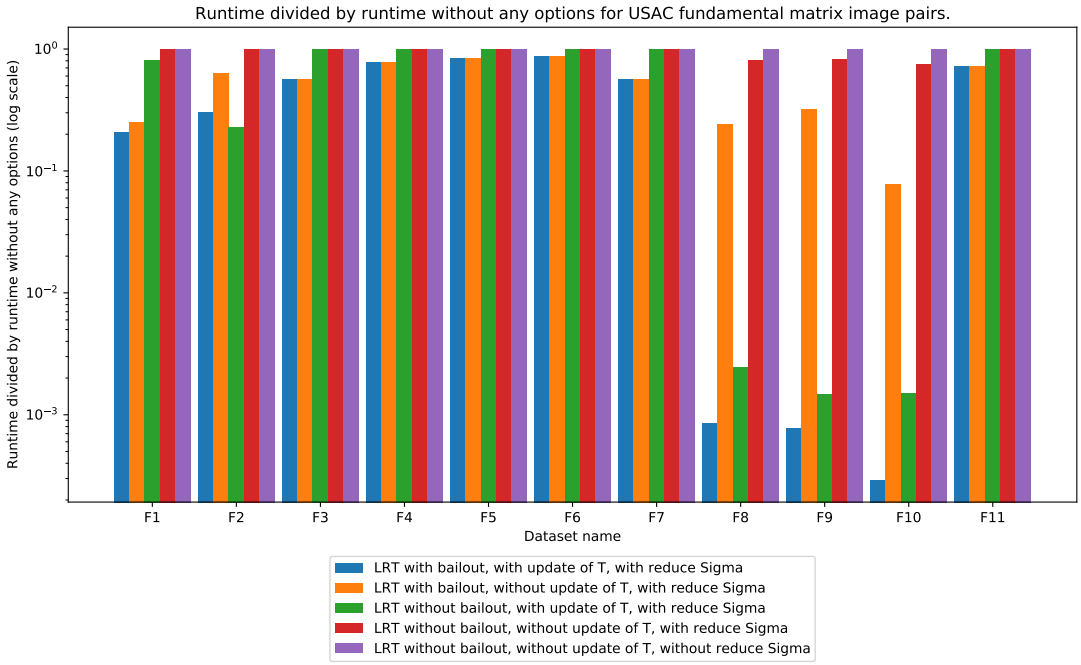


Figure 3.4: Ratios of LRT Runtime with different options enabled over LRT runtime without any options. Results are presented across all available USAC [73] datasets for fundamental matrix estimation. The ratio is presented in logscale.

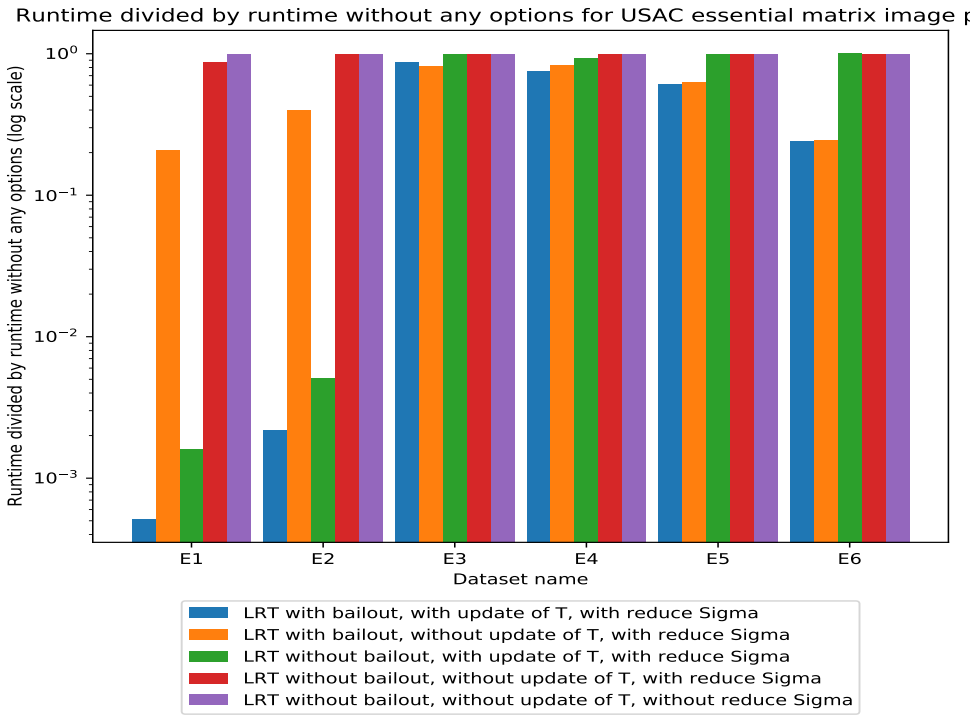


Figure 3.5: Ratios of LRT Runtime with different options enabled over LRT runtime without any options. Results are presented across all available USAC [73] datasets for essential matrix estimation. The ratio is presented in logscale.

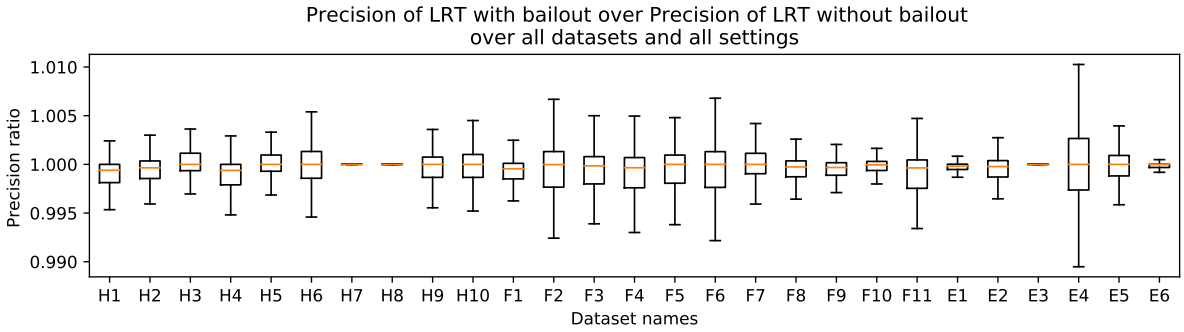


Figure 3.6: Boxplot of the ratio of the precision of LRT with the early bailout strategy and without the bailout. Results are presented across all available USAC [73] datasets and over a high range of semi-artificial dataset settings. `H` stands for homography estimation, `F` for fundamental matrix estimation and `E` for essential matrix estimation with same numbering as USAC.

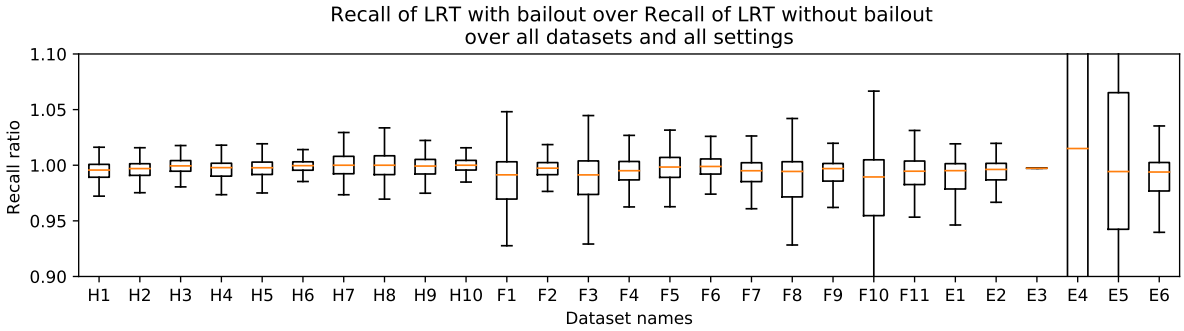


Figure 3.7: Boxplot of the ratio of the recall of LRT with the early bailout strategy and without the bailout. Results are presented across all available USAC [73] datasets and over a high range of semi-artificial dataset settings. `H` stands for homography estimation, `F` for fundamental matrix estimation and `E` for essential matrix estimation with same numbering as USAC. Dataset $E4$ and $E4$ are cut as they spread much more than others and made them hard to read. For $E4$ the quartiles are $[0.89, 1.15]$ and the whiskers are $[0.51, 1.56]$. For $E5$ the quartiles are $[0.94, 1.07]$ and the whiskers are $[0.78, 1.25]$.

3.4 Classification performance

In this section we observe the classification performance of each algorithms using precision and recall, as presented in Section 2.2. However, first observations, see Figure 3.8, reveal no change in trade-off between the two metrics. Usually, algorithms will perform better in precision than recall, but it will be a consistent behaviour without any increase in one metric while the other decreases. This is why we summarise most other observations using the F1-Score, as it represents the global performance of an algorithm. If interesting differences between the two metrics appear, it will be discussed.

Our second observation is that algorithms do not change behaviour across two-view geometry tasks. A given algorithm might show different performance according to the

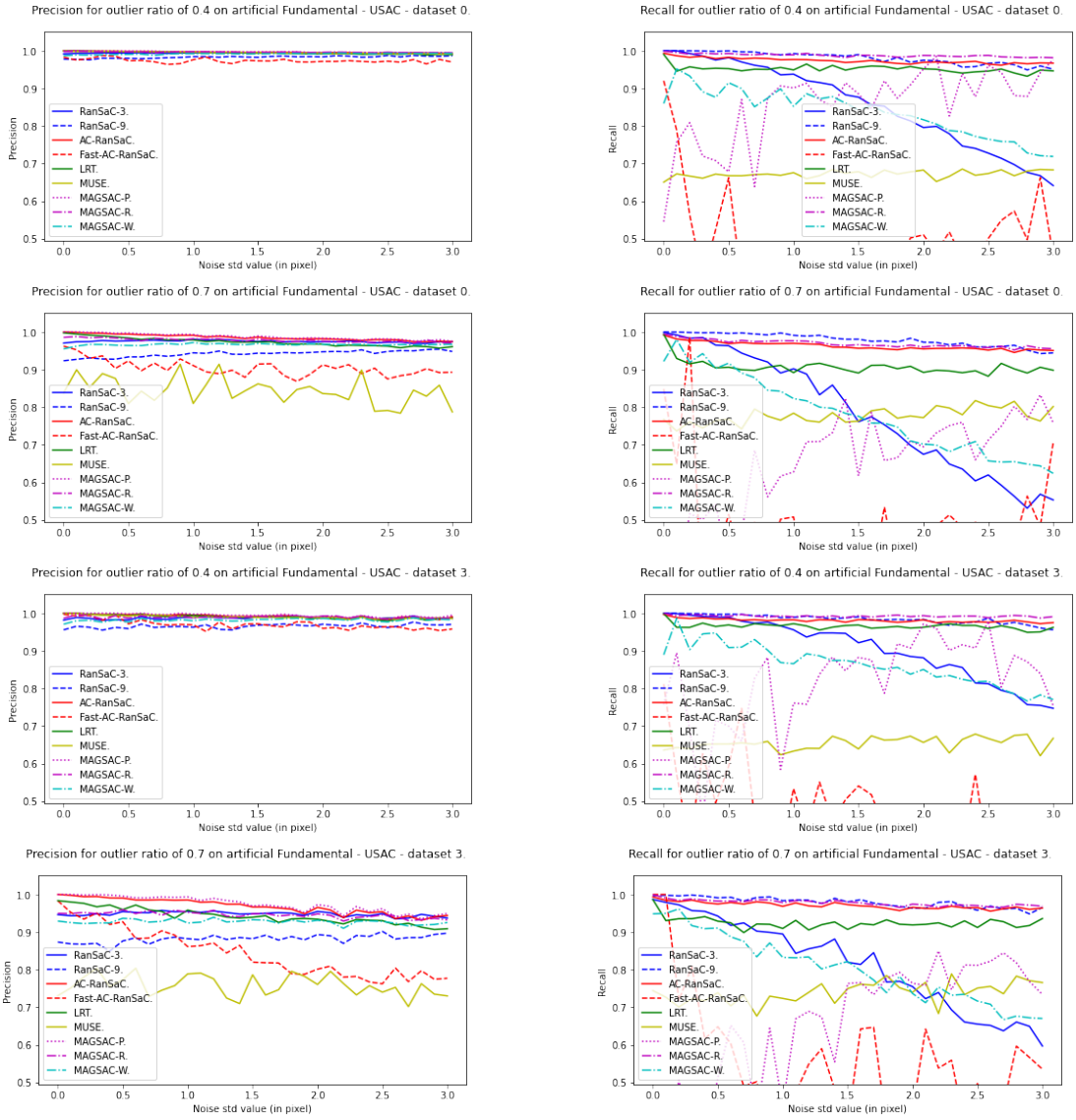


Figure 3.8: Precision (left) and recall (right) of all tested algorithms as a function of inleir noise on the Fundamental USAC dataset for various image pair and outlier ratios.

image pair or test setting but its general tendencies are not impacted by the estimation problem, be it homography, fundamental matrix or essential matrix. On the other hand, the PnP problem proves a challenge for some algorithms.

Figures 3.9 and 3.10 illustrate the typical behaviours with low and high outlier ratios on a variety of estimation problems and datasets.

Classic RanSaC with two different thresholds shows the expected behaviour with good performance for easy settings that quickly degrades as the inlier noise and outlier ratio increases. This behaviour can be observed across all estimation problems and all image pairs. On the one hand, the one with the lowest threshold has higher precision than the one with higher threshold with huge drops of performance for high noise or outlier ratio. It gives a baseline for the performance of a conservative RanSaC with low threshold. On the other hand, concerning recall, the high threshold one performs better than the other

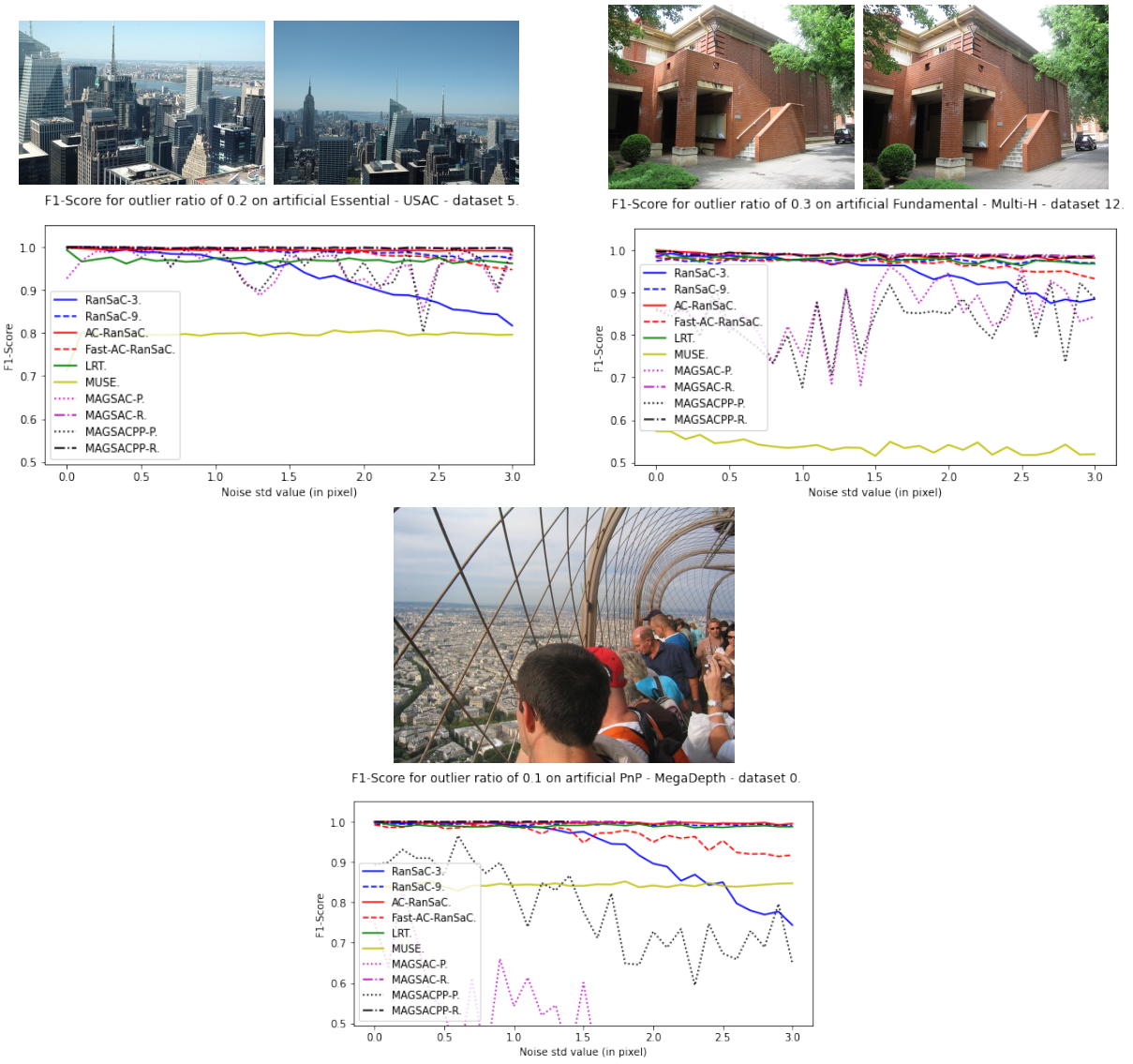


Figure 3.9: Typical F1-score evolution over inlier noise for low outlier ratios. Estimation problem, dataset name and image pair number can be found in each graph's title. `Magsac-P`, `Magsac-R` and `Magsac-W` correspond to the metrics presented in Section 2.2.

and, moreover, its recall remains high while for a 3 pixels threshold, it drops significantly. This gives the baseline for a permissive RanSaC with high threshold.

The MUSE algorithm will often present poor performance, mostly because of its selection of thresholds. Its performance is similar across all test cases and might perform worse than baseline in easy to normal settings. It usually selects quite small thresholds which impacts significantly its recall compared to other algorithms. However, this ensures it keeps high precision, even in the most complex settings, reaching more than 95% of good selections in some settings.

AC-RanSaC shows good performance for all test settings and estimation problems. For easy to medium settings it will often be amongst the best performing algorithm and its performance drop for complex settings is lower than most other algorithms.

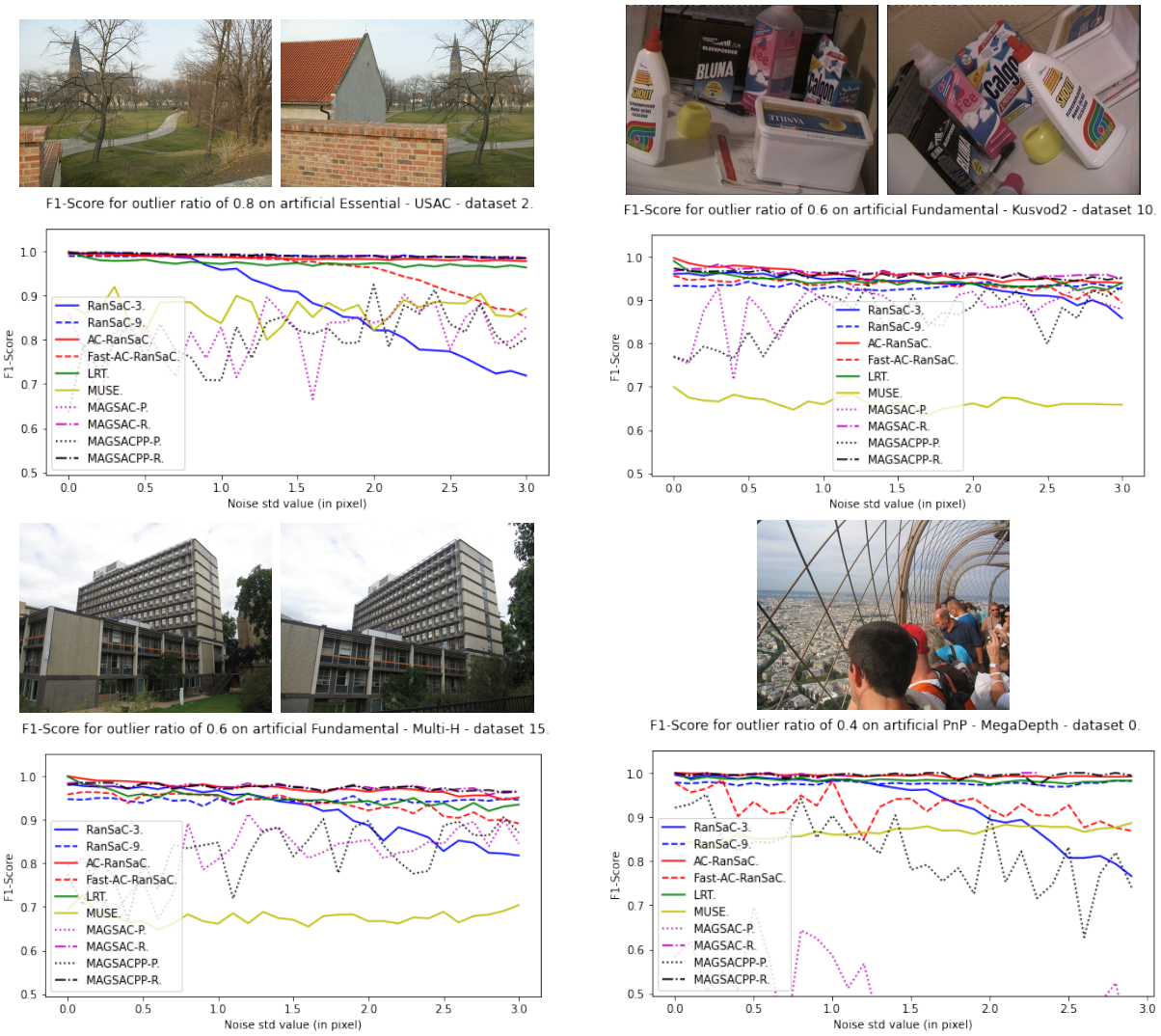


Figure 3.10: Typical F1-score evolution over inlier noise for high outlier ratios. Estimation problem, dataset name and image pair number can be found in each graph's title. `Magsac-P`, `Magsac-R` and `Magsac-W` correspond to the metrics presented in Section 2.2.

For LRT, its performance is on par, or just below, the best performing algorithms for easy to medium settings. However, in hard settings, with high outlier ratios and high noise levels, it presents significant performance drop though rarely below baseline.

Fast-AC-RanSaC shows performance slightly above LRT in easy to medium settings. However, for the hardest test conditions, it shows similar performance drop with none of the two consistently being better across datasets. Its performance is usually explained by good recall but lesser precision than normal AC-RanSaC which can be explained by its reduced choice of possible thresholds. The main issue with Fast-AC-RanSaC is reliability as in some runs it will fail to find any good model and might stop on a contaminated model or too high a threshold.

MAGSAC does not present a clear inlier/outlier threshold and needs different metrics to analyse its classification performance, see Section 2.2. However, using all three meth-

ods to compute precision and recall, it usually performs better than all other algorithms in two-view geometry tasks. It can even keep above 90% precision and above 80% recall in the highest inlier noise levels and outliers ratios. We keep in mind that `Magsac-P`, corresponding to $R_{P(\text{AC-RanSaC})}$, and `Magsac-R`, corresponding to $P_{R(\text{AC-RanSaC})}$, depend on the performance of AC-RanSaC. High precision and high recall in those values mean MAGSAC can perform better than AC-RanSaC if a well chosen threshold is found. However, for the PnP task, MAGSAC fails to present satisfactory results. It fails to find a good model most of the time and when it does its recall is below baseline.

MAGSAC++ almost consistently performs slightly better than MAGSAC with small variations. It never performs worse than MAGSAC. It does keep the same troubles with PnP tasks and will show extremely low recall values when it finds a good model. For both algorithms, lifting the runtime limit does not solve this issue.

3.5 Validity of method and algorithm choice

The behaviour of algorithms across different datasets, different images for a given estimation tasks seems solely impacted by the semi artificial dataset parameters. Indeed, while specific values might change, the ordering of algorithms, the drops in performance, the runtime evolution, are consistent across all test cases and depend on the test setting and not input task or images. We conclude that our data generation methodology is able to reveal intrinsic qualities and characteristics of the tested algorithms. However, some differences might appear for specific estimation tasks so we need to test algorithms accordingly.

Both StaRSaC and MUSE exhibit poor performance compared to other algorithms. The StaRSaC algorithm offers very small to no increase in performance compared to RanSaC, even with a poorly chosen threshold, and it is way too slow to justify its use in almost all cases. MUSE offers high speed and high stability but usually, it will perform worse than newer fasts methods. It remains interesting when the only focus is precision and not recall as it selects conservative thresholds.

Then, we separate algorithms according to speed to compare their performance. Firstly because there is a difference in application for fast algorithms that can be included in real time applications if they consistently have runtime below a tenth of a second. Secondly because the speed of an algorithm has a significant impact on the classification performance robustness as most fast algorithms will show bigger performance drop in difficult test settings than slower algorithms. The speed of an algorithm is, in most cases, explained by the two main steps at each iteration: the computation of the model from the minimal sample and the computation of the residual to estimate the quality of computed model. The second step usually represents most of the runtime as estimating a model from a minimal sample is usually extremely fast. However, algorithms like AC-RanSaC

and MAGSAC that require a post processing after residual computation, like sorting or σ-consensus, may show significant slow down. For example, for AC-RanSaC the sorting step required to compute the NFA at each possible threshold, and can frequently take as much time as the residual evaluation. It is also impacted by the reserved number of iterations, as it will always at least do 10% of the maximum number of allowed iterations. On the other hand, the σ-consensus procedure is, in easy settings, improving the runtime by reducing the number of iterations required. For example, LRT is a fast algorithm in easy settings, as its early bailout strategy helps it skip useless computation of residuals. But for hard settings, it can be slowed down by missing too many good models. The new Fast-AC-RanSaC algorithm offers a good compromise for a fast algorithm as it is usually fast enough to perform real-time operations and its runtime is very stable across settings.

For fast algorithms LRT performs slightly worse than Fast-AC-RanSaC in a few cases, but is almost always faster for easy cases. As it is very sensitive to the complexity of the task, it might be better to use only when the setting is known and the user may prefer the slowest but more stable Fast-AC-RanSaC when needing a fast algorithm.

For slow algorithms, AC-RanSaC is one of the slowest but most stable and consistent solution. Baring the most complex image pairs and generation parameters it always offers good precision and recall. MAGSAC is almost always slower or less effective or both than MAGSAC++, as expected as the second is presented as an improvement on the first one. For two-view geometry, MAGSAC++ performs almost always better than AC-RanSaC, producing good results even when other algorithms fail. It is also usually faster so it is a very stable and powerful solution. On the other hand, for the PnP problem, neither MAGSAC nor MAGSAC++ produce satisfactory results. The σ-consensus step seems to not be well designed with the EPNP estimator.

To conclude, a user who needs speed should prefer Fast-AC-RanSaC, a user who needs precision AC-RanSaC and for robustness in two-view geometry tasks MAGSAC++. All these remarks are summarised in table 3.2. As in the next chapter we focus mainly on the PnP-problem in ColMap, we implemented Fast-AC-RanSaC, AC-RanSaC and LRT in the tests to validate those observations.

Table 3.2: Best algorithm for different use-cases.

Use-case	Setting complexity	Algorithms, by order of performance
Runtime only	easy	LRT, MUSE, RanSaC
Runtime only	hard	MUSE
Fast and reliable	easy	LRT, MUSE, RanSaC
Fast and reliable	hard	MUSE, Fast-AC-RanSaC
Robustness (2-view)	Easy and hard	MAGSAC++, MAGSAC, AC-RanSaC
Robustness (PnP)	Easy and hard	AC-RanSaC, Fast-AC-RanSaC, LRT

Chapter 4

Using Adaptative RanSaC Methods in Colmap

Chapter 3 presented a benchmark of different Multi-View Stereo (MVS) and Structure-from-Motion (SfM) tasks, including the PnP problem. It showed which threshold-free algorithms perform better on for each task independently and on various scenarios. Some algorithms, like MAGSAC [6], LRT [17] and AC-RanSaC [60, 61] presented far better performance than the baseline in most settings. This could lead to better performance in a full reconstruction pipeline if used to replace the classical procedure and remove the need for a user threshold. Indeed, a reconstruction pipeline can be presented with large quantities of data including lots of variation in view-points and from potentially diverse sources which prevents a unique choice of threshold. In this chapter we try to see the impact of the threshold-free algorithms on a reconstruction pipeline.

Instead of building a pipeline from scratch, we are using ColMap, proposed in [79], which we modify to include different threshold-free RanSaC methods. This implementation is open-source and in C++ which will ease integration of previous work. It also is still considered state-of-the-art [21] for 3D-reconstruction and used to generate training data for many deep learning pipelines, so improving upon it could lead to improving the state-of-the-art. The fist section 4.1 presents how a basic reconstruction pipeline works and how ColMap improves upon it, then we present in section 4.2 how we generate semi-artificial data following the principle described in Chapter 2, then we present our experiments in section 4.3.

4.1 The ColMap pipeline

A 3D-reconstruction pipeline in the most basic form takes as input a set of photos of a scene and outputs a 3D point cloud as well as the position of each camera around this point cloud. The input set of photos can contain different views of the same scene from various angles, positions and cameras, with varying illumination, some non rigid transformation, like movement of some elements in the scene, and can include images that do not overlap at all; for example when taking a full 360-view of a building, the front and back pictures will not overlap. The cameras can be calibrated or not, and when they

are not, the calibration can be guessed or estimated alongside the rest of the computation.

The first step of the pipeline is to extract features from each picture using a feature detector and descriptor. ColMap uses SIFT [52, 53] to generate the features. It has a lot of different options depending on the number of images and availability of GPU to decide how to compute matches between images. If the number of images is above a few hundred, it will avoid doing exhaustive matching between all images. It mostly uses Fast Library for Approximate Nearest Neighbors (FLANN) [67], a collection of nearest neighbours algorithms which can choose the best one. Once matches are computed between image pairs, it is necessary to add a geometric verification step, to remove the bulk of outliers generated by improper matching. This step is performed by estimating homographies, fundamental matrices or essential matrices using the matches and a RanSaC algorithm. ColMap adds a series of tests to determine more information about the image pair: it checks whether this is a panoramic setup or a planar scene or a generic camera movement, if the image is corrupted by watermarks or other added elements and whether the calibration is reliable and uses the LO-RanSaC [16, 42] algorithm to improve the estimations. For any image pair that is tested, first the fundamental matrix is estimated and n_F inliers are found; if it is at least N_F inliers $n_F > N_F$, the pair is said geometrically verified. Then an homography is estimated and its number of inliers n_H is compared to the fundamental matrix threshold: if $n_H/n_F < \epsilon_{HF}$[1], for a threshold ϵ_{HF}, the pair is deemed to represent a general scene with moving cameras. When the calibration is available, the same is done to validate the intrinsic parameters: an essential matrix is estimated and its number of inliers n_E is compared to the number of inliers of the fundamental matrix: if $n_E/n_F > \epsilon_{EF}$, for a threshold ϵ_{EF}, the cameras calibration is validated. If the calibration is valid but $n_H/n_F > \epsilon_{HF}$ the essential matrix is decomposed to find the rotation and translation and the pair is classified as either a planar scene or a pure rotation. Finally, an analysis of watermarks, timestamps, frames (WTF [35, 94]) is done, using a similarity transformation, which yields n_S inliers. If this number is too high, $n_S/n_F > \epsilon_{SF}$ or $n_S/n_E > \epsilon_{SE}$, for given thresholds ϵ_{SF} and ϵ_{SE}, the image pair is not used in the reconstruction. All those tests will help the reconstruction by better guiding the process, mainly by choosing an initial image pair with valid calibration and that is not panoramic. Once all those steps are done, the scene graph is completed, where image are linked to each other by a geometric validation and their associated inliers.

In order to initialise the reconstruction, the first image pair has to be carefully selected. This is done by choosing an image pair in a dense part of the scene graph, where plenty of other images can be added to increase the robustness of the estimation. As said above, if possible, the initial image pair is also chosen among calibrated image pairs that are not

[1] [79] uses the threshold N_F in this inequation, found in section 4.1, but in its implementation it is indeed the number of inliers n_F that is used. The rest of the paragraph presents other slight differences with the original paper for the inequations but what is presented here is based on the available implementation.

panoramic views. Then, to add a new image, the Perspective-from-n-Points problem is solved, by using the 2D matches between the new image and previously registered images to establish 2D-3D correspondances with already reconstructed points. Those 2D match strings are called feature tracks. As some matches can be outliers, a RanSaC method is used here to avoid contaminating the reconstructed points with outliers which might ruin the rest of the reconstruction. ColMap uses the LO-RanSaC algorithm for this step as well. The strategy used to choose which image is added to the reconstruction next is crucial as it will impact each further image registration. ColMap proposes a strategy based around favouring images that see a lot of points but distributed around the whole image instead of concentrated in a small section. All images that see at least N_t points will be considered. Then they are discretised are different scales $\{K_l = 2^l,\ l \in [L]\}$, for a given L, using a square grid of K_l^2 bins, where each bin's value is set to 1 if at least one 2D-3D match is in the bin, and 0 otherwise. Each bin will contribute a weight $w_l = K_l^2$ to the score of the image, such that higher bigger bins contribute less to the overall score. The image with best score is selected. Once the position of the image is estimated using the PnP method, with P3P as minimal solver and EPNP as non minimal solver for the optimisation step, see sections 1.2.3 and 1.4.2, new points can be triangulated into the reconstruction. This step is a precarious one, as using corrupted feature tracks can lead to high outlier ratios very quickly. To avoid testing all pairwise combination which would be too time consuming, ColMap uses a RanSaC based estimation. Given a feature track $\mathcal{T} = \{T_n,\ n \in [N_T]\}$, with N_T the length of the track, T_n a measurement of image I_n including a normalised observation $\bar{x}_n \in \mathbb{R}^2$ and the position of the camera $P_n = [R^T, -R^T t]$ with $R \in SO(3)$ the rotation and $t \in \mathbb{R}^3$ the translation. The metric to maximise in the RanSaC algorithm is the support of a two-view triangulation in the feature track. Given a triangulated point X_{ab} from two different images (I_a, I_b), a measurement T_n belongs to the support of X_{ab} if its depth is positive and:

$$\left| \bar{x}_n - \begin{pmatrix} x'/z' \\ y'/z' \end{pmatrix} \right| < t, \tag{4.1}$$

with t a given threshold and $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = P_n \begin{pmatrix} X_{ab} \\ 1 \end{pmatrix}$. The triangulated point $X_{ab} \sim \tau(T_a, T_b)$, with τ the triangulation method, Direct Linear Transformation [32] for ColMap, needs to follow two constraints to be well-conditioned. First, it needs to have a high enough triangulation angle α, $\cos\alpha = \frac{t_a - X_{ab}}{|t_a - X_{ab}|} \cdot \frac{t_b - X_{ab}}{|t_b - X_{ab}|}$. Then it needs to respect the chirality constraints, that is to have a positive depth in both images. Using RanSaC to estimate feature tracks will provide good triangulated points. As a feature track can be linked to multiple real world points, RanSaC is launched iteratively, by removing the consensus set of the previous point each time, until there are fewer than 3 points. Adding images sequentially will create drift as small errors in registration and triangulation will accumu-

late, this is why Bundle Adjustment (BA) is performed to optimise camera and 3D points parameters jointly. BA is one of the most time consuming steps of running a reconstruction pipeline, so ColMap only runs a global BA each time enough images are added to the model. However, a local BA step is performed after each image registration. As there are usually far fewer images than points, it is possible to optimise first the camera positions and then compute the 3D point positions. To perform BA, ColMap uses the Ceres solver [3], more details about the parameters can be found in the original citation. To further improve the reconstruction and avoid drift, points that are too far from the model are removed after each BA, as well as degenerate cameras. Triangulation is done both before and after the BA step, by continuing tracks that have been abandoned previously. To further avoid contamination of the BA by outliers, all those steps, re-triangulation, BA and filtering, are done iteratively, until the number of filtered or modified points is low enough, usually this takes two iterations. Finally, to improve the speed of BA, ColMap clusters view points when they are redundant. This stems from two observations: first internet photos tend to focus around a few interesting view points, offering dense locations and sparse images between those, and second that BA will optimise newer elements more while leaving most of the scene unchanged. Images that have been added recently or that have enough matches with high reprojection error will be left out of any groups, so that their parameters can be optimised more efficiently. Other images will be grouped in N_G groups, which will be represented by a single set of parameters. To create the groups, each image I_i will have a binary vector v_i of visibility of the N_X world points, $v_i(n) = 1$ if the n^{th} point is visible from I_i and 0 otherwise. Images are then ordered in decreasing order of number of points visible. The first image I_a will seed the first group, and the next image I_b will be included in the group if $V_{ab} = \frac{|v_a \wedge v_b|}{|v_a \vee v_b|}$ is greater than a threshold V and the group of I_a is not full yet. Otherwise, I_b will seed another group. To speed up the process, only images spatially close are considered.

4.2 Experimental setup

The ColMap pipeline uses the LO-RanSaC algorithm [16, 42] to perform multiple tasks, from geometric verification to image registration and triangulation of new points. As the PnP problem is solved at each image registration and impacts the quality of the registration of each image, we chose to test replacing the LO-RanSaC algorithm of this task by threshold-free alternatives. The algorithm we chose are AC-RanSaC [60, 61], Fast-AC-RanSaC [65], LRT [17], because they are the most reliable for the PnP task, see table 3.2. Our objective is to see whether the quality of the reconstruction can be improved by using more carefully chosen inlier/outlier thresholds. To follow the same logic as in chapters 2 and 3, we use semi-artificial data to study the behaviour of ColMap.

4.2.1 Generating inlier data

We use the same method as described in Section 2.3.1 to generate semi-artificial data. First we use real data to evaluate a model, then we correct data to fit this model and add noise. This is the model we call ground truth. To do so, we run ColMap on a set of images, with nothing but their calibration available, and let it recompute the SIFT [52, 53] features, the matches and the geometric verifications that serve as input to the pipeline, and the output will be the position of the cameras, and the 3D positions of the inlier elements.

We take both the input database, including all SIFT features with their matches and verifications and the output 3D positions and cameras positions to use for clean-up. First, we filter the geometrically verified match to only keep the one that are linked to a 3D point. This creates new feature tracks, linked to a 3D point. The features of each track are then replaced by the projection of the 3D point onto their camera using the output camera positions and calibration. On this reprojection step, a chirality test is added to avoid corruption of the feature track if one camera has bogus parameters or a 3D point is wrongly positioned. Once the features are perfectly matched to the 3D point, a perturbation of uniform noise of standard deviation σ is added. This step allows us to make sure the inliers are truly linked to a 3D point and we have control over the noise level.

4.2.2 Generating outlier data

Similarly, the same logic as in Section 2.4.1 is used but with adaptations as we cannot treat each image individually but we need to work on the feature tracks to ensure the PnP estimation is properly disturbed and artificial outliers are not filtered out immediately by the various post-processing of ColMap. First, we identify which image pair is used to seed the reconstruction, as this step is not our focus, we do not wish to disturb this reconstruction. Then, for each image pair, we select randomly a side of the pair, avoiding to select the images used for reconstruction, and will modify the features on this side. This will break some of the feature tracks by adding outliers to them, disturbing the registration process. A proportion of the matches are selected and the feature on the chosen side will be replaced by an outlier, using the same method used to generate an outlier in the PnP experiment, see Section 2.4.2. First the 3D point associated with the feature is projected onto the image, then the direction of perturbation is chosen uniformly, and the offset is chosen uniformly to ensure the point is further away from its true match than the biggest noise applied to inliers. This new feature is added to the feature pool and inserted instead of the correct one in the feature track.

The modified inliers and generated outliers are then saved to a database, alongside the geometric verifications of the image pairs, so that ColMap does not recompute those

from scratch when running experiments.

4.2.3 Experimental parameters

Datasets used:

Different datasets were used to run experiments with ColMap. First, the Gerrard Hall dataset, which is provided by the authors of ColMap as a test dataset, and can be downloaded from `https://demuc.de/colmap/datasets/`. It is a set of 100 images of the Gerrard hall at University of North Carolina, Chapel Hill. This dataset covers a 360-view of the building and was used to create different datasets, to analyse the impact of sparsity of view on the chosen algorithms. Datasets were subsampled at different rates, by selecting images regularly around the building. The datasets are named sub$X$ meaning that the dataset contains X images out of the 100 available with $X \in [20, 40, 60, 80]$ and sub2 containing half the images. See Figure 4.1 for examples of images of this dataset.

Figure 4.1: Example of images from the 100 images of the Gerrard Hall Dataset.

Other test datasets are extracts of the MegaDepth [46] dataset. A few of the different scenes have been selected and images have been sampled from those to create small test sets of various complexity. The first set is Mega01 containing 28 images of the Eiffel Tower in Paris, see Figure 4.2. Mega11 contains 47 images of the National Gallery in London, see Figure 4.3. Mega21 contains 22 image of the London Eye, see Figure 4.4. Mega31 contains 42 images of Big Ben in London, see Figure 4.5. Mega41 contains 116 images of the outside of *Notre-Dame* in Paris, see Figure 4.6, and Mega42 contains 38 images of the inside of the cathedral, see Figure 4.7. Mega52 contains 38 images of the *Nike* of Samothrace, in the Louvre, Paris, see Figure 4.8. The datasets contain various internet photos, captured with various devices and represent more in-the-wild situations than the Gerrard Hall setup.

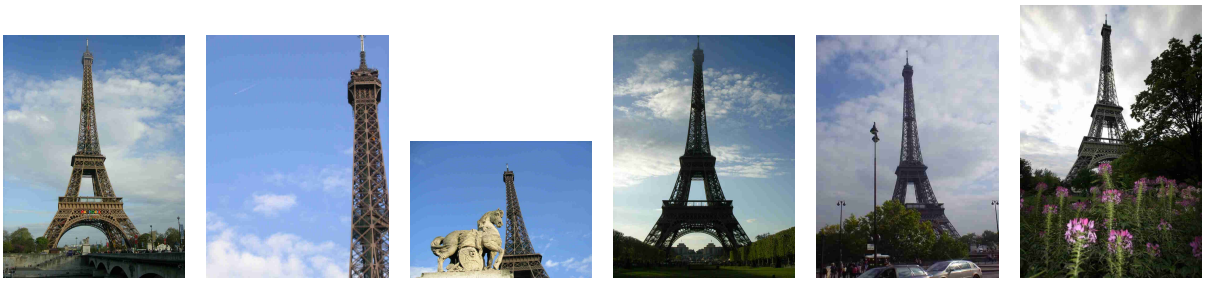


Figure 4.2: Example of images from the 28 images of the Mega01 dataset.

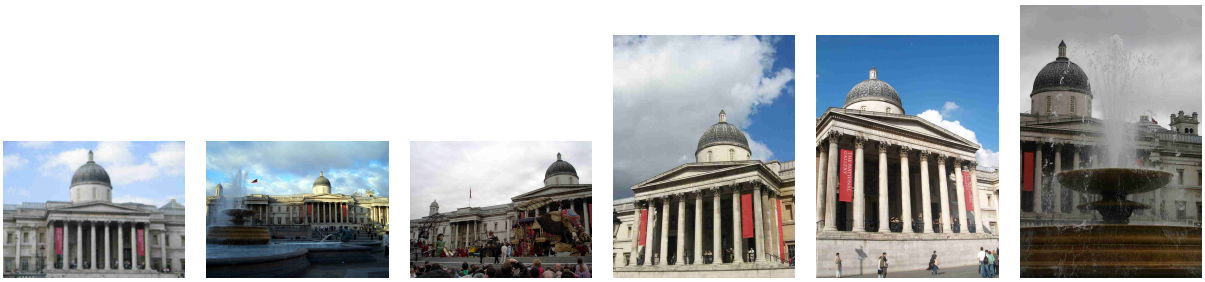


Figure 4.3: Example of images from the 47 images of the Mega11 dataset.

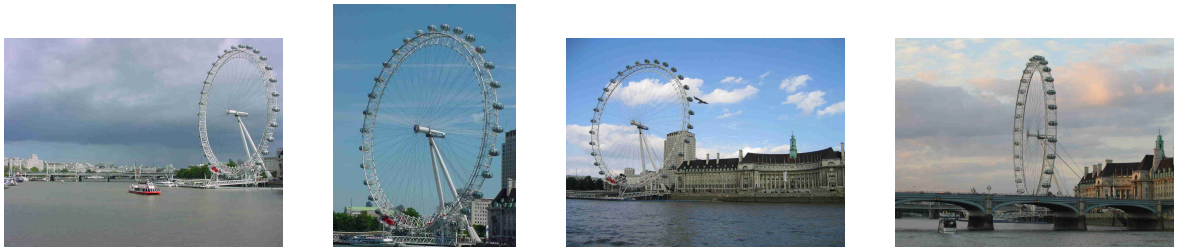


Figure 4.4: Example of images from the 22 images of the Mega21 dataset.

Figure 4.5: Example of images from the 42 images of the Mega31 dataset.

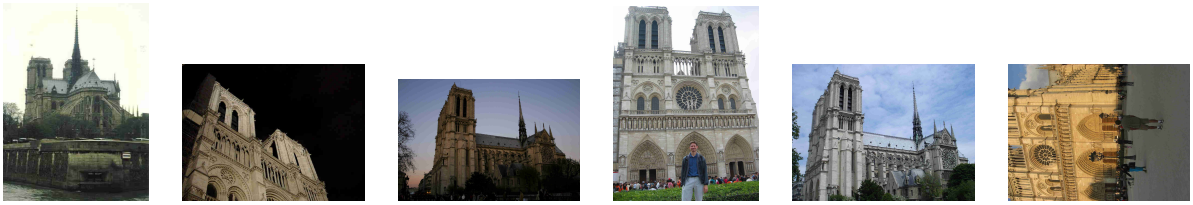


Figure 4.6: Example of images from the 116 images of the Mega41 dataset.

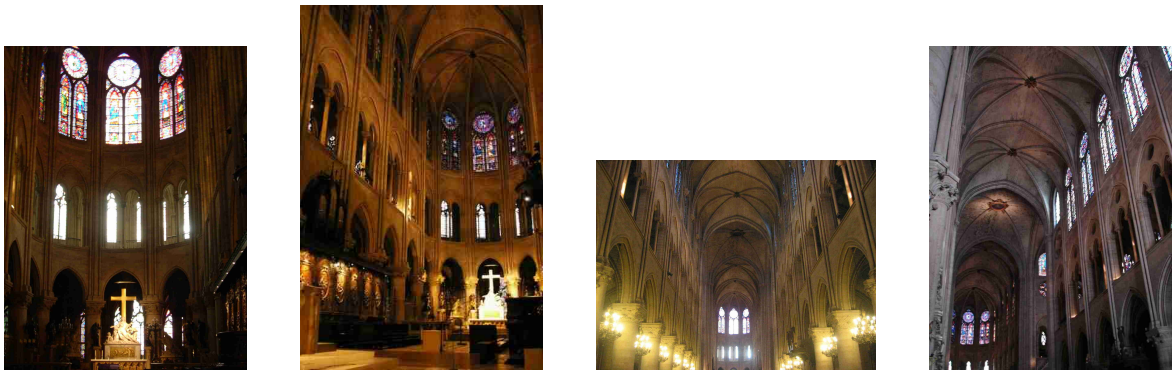


Figure 4.7: Example of images from the 38 images of the Mega42 dataset.

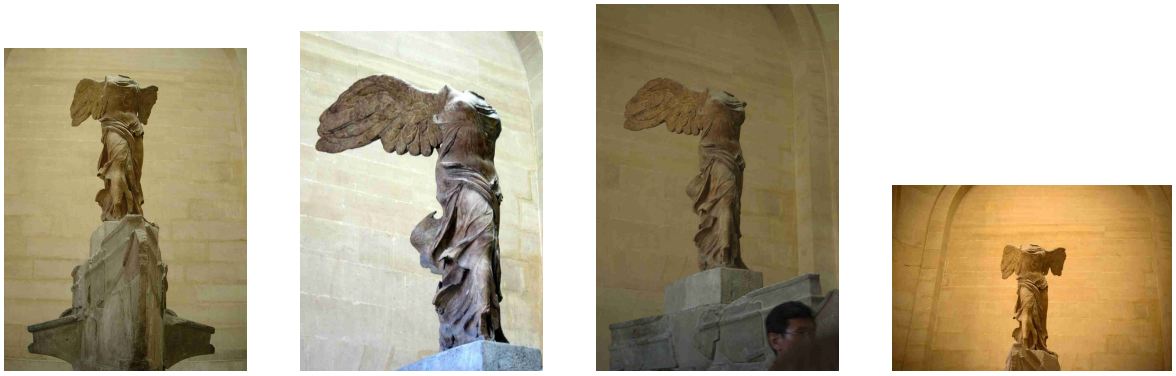


Figure 4.8: Example of images from the 38 images of the Mega52 dataset.

Generation parameters:

ColMap being slower to run than other experiments and more robust to small variations, we used a broader grid of hyperparameters to study its behaviour. The model used as ground truth is checked by hand to ensure the ColMap run was successful before generating semi-artificial data. The inlier noise varies from 0 to 6 pixels by increments of 0.5 pixels. The outlier ratio varied from 0 to 90% by increments of 10%. For each experimental setups, 10 different datasets were generated and ColMap was run once on each, to avoid peculiar cases disturbing the observations.

ColMap hyperparameters:

The different algorithms have following hyperparameters: LO-RanSaC uses a max error of $\sigma_{max} = 12$ and confidence $p_{II} = 0.99999$ to end computation. LRT uses max inlier ratio of $\sigma_{max} = 32$, with confidence with respect to Type I Error $p_I = 0.99$, confidence with respect to Type II Error $p_{II} = 0.99$ and confidence with respect to Type II Error due to early bailout $p'_{II} = 0.95$. AC-RanSaC and Fast-AC-RanSaC use max inlier ratio of $\sigma_{max} = 32$. All algorithms are run with at least 100 iterations and maximum 10000.

The max errors of each algorithm is set well above the max inlier noise of the generated data voluntarily. For LO-RanSaC this is by design, to use a very high value, as it should

work in most scenarios, and inlier noise above this value would not make much sense. For adaptative methods, this is leveraging the ability of those methods to choose a smaller threshold, adapted to the inlier noise level.

4.3 Observation

4.3.1 Time analysis

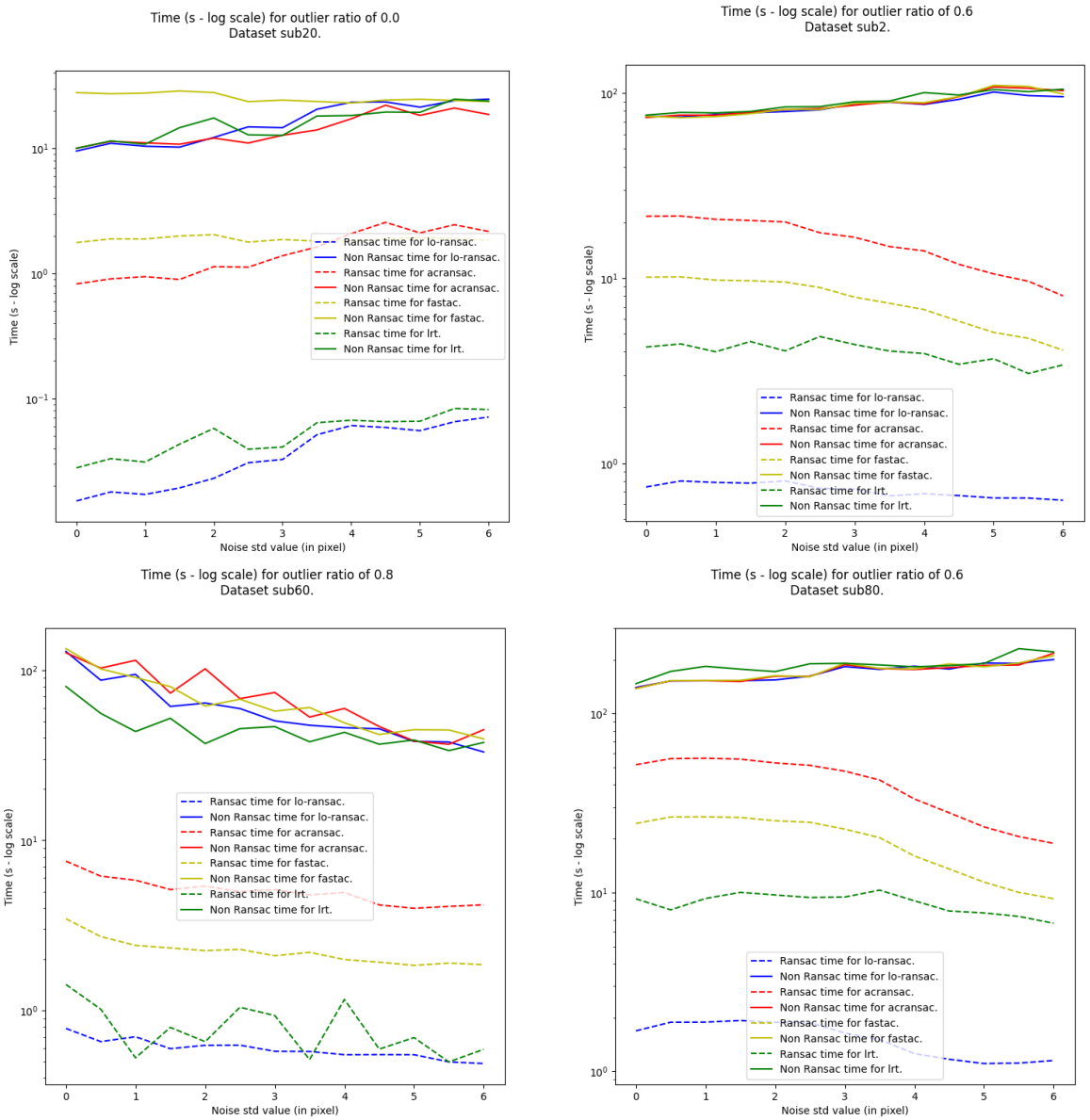


Figure 4.9: Runtime performance of ColMap for various datasets and outlier ratios, in log scale for the runtime is seconds, as a function of the inlier noise level in pixels for all tested algorithms. `Ransac time` measures the time spent only in the RanSaC algorithm for the PnP estimation and `Non Ransac time` measures the rest of the ColMap runtime. From left to right and top to bottom, dataset sub20 with no outliers, dataset sub2 with 60% outliers, dataset sub60 with 80% outliers and dataset sub80 with 60% outliers.

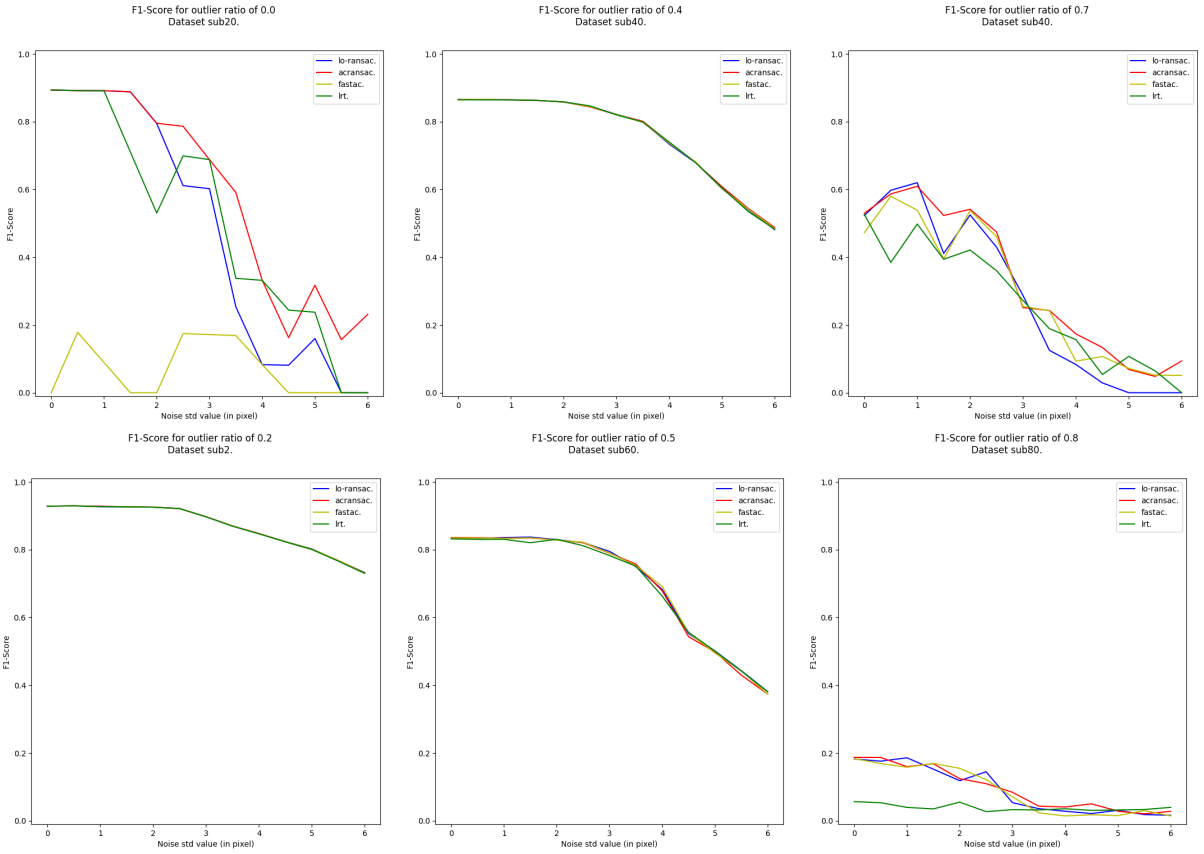


Figure 4.10: F1-Score of ColMap for various datasets and outlier ratios as a function of the inlier noise level in pixels for all tested algorithms. From left to right and top to bottom, dataset sub20 with no outliers and with 10% outliers, dataset sub40 with 40% outliers, dataset sub2 with 20% outliers, dataset sub60 with 80% outliers and dataset sub80 with 80% outliers.

As in the previous chapter, we start by analysing the runtime of the different algorithms. We measure both the time spent by ColMap in the RanSaC procedure linked to the PnP estimation specifically and the global runtime, minus this element, called `Non Ransac time`. As we provide the pipeline with the geometrically verified features, the rest of the runtime is dominated by the Bundle Adjustment and re-triangulation procedures by up to two orders of magnitudes. The algorithms we chose, LRT, AC-RanSaC, Fast-AC-RanSaC can be slower than RanSaC in most scenarios, see Section 3.3, and we hope that the increase in quality will result in a faster Bundle Adjustment and a faster processing of the whole pipeline.

Figure 4.9 presents runtime in log scale as a function of the inlier noise level for different datasets and different outlier ratios and is representative of the majority of observed situations regarding runtime. Performances of the chosen RanSaC algorithms are coherent with previous observations of Section 3.3, with LRT being faster than Fast-AC-RanSaC which is faster than AC-RanSaC. We also see little to no difference in runtime for the `Non Ransac time` metric, with most algorithms using the same amount of time for the post-processing steps. This means that the adaptative method will be slightly

slower overall, as the combined time will be impacted by the RanSaC time, while still dominated by the Non RanSaC time.

4.3.2 Retrieval performance

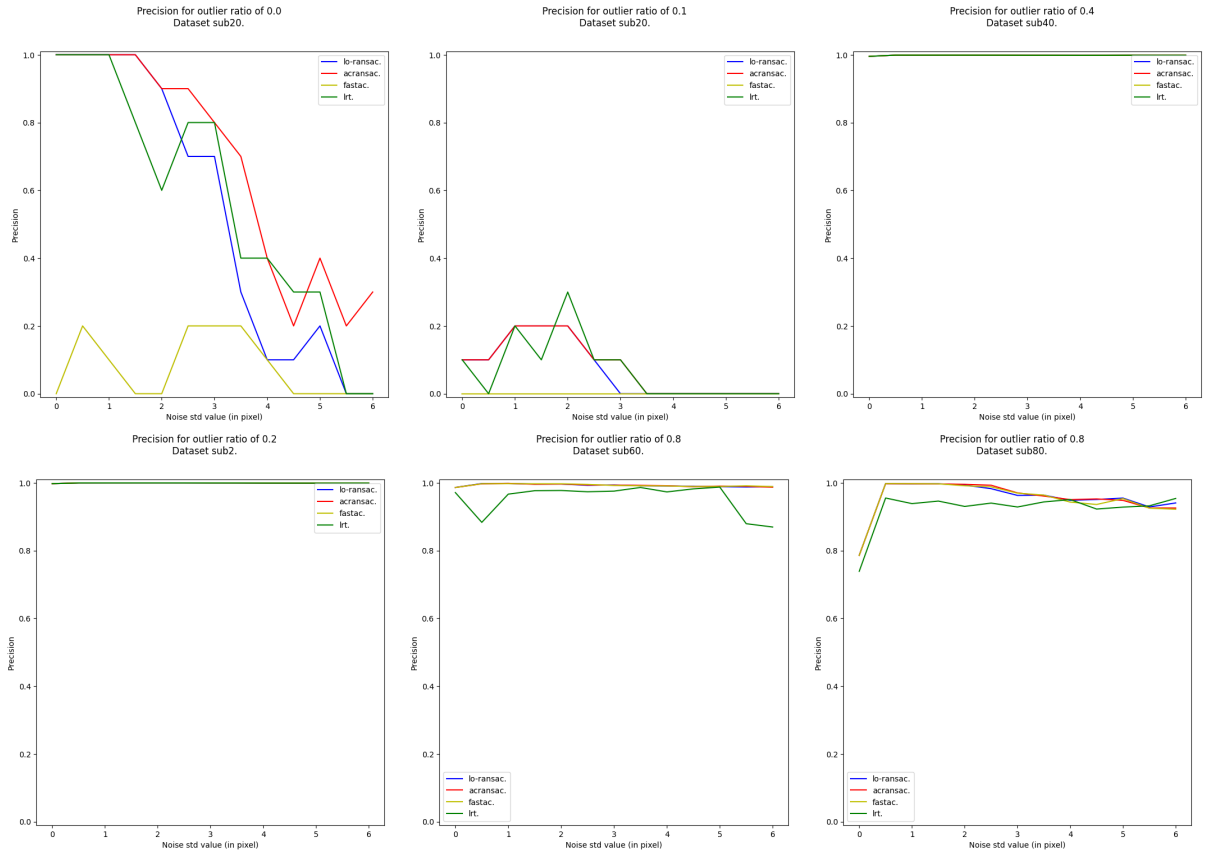


Figure 4.11: Precision performance of ColMap for various datasets and outlier ratios as a function of the inlier noise level in pixels for all tested algorithms. From left to right and top to bottom, dataset sub20 with no outliers and with 10% outliers, dataset sub40 with 40% outliers, dataset sub2 with 20% outliers, dataset sub60 with 80% outliers and dataset sub80 with 80% outliers.

The same metrics are studied for ColMap regarding the efficiency of the reconstruction: the precision and the recall. The ratio of reconstructed points that are truly inliers and the ratio of inliers that have been reconstructed gives good insight in how the algorithms successfully retrieve data to enhance the reconstruction. The default ColMap algorithm uses a non adaptative RanSaC, LO-RanSaC, with an inlier/outlier threshold of 12 pixels whatever the situation, using an adaptative method could improve the reconstruction if inliers and outliers are best separated and this is what those metrics would reveal. Another metric we included is the number of images seen, which corresponds to the number of image included in the final reconstruction. When ColMap cannot find a good enough reconstruction, it will not register the image and try another one. For complex setups the

number of image seen will drop and the better performance of some algorithms might be able to register more images.

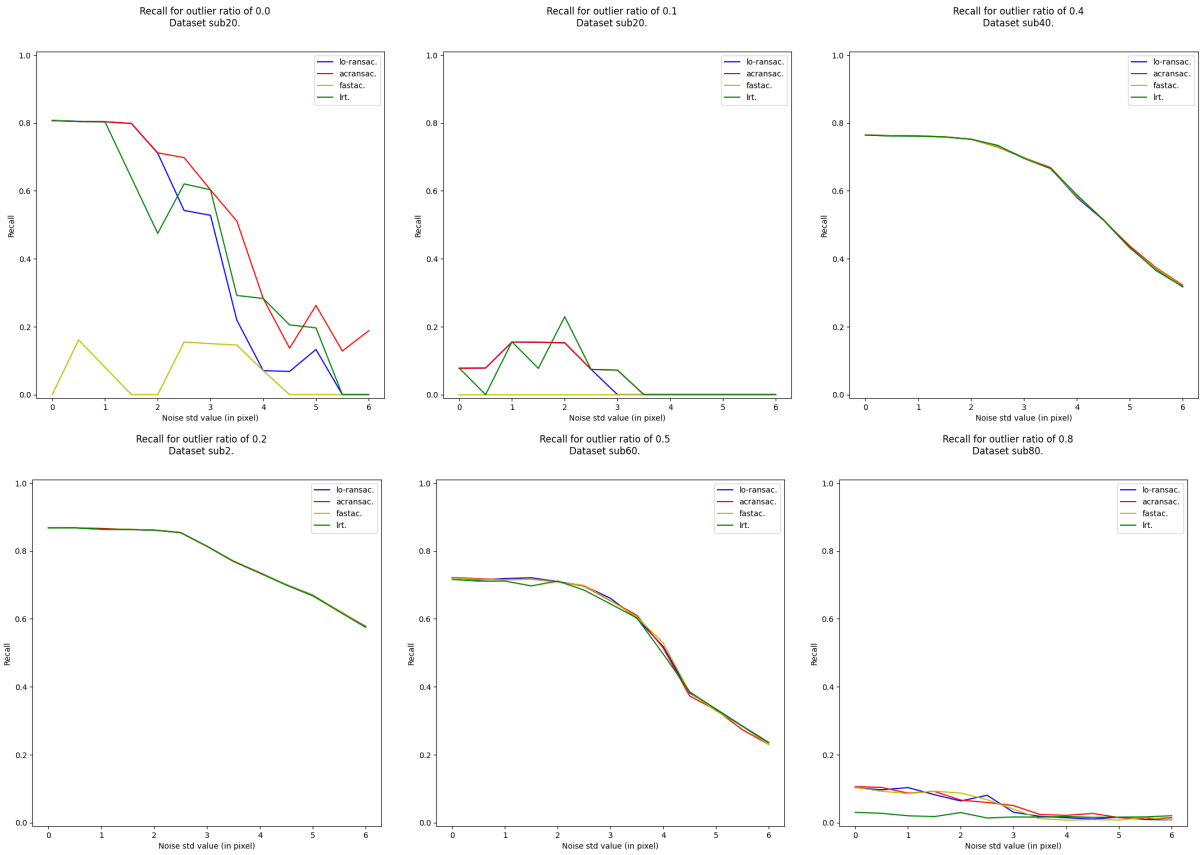


Figure 4.12: Recall performance of ColMap for various datasets and outlier ratios as a function of the inlier noise level in pixels for all tested algorithms. From left to right and top to bottom, dataset sub20 with no outliers and with 10% outliers, dataset sub40 with 40% outliers, dataset sub2 with 20% outliers, dataset sub60 with 50% outliers and dataset sub80 with 80% outliers.

First, Figure 4.10 presents the F1-Score for various setups. The main observation we can draw from the presented data is the lack of difference between algorithm performance in the vast majority of setups. Except in the most challenging scenarios, the difference between adaptative and non-adaptative algorithms is almost non-existent. For high outlier ratios, or with high view point changes AC-RanSaC consistently performs better than other algorithms but only in a few extreme cases.

Behaviour in precision and recall is very different, so we present both separately next. When looking at all datasets besides sub20, the precision remains very high for almost all setups, even high outliers ratios, as presented in Figure 4.11. This precision performance is better than the one of normal PnP tasks which means that the different steps of the pipeline help removing outliers from the reconstructed points. However, when looking at recall, see Figure 4.12, we observe sharp performance drops, usually around 50% outliers or above 2.5 pixels of inlier noise, earlier when the view points are further apart. Recall

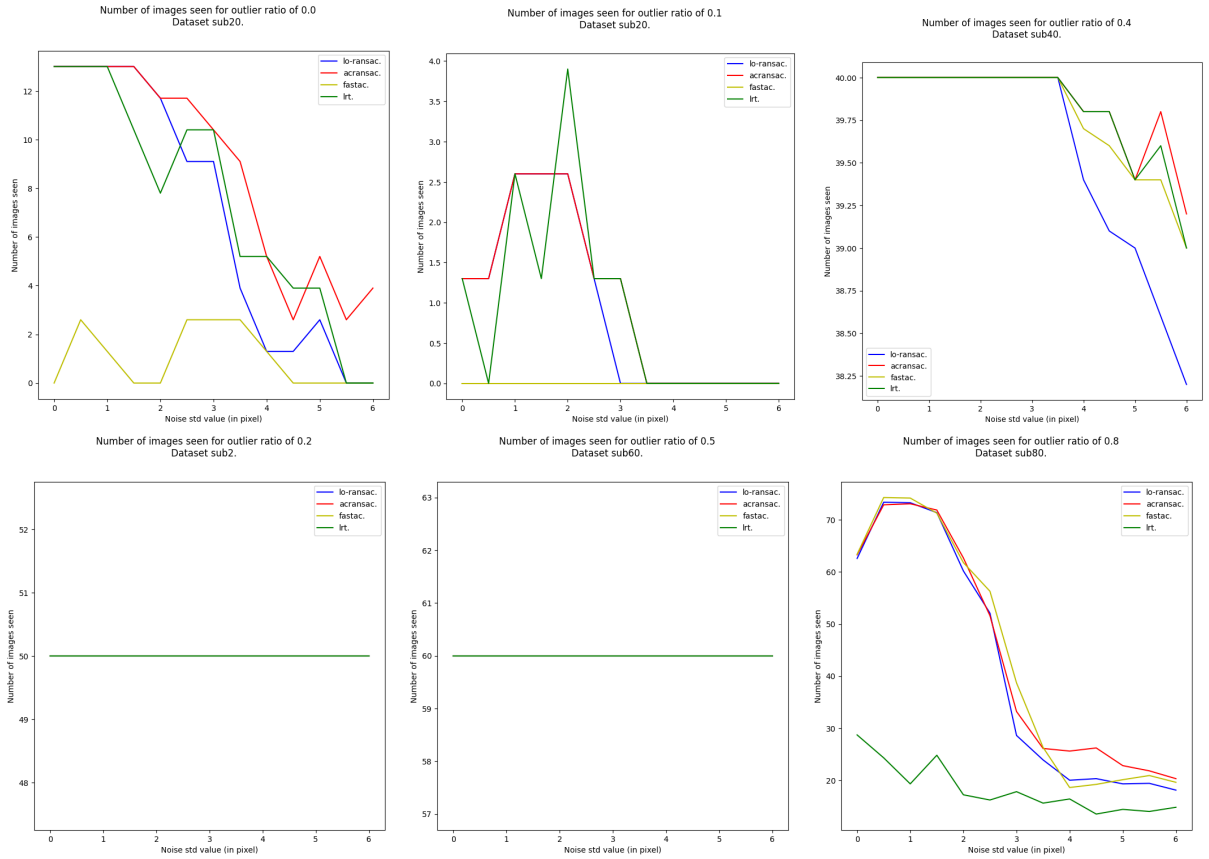


Figure 4.13: Number of images registered by ColMap for various datasets and outlier ratios as a function of the inlier noise level in pixels for all tested algorithms. From left to right and top to bottom, dataset sub20 with no outliers and with 10% outliers, dataset sub40 with 40% outliers, dataset sub2 with 20% outliers, dataset sub60 with 50% outliers and dataset sub80 with 80% outliers.

are partly due to the number of images registered, Figure 4.13 presents the number of images registered for the same setups as Figure 4.12 and we observe that for complex cases, like with very few images or high outlier ratio, the number of images registered drops quickly and recall is reduced as well. However, for more reasonable view points and outliers ratios, recall can drop while most images are still registered for high outlier ratios. This leads to lower F1-Score than PnP done outside of the ColMap pipeline for the tested algorithms.

4.4 Conclusion

The use of adaptative threshold methods in the PnP step of ColMap does yield more robust results but only when faced with the most extremes experimental setups. Those situations are rare in normal usage of ColMap as there are checks, especially in the initial geometric verification steps, to ensure not too many outliers are propagated to the rest of the pipeline. The efficiency of the method to retrieve more and better matches does not

allow the Bundle Adjustment and Re-triangulation steps to run faster and compensate for the increased runtime of the RanSaC methods.

Here, we tried to use adaptative RanSaC for the PnP task of ColMap. We managed to remove an hyperparameter of ColMap and keep the same quality of results which will ease the use of the software. However, the default, very conservative value of this parameter allows ColMap to perform similarly to adaptative methods except in the most difficult cases. The impact being very limited, we want in future work to explore the impact adaptative methods would have on the initial two view geometry steps, as it is the step that initialise the rest of the pipeline, a small improvement on this step could potentially improve the end result or the efficiency of the pipeline, especially in complex situations.

Conclusion

Traditional 3D reconstruction pipelines still rely on non-adaptative robust estimators to perform all camera pose estimations. In order to switch to adaptative methods, it is required to have a reliable benchmark of the different existing algorithms. In this thesis, we developed a novel benchmarking method using semi-artificial data to answer this issue. To avoid the problem of selecting the distribution for the artificial model generation and inlier generation, we rely on real data processed through AC-RanSaC and then corrected to get a ground truth with a realistic model with perfectly aligned inliers. Then it can be disturbed with inlier noise and carefully generated outliers to benchmark algorithms across a wide variety of tasks and challenges. We used this method to benchmark various algorithms using different input datasets. When observing the results of the benchmark, it appeared clearly that the input dataset was affecting the specific results of each algorithms but not the generic tendencies, like one algorithm yielding better precision than the other for a range of noise levels and outlier ratios, then experiencing more drops in performance around the same values across multiple datasets. This means our dataset generation method is robust enough to reveal behaviour of algorithms with respect to inlier noise levels and outlier ratios and not to the intrinsic difficulty of an image pair or PnP problem.

Thanks to our new method, we benchmarked a variety of adaptative RanSaC methods on homography, epipolar geometry and PnP estimation tasks. This benchmark allowed to get valuable insight on the various adaptative algorithms and their performance. For example, we could analyse precisely the impact of early bailout on LRT's performance. We also established that LRT, MUSE and RanSaC are the fastest algorithms for easier tasks and that only MUSE retains this speed at higher difficulties. We showed that MAGSAC++ and MAGSAC are the most robust algorithm regardless of the setting for two-view geometry exhibit poor performance for PnP tasks. AC-RanSAC is almost always slower than other algorithms but shows consistently good results across all tasks and setting complexity, and Fast-AC-RanSaC runs faster than AC-RanSaC but at the price of lower performance across the board. Such detailed analysis of the behaviour of algorithms had not been done before, as previous benchmarks usually focus on testing algorithms across a wide variety of datasets and observe which one obtain the best average performances, without looking at the intrinsics of the dataset in detail.

Using the insight gained from the PnP benchmark, we were able to choose LRT, Fast-AC-RanSaC and AC-RanSaC to implement in the ColMap software and test the impact of Adaptative methods on a full reconstruction pipeline. To measure the impact of each RanSaC separately we only substituted the robust estimation of the PnP problem in the

whole pipeline. The resulting analysis showed little impact in term of performance from the tested algorithms. All algorithms obtained similar level of quality as the default LO-RanSaC with a slight increase in global runtime, fully due to the increased runtime of the RanSaC step. The only exception being AC-RanSaC which performs better than LO-RanSaC on the most extreme cases where performance drops fast, cases which should be rare as initialisation should prevent them from happening, either by failing before the PnP step or cleaning the data before hand. The efficiency of the Bundle adjustment and other post-processing implemented by ColMap removes the gain of using an adaptative method in most cases. We still were able to remove an hyperparameter of ColMap without any drop in classification performance and with a small increase in runtime but the impact is not significant enough.

In future work, we would like to see the impact of adaptative method on the initialisation step of the reconstruction pipeline. Our experiment revealed that wide angles between initial images are still very hard to tackle for the ColMap software and that it will quickly fail to find any good model to initialise the reconstruction. Using a more robust method might prove beneficial, allowing to diminish the number of images required to get a full reconstruction, and tackling more challenging datasets with sparser views. Moreover, we hope to increase the quality of the reconstruction in complex cases which could then be used as more accurate training data for Machine Learning solutions and thus improve the State-of-the-Art for those algorithms as well.

Bibliography

[1] Yousset I Abdel-Aziz, Hauck Michael Karara, and Michael Hauck, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry", in: *Photogrammetric engineering & remote sensing* 81.2 (2015), pp. 103–107.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski, "Building rome in a day", in: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[3] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team, *Ceres Solver*, version 2.1, Mar. 2022, URL: `https://github.com/ceres-solver/ceres-solver`.

[4] Adnan Ansar and Konstantinos Daniilidis, "Linear pose estimation from points or lines", in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.5 (2003), pp. 578–589.

[5] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors", in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5173–5182.

[6] Daniel Barath, Jiri Matas, and Jana Noskova, "MAGSAC: marginalizing sample consensus", in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10197–10205.

[7] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas, "MAGSAC++, a fast, reliable and accurate robust estimator", in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1304–1312.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (SURF)", in: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[9] G. Bradski, "The OpenCV Library", in: *Dr. Dobb's Journal of Software Tools* (2000).

[10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "Brief: Binary robust independent elementary features", in: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, Springer, 2010, pp. 778–792.

[11] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al., "Efficient geometry-aware 3D generative adversarial networks", in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16123–16133.

[12] Yu Chen, Yisong Chen, and Guoping Wang, "Bundle adjustment revisited", in: *arXiv preprint arXiv:1912.03858* (2019).

[13] Jongmoo Choi and Gerard Medioni, "StaRSaC: Stable random sample consensus for parameter estimation", in: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 675–682, DOI: 10.1109/CVPR.2009.5206678.

[14] O. Chum and J. Matas, "Matching with PROSAC - progressive sample consensus", in: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 220–226 vol. 1, DOI: 10.1109/CVPR.2005.221.

[15] Ondrej Chum, Tomas Werner, and Jiri Matas, "Two-view geometry estimation unaffected by a dominant plane", in: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, IEEE, 2005, pp. 772–779.

[16] Ondřej Chum, Jiři Matas, and Josef Kittler, "Locally optimized RANSAC", in: *Joint Pattern Recognition Symposium*, Springer, 2003, pp. 236–243.

[17] Andrea Cohen and Christopher Zach, "The Likelihood-Ratio Test and Efficient Robust Estimation", in: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2282–2290, DOI: 10.1109/ICCV.2015.263.

[18] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018, URL: http://www.blender.org.

[19] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection", in: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, Ieee, 2005, pp. 886–893.

[20] Jiri Matas Daniel Barath and Levente Hajder, "Multi-H: Efficient recovery of tangent planes in stereo images", in: *Proceedings of the British Machine Vision Conference (BMVC)*, ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith, BMVA Press, Sept. 2016, pp. 13.1–13.13, ISBN: 1-901725-59-6, DOI: 10.5244/C.30.13, URL: https://dx.doi.org/10.5244/C.30.13.

[21] François Darmon, "Deep Learning based 3D reconstruction: supervision and representation", PhD thesis, École des Ponts ParisTech, 2022.

[22] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan, "Depth-supervised nerf: Fewer views and faster training for free", in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12882–12891.

[23] Michel Dhome, Marc Richetin, J-T Lapreste, and Gerard Rives, "Determination of the attitude of 3D objects from a single perspective view", in: *IEEE transactions on pattern analysis and machine intelligence* 11.12 (1989), pp. 1265–1278.

[24] Olivier D Faugeras, "What can be seen in three dimensions with an uncalibrated stereo rig?", in: *Computer Vision—ECCV'92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2*, Springer, 1992, pp. 563–578.

[25] Jean-Charles Faugère, Guillaume Moroz, Fabrice Rouillier, and Mohab Safey El Din, "Classification of the perspective-three-point problem, discriminant variety and real solving polynomial systems of inequalities", in: *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, 2008, pp. 79–86.

[26] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", in: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[27] Martin A Fischler and Robert C Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", in: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[28] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng, "Complete solution classification for the perspective-three-point problem", in: *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), pp. 930–943.

[29] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] Robert M Haralick, Chung-nan Lee, Kars Ottenburg, and Michael Nölle, "Analysis and solutions of the three point perspective pose estimation problem.", in: *CVPR*, vol. 91, 1991, pp. 592–598.

[31] Chris Harris, Mike Stephens, et al., "A combined corner and edge detector", in: *Alvey vision conference*, vol. 15, 50, Citeseer, 1988, pp. 10–5244.

[32] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd, ISBN 978-0521540513, Cambridge University Press, 2004.

[33] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[34] Richard I Hartley, "In defense of the eight-point algorithm", in: *IEEE Transactions on pattern analysis and machine intelligence* 19.6 (1997), pp. 580–593.

[35] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm, "Reconstructing the world* in six days*(as captured by the yahoo 100 million image dataset)", in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3287–3295.

[36] Radu Horaud, Bernard Conio, Olivier Leboulleux, and Bernard Lacolle, "An analytic solution for the perspective 4-point problem", in: *Computer Vision, Graphics, and Image Processing* 47.1 (1989), pp. 33–44.

[37] Michal Irani and Prabu Anandan, "Parallax geometry of pairs of points for 3D scene analysis", in: *Computer Vision—ECCV'96: 4th European Conference on Computer Vision Cambridge, UK, April 15–18, 1996 Proceedings, Volume I 4*, Springer, 1996, pp. 17–30.

[38] Hossam Isack and Yuri Boykov, "Energy-Based Geometric Multi-model Fitting", in: *International Journal of Computer Vision (IJCV)* 97.2 (Apr. 2012), pp. 123–147, ISSN: 1573-1405, DOI: 10.1007/s11263-011-0474-7, URL: https://doi.org/10.1007/s11263-011-0474-7.

[39] Maksym Ivashechkin, Daniel Barath, and Jiři Matas, "Vsac: Efficient and accurate estimator for h and f", in: *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 15243–15252.

[40] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun, "Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction", in: *ACM Transactions on Graphics* 36.4 (2017).

[41] Anton Konouchine, Victor Gaganov, and Vladimir Veznevets, "AMLESAC: A new maximum likelihood robust estimator", in: *Proc. Graphicon*, vol. 5, 2005, pp. 93–100.

[42] Karel Lebeda, Jirı Matas, and Ondrej Chum, "Fixing the locally optimized ransac–full experimental evaluation", in: *British machine vision conference*, vol. 2, Citeseer, 2012.

[43] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua, "Epnp: An accurate o (n) solution to the pnp problem", in: *International journal of computer vision* 81.2 (2009), pp. 155–166.

[44] Annick M Leroy and Peter J Rousseeuw, "Robust regression and outlier detection", in: *Wiley series in probability and mathematical statistics* (1987).

[45] Kenneth Levenberg, "A method for the solution of certain non-linear problems in least squares", in: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[46] Zhengqi Li and Noah Snavely, "MegaDepth: Learning single-view depth prediction from internet photos", in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2041–2050.

[47] Tony Lindeberg, "Feature detection with automatic scale selection", in: *International journal of computer vision* 30 (1998), pp. 79–116.

[48] Tony Lindeberg, "Image matching using generalized scale-space interest points", in: *Journal of mathematical Imaging and Vision* 52 (2015), pp. 3–36.

[49] Tony Lindeberg, *Scale-space theory in computer vision*, vol. 256, Springer Science & Business Media, 2013.

[50] Tony Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales", in: *Journal of applied statistics* 21.1-2 (1994), pp. 225–270.

[51] H Christopher Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", in: *Nature* 293.5828 (1981), pp. 133–135.

[52] David G Lowe, "Object recognition from local scale-invariant features", in: *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, Ieee, 1999, pp. 1150–1157.

[53] David G. Lowe, "Distinctive image features from scale-invariant keypoints", in: *International Journal of Computer Vision (IJCV)* 60.2 (2004), pp. 91–110.

[54] C-P Lu, Gregory D Hager, and Eric Mjolsness, "Fast and globally convergent pose estimation from video images", in: *IEEE transactions on pattern analysis and machine intelligence* 22.6 (2000), pp. 610–622.

[55] Luca Magri and Andrea Fusiello, "T-Linkage: A Continuous Relaxation of J-Linkage for Multi-model Fitting", in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014, pp. 3954–3961, DOI: 10.1109/CVPR.2014.505.

[56] Donald W Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", in: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[57] J. Matas and O. Chum, "Randomized RANSAC with sequential probability ratio test", in: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, 2005, 1727–1732 Vol. 2, DOI: 10.1109/ICCV.2005.198.

[58] James V. Miller and Charles V. Stewart, "MUSE: Robust Surface Fitting using Unbiased Scale Estimates", in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Nov. 1996), pp. 300–306, DOI: 10.1109/CVPR.1996.517089.

[59] Dmytro Mishkin, Jiri Matas, and Michal Perdoch, "MODS: Fast and robust method for two-view matching", in: *Computer Vision and Image Understanding* (2015), pp. - , ISSN: 1077-3142, DOI: http://dx.doi.org/10.1016/j.cviu.2015.08.005, URL: http://www.sciencedirect.com/science/article/pii/S1077314215001800.

[60] Lionel Moisan, Pierre Moulon, and Pascal Monasse, "Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers", in: *Image Processing On Line (IPOL)* 2 (2012), pp. 56–73.

[61] Lionel Moisan, Pierre Moulon, and Pascal Monasse, "Fundamental Matrix of a Stereo Pair, with A Contrario Elimination of Outliers", in: *Image Processing On Line (IPOL)* 6 (2016), pp. 89–113.

[62] Lionel Moisan and Bérenger Stival, "A probabilistic criterion to detect rigid point matches between two images and estimate the fundamental matrix", in: *International Journal of Computer Vision (IJCV)* 57.3 (2004), pp. 201–218.

[63] Pierre Moulon, *AC-RanSaC implementation*, https://github.com/pmoulon/IPOL_AC_RANSAC, [Online; accessed 22-January-2021], 2012.

[64] Pierre Moulon, Pascal Monasse, and Renaud Marlet, "Adaptive structure from motion with a contrario model estimation", in: *Proceedings ot the Asian Conference of Computer Vision (ACCV)*, Springer, 2012, pp. 257–270.

[65] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet, "OpenMVG: Open multiple view geometry", in: *International Workshop on Reproducible Research in Pattern Recognition*, Springer, 2016, pp. 60–74.

[66] Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet, "OpenMVG: Open multiple view geometry", in: *International Workshop on Reproducible Research in Pattern Recognition*, Springer, 2016, pp. 60–74.

[67] Marius Muja and David G Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.", in: *VISAPP (1)* 2.331-340 (2009), p. 2.

[68] Kai Ni, Hailin Jin, and Frank Dellaert, "GroupSAC: Efficient consensus in the presence of groupings", in: *2009 IEEE 12th International Conference on Computer Vision*, IEEE, 2009, pp. 2193–2200.

[69] David Nistér, "An efficient solution to the five-point relative pose problem", in: *IEEE transactions on pattern analysis and machine intelligence* 26.6 (2004), pp. 756–770.

[70] David Nistér, "An efficient solution to the five-point relative pose problem", in: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 26.6 (2004), pp. 756–770.

[71] Long Quan and Zhongdan Lan, "Linear n-point camera pose determination", in: *IEEE Transactions on pattern analysis and machine intelligence* 21.8 (1999), pp. 774–780.

[72] Julien Rabin, Julie Delon, Yann Gousseau, and Lionel Moisan, "MAC-RANSAC: a robust algorithm for the recognition of multiple objects", in: *Proceedings of the Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)* (2010), pp. 51–58.

[73] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm, "USAC: a universal framework for random sample consensus", in: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.8 (2012), pp. 2022–2038.

[74] Rahul Raguram and Jan-Michael Frahm, "Recon: Scale-adaptive robust estimation via residual consensus", in: *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1299–1306.

[75] Clément Riu, Vincent Nozick, and Pascal Monasse, "Automatic RANSAC by Likelihood Maximization", in: *Image Processing On Line* 12 (2022), pp. 27–49.

[76] Clément Riu, Vincent Nozick, and Pascal Monasse, "Automatic Threshold RanSaC Algorithms for Pose Estimation Tasks", in: *International Joint Conference on Computer Vision, Imaging and Computer Graphics*, Springer, 2022, pp. 1–20.

[77] Clément Riu, Vincent Nozick, Pascal Monasse, and Joachim Dehais, "Classification performance of RanSaC algorithms with automatic threshold estimation", in: *17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022)*, vol. 5, Scitepress, 2022, pp. 723–733.

[78] Peter J Rousseeuw and Annick M Leroy, *Robust regression and outlier detection*, John wiley & sons, 2005.

[79] Johannes L. Schonberger and Jan-Michael Frahm, "Structure-from-motion revisited", in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.

[80] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus, "Indoor segmentation and support inference from rgbd images", in: *European conference on computer vision*, Springer, 2012, pp. 746–760.

[81] Noah Snavely, Steven M Seitz, and Richard Szeliski, "Photo tourism: exploring photo collections in 3D", in: *ACM siggraph 2006 papers*, 2006, pp. 835–846.

[82] Thomas Sølund, Anders Glent Buch, Norbert Krüger, and Henrik Aanæs, "A Large-Scale 3D Object Recognition Dataset", in: *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 73–82, DOI: 10.1109/3DV.2016.16.

[83] C.V. Stewart, "MINPRAN: a new robust estimator for computer vision", in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.10 (1995), pp. 925–938, DOI: 10.1109/34.464558.

[84] Charles V. Stewart, "MINPRAN: A new robust estimator for computer vision", in: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.10 (1995), pp. 925–938.

[85] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers, "A benchmark for the evaluation of RGB-D SLAM systems", in: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580, DOI: 10.1109/IROS.2012.6385773.

[86] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis", in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[87] Roberto Toldo and Andrea Fusiello, "Image-consistent patches from unstructured points with J-linkage", in: *Image and Vision Computing* 31 (2013), pp. 756–770.

[88] Roberto Toldo and Andrea Fusiello, "Real-time incremental j-linkage for robust multiple structures estimation", in: *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, vol. 1, 2, 2010, p. 6.

[89] Philip Hilaire Torr, Slawomir J Nasuto, and John Mark Bishop, "Napsac: High noise, high dimensional robust estimation-it's in the bag", in: *British Machine Vision Conference (BMVC)*, vol. 2, 2002, p. 3.

[90] Bill Triggs, "Camera pose and calibration from 4 or 5 known 3d points", in: *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, IEEE, 1999, pp. 278–284.

[91] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon, "Bundle adjustment—a modern synthesis", in: *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, Springer, 2000, pp. 298–372.

[92] Shinji Umeyama, "Least-squares estimation of transformation parameters between two point patterns", in: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.04 (1991), pp. 376–380.

[93] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser, "Ibrnet: Learning multi-view image-based rendering", in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699.

[94] Tobias Weyand, Chih-Yun Tsai, and Bastian Leibe, "Fixing wtfs: Detecting image matches caused by watermarks, timestamps, and frames in internet photos", in: *2015 IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2015, pp. 1185–1192.

[95] Hoi Sim Wong, Tat-Jun Chin, Jin Yu, and David Suter, "Dynamic and hierarchical multi-structure geometric model fitting", in: *2011 International Conference on Computer Vision*, 2011, pp. 1044–1051, DOI: 10.1109/ICCV.2011.6126350.

[96] Changchang Wu, "Towards linear-time incremental structure from motion", in: *2013 International Conference on 3D Vision-3DV 2013*, IEEE, 2013, pp. 127–134.

[97] Gehua Yang, Charles V Stewart, Michal Sofka, and Chia-Ling Tsai, "Registration of challenging image pairs: Initialization, estimation, and decision", in: *IEEE transactions on pattern analysis and machine intelligence* 29.11 (2007), pp. 1973–1989.

Contents

Abstract **1**

Short Abstract - English Version . . . 1
Résumé court - Version Française . . . 2
Résumé substantiel - Version Française . . . 3

Introduction **13**

1 Traditional Reconstruction Pipeline **19**

1.1 Notations . . . 20
1.2 Model estimators . . . 22
1.2.1 Homography estimation . . . 23
1.2.2 Epipolar Geometry . . . 25
1.2.3 Perspective-from-n-Points . . . 27
1.3 Feature detection and matching . . . 29
1.4 Robust fitting methods . . . 32
1.4.1 The original RanSaC algorithm . . . 33
1.4.2 Early improvements of RanSaC . . . 37
1.4.3 Threshold-free algorithms . . . 48
1.4.4 Multi model methods . . . 56
1.5 Bundle Adjustment . . . 57

2 Semi-Artificial Data Generator **61**

2.1 Reasoning and motivation . . . 61
2.2 Metrics . . . 65
2.3 Model and inlier generation . . . 67
2.3.1 Generic method . . . 67
2.3.2 Generation for MVS and SfM . . . 70
2.4 Outlier generation . . . 73
2.4.1 Choice of outlier distribution . . . 73
2.4.2 For MVS and SfM . . . 74

3 Adaptative RanSaC Benchmark **79**

3.1 Tested algorithms . . . 79
3.2 Benchmark parameters and datasets . . . 83
3.3 Time . . . 85

3.3.1 Runtime analysis 86
3.3.2 Using early bailout to improve runtime 87
3.4 Classification performance 91
3.5 Validity of method and algorithm choice 95

4 Application to ColMap **97**
4.1 The ColMap pipeline 97
4.2 Experimental setup 100
4.2.1 Generating inlier data 101
4.2.2 Generating outlier data 101
4.2.3 Experimental parameters 102
4.3 Observation 105
4.3.1 Time analysis 105
4.3.2 Retrieval performance 107
4.4 Conclusion 109

Conclusion **111**

Bibliography **121**