Visual SLAM with automatic map update in dynamic environments

Youssef Bouaziz







UNIVERSITY of Clermont Auvergne - Clermont Ferrand

EDSPI DOCTORAL SCHOOL

P H D T H E S I S

to obtain the title of

DOCTOR OF THE UNIVERSITY

of Clermont Auvergne University - Clermont-Ferrand, France

Speciality : Computer Science

Defended by

Youssef Bouaziz

Visual SLAM with automatic map update in dynamic environments

Defended on

July 7, 2021

Jury :

Jury president:	Rémi Boutteau	-	Université de Rouen Normandie
Co-supervisor:	Guillaume Bresson	-	Institut Vedecom
Director:	Michel Dhome	-	CNRS
Reviewer:	Simon Lacroix	-	LAAS/CNRS
Reviewer:	Michèle Rombaut	-	Université Grenoble Alpes
Co-supervisor:	Eric Royer	-	Université Clermont Auvergne

Dédicaces

Du profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers,

À MA CHÈRE MÈRE

" Que ce travail soit l'expression de ma reconnaissance pour tes sacrifices consentis, ton soutien moral et matériel que tu n'as cessé de prodiguer.
Tu as tout fait pour mon bonheur et ma réussite.
Que dieu te préserve en bonne santé et t'accorde une longue vie "

À MON CHER PÈRE

" Je dédie cet événement marquant de ma vie à la mémoire de mon père disparu trop tôt. J'espère que, du monde qui est sien maintenant, il apprécie cet humble geste comme preuve de reconnaissance de la part d'un fils qui a toujours prié pour le salut de son âme "

À MA CHERE FÈMME

Pour la patience et le soutien dont elle a fait preuve pendant cette thèse, pour toute l'affection qu'elle m'a témoignée, pour ses précieux encouragements et pour sa présence malgré la distance qui nous sépare.

À MON FRÈRE ET MES SŒURS

" Vous étiez toujours présents pour m'aider et m'encourager.
Sachiez que vous serez toujours dans mon cœur "

À TOUS MES AMIS…

" A tous mes amis qui n'ont cessé de m'encourager et de me soutenir "

Youssef

Remerciements

Je tiens tout d'abord à exprimer mes sincères et chaleureux remerciements à mon directeur de thèse M. Michel Dhome et mes encadrants M. Éric Royer et M. Guillaume Bresson pour leur constante disponibilité, leur aide et leur encouragement ainsi que pour m'avoir fait bénéficier amplement de leur rigueur scientifique, de leurs critiques objectives et de leurs conseils avisés, aussi pour leurs qualités humaines qui suscitent mon admiration et mon profond respect.

J'exprime également ma profonde gratitude aux membres du jury Mr. Rémi Boutteau, Mr. Simon Lacroix et Mme. Michèle Rombaut pour avoir accepté d'examiner ce travail.

Je tiens aussi à témoigner ma profonde gratitude à tous les enseignants qui ont participé à mon évolution scientifique durant mon cursus scolaire et universitaire.

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du programme "Investissements d'Avenir" dans le cadre du Laboratoire d'Excellence IMobS3 (ANR-10-LABX-0016) et de l'Initiative d'Excellence IDEX-ISITE CAP 20-25 (ANR-16-IDEX-0001)

Abstract:
Appearance changes are one of the most challenging problems for the visual localization of an autonomous vehicle in outdoor environments. Data association between the current image and the landmarks in the map can be difficult if the map was built with different environmental conditions. The work developed in this PhD thesis is focused on life-long navigation for autonomous shuttles in outdoor environments where the appearance is subject to substantial changes.

In this thesis, we put forward several contributions to improve localization robustness and accuracy in long-term scenarios. The proposed approaches present a solution to build and use multi-session maps that incorporate sequences recorded in different conditions (day, night, fog, snow, rain, change of season, etc.). We can categorize our proposed approaches in two ways: keyframes retrieval and map management.

The keyframes retrieval process is carried out online during visual localization, in which, we take advantage of a visual landmark map composed of N sequences gathered at different times and conditions. During each localization session, we exploit information collected in the beginning of the trajectory to compute a ranking function which will be used in the rest of the trajectory to retrieve from the map the keyframes that maximise the number of matched points. The retrieval depends on the geometric distance between the pose of the keyframe and the current pose of the vehicle, and the similarity of this keyframe with the current environmental condition. In the mapping phase, covering all conditions by constantly adding data to the map leads to a continuous growth in the map size which in turn deteriorates the localization speed and performance. Therefore, we employ a map management phase which has the aim of maintaining a long-term reliable map with a fixed size throughout the frequent runs. To efficiently reduce the size of the map without significantly degrading the localization performance, we tend to remove superfluous data from the map. Accordingly, we have proposed different map management strategies by using different kind of information, such as the traversals similarity information computed during the keyframes retrieval process or the sun's position information related to each traversal, to analyse the data redundancy in the map.

Throughout this report, we provide the results of our experiments which were performed on real data collected with our autonomous shuttle as well as on a widely used public dataset. By comparing our proposed keyframes retrieval technique with a simple strategy where the keyframes are retrieved only according to their geometric distance to the vehicle pose, we have demonstrated that our approach has significantly improved localization performance in different challenging conditions. On several aspects related to map management, we demonstrated that our proposed strategies have outperformed a state-of-the-art map management approach.

We have also presented a new dataset that contains challenging environmental conditions which we make available to the community in the hope that it will be useful to other people working in this field.

Keywords:
Visual-Based Navigation, Computer Vision for Transportation, SLAM, Life-Long Navigation, Map Management, Landmark Retrieval, Autonomous Shuttles

Résumé étendu en français :

Ces dernières années, le SLAM visuel (localisation et cartographie simultanées) s'est avéré être un algorithme très puissant pour la localisation en temps réel de systèmes robotiques dans différentes applications. Il permet d'extraire des images d'une ou plusieurs des caméras embarquées du robot pour calculer simultanément la trajectoire des caméras (ce qui revient à calculer la trajectoire du robot) et la structure de la scène dans laquelle il opère.

Pour garantir le meilleur compromis vitesse/précision, la majorité des algorithmes de SLAM visuel sont basés sur une représentation éparse de l'environnement. En faisant correspondre des primitives visuelles (amers) sur des images successives, il est possible d'estimer la position 3D de ces primitives ainsi que les poses (position, orientation) des caméras. De nombreux progrès, tant dans la vitesse que dans la qualité de la trajectoire et de la reconstruction 3D, ont été réalisés ces dernières années. Cependant, la localisation par SLAM visuel présente encore plusieurs défis, nous en présentons quelques-uns dans la Figure I.

(a) Changement de saison ([Bai 2018]).

(b) Changement d'éclairage.

(c) Changement d'angle du point de vue.

FIGURE I – Exemples de défis pour le SLAM visuel.

Les changements d'apparence restent un défi crucial pour le SLAM visuel. Comme illustré dans la Figure I, les informations visuelles affectées par les changements saisonniers, les changements d'éclairage ou les changements de point de vue peuvent entraîner des difficultés majeures lors de l'association des données entre l'image actuelle et les amers qui se trouvent dans la carte. Par conséquent, la relocalisation dans des zones précédemment cartographiées peut être une tâche difficile car l'apparence de l'environnement change sans cesse dans les scénarios extérieurs et un tel phénomène peut entraîner une localisation inexacte ou erronée. Dans certaines applications comme les voitures autonomes, la partie d'estimation de la pose du processus de SLAM est très délicate et doit être prise avec beaucoup de prudence car même une petite erreur dans l'estimation de la pose du véhicule peut entraîner

un accident dangereux. Pour assurer une navigation sécurisée et sûre à long terme dans des environnements dynamiques, les véhicules autonomes doivent faire face à un tel défi.

L'association des données entre l'image actuelle et les amers dans la carte peut être difficile si la carte a été construite dans des conditions environnementales différentes. Les travaux développés dans cette thèse de doctorat se concentrent sur la localisation à long terme pour les navettes autonomes dans des environnements extérieurs où l'apparence est soumise à des changements substantiels.

Dans nos travaux, nous proposons plusieurs contributions pour améliorer la robustesse et la précision de la localisation dans des scénarios à long terme. Les approches proposées présentent une solution pour construire et utiliser des cartes multi-sessions qui intègrent des séquences enregistrées dans différentes conditions (jour, nuit, brouillard, neige, pluie, changement de saison, etc.). Nous pouvons catégoriser nos approches proposées en deux volets : la récupération des images clés et la gestion des cartes.

La récupération des images clés est effectuée lors de la localisation visuelle, nous exploitons une fonction de classement pour extraire les informations les plus pertinentes de la carte. Cette fonction de classement est conçue pour prendre en compte la pose du véhicule et les conditions environnementales actuelles. Dans la phase de cartographie, couvrir toutes les conditions en ajoutant constamment des données à la carte conduit à une croissance continue de la taille de la carte qui à son tour détériore la vitesse et les performances de localisation. C'est pour cette raison que nous avons conçu nos stratégies de gestion de la carte qui visent à limiter la taille de la carte tout en la gardant aussi diversifiée que possible.

L'estimation de la probabilité d'appariement entre l'image actuelle et une image clé de la carte est une tâche difficile car, comme l'illustre la Figure II, ces deux images peuvent avoir été enregistrées à différents points de vue, dans des conditions environnementales différentes, ou même les deux à la fois.

En conséquence, la première contribution de cette thèse consiste en une solution de récupération d'images clés face au problème posé dans la Figure II. Nous proposons une approche de localisation capable de tirer parti d'une carte visuelle composée de N séquences rassemblées à des moments différents. Cette approche peut être appliquée en particulier aux navettes autonomes car l'utilisation de N séquences ayant la même trajectoire pour construire une carte convient à un tel usage. En général, les algorithmes de SLAM récupèrent les images clés utilisées pour la localisation en fonction de leur distance géométrique par rapport à la pose du véhicule. Cependant, cette technique a montré une faiblesse dans la localisation à long terme car elle ne prend pas en compte les changements de l'environnement. L'exemple présenté dans la Figure III illustre un cas de mauvaise localisation où le SLAM a récupéré l'image clé la plus proche (encerclée) de la pose estimée du véhicule (violet). Un tel cas peut conduire à un échec de localisation car l'image clé récupérée et l'image actuelle ne partagent pas les mêmes conditions environnementales (correspondance entre jour et nuit).

Cela nous incite à développer une nouvelle stratégie de récupération d'images

(a) Points de vue différents.

(b) Conditions environnementales différentes.

(c) Points de vue et conditions environnementales différents.

FIGURE II – Un exemple de différents cas de mise en correspondance de l'image actuelle avec une image clé dans la carte. Dans (a), l'image actuelle (en haut) et l'image clé (en bas) ont des points de vue différents. Dans (b), les deux images ont un même point de vue mais des conditions environnementales différentes. Dans (c), les deux images ont à la fois un point de vue différent et des conditions environnementales différentes.

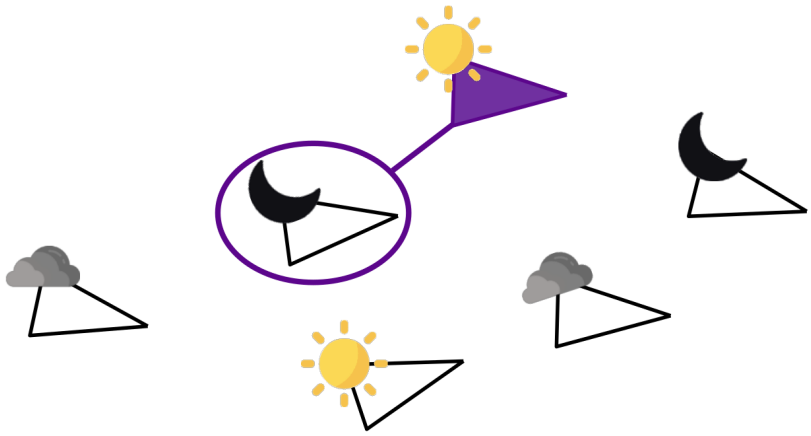


FIGURE III – Un exemple d'un cas de mauvaise localisation où le SLAM a récupéré l'image clé la plus proche qui a été enregistrée dans des conditions environnementales différentes.

clés afin d'adapter notre algorithme SLAM aux opérations à long terme. Au cours du processus de localisation, nous visons à maximiser le nombre de points appariés en récupérant des expériences pertinentes à partir d'une carte qui intègre de nombreuses conditions environnementales comme illustré dans la Figure IV. Notre

approche proposée tire parti des statistiques recueillies dans les premiers mètres d'une traversée pour calculer une fonction de classement probabiliste. Cette fonction est utilisée dans le reste de la traversée pour attribuer des scores de pertinence à un ensemble d'images clés dans la carte. Ces scores sont calculés par la fonction de classement en tenant compte des conditions environnementales de chaque image clé et de sa distance géométrique à la pose estimée du véhicule.

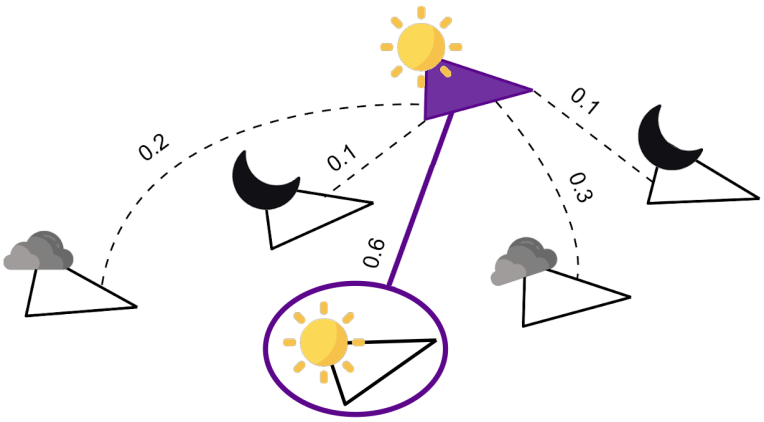


FIGURE IV – Les scores attribués aux images clés par la fonction de classement. Celle qui a le score le plus élevé (encerclée) sera récupérée et utilisé pour la localisation.

Dans la phase de cartographie, la création d'une carte qui couvre toutes les conditions environnementales en ajoutant sans cesse des amers à cette carte peut aider à améliorer les performances de localisation dans des conditions changeantes. Cependant, cela conduit également à une croissance incessante de la taille de la carte, qui est relative au nombre de traversées effectuées par la navette (expériences). Cela signifie que la localisation après plusieurs traversées nécessite une quantité immense de mémoire pour stocker la carte, ainsi qu'un processeur haut de gamme pour trouver des correspondances entre les points de l'image actuelle et les amers qui se trouvent dans une énorme base de données. En d'autres termes, la localisation en temps réel à long terme sera impossible après un certain nombre de sessions de localisation. Par conséquent, la deuxième contribution de cette thèse consiste en plusieurs stratégies de mise à jour des cartes qui ont été développées pour éviter de tels cas et garantir une localisation fiable en temps réel à long terme :

1) Nous avons exploité des scores de ressemblance entre les traversées calculées par l'approche de récupération des images clés pour déterminer quelle traversée a la plus grande similitude avec les autres pour la supprimer éventuellement.
2) Nous avons exploité les informations relatives à la position du soleil pour calculer la corrélation des traversées afin de produire une carte avec un nombre minimum de traversées ayant des conditions environnementales diverses.
3) Nous avons proposé une amélioration pour le travail de Mühlfellner *et al.* [Mühlfellner 2016]. Dans leur travail, Mühlfellner *et al.* attribuent des scores

aux amers en fonction du nombre de sessions de localisation différentes dans lesquelles ils apparaissent, ensuite, ils suppriment les amers ayant les scores les plus bas dans un processus exécuté hors ligne. De même, nous avons proposé de supprimer les amers ayant les scores les plus bas dans la carte tout en conservant un nombre uniforme d'amers à chaque traversée.

Toutes ces stratégies ont en commun leur capacité à produire une carte compressée et fiable qui assure une localisation à long terme sans, ou avec très peu d'erreurs de localisation.

Dans la Figure V, nous présentons un diagramme expliquant le mécanisme de fonctionnement de la récupération des images clés et le processus de gestion de la carte ainsi que la connexion entre eux. Notre système utilise une carte multi-session qui contient plusieurs traversées enregistrées dans différentes conditions environnementales. Chaque fois qu'une nouvelle image arrive, l'algorithme de récupération des images clés doit extraire des données cartographiques pertinentes pour la condition environnementale actuelle. Ces données récupérées sont ensuite utilisées pour calculer la pose du véhicule. Après la fin de la session de localisation, notre algorithme de gestion de carte sera exécuté hors ligne pour mettre à jour la carte en ajoutant de nouvelles données ou en supprimant les informations superflues.

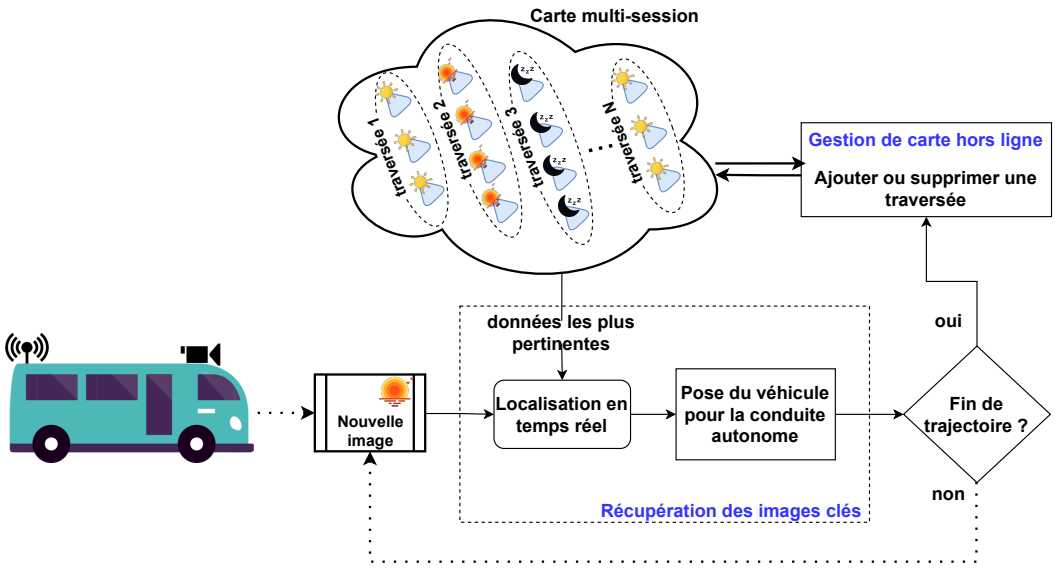


FIGURE V – Un schéma représentant le mécanisme de fonctionnement de la récupération des images clés et de la gestion de la carte proposées dans ce manuscrit.

La figure montre deux parties principales qui correspondent aux deux contributions fournies dans cette thèse. La première partie, qui correspond à la question de la recherche des amers pertinents pour la localisation, a été abordée dans de nombreux travaux récents. Sutherland and Thompson [Sutherland 1993] ont été parmi les premiers chercheurs à étudier ce problème. Ils ont démontré l'existence d'une forte dépendance entre l'erreur de localisation et la configuration des amers sélectionnés.

À cet égard, des efforts intensifs ont été consacrés à la sélection de l'ensemble optimal d'amers afin de réduire les erreurs de localisation. Par exemple, Mühlfellner *et al.* [Mühlfellner 2016] ont proposé une approche de sélection d'amers dans laquelle un score est attribué à chaque amer en fonction du nombre de fois où il a été observé. Les amers avec des scores élevés sont considérés comme utiles pour la localisation et par conséquent récupérés lors de la phase de localisation.

Dans le même esprit, Bürki *et al.* [Bürki 2016] ont développé un système de SLAM distribué avec une carte décentralisée sur le cloud pour des scénarios coopératifs multi-véhicules. Leurs travaux se sont concentrés sur le développement d'une solution en temps réel pour les environnements à bande passante limitée. Ils ont proposé une approche qui réduit le flux de données à distance en extrayant de la carte distribuée uniquement les informations pertinentes pour la localisation dans les conditions environnementales actuelles. L'approche proposée consiste, dans un premier temps, à créer une sorte de graphe qui représente la co-observabilité des amers dans toutes les traversées passés. Ensuite, ils utilisent une fonction de classement dont le but est de trouver la probabilité qu'un amer potentiel soit observé dans les conditions environnementales actuelles.

Contrairement à ces travaux qui ont abordé ce problème en sélectionnant un ensemble d'amers sur la carte, nous abordons la récupération d'une manière basée sur les images clés. Cela signifie que nous n'envisageons pas de récupérer les amers directement à partir de la carte, mais plutôt que nous recherchons les images clés les plus pertinentes pour la localisation et nous récupérons leurs amers associés. En fait, cette formulation d'images clés du problème de récupération des amers nous a permis de concevoir une fonction de classement probabiliste qui prend en compte non seulement les conditions environnementales des images clés (et leurs amers associés), mais aussi leurs distances géométriques par rapport à la pose du véhicule. Cela signifie également qu'au lieu de calculer un score pour chaque amer séparément, ce qui est coûteux et inutile puisque tous les amers associés à une même image clé partagent la même condition environnementale, la fonction de classement proposée calcule et attribue un score à chaque image clé, ce qui permet de minimiser le temps de calcul.

Afin de récupérer uniquement les informations pertinentes pour la localisation, nous avons conçu une fonction de classement qui est partiellement construite avec des données hors ligne. Le rôle de cette fonction est de récupérer à partir de la carte les images clés les plus pertinentes en tenant compte de deux facteurs principaux :

— La distance géométrique entre l'estimation de la pose du véhicule et la pose associée à l'image clé K_j qui est utilisée pour la localisation.

— Les conditions environnementales de l'image clé K_j ainsi que les conditions environnementales actuelles (éclairage, météo, saison,...).

La fonction de classement est calculée pendant les premiers mètres de la trajectoire. Après cela, elle sera utilisée pour estimer la probabilité qu'un point pt extrait de l'image courante I_i trouve une correspondance dans l'image clé K_j extraite de la

carte. La fonction de classement est décrite par l'Équation (1) :

$$P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot f_c(I_i, K_j) \tag{1}$$

$f_{dist}(I_i, K_j)$ est une fonction qui définit un score pour faire correspondre l'image courante I_i avec l'image clé K_j récupérée de la carte en supposant que les deux ont été prises dans des conditions similaires (météo, éclairage,...). Cela implique que le calcul de ce score ne dépend que de la distance géométrique entre les deux images. En d'autres termes, f_{dist} est défini comme la fonction qui calcule le pourcentage d'inliers en fonction de la distance longitudinale, latérale et angulaire entre les poses de deux images. En revanche, $f_c(I_i, K_j)$ suppose que les poses des deux images sont identiques ; il ne prend donc en compte que les changements d'apparence pour attribuer le score de correspondance entre les deux images I_i et K_j.

Le calcul de $P(pt \in I_i, K_j)$ ainsi que le calcul/mise à jour de la fonction f_c sont effectués en ligne lors de la re-localisation. Le terme re-localisation réfère au processus de localisation du véhicule dans un endroit précédemment cartographié.

Dans la Figure VI, nous présentons un diagramme qui explique le processus de re-localisation à l'aide de la fonction de classement proposée. Cette fonction de classement ($P(pt \in I_i, K_j)$) prend en entrée l'image courante (et sa pose prédite selon les données d'odométrie) pour récupérer de la carte multi-session l'image clé (et ses amers associés) qui a le score de similarité le plus élevé. Une nouvelle pose sera calculée en faisant correspondre l'image actuelle et l'image clé récupérée. Pour assurer la cohérence de f_c avec les changements des conditions environnementales, elle sera mise à jour tout au long de la trajectoire. Cette tâche de mise à jour est effectuée en parallèle avec la récupération de l'image clé la plus pertinente pour la localisation (selon la fonction de classement) en récupérant de la carte une autre image clé (choisie, chaque fois, au sein d'une traversée différente de manière circulaire) et en l'utilisant pour mettre à jour f_c.

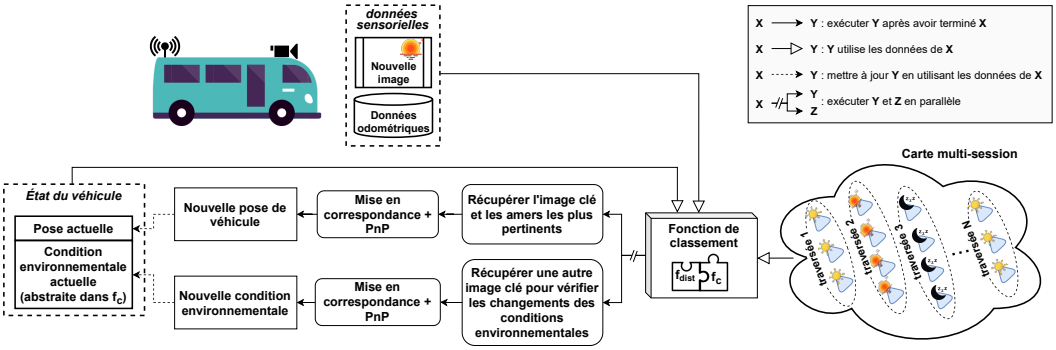


FIGURE VI – Un diagramme représentant le mécanisme de fonctionnement de la partie de re-localisation.

Comme indiqué dans l'Équation (1) ainsi que dans la Figure VI, la fonction de classement est constituée de deux fonctions f_{dist} et f_c. La fonction $f_{dist}(I_i, K_j)$ a été calculée hors ligne avec l'utilisation de certaines données collectées spécifiquement à

cet effet. Elle modélise le mouvement de la caméra entre la pose d'acquisition de I_i et de K_j par rapport aux déplacements longitudinaux, latéraux et angulaires (angle de lacet). La fonction $f_c(I_i, K_j)$ ne dépend pas de l'emplacement d'acquisition des images I_i et K_j, mais ne prend en compte que la condition de l'environnement pour estimer le score d'appariement entre I_i et K_j. En conséquence, nous en déduisons que f_c est dépendante de la traversée, cela signifie que la valeur de f_c entre I_i et toute image appartenant à la même traversée que K_j aura la même valeur que $f_c(I_i, K_j)$ (dans le cas normal où il n'y a pas de changement brusque dans les conditions météorologiques ou dans l'éclairage).

Par conséquent, il est plus approprié de concevoir la fonction f_c comme une matrice F_c de deux dimensions et qui a la forme de $[N \times N]$ (où N est le nombre de traversées existant dans la carte). La matrice F_c peut être formalisée comme suit : $f_c(I_i, K_j) = F_c(trav(I_i), trav(K_j))$, où la fonction $trav(X)$ se réfère à l'indice de la traversée à laquelle l'image X appartient (puisque I_i est l'image courante, $trav(I_i)$ fait référence à la traversée courante). La valeur de $F_c(trav(I_i), trav(K_j))$ correspond au score de similitude (qui dépend des conditions environnementales) entre la traversée qui contient I_i et la traversée qui contient K_j.

Afin de définir et de calculer la matrice F_c, nous utilisons notre carte globale (qui est composée de N séquences ayant des conditions environnementales différentes) comme carte de référence pour effectuer une re-localisation. Lors de cette phase de re-localisation, nous consacrons le début de la trajectoire au calcul de la fonction. D'après l'Équation (1), nous avons :

$$F_c(trav(I_i), trav(K_j)) = \frac{P(pt \in I_i, K_j)}{f_{dist}(I_i, K_j)} \tag{2}$$

Notre idée consiste à faire apprendre la matrice F_c au début de la trajectoire selon l'Équation (2). Puisque la fonction de classement n'étant pas encore définie, nous avons mesuré le taux d'inlier en appariant l'image I_i avec l'image clé K_j et l'avons utilisée pour remplacer $P(pt \in I_i, K_j)$ dans l'Équation (2).

Après avoir calculé la matrice F_c au début, le but dans le reste de la trajectoire est de trouver l'image clé K^* qui maximise la probabilité d'appariement définie dans l'Équation (3) :

$$K^* = \underset{K}{\operatorname{argmax}}\, P(pt \in I_i, K) \tag{3}$$

K^* correspond à l'image clé encerclée dans l'exemple montré dans la Figure IV. La recherche de K^* dans toute la carte est coûteuse. Par conséquent, nous choisissons de chaque traversée l'image clé $K^l, l \in [1, N]$ qui a la distance géométrique minimale par rapport à l'estimation de la pose actuelle du véhicule. Pour toutes les images clés $K^l, l \in [1, N]$, nous calculons la probabilité d'appariement avec l'Équation (1) pour récupérer celle qui a le score le plus élevé (K^*) avec l'Équation (3). K^* sera utilisé pour la mise en correspondance avec l'image actuelle et pour le calcul et l'optimisation de la pose.

Il est intéressant de mettre à jour les valeurs de la matrice F_c régulièrement après les premiers mètres de la trajectoire. En effet, cette mise à jour peut être utile lorsque l'état de l'environnement change entre le début et la fin de la session de re-localisation. Ce processus de mise à jour fonctionne en parallèle avec le processus de récupération des images clés. Pour cette raison, nous visons à éviter de ralentir la re-localisation en mettant à jour F_c pour une seule traversée à chaque itération.

Nous avons évalué notre approche sur le jeu de données d'Oxford [Maddern 2017] ainsi que sur notre propre jeu de données qui a été enregistré avec notre véhicule expérimental et qui contient, pour le moment, 127 séquences collectées sur deux ans dans lesquelles le véhicule a suivi le même chemin autour d'un parking avec de légères déviations latérales et angulaires. Ce jeu de données contient diverses conditions environnementales dues aux changements de luminance, de l'heure de la journée, de saisons et de véhicules stationnés, et chaque séquence mesure environ 200 m de longueur. Une vue aérienne du parking où nous avons enregistré notre jeu de données est présentée dans la Figure VII. Plus de détails sur le jeu de données IPLT sont fournis dans l'Annexe A.

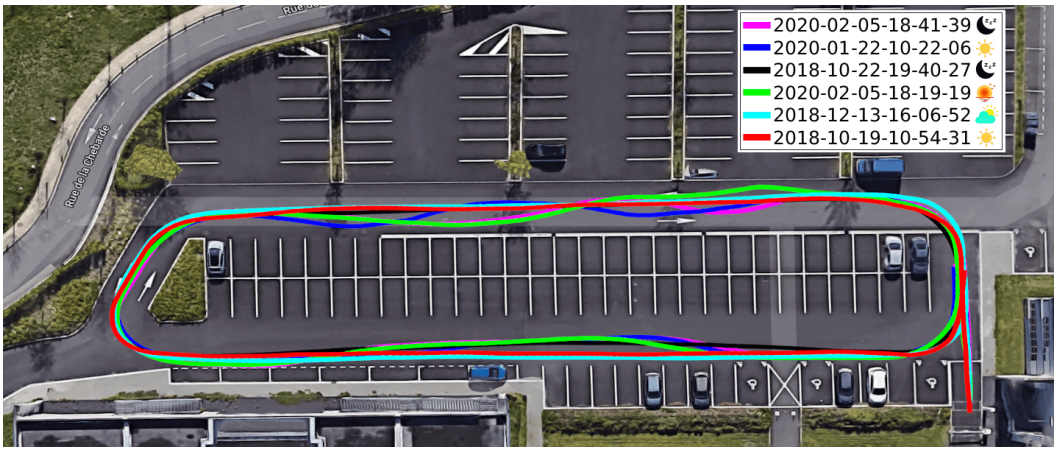


FIGURE VII – Une vue aérienne du parking où nous avons enregistré le jeu de données IPLT. Les courbes colorées représentent un exemple du chemin parcouru lors de l'enregistrement de 6 séquences différentes.

Pour le jeu de données IPLT, nous avons sélectionné 103 séquences ayant différentes heures de la journée, avec des conditions météorologiques diverses (pluie, soleil, couvert) et quelques déviations latérales et angulaires, 10 d'entre elles ont été utilisées pour la construction de la carte globale d'IPLT (séquences de cartographie). On peut voir sur la Figure VIII un aperçu des images prises de séquences de cartographie.

Pour le jeu de données d'Oxford, nous avons identifié un segment d'environ 1,6 km de long dans lequel le véhicule a suivi le même itinéraire et la même direction dans 20 séquences différentes. Huit d'entre elles ont été utilisées pour la construction de la carte globale d'Oxford. La Figure IX illustre un aperçu des images prises à partir des séquences de cartographie d'Oxford.

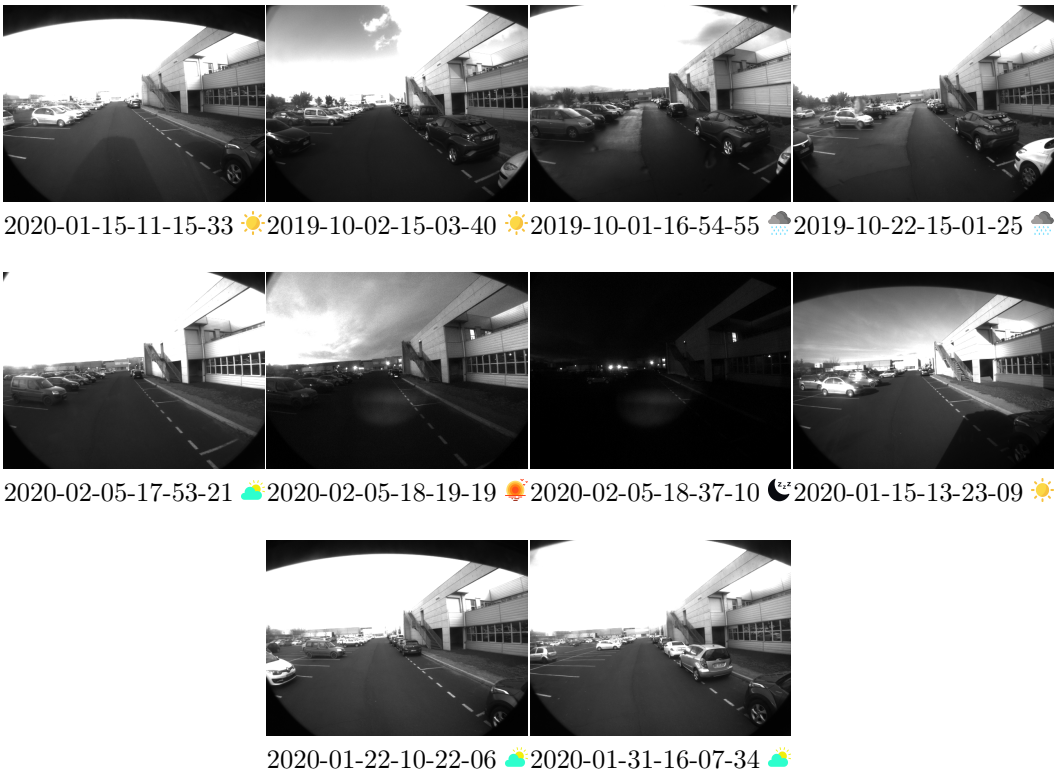


FIGURE VIII – Un aperçu des images de jeu de données IPLT utilisé pour la construction de la première carte globale. Pour chaque séquence, nous indiquons la date d'acquisition au format « aaaa-mm-jj-HH-MM-SS » et symbolisons la condition environnementale par une petite icône.

Pour les deux jeux de données, nous comparons les résultats obtenus en utilisant $\tau_1 = f_{dist}(I_i, K_j)$ comme fonction de classement (qui ne prend en compte que la distance géométrique comme critère pour attribuer des scores aux images clés) avec les résultats obtenus en utilisant notre fonction de classement proposée $\tau_2 = P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot F_c(trav(I_i), trav(K_j))$ (qui prend en compte la distance géométrique et les conditions environnementales des images clés).

Nous présentons le nombre moyen d'inliers observés dans chaque séquence ainsi que le nombre d'échecs de localisation par km comme critères de comparaison. Pratiquement, nous avons constaté que la localisation peut être considérée comme fiable lorsqu'il y a au moins 30 points appariés entre l'image courante et la base de données, en dessous de ce seuil, nous considérons un échec de localisation. Il s'agit d'un seuil de précaution pour assurer la sécurité de notre navette autonome [Royer 2016].

Pour le jeu de données IPLT, nous utilisons 93 séquences ayant des conditions environnementales différentes pour la phase de test. Ces séquences de test ont été classifiées manuellement en 5 classes différentes en fonction de leurs conditions environnementales : « soleil », « couvert », « pluie », « crépuscule » et « nuit ». Ces 5 classes contiennent respectivement 13, 39, 12, 17 et 11 séquences. La classe

2014-06-24-14-47-45 2014-06-26-09-53-12 2014-12-05-11-09-10 2015-02-20-16-34-06

2015-07-08-13-37-17 2014-07-14-14-49-50 2014-11-25-09-18-32 2015-05-19-14-06-38

FIGURE IX – Un aperçu des images du jeu de données d'Oxford utilisé pour la construction de la carte d'Oxford.

« globale » regroupe l'ensemble des 93 séquences. La figure X présente une comparaison entre les performances de localisation sur la carte d'IPLT en utilisant τ_1 et τ_2 comme fonctions de classement.

Globalement, nous constatons que notre approche de récupération d'images clés a considérablement augmenté le nombre d'inliers observés lors de la re-localisation pour toutes les classes. Nous remarquons également qu'avec notre approche, nous avons réussi à réduire le nombre de mauvaises localisations par km de $61,24$ à seulement $0,16$ (soit à peu près $400\times$ moins). Un tel critère (le nombre d'échecs de localisation) est très important dans les applications des navettes autonomes car un échec de localisation conduit à l'interruption du système de conduite autonome et nécessite des interventions manuelles. Par exemple, si la navette parcourt 100 km par jour, alors notre approche permet de réduire les interventions manuelles quotidiennes de plus de 6 000 à seulement 16.

Pour évaluer les performances de notre approche sur le jeu de données d'Oxford, nous avons utilisé 12 séquences de test avec des conditions environnementales différentes. Ces séquences de test ont été classifiées en 4 classes : « soleil », « couvert », « pluie » et « neige ». La classe « soleil » contient 5 séquences, la classe « couvert » contient 4 séquences, la classe « pluie » contient 2 séquences et la classe « neige » ne contient qu'une seule séquence.

Dans la Figure XI, nous comparons les performances de re-localisation de τ_1 et τ_2 sur la carte globale d'Oxford qui comprend 8 séquences (comme présenté dans la figure IX). La comparaison a été effectuée par rapport au nombre moyen d'inliers par image et aux échecs de localisation par km, comme nous l'avons fait pour le jeu de données IPLT.

Selon la figure, τ_2 a réussi à augmenter le nombre d'inliers et à réduire le nombre d'échecs de localisation de 9 échecs par km à seulement 0,5. Dans ce jeu de données, nous considérons une augmentation moins significative en termes d'inliers par rapport à l'augmentation observée sur le jeu de données IPLT (Figure X). Cela est dû

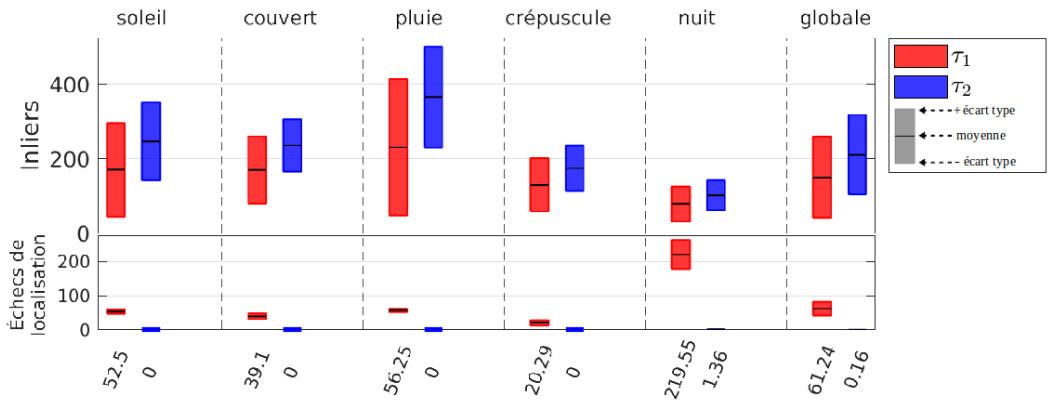


FIGURE X – Une comparaison entre les deux fonctions de classement τ_1 et τ_2 sur la carte IPLT à l'aide des 93 séquences de test du jeu de données IPLT. La comparaison se fait selon deux critères : le nombre moyen d'inliers par image et le nombre moyen d'échecs de localisation par km. Chaque case représente la valeur moyenne + et - l'écart type des inliers (ou échecs de localisation) enregistrés lors de la re-localisation en utilisant toutes les séquences de la classe correspondante sur la carte globale. La couleur des cases indique quelle fonction de classement a été utilisée pour enregistrer ces valeurs. Pour une meilleure lisibilité des valeurs des échecs de localisation, nous représentons le nombre moyen d'échecs par km de localisation sur l'axe des abscisses du bas.

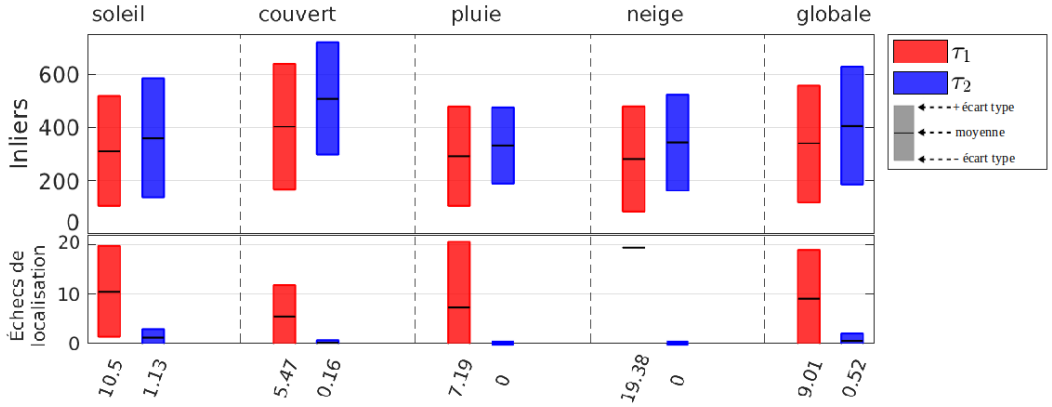


FIGURE XI – Une comparaison entre les deux fonctions de classement τ_1 et τ_2 sur la carte globale d'Oxford à l'aide des 12 séquences de test de ce jeu de données.

au fait que le jeu de données IPLT contient plus de séquences avec plus de changements dans les conditions environnementales et il comprend des séquences avec des déviations latérales et angulaires qui ont un impact significatif sur les performances de τ_1.

Malgré cette amélioration significative des performances de localisation, les techniques de récupération des amers, comme notre approche définie par τ_2, ne sont pas suffisantes pour assurer une navigation à long terme pour les robots autonomes. Ces types d'approches atteignent leurs limites lorsque la taille de la carte augmente

considérablement, ce qui entraîne une augmentation significative du temps de calcul. Par conséquent, la deuxième contribution de notre thèse vise à résoudre ce problème. A cet égard, nous proposons différentes techniques de gestion de la carte dont le rôle est d'éviter l'expansion continue de la carte.

D'autres travaux ont également abordé le problème de la gestion des cartes ces dernières années [Biber 2005], [Mühlfellner 2016], [Dymczyk 2015], [Bürki 2018], [Krajník 2016], [Halodová 2019], ... Certains de ces travaux réduisent la taille de la carte selon une politique qui suggère de supprimer les amers qui ont des taux d'observation faibles [Mühlfellner 2016, Dymczyk 2015]. Certains autres travaux ont proposé d'éliminer les amers qui ont été incorporés dans les échecs de localisation [Bürki 2018] et ont un taux élevé de correspondances incorrectes [Halodová 2019].

Le travail effectué dans cette thèse aborde également le même problème de différentes manières. Afin d'assurer une cohérence à long terme de la carte, nous avons proposé trois différentes techniques de gestion de carte dans le but de limiter sa taille au cours des multiples parcours de la navette autonome, tout en évitant de dégrader les performances de la re-localisation.

La première technique de gestion de la carte correspond à une amélioration de l'approche « Summary Maps » (désignée par SM) proposée par Mühlfellner *et al.* [Mühlfellner 2016]. Dans ce travail, les auteurs donnent des scores aux amers en fonction du nombre de sessions de localisation différentes dans lesquels ils apparaissent, et ils suppriment celles qui ont les scores les plus bas dans un processus hors ligne. Dans cette approche, nous avons identifié une limitation qui peut être définie comme un problème de biais vers des conditions environnementales plus fréquemment rencontrées. Ce biais se produit lorsqu'un ensemble de séquences avec des conditions environnementales similaires est mélangé avec une séquence ayant une condition environnementale étrange dans la même carte (par exemple, un ensemble de séquences de jour est inclus dans la même carte avec une séquence de nuit). Dans ce cas, les amers observés dans la condition environnementale étrange se voient attribuer des scores faibles par la fonction de classement car ils n'ont pas été observés aussi fréquemment que les amers des autres séquences, ce qui signifie qu'ils seront filtrés lors de l'étape de compression de la carte.

Par conséquent, nous proposons une amélioration de la technique « Summary Maps » proposée par Mühlfellner *et al.*. Notre version améliorée consiste à imposer une contrainte qui assure la distribution d'un nombre uniforme d'amers dans toutes les traversées après la compression de la carte. Par conséquent, nous appelons notre technique « Uniform Summary Maps » (et la désignons par USM).

La deuxième technique de gestion de la carte est une technique basée sur les traversées. Cela signifie qu'elle est utilisée pour limiter le nombre de traversées dans la carte à un nombre fixe $\hat{N}$. Cette valeur ($\hat{N}$) représente le nombre de traversées à maintenir dans la carte après avoir effectué la gestion de la carte. Pour cette raison, nous utilisons les informations relatives à la position du soleil pour calculer la corrélation entre les traversées afin de produire une carte ayant $\hat{N}$ traversées avec diverses conditions environnementales. Différentes valeurs de $\hat{N}$ ont été choisies dans

nos expériences pour évaluer l'efficacité de notre approche dans différents cas.

La méthodologie de notre approche est décrite dans la Figure XII. Après chaque session de localisation, nous testons si le nombre de traversées dans la carte (N) est supérieur au nombre prédéfini $\hat{N}$. Si tel est le cas, notre algorithme de gestion de carte sera exécuté hors ligne pour réduire la taille de la carte. $N > \hat{N}$ signifie que la carte contient une traversée supplémentaire ($N = \hat{N} + 1$), ce qui implique aussi qu'elle occupe un espace mémoire supplémentaire. Dans ce cas, notre approche doit agir hors ligne pour limiter la taille de la carte, c'est-à-dire, elle doit décider quelle traversée a le plus de similitude avec les autres pour finalement la supprimer.

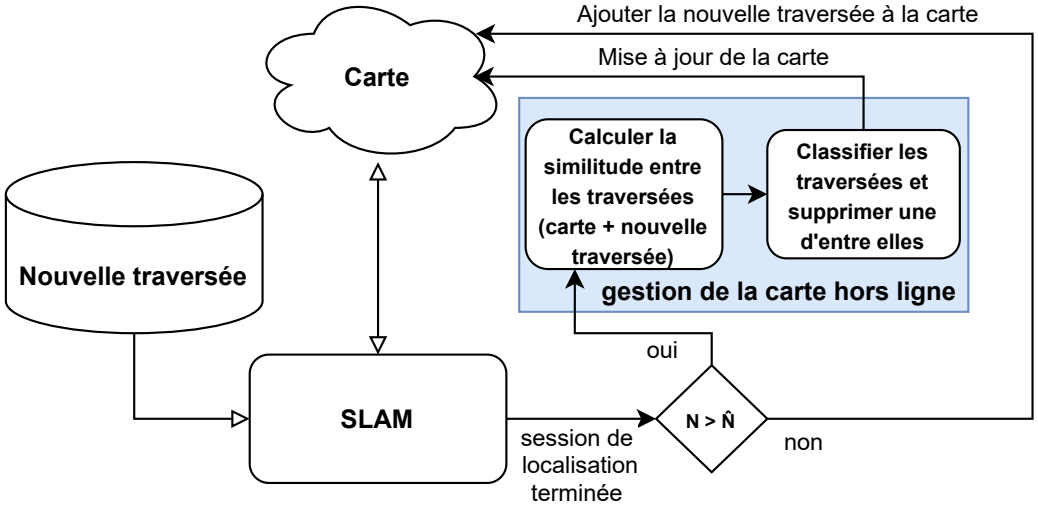


FIGURE XII – Un schéma illustrant le processus de la deuxième technique de gestion de la carte proposée dans cette thèse. Une nouvelle traversée est localisée et ajoutée à la carte existante. Après avoir terminé la re-localisation et la cartographie par SLAM, l'algorithme de gestion de la carte est exécuté hors ligne pour vérifier si la carte doit être compressée ou non (si $N > \hat{N}$). Si la réponse est oui, l'algorithme doit agir en deux parties. Tout d'abord, il doit calculer les similitudes entre les N traversées dans la carte ($N = \hat{N} + 1$). Deuxièmement, il utilise ces informations de similarité pour classifier toutes les traversées et sélectionner celle qui doit être supprimée.

Ce mécanisme comprend deux parties principales, le calcul de similarité et la classification. Dans la partie calcul de similarité, nous profitons de la position du soleil pour comparer la ressemblance entre les traversées de la carte afin de déterminer celles à conserver et celles à supprimer. Pour chaque traversées dans la carte, nous utilisons l'algorithme de l'Almanach astronomique [Michalsky 1988] pour calculer les coordonnées sphériques du soleil correspondantes : l'angle d'élévation solaire (el) et l'angle d'azimut solaire (az). Après avoir calculé toutes les positions du soleil associées aux N traversées dans la carte, nous générons une matrice de similarité D ayant la forme de $[N \times N]$. D est une matrice de distance symétrique 2D contenant les distances, prises par paires, entre les positions solaires calculées des N traversées.

Nous avons testé deux variantes de la matrice de similarité D. La première variante, D_{el}, est construite en calculant les distances entre les angles d'élévation

de chaque paire de traversées, et la deuxième variante, D_{az_el}, est construite en utilisant à la fois les angles d'élévation et d'azimut. Pour chaque traversée, on calcule le vecteur cartésien à partir des coordonnées sphériques, ensuite, on calcule l'angle entre chaque paire de vecteurs cartésiens.

Après avoir calculé D_{el} et D_{az_el}, un algorithme de classification hiérarchique [Szekely 2005] est utilisé pour classifier la matrice D (on applique cette méthode de la même manière sur D_{el} et sur D_{az_el}) afin de sélectionner la traversée à supprimer. Nous désignons notre technique par MMSE lorsqu'on utilise uniquement l'élévation solaire (lorsqu'on utilise la matrice D_{el}) et par MMSAE lorsqu'on utilise à la fois l'azimut et l'élévation solaires (lorsqu'on utilise la matrice D_{az_el}).

La localisation dans la nuit peut être considérée comme un cas particulier car il n'y a pas de lumière fournie par le soleil après le crépuscule astronomique. Par conséquent, les lampadaires sont presque la seule source de lumière, ce qui signifie que toutes les séquences enregistrées pendant la nuit sont visuellement similaires. Cela signifie qu'il est inutile de conserver plusieurs traversées de nuit dans la carte, et d'autre part, la suppression de toutes les traversées de nuit peut conduire à une dégradation significative des performances de localisation pour les séquences de nuit. Pour cette raison, nous imposons une nouvelle contrainte à notre algorithme : la traversée avec l'angle d'élévation du soleil le plus bas n'est pas désignée comme la traversée à supprimer (les traversées de nuit ont un angle d'élévation du soleil négatif), toute autre traversée de nuit sera supprimée. Cela garantit que la carte produite n'intégrera qu'une seule traversée de nuit.

Dans la troisième technique de gestion de la carte, nous proposons une méthode basée sur la matrice F_c définie précédemment. Comme dans la technique précédente, l'idée ici est de réduire la taille de la carte tout en la gardant aussi diversifiée que possible pour couvrir un nombre maximum de conditions environnementales différentes.

Le principe de cette approche est très similaire à celui proposé dans la technique précédente. La principale différence entre elles réside dans les données utilisées pour calculer la similitude entre les traversées. Par conséquent, la procédure de gestion de carte proposée ici consiste à limiter le nombre total de traversées dans la carte (N) à un nombre prédéfini de traversées $\hat{N}$. Lorsque le nombre de traversées dans la carte N dépasse le nombre prédéfini $\hat{N}$ ($N = \hat{N} + 1$), notre algorithme doit choisir un parcours à supprimer de la carte. Le choix dépend principalement de la matrice F_c définie par l'algorithme de récupération des images clés. Cependant, comme indiqué précédemment, F_c est régulièrement mise à jour le long de la trajectoire. Par conséquent, nous calculons sa valeur moyenne $\bar{F}_c$ à la fin de chaque session de re-localisation :

$$\bar{F}_c = \frac{1}{n}\sum_{i=1}^{n} F_c^i \tag{4}$$

où n est le nombre total d'images dans la trajectoire et F_c^i est la valeur de la matrice F_c à l'itération i.

Pour effectuer une classification hiérarchique sur la matrice $\bar{F}_c$, nous devons

d'abord la convertir en une matrice de distance. Puisque $\bar{F_c}$ est symétrique et que ses valeurs sont comprises entre 0 et $1/max(f_{dist})$, nous pouvons obtenir une matrice de distance en la normalisant comme suit :

$$D = J - \bar{F_c} \cdot max(f_{dist}) \tag{5}$$

où J est une matrice de uns avec la même dimension que $\bar{F_c}$ ($[N \times N]$). La matrice résultante D est une matrice de distance avec une diagonale nulle et toutes ses valeurs appartiennent à $[0, 1]$.

Nous avons utilisé l'algorithme de classification hiérarchique pour classifier la matrice normalisée D exactement de la même manière que nous l'avons fait avec D_{el} et D_{az_el} dans la technique précédente. Nous désignons cette technique qui est basée sur la matrice F_c par MMFc.

Nous avons testé nos techniques de gestion de carte (USM, MMSE, MMSAE, MMFc) sur les mêmes jeux de données que ceux utilisés pour évaluer la technique de récupération d'images clés. Nous avons également utilisé la même division pour les séquences de cartographie et les séquences de test (10 séquences de cartographie et 93 séquences de test pour l'ensemble de données IPLT, et 8 séquences de cartographie et 12 séquences de test pour l'ensemble de données Oxford RobotCar).

Dans la partie d'évaluation, nous visons à limiter le nombre de traversées utilisées pour construire la carte globale à un nombre prédéfini $\hat{N}$. En d'autres termes, nous cherchons à choisir la meilleure carte à $\hat{N}$-sessions parmi un total de N traversées (nous appelons une carte à $\hat{N}$-sessions une carte constituée à partir de $\hat{N}$ traversées).

Ce mode d'évaluation convient uniquement aux techniques basées sur les traversées (MMSE, MMSAE et MMFc), le concept de « Summary Maps » (SM) et de « Uniform Summary Maps » (USM) est assez différent des autres techniques puisque dans ces deux approches, la compression de la carte est effectuée d'une manière basée sur les amers, où les amers les moins pertinents sont supprimés. Par conséquent, pour comparer équitablement les deux techniques SM et USM avec les autres approches proposées (MMSE, MMSAE et MMFc), nous recherchons le taux de compression $\hat{r}$ qui génère une carte avec approximativement le même nombre d'amers incorporés dans une carte à $\hat{N}$-sessions :

$$\hat{r} = \frac{n_{\text{amers}}(M_N)}{n_{\text{amers}}(M_{\hat{N}})} \tag{6}$$

où M_X est une carte contenant X traversées (ou encore carte à X-sessions) et le terme $n_{\text{amers}}(M)$ désigne le nombre d'amers contenus dans la carte M.

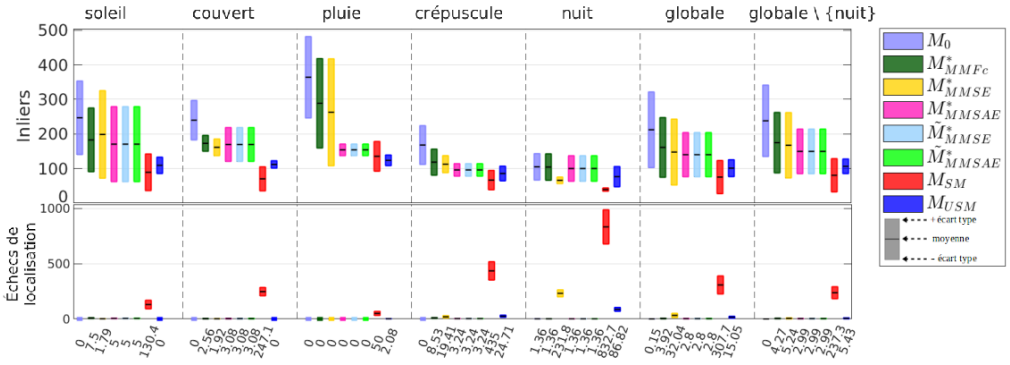
De cette manière, nous nous assurons que les cartes générées avec les techniques SM, USM, MMSE, MMSAE et MMFc sont équivalentes en termes de nombre d'amers (et respectivement en termes d'occupation de mémoire), ce qui garantit une comparaison équitable entre les performances de localisation après la compression de la carte.

Pour évaluer nos différentes approches, nous continuons à effectuer le SLAM

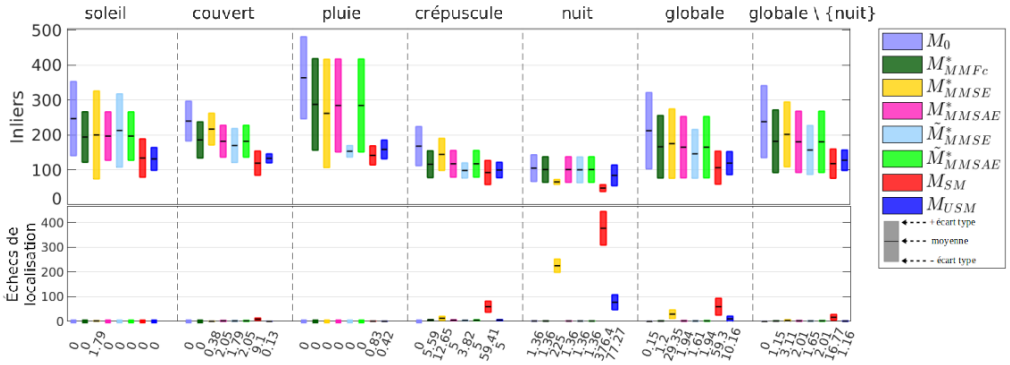
sur les séquences de cartographie, une par une, et à la fin de chaque session, nous utilisons nos différentes techniques pour réduire la taille de la carte. Considérant que l'ordre dans lequel les traversées sont ajoutées à la carte peut influencer la carte résultante, nous testons notre approche avec plusieurs permutations dans l'ordre des N traversées (on a testé 100 000 différentes permutations des 10 traversées de IPLT et toutes les permutations possibles de 8 traversées d'Oxford : 8! = 40 320). Cela conduit à la génération de plusieurs cartes à $\hat{N}$-sessions lorsque nous exécutons chaque technique de gestion de la carte (MMSE, MMSAE ou MMFc) plusieurs fois avec différentes permutations dans l'ordre des N traversées . Ainsi, nous choisissons la carte à $\hat{N}$-sessions la plus reproduite (notée M^*) et l'utilisons pour évaluer les performances de localisation avec les séquences de test. Ceci n'est possible qu'avec les techniques basées sur les traversées : MMSE, MMSAE et MMFc. Comme SM et USM ne sont pas des techniques basées sur les traversées, nous procédons simplement à une compression de carte après la fin de chaque session de SLAM avec un taux de compression $\hat{r}$ calculé par l'Équation (6).

Afin d'évaluer l'influence de la compression de la carte sur la performance de localisation avec le jeu de données IPLT, nous comparons les performances de localisation sur la carte globale d'IPLT M_0, qui est composée de $N = 10$ traversées, avec les performances de localisation sur la carte à $\hat{N}$-sessions la plus reproduite par chaque technique de gestion de la carte : M^*_{MMSE}, M^*_{MMSAE} et M^*_{MMFc} ainsi que les deux cartes produites par la technique classique de [Mühlfellner 2016] et celle après nos améliorations (M_{SM} et M_{USM}). Nous incluons également dans la comparaison deux autres cartes à $\hat{N}$-sessions, notées $\widetilde{M}^*_{MMSE}$ et $\widetilde{M}^*_{MMSAE}$, qui ont été reproduites avec les approches MMSE et MMSAE tout en imposant la contrainte qui assure le maintien d'une séquence de nuit dans la carte. Nous avons utilisé trois valeurs différentes de $\hat{N}$ pour démontrer l'efficacité de nos approches sur différentes configurations : $\hat{N} = \{3, 4, 5\}$.

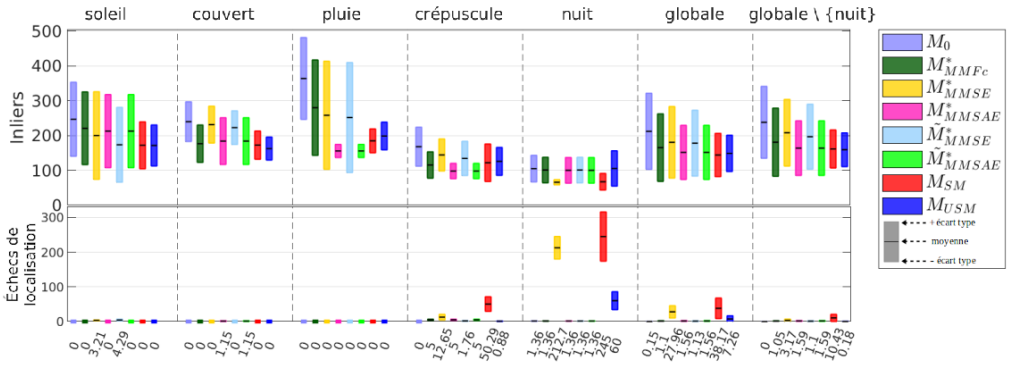
Dans la Figure XIII, nous présentons le nombre moyen d'inliers par image et le nombre moyen d'échecs de localisation par km enregistrés en re-localisant sur les différentes cartes. Les séquences de test ont été classifiées manuellement en 5 classes différentes comme expliqué précédemment. La classe « globale » contient les 93 séquences de test et la classe « globale $\setminus$ {nuit} » contient toutes les séquences à l'exclusion des séquences de nuit (82 séquences). La Figure XIII montre que la performance de localisation a globalement diminué après avoir limité la carte à $\hat{N} = 5$ traversées et a diminué davantage pour $\hat{N} = 4$ et 3 traversées. Pour tous les choix de $\hat{N}$, la performance de localisation sur la carte M^*_{MMSE} avec les séquences de nuit était très faible puisque cette carte n'inclut aucune traversée de nuit. Ce n'est pas le cas pour les cartes à $\hat{N}$-sessions M^*_{MMSAE} et M^*_{MMFc} qui intègrent une traversée de nuit. Les deux cartes $\widetilde{M}^*_{MMSE}$ et $\widetilde{M}^*_{MMSAE}$ incluent également une traversée de nuit grâce à la contrainte imposée. Par conséquent, la performance de localisation avec les séquences de nuit sur ces cartes est fiable (1.36 échecs de localisation par km, soit la même valeur enregistrée sur M_0).



(a) $\hat{N} = 3$



(b) $\hat{N} = 4$



(c) $\hat{N} = 5$

FIGURE XIII – Comparaison des performances de localisation entre M_0 et chacune des cartes produites par les différentes approches en utilisant les 93 séquences de test de jeu de données IPLT. Les sous-figures (a), (b) et (c) montrent les performances de localisation lors du choix de 3, 4 et 5 comme valeurs pour le paramètre $\hat{N}$, respectivement. Chaque case dans la figure représente la valeur moyenne + et - l'écart type des inliers (ou échecs de localisation) enregistrés lors de la re-localisation en utilisant toutes les séquences de la classe correspondante. La couleur de chaque case indique quelle technique de gestion de la carte a été utilisée pour produire la carte où la localisation a été effectuée. Pour une meilleure lisibilité des valeurs des échecs de localisation, nous représentons le nombre moyen d'échecs de localisation par km sur l'axe des abscisses du bas.

En revanche, la carte M_{SM} a montré une faiblesse majeure dans la localisation, notamment pour les séquences de nuit. Cela confirme que cette approche n'est pas bien adaptée lorsque les amers ne sont pas uniformément répartis sur les différentes conditions environnementales comme le cas de la carte globale M_0 d'IPLT, qui ne contient qu'une seule séquence de nuit (Figure VIII). Cela signifie que les amers observés pendant la nuit se voient attribuer un score faible puisqu'ils n'ont été observés qu'au cours d'une seule session (nous avons exprimé ce phénomène par un biais vers des conditions environnementales plus fréquemment rencontrées). Comme prévu, les performances de localisation ont été améliorées, en particulier pour les séquences de nuit, lors de l'utilisation de la carte M_{USM} qui comprend un nombre uniforme d'amers sur les différents traversées. Cependant, même avec l'amélioration, l'approche USM n'a pas été en mesure de fournir une très bonne performance de localisation pour les séquences de nuit. Cela signifie que la politique de classement adoptée dans les deux approches (SM et USM), qui supprime les amers en fonction de leur taux d'observation dans différentes sessions, ne permet pas de compresser efficacement la carte pour assurer une navigation à long terme.

En global, les performances de localisation ont été dégradées lors de la compression de la carte avec différentes valeurs de $\hat{N}$. Cependant, même avec cette dégradation, la localisation peut toujours être considérée comme fiable avec l'utilisation de certaines de nos techniques de gestion de la carte proposées, en particulier MMFc, qui fournit visiblement des résultats légèrement meilleurs que les autres méthodes. En utilisant MMFc avec $\hat{N} = 5$, nous considérons 1,1 échecs de localisation par km sur toutes les séquences de test (classe « globale »). Cela signifie que seulement 20 images parmi les 93 séquences (qui incorporent un total de 159 074 images) correspondaient à moins de 30 inliers (c'est-à-dire, $\sim 1,26e^{-4}$ échecs par image).

Comme nous l'avons fait avec le jeu de données IPLT, nous évaluons l'influence de la compression de la carte sur la performance de localisation avec le jeu de données d'Oxford avec $\hat{N} = 3$. Dans la Figure XIV, nous comparons les performances de localisation sur la carte globale d'Oxford M_0, qui est composée de $N = 8$ traversées, avec les performances de localisation sur les cartes à $\hat{N}$-sessions les plus reproduites pour chaque technique de gestion de la carte. Nous comparons également ces techniques avec les approches SM et USM. Les séquences utilisées pour la cartographie et pour le test n'intègrent aucune séquence de nuit dans ce jeu de données. Par conséquent, nous n'incluons pas les cartes $\widetilde{M}^*_{MMSE}$ et $\widetilde{M}^*_{MMSAE}$ dans la comparaison (les cartes qui ont été obtenues en imposant la contrainte qui assure le maintien d'une séquence de nuit dans la carte). La comparaison est effectuée en fonction de la moyenne des inliers par séquence et du nombre d'échecs de localisation par km.

Cette figure montre que les performances de localisation sur le jeu de données d'Oxford ont légèrement diminué avec $\hat{N} = 3$ lors de l'utilisation de nos différentes approches. Ce type de dégradation est attendu après compression de la carte de 8 traversées à seulement 3. D'après la figure, il y a un taux d'échec significatif pour les séquences de la classe « pluie ». Cela peut s'expliquer par le fait que les séquences de test incluent une séquence (2014-11-21-16-07-03 🌧️🌅) qui a été enregistrée au début du crépuscule et lors d'une forte pluie. Par contre, la localisation sur la classe

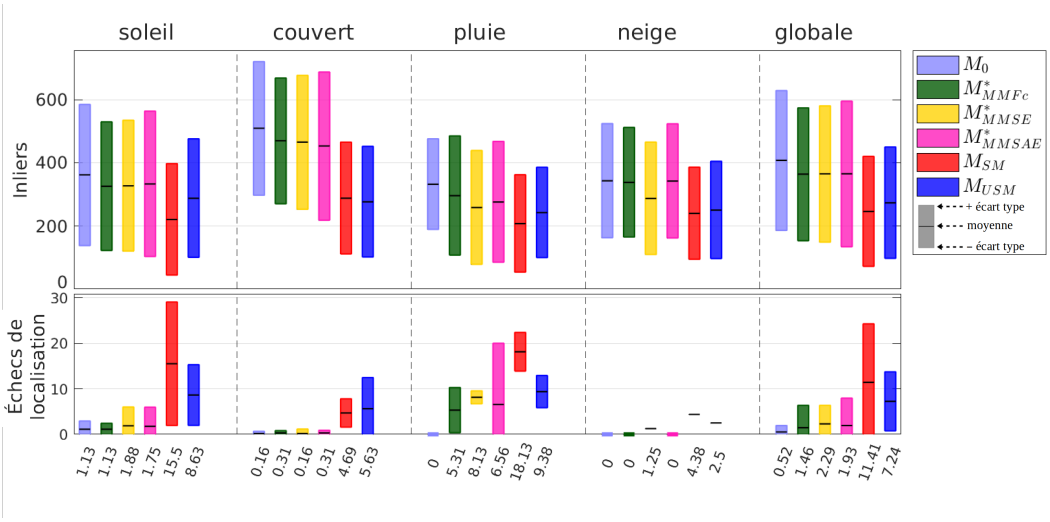


FIGURE XIV – Comparaison des performances de localisation entre M_0 et chacune des cartes produites par les différentes approches en utilisant les 12 séquences de test du jeu de données d'Oxford. $\tilde{M}^*_{MMSE}$ and $\tilde{M}^*_{MMSAE}$ sont exclus de la comparaison car ce jeu de données ne comporte aucune séquence de nuit.

« neige » (cette classe ne contient qu'une seule séquence 2015-02-03-08-45-10 🌨) n'a conduit à aucun échec de localisation avec M^*_{MMFc} et M^*_{MMSAE}. Ces deux cartes à $\hat{N}$-sessions contiennent la séquence 2015-02-20-16-34-06 🌧 qui est visuellement similaire à cette séquence de neige (2015-02-03-08-45-10 🌨) qui a été enregistrée lors d'une légère chute de neige.

Dans ce rapport, nous avons présenté trois algorithmes différents qui peuvent être utilisés pour un processus de localisation à long terme. Ces trois algorithmes ont en commun leur capacité à limiter la taille de la carte pour éviter son inflation continue et chacun d'eux est conçu pour utiliser un type différent d'informations pour supprimer les données superflues de la carte. Dans le premier algorithme, nous avons proposé une amélioration de la technique de « Summary Maps » de l'état de l'art pour éviter d'omettre des amers observés dans des conditions environnementales rarement rencontrées. Dans le deuxième algorithme, nous avons utilisé les informations de position du soleil comme guide pour filtrer les traversées et leurs amers associés qui ont été enregistrés dans des conditions d'éclairage similaires. Le dernier algorithme est basé sur certains concepts fondamentaux proposés dans l'approche de récupération d'images clés. Il utilise la matrice de similarité F_c comme information pour classifier les traversées dans la carte et supprimer celles les moins importantes en fonction de leur ressemblance.

Pour tester l'efficacité de nos approches proposées, nous avons évalué les performances de localisation à l'aide de deux jeux de données. À l'exception de notre technique améliorée de « Summary Maps » (USM) qui n'a pas donné de très bons résultats, nos approches se sont avérées capables de parvenir à une navigation fiable à long terme sur deux jeux de données différents tout en limitant la taille de la carte. Bien que les performances de localisation aient été légèrement réduites après

la compression de la carte, nos approches proposées ont réussi à limiter la complexité de calcul. Nous avons également démontré que nos différentes techniques ont nettement surpassé la performance de localisation avec une approche de l'état de l'art (« Summary Maps »).

Nous avons également enregistré le jeu de données IPLT, que nous avons mis à la disposition de la communauté dans l'espoir qu'il sera utile à d'autres chercheurs travaillant dans le domaine de la navigation à long terme. Ce jeu de données a été enregistré sur une période de deux ans dans un parking et comprend, à ce jour, 127 séquences enregistrées dans diverses conditions environnementales (luminance, météo, changements saisonniers ...).

Mots clés :
Navigation Visuelle, Vision par Ordinateur pour les Transports, SLAM, Navigation à Long Terme, Gestion de la Carte, Récupération des Amers, Navettes Autonomes

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

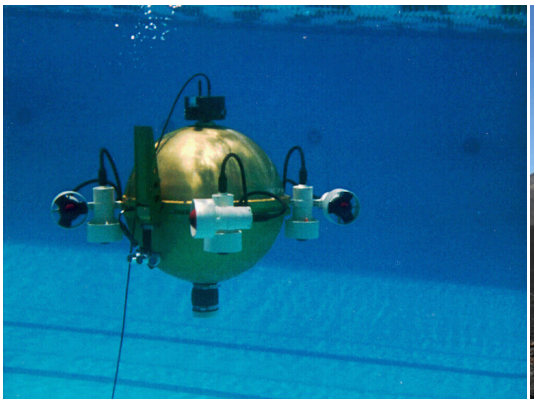
1.1 Context of the thesis

The work done in this PhD thesis is focused on visual-based long-term localization under challenging conditions. In recent years, visual SLAM (Simultaneous Localization And Mapping) has proven to be a very powerful algorithm for real-time localization of robotic systems in different applications. It makes possible to extract images from one or more of the robot on-board cameras to simultaneously calculate the trajectory of the cameras (i.e., calculate the trajectory of the robot) and the structure of the scene in which it operates.

The SLAM algorithm is a central topic in robotic research field and its applications are becoming more and more numerous. Actually, SLAM applications are covering a wide number of fields, ranging from smart vacuum cleaner robots [Hasan 2014], to the exploration of dangerous areas [Holmquist 2017, Weidner 2017, Vanicek 2018, Appapogu 2019] (e.g., areas of natural disasters, military combat zones, caves, underwater exploration, etc.) and the mapping of inaccessible environments to humans (e.g., NASA's Mars Exploration Rover 2003 [Di 2008] and 2020[1] missions, China's Chang'E-3 [WANG 2014] and Chang'E-4 [Liu 2020] missions). Figure 1.1 presents some examples of SLAM applications in various fields.

Recently, Visual SLAM is becoming a fundamental component in autonomous driving applications due to its inexpensive setup requirements and its efficiency in localizing vehicles without any prior knowledge of the operating environment. Most of the current SLAM algorithms are adapted to environments that are supposed to be rigid and static over time. However, in real-life scenarios, the environment is constantly changing. The automatic update of the 3D model of the scene is a difficult but crucial problem. In an urban scene, we can distinguish several types of

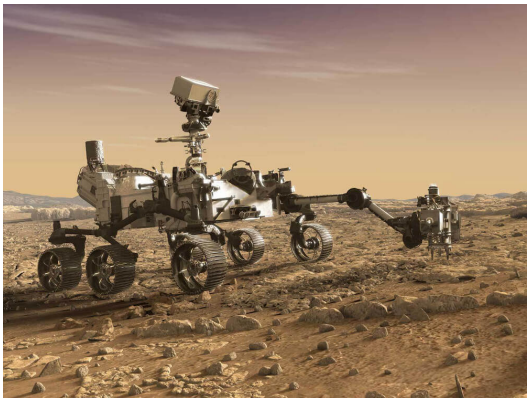
[1] https://mars.nasa.gov/mars2020/



(a) Submarine exploration [Antonelli 2001].

(b) Volcano exploration [Caltabiano 2005].

(c) Caves exploration [Gallant 2016].



(d) Space exploration (NASA's 2020 Perseverance Rover).

Figure 1.1: Example of SLAM application areas.

changes that must be treated differently. Figure 1.2 illustrates an example of some types of elements that can be found in an urban scene. Such a scene may contain:

(a) some elements that have a long-term durability (e.g. buildings, street furniture, pavement, etc.). These elements are modified rarely and they can stay unchanged for long periods.

(b) some elements that are temporary (such as parked cars) but they are always found in the same areas.

(c) other elements that have a stable geometry but change their appearance (e.g. a repainted building, an advertising panel whose poster is replaced constantly, etc.).

In this thesis, we are developing a navigation system for autonomous shuttles. In such a context, driverless shuttles are frequently revisiting the same places at different times of the day. This means that they are very likely to experience many

(a) Example of unchanged elements in a scene [Naseer 2014].

(b) Example of temporary parked cars.

(c) Example of an appearance change for a geometrically stable element.

Figure 1.2: Example of some types of elements that can be found in an urban scene.

different environmental conditions which can deteriorate the localization performance even when revisiting familiar places. In this thesis, we aim to improve the robustness of long-term navigation in such dynamic environments.

1.2 Problematic

The main challenge of mobile robotics is to design a system capable of localizing itself autonomously, quickly and safely, in an unfamiliar environment that can be dynamic, difficult and large, based only on the sensory data. SLAM algorithms have been attempting to accomplish this task for several years. The first solutions proposed for SLAM date back to the 1980s with the work of Smith and Cheeseman [Smith 1986]. Since then, this algorithm has reached a significant level of maturity, with a great number of real-time solutions reported by the state of the art. In the last few years, cameras have become one of the most commonly used types of sensors in SLAM thanks to their inexpensive setup requirements and their efficiency. Accordingly, Visual-SLAM has widely drawn attention of researchers and it is seeing a growing number of real-time applications in different disciplines.

To guarantee the best speed/precision compromise, the majority of visual SLAM algorithms are based on a sparse representation of the environment. By matching visual primitives on successive images, it is possible to estimate the 3D position of these primitives and the poses (position, orientation) of the cameras. Much progress, both in the speed and in the quality of the trajectory and of the 3D reconstruction, has been made over the past years. However, localization by visual SLAM still has multiple challenges, we present some of them in Figure 1.3.

As illustrated in Figure 1.3, appearance changes remain a crucial challenge for visual SLAM. Visual information affected by seasonal changes, lighting changes or

(a) Seasonal change ([Bai 2018]).

(b) Lighting change.

(c) Viewpoint angle change.

Figure 1.3: Examples of challenges for visual SLAM.

viewpoint changes can result in major difficulties when associating data between the current image and the landmarks in the map. Therefore, re-localization in previously mapped areas can be a hard task since the appearance of the environment is changing ceaselessly in outdoor scenarios and such a phenomenon can result in inaccurate or erroneous localization. In some applications like self-driving cars, the pose estimation part of the SLAM process is very delicate and must be taken very carefully because even a small error in estimating the pose of the vehicle can result in a dangerous accident. To ensure a secure and safe lifelong navigation in dynamic environments, autonomous vehicles must cope with such a challenge.

In previous work done at Institut Pascal [Royer 2016], Royer *et al.* employed a driverless shuttle for three months on an industrial site, totaling nearly 1500 km of autonomous travel. During this experience, we identified some long-term difficulties for autonomous navigation. Generally, SLAM algorithms retrieve the keyframes that are used for localization based on their geometric distance to the vehicle pose. However, this technique has shown a weakness in long-term localization because it does not take into account environmental changes.

Building a map that covers all environmental conditions by continuously adding landmarks to this map can help improving localization performance in changing conditions. However, this will result, also, in a ceaseless growth of the map's size that is relative to the number of traversals (experiences) performed by the vehicle. This means that after several traversals, localization will require an immense storage space for the map and high-end CPU to find matching points between the current image and the corresponding landmarks in a vast database. In other words, real-time long-term localization will be impossible after a certain number of localization sessions. Thus, a map update strategy is required to prevent such cases and to ensure reliable real-time long-term localization.

1.3 Contributions and organization of the manuscript

The work carried out during this thesis is structured in two main contributions: keyframes retrieval and map management. In the keyframes retrieval, we propose a localization approach able to take advantage of a visual landmark map composed of N sequences gathered at different times.

Generally, basic SLAM algorithms retrieve the keyframes that are used for localization based on their geometric distance to the vehicle pose. However, this technique has shown a weakness in long-term localization because it does not take into account environmental changes. This incents us to develop a new strategy for keyframe retrieval in order to adapt our SLAM algorithm [Lébraly 2011], [Royer 2016] for long-term operation. During the localization process, we aim to maximize the number of matched points (i.e., ameliorate the localization performance) by retrieving relevant experiences from a map that integrates numerous environmental conditions.

The second contribution of this thesis consists in a map management operation whose aim is to maintain a reliable map with a fixed size throughout the frequent runs of the vehicle. This reliable map must ensure long-term localization with no, or very few, localization errors. We have proposed three different map management techniques in this thesis:

1) We exploited scores of resemblance between the traversals computed by our keyframes retrieval approach to determine which traversal has the highest similarity to the others to remove it eventually.

2) We exploited information related to the sun's position to compute the traversals correlation in order to produce a map with a minimum number of traversals having diverse environmental conditions.

3) We proposed an improvement for the work of Mühlfellner *et al.* [Mühlfellner 2016]. In their work, Mühlfellner *et al.* are scoring landmarks according to the number of different localization sessions in which they appear and they are removing the landmarks with the lowest scores in an offline process. Similarly, we proposed to remove the landmarks with the lowest scores from the map while keeping a uniform number of landmarks on each traversal.

In Figure 1.4, we present a diagram explaining the operating mechanism of both the keyframes retrieval and the map management processes and the connection between them. Our system uses a multi-session map that incorporates multiple traversals recorded in different environmental conditions. Each time a new frame arrives, the keyframes retrieval algorithm must extract, from the map, data relevant to the current environmental condition. These retrieved data are then used to compute the vehicle pose. After the end of the localization session, our map management algorithm will be invoked offline to update the map by adding new data or removing superfluous information.

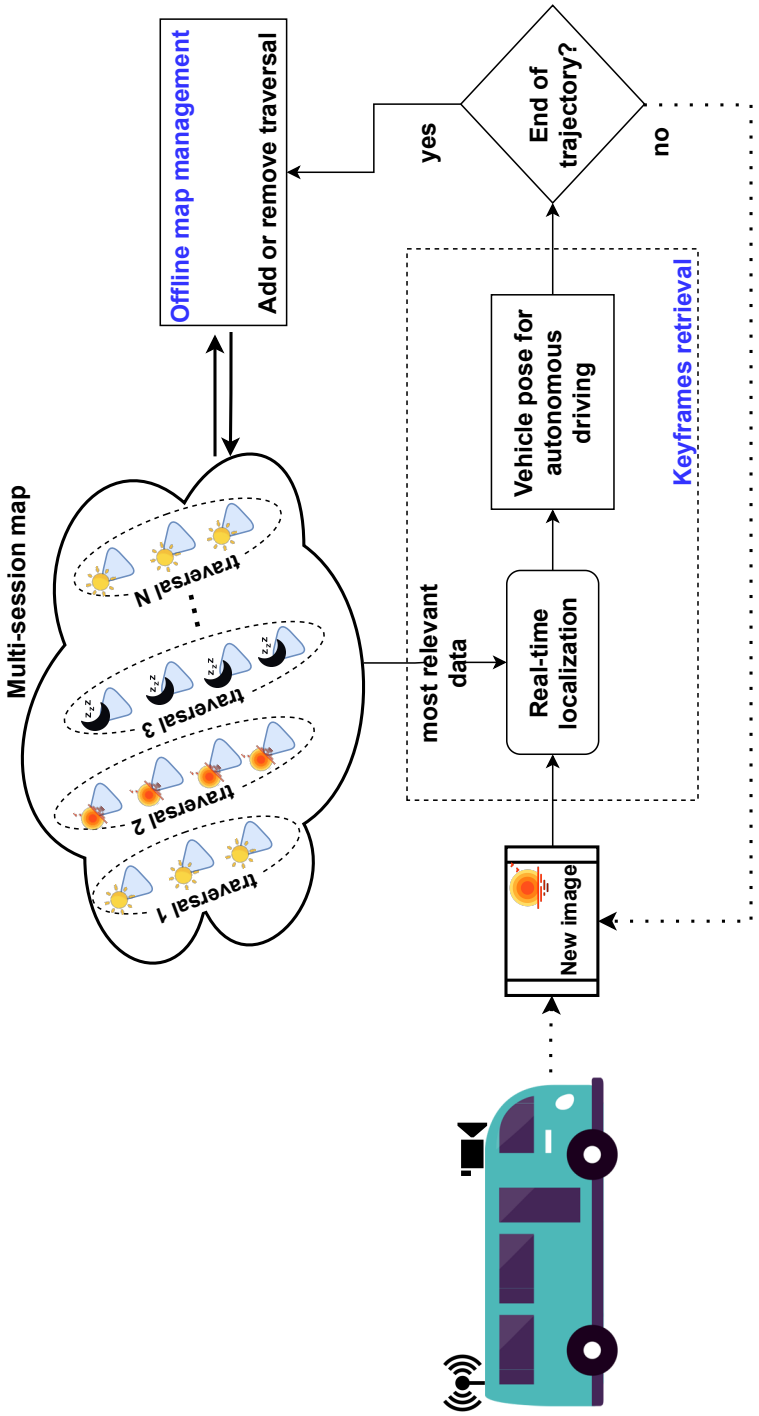


Figure 1.4: A diagram representing the operating mechanism of the keyframes retrieval and the map management proposed in this document.

This document is organized as follows. In Chapter 2, we present the state of the art in relation with our problematic. Some SLAM basics are introduced in Section 2.2. Section 2.3 presents the main works that led to the continuous improvement of the visual SLAM over the last few years. In Section 2.4, we present the state of the art in relation with our problematic, namely the lifelong navigation in dynamic environments. The three sub-sections incorporated in the Section 2.4 are highlighting:

1) some different types of feature descriptors that are employed to improve the robustness of visual SLAM in changing conditions.

2) some methods in relation with landmark retrieval.

3) some map management techniques to reduce the size of the map.

In Chapter 3, we propose a keyframes retrieval approach that takes advantage of statistics gathered in the first few meters of a traversal in a given location to compute a probabilistic ranking function. This ranking function is used in the rest of the traversal to retrieve from the map the most relevant keyframes, taking into account the current environmental conditions. In order to ensure our ranking function consistency, we keep updating it regularly throughout the trajectory.

Chapter 4 presents our proposed map management strategies. All the proposed strategies have the aim of reducing the size of the multi-session map without considerably degrading the localization performance. In the experimental part of this chapter, we compared the localization performance for each of these techniques with the state-of-the-art approach proposed by Mühlfellner *et al.* [Mühlfellner 2016].

Finally, Chapter 5 presents the conclusion and the perspectives of this research.

1.4 Publications

The work presented in this thesis has been the subject of the following publications:

- **Journals:**

  Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome: "Map management for robust long-term visual localization of an autonomous shuttle in changing conditions": Submitted to MTAP journal "Multimedia Tools and Applications" Special Issue on Machine Vision Theory and Applications for Cyber Physical Systems, 2021.(Impact factor 2.3 JCR). **On going**

- **International Conferences:**

  Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome: "Over two years of challenging environmental conditions for localization: The IPLT dataset". ICINCO 2021: "18th International Conference on Informatics in

Control, Automation and Robotics".

Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome: "Keyframes retrieval for robust long-term visual localization in changing conditions". SAMI 2021: IEEE 19th World Symposium on Applied Machine Intelligence and Informatics.

Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome: "Long-term localization with map compression based on solar information". Submitted to ICIRA 2021: "The 14th International Conference on Intelligent Robotics and Applications". **On going**

- **National Conferences:**

Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome: "Institut Pascal Long-Term dataset". RFIA 2020.

Chapter 2

State of the art

Contents

2.1 Introduction

The goal of this chapter is to present the state of the art and some basics related to our problem, with intense focus on SLAM solutions that solve the problem of long-term visual localization. We have divided this chapter into several sections. Section 2.2 presents some fundamentals on SLAM. In Section 2.3, we highlight the important works that has led to the continuous improvement of SLAM in recent years, with more focus on visual-based solutions. In Section 2.4, we present the state of the art in relation with our problematic, namely the lifelong navigation in dynamic environments. We classified the state-of-the-art contributions in this regard into three main categories, which are presented in sub-sections 2.4.1, 2.4.2 and 2.4.3. Finally, section 2.5 gives the conclusion of this chapter.

2.2 Fundamentals on SLAM

The Simultaneous Localization and Mapping (SLAM) problem addresses two important issues in mobile robotics. These two problems can be formulated into two questions, the first question is: *where am I?* The answer to this question defines the localization part of the robot. The second question concerns the characteristics of

the robot's environment: *what does the environment in which I operate look like?* The answer to this question defines the mapping part. Thus, the SLAM problem is formulated in Figure 2.1.

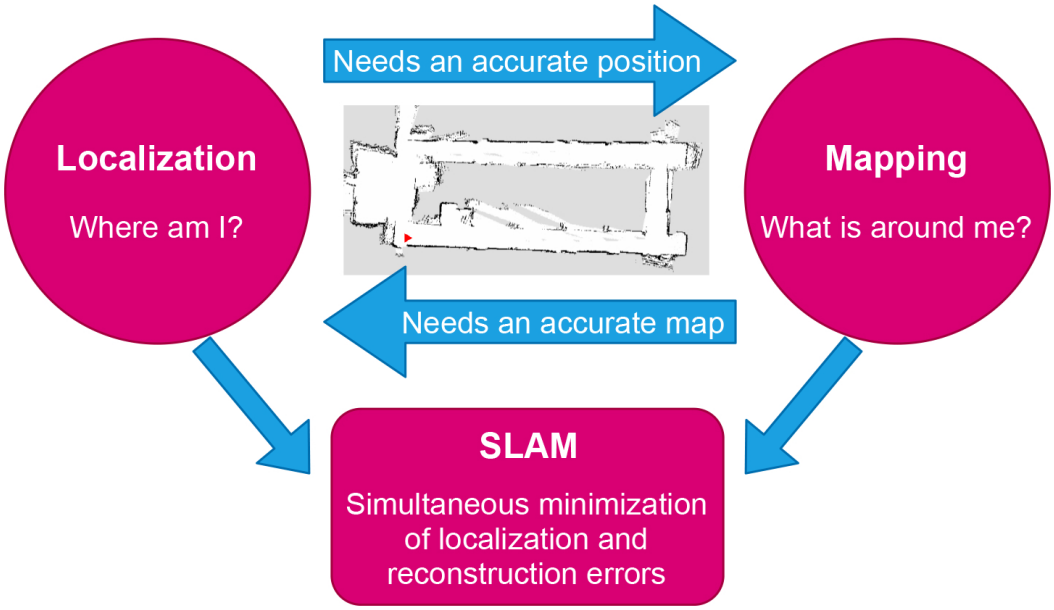


Figure 2.1: A diagram formulating the SLAM problem.

With a SLAM system, a robot placed in an unknown environment must create a map of the environment and simultaneously attempt to localize itself with respect to that map. This robotic system has several sensors that help it retrieve the information it needs. As formulated in Figure 2.1, this task appears to be a "chicken-and-egg" problem since the robot needs an accurate map to precisely localize itself, but at the same time it needs to know its exact location in order to be able to construct an accurate map. In order to make it easier to deal with this problem, the researchers formulated the previous two problems into one problem: *Where am I located in the most likely map of the environment that I have observed so far?* This formulation helped minimize error in both the localization and mapping tasks.

The state of the robot is described by its own position and orientation, although other secondary parameters may also be included in this description, such as its speed and the calibration parameters of its sensors. On the other hand, the map is a representation of distinctive primitives observed in the environment, such as points of interest (e.g., the position of landmarks or obstacles). This representation is used to describe the environment in which the robot operates. The mapping part helps to limit the error made in estimating the robot state. Without a map, the state of the robot would quickly drift over time. On the other hand, in the presence of a map, the robot can correct its accumulated drift by revisiting known areas (this process is called loop closure).

SLAM aims to provide a consistent representation of the environment based on measurements of motion and loop closure. Therefore, loop closure is a very

important component in this process. A mapping system that omits the loop closure step is in some ways similar to odometry. In early applications, odometry was obtained from wheel encoders. The position estimate obtained from odometer data drifts rapidly, which increases the localization error after a few meters. This was one of the main reasons for the development of SLAM. The observation of landmarks is useful for reducing the errors that are related to the odometry. A robot that performs odometry and neglects the loop closure interprets the mapping domain as an endless world in which it explores an infinite number of new domains. In Figure 2.2, we show an example of a mapping case where the SLAM made a significant drift that was corrected in the loop closure step.

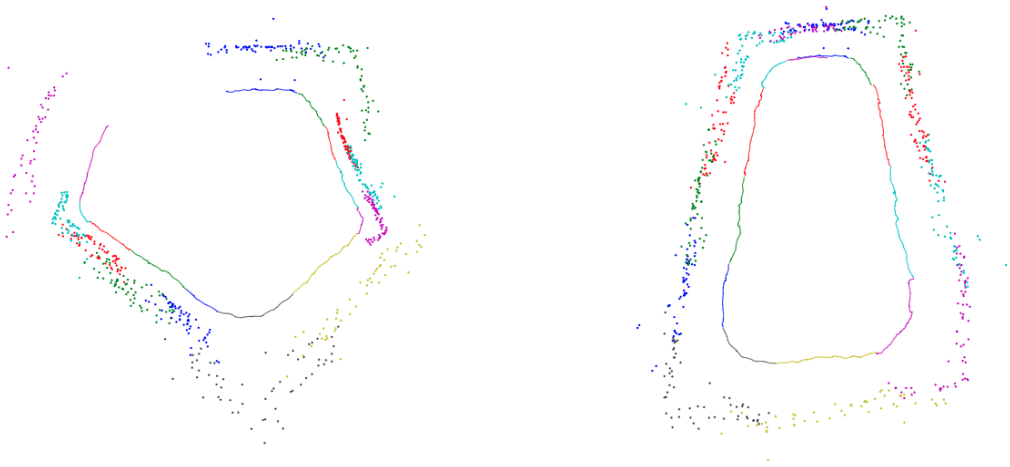


Figure 2.2: Pre and post loop closing [Bokovoy 2017]. Left: A raw map created with SLAM. The inner curve represents the trajectory of a mobile robotic system. The outer points represent the landmarks of the map. Right: The trajectory and the map were optimized with a loop closure algorithm.

Thus, loop closure is necessary in SLAM algorithms because many applications require reconstructing a consistent world-scale map. For example, in many exploration applications, the purpose of the robot is to explore an environment and return a consistent map to the human operator.

The classical architecture of a modern SLAM system consists of three main components: the sensor data acquisition part, the front-end part, and the back-end part of SLAM. Data can be collected from two types of sensors: proprioceptive and exteroceptive. The front-end part is summarized in mathematical models to interpret the sensor data, while the back-end part, which corresponds to the SLAM core, performs inference from the abstract data generated by the previous phase. The back-end process can also provide feedback to the front-end part for loop closure detection and verification. This architecture is summarized in Figure 2.3 and explained in the following sub-sections.

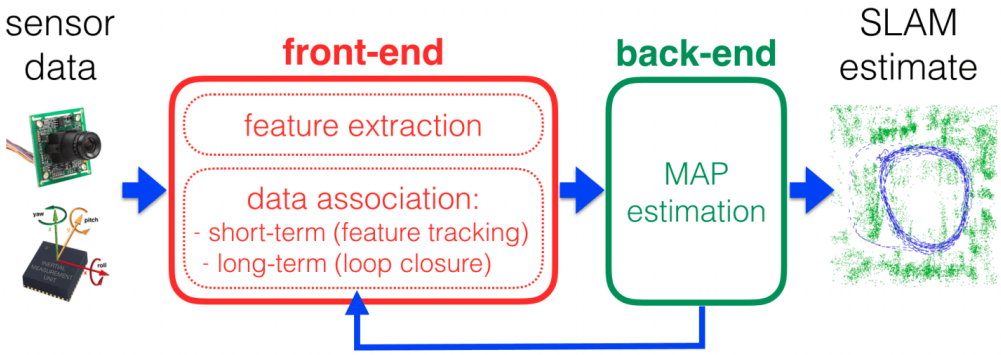


Figure 2.3: The architecture of a classic SLAM system [Cadena 2016].

2.2.1 Sensor data acquisition phase

A wide variety of sensors can be used in navigation. Sensors are traditionally divided into two categories depending on whether they measure the state of the robotic system itself (proprioceptive sensors) or the state of its environment (exteroceptive sensors).

2.2.1.1 Proprioceptive sensors

We can give some examples of proprioceptive sensors: inertial measurement unit (IMU), gyroscopes, odometers, counters, etc. These sensors provide measurements that correspond to variations in position, orientation and increments. An IMU provides information on acceleration, a gyroscope provides information on angular velocity, and an odometer provides information on the cumulative distance traveled. Information coming from these sensors are generally used with the mathematical motion model of the mobile robot to predict the followed trajectory. However, there is an intrinsic limitation to the use of this technique. In fact, measurement noise accumulates for each prediction computed according to the vehicle model, which ultimately leads to a highly noisy position estimate. In practice, position estimation based solely on this principle is impractical during long journeys (we can exclude certain very expensive inertial units that can provide very low measurement noise). Nevertheless, these sensors have some interesting characteristics that make them valuable tools for instantaneous measurements:

- a high sampling frequency.
- passivity (no emission of electromagnetic or other waves).
- independence from the external environment. For example, a gyroscope works equally well on land, in the air or under the sea.

2.2.1.2 Exteroceptive sensors

They can be classified into two categories. There are those that provide an absolute position in the terrestrial reference frame (GNSS) and those that provide a local position and/or orientation in a given coverage area.

GNSS GNSS stands for Global Navigation Satellite System. It is a set of components based on a constellation of artificial satellites that allow a user to be provided with their 3D position, 3D velocity and time via a small portable receiver. This category of global positioning systems is characterized by its metric precision, its global coverage and the compactness of the terminals, and also by its sensitivity to obstacles located between the receiving terminal and the satellites.

Examples of GNSS include Europe's Galileo, the USA's NAVSTAR Global Positioning System (GPS), Russia's Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) and China's BeiDou Navigation Satellite System.

The first satellite positioning system was developed by the United States with TRANSIT back in 1964 for military use only, then the Global Positioning System (GPS) has became operational since 1995 using a constellation of 24 satellites, describing the same orbit twice a day. Later, a few more satellites were added to this constellation (see Figure 2.4). Among this satellite constellation, between 5 and 8 satellites are always visible from any point on Earth [Arora 2016], allowing good positioning accuracy to be achieved through trilateration.

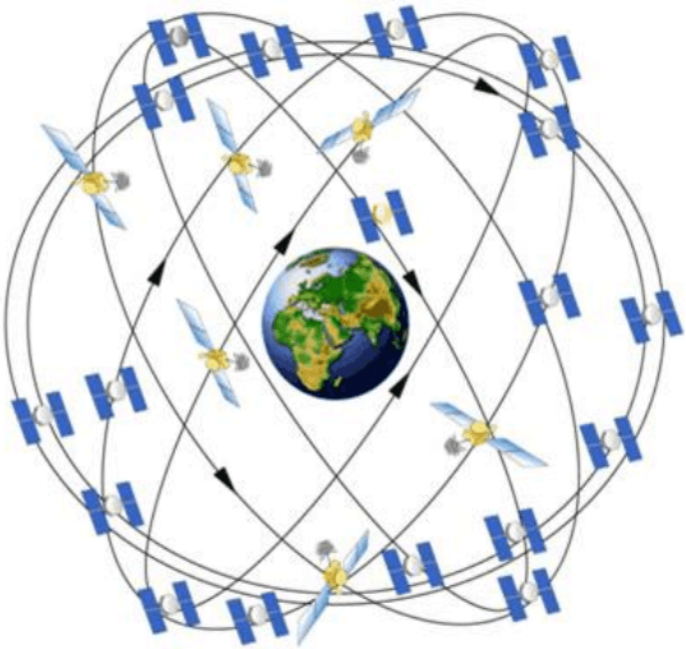


Figure 2.4: GPS constellation [Raghu 2014].

GPS suffers from multiple limitations, such as the coverage quality of some zones (e.g. when traveling in closed or covered areas such as tunnels, in very isolated geographical areas, etc.), the interference of information, the multiple reflections related to infrastructure and the presence of non-covered areas (e.g., in space exploration).

RADAR RADAR — RAdio Detection And Ranging — allows to obtain an estimate of the linear velocity of the mobile robot with respect to an object in the scene. It uses the principle of the Doppler effect to measure speed. It emits a continuous electromagnetic wave which is reflected by any target in the pointed direction. By Doppler effect, this reflected wave has a frequency slightly different from that emitted: higher frequency for mobile objects approaching the radar and lower for those moving away. By measuring the frequency difference between the emitted wave and the returned one, we can calculate the speed of the target.

LIDAR LIDAR — LIght Detection And Ranging — works on the same principle as RADAR, but uses light instead of electromagnetic waves.

Vision sensor (Camera) Through the image sequences that it provides, a camera can offer a lot of information on the environment crossed by a mobile robot. Despite the fact that the processing necessary to make use of the information provided by this sensor is usually time-consuming and complicated to set up, cameras have the advantage of being the only sensors that provide sensory information from the environment that is very similar to that perceived by humans.

When used alone, a camera provides two-dimensional information. However, by calculating the optical flow between multiple images, it is possible to measure the camera angle or its distance from the perceived landmarks [Alberto 2009], or even its speed. Several studies focused on using a single camera for localization [Mouragnon 2006, Eade 2006, Davison 2007]. However, in order to improve the results or make the treatments less heavy, two or more cameras are usually used. We talk about stereo-vision when two cameras are pointed in the same direction with an overlapping field of view (FoV).

2.2.2 Front-end

In robotic applications, it might be difficult to describe sensor measurements directly as an analytic function of the state, as required by the back-end (SLAM core). If the raw sensor data are images, it might be difficult to express the intensity of each pixel as a function of the state of the SLAM core. This problem arises because we are unable to conceive a sufficiently general representation of the environment. Even in the presence of such a representation, it would be difficult to write an analytic function that relates the measurements (observations) to the parameters of that representation. For this reason, before running the SLAM core, it is recommended to have a pre-processing module (front-end) whose task is to extract features from the raw sensor data.

Landmark extraction is an important pre-processing phase for SLAM algorithms. Landmark detectors detect a minimum number of landmarks from a large stream of raw data coming from an exteroceptive sensor. There are different types of landmarks depending on the exteroceptive sensor used. For lidars, landmarks are either lines, corners, or even local maxima corresponding to walls or objects. For a camera, landmarks are pixels extracted from either outlines, corners, static objects in the scene, etc. Visual landmark extraction algorithms are usually divided into two main processes:

- The detection process: this process involves the extraction of points from the space using the images that are captured from the environment. This extraction process must be invariant to changes in scale and viewpoint. In other words, a moving robotic system must be able to detect the same points even if they are perceived from different angles and distances.

- The description process: the task of this process is to describe the appearance of a detected point based on several visual information. A visual descriptor computed on a point must also be invariant to scale and viewpoint changes. Therefore, this process allows the recognition of a same point that is perceived from different scales and viewpoints, which is a common operating scenario for a moving robotic system.

Several visual-based landmark detection and extraction algorithms have been proposed and used in the context of visual SLAM. For example, the Scale Invariant Feature Transform (SIFT) [Lowe 2004] is proposed for detecting and describing salient key-points from images for object recognition applications. As indicated in the name of this descriptor, the features produced by this algorithm are supposed to be invariant to image scale and partially invariant to illumination changes and viewpoint changes. More recently, another feature description method, called the Speeded Up Robust Features (SURF), has been proposed by Bay *et al.* [Bay 2006]. SURF is computationally simpler and faster than SIFT [Sheena 2016], therefore, in many recent real-time visual SLAM applications as [Murillo 2007] and [Engelhard 2011], SURF was used in the pre-processing step for landmark extraction. Despite the fact that SIFT and SURF were designed to be able to match features with a certain level of invariance to illumination changes, some studies [Petry 2013, Diaz-Escobar 2018, Xiang 2020] demonstrated that both of them suffer from strong illumination changes.

Unlike SIFT and SURF, which are used for both detection and description processes, corner detectors such as Harris [Harris 1988] and Shi-Tomasi [Shi 1994] can be used for the landmark detection task in SLAM applications. For example, Davison *et al.* [Davison 2002] have proposed a monocular SLAM system that uses the Harris corner detector for feature detection and they have described landmarks using 15×15 patches centered on the points. Similarly, the SLAM system used in this thesis uses the Harris corner detector to detect landmarks on the images. These landmarks are then matched with

ZNCC — Zero-mean Normalized Cross-Correlation — computed on 11×11 pixel windows around each of them.

To sum up, in a vision-based SLAM system, the front-end part extracts the pixel location in the image of some distinguishable points in the environment. These pixel observations can now be easily modelled by the SLAM core. The front-end part is also responsible for the data association task, this means that its job is to match each measurement with a particular 3D point in the environment. In other words, the data association module tries to identify an observation z_k in a set of unknown variables $\mathcal{X}_k$ such that $z_k = h_k(\mathcal{X}_k)$. Finally, the front-end module can also provide an initial estimate of the variables. For example, in a monocular SLAM, the front-end part can take care of the initialization of landmarks by triangulation. The type of calculation performed in the front-end part strongly depends on the type of the used sensor since the notion of landmark changes from one sensor to another.

According to Figure 2.3, the data association module in the front-end can act in two modes: in short-term manner and in long-term one. The short-term data association means that the front-end module is responsible for associating corresponding features in consecutive sensor measurements. For example, in a monocular SLAM, the short-term data association module would take into account that two different pixel measurements in two consecutive frames may be referring to a same 3D point. After data association, it is common in visual SLAM algorithms to perform a nonlinear optimization called Bundle Adjustment (BA) [Triggs 1999] to minimize the reprojection error. On the other hand, long-term data association which refers to loop closure is in charge of associating new measurements to older landmarks. We notice in the figure that the back-end usually feeds back to the front-end about information in relation to loop closure detection and validation.

2.2.3 Back-end

The present hierarchy of SLAM was first introduced in the work of Lu and Milios [Lu 1997] followed by the work of Gutmann and Konolige [Gutmann 1999]. SLAM in general is formulated as a posterior estimation problem and can be represented by a graph to demonstrate the independence between variables. Supposing that we want to look for an estimate of a variable $\mathcal{X}$. In the context of SLAM, $\mathcal{X}$ is a variable that includes the trajectory of the robot (as a set of discrete points) and the positions of landmarks in the scene. Given a set of observations $Z = \{z_k; k = 1, \ldots, m\}$ such that each observation can be expressed as a function of $\mathcal{X}$: $z_k = h_k(\mathcal{X}_k) + \varepsilon_k$, where h_k is a known function (observation model) and ε_k is the measurement error. In the processing phase of the SLAM core, $\mathcal{X}$ is calculated to maximize the posterior $p(\mathcal{X}|Z)$. According to Bayes theorem:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\ p(\mathcal{X}|Z) = \underset{\mathcal{X}}{\operatorname{argmax}}\ p(Z|\mathcal{X})p(\mathcal{X}) \tag{2.1}$$

$p(Z|\mathcal{X})$ is the likelihood of the measurements Z given the variable $\mathcal{X}$, $p(\mathcal{X})$ represents the previous probability of $\mathcal{X}$ which includes any prior knowledge about

this variable. In the case of the lack of previous information, $p(\mathcal{X})$ becomes a uniform probabilistic distribution and can be excluded from the factorization.

Supposing that the measurements Z are independent, this means that the measurements noises are not correlated and therefore equation (2.1) can be factorized as follows:

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\ p(\mathcal{X}) \prod_{k=1}^{m} p(z_k|\mathcal{X}) = \underset{\mathcal{X}}{\operatorname{argmax}}\ p(\mathcal{X}) \prod_{k=1}^{m} p(z_k|\mathcal{X}_k) \tag{2.2}$$

This representation can be interpreted as a factor graph[1] [Kschischang 2001]. The variables correspond to the nodes of the graph, the two terms $p(z_k|\mathcal{X}_k)$ and $p(\mathcal{X})$ are called factors. These factors encode the probabilistic constraints throughout a sequence of nodes. The factorized graph is a graphical representation which encodes the dependencies between the factors and the corresponding variable $\mathcal{X}$.

A first advantage of the interpretation by factor graph is that it allows an insightful visualization of the problem. Figure 2.5 shows an interpretation of a simple SLAM problem by a factor graph representation. The figure shows the different variables in a SLAM system, including the position of the robot, the position of the landmarks and the factors that impose the constraints between the different variables. As a second advantage, the factorized graph model serves to give a general representation of such a complex problem integrating several heterogeneous factors and variables.

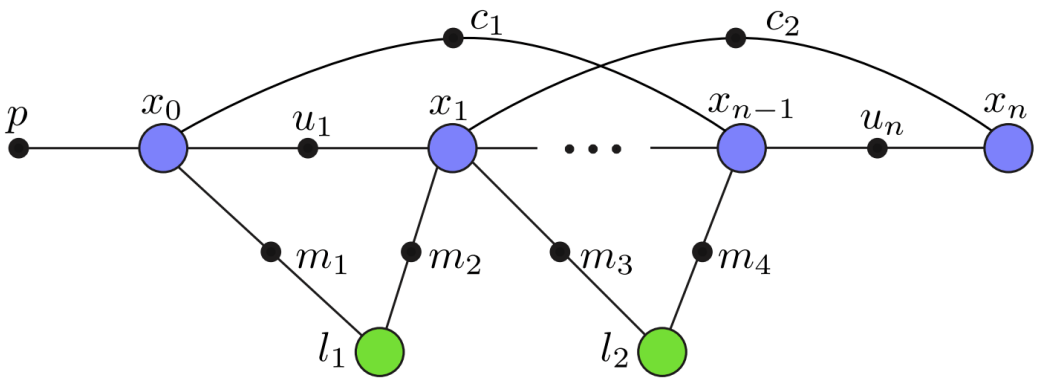


Figure 2.5: Factor graph formulation of the SLAM problem [Kaess 2012]. The colored circles represent the variable nodes (x_i nodes represent the poses of the cameras and l_i nodes represent the landmark positions), the small solid black circles represent the factor nodes (p denotes prior factors, u_i represent the odometry measurements, c_i represent a loop closure constraint and m_i represent landmark measures).

After finishing data association by the front-end part, the back-end part performs a bundle adjustment step to minimize the reprojection error. The BA seeks a

[1] A factor graph is a bipartite graph made up of two types of nodes: nodes representing variables and nodes representing factors. The variable nodes are connected to the factor nodes by edges in the graph, where each factor node is a function of all the variables to which it is connected.

maximum likelihood solution of the observations by using all the measurements over the entire evolution of the robot. The goal of BA is to minimize the error between the predicted observation and the image measurements of n 3D points observed from m sensor poses (Usually expressed with 6 DoF — Degrees of Freedom —). The measures and parameters to be estimated are considered to be normally distributed and the problem is often solved with nonlinear least squares minimization techniques such as the Gauss-Newton method.

In a usual back-end part (SLAM core) of a SLAM system, we can distinguish 3 basic modules: Initialization, Tracking and Mapping:

- Initialization: To start a SLAM session, it is necessary to define the coordinate system for the vehicle estimate pose and the 3D reconstruction. This means that the first step in this initialization phase is to determine the global coordinate system, then, to reconstruct a part of the environment as an initial map with respect to this determined global coordinate system. After a successful initialization, the tracking and mapping are performed to continuously estimate the vehicle poses in the reconstructed initial map.

- Tracking: In this phase, the SLAM takes into account the initial vehicle pose x_{i-1} and the reconstructed initial map to track features in the recorded images to estimate the camera pose x_i with respect to the map. To accomplish this task, it is crucial for visual SLAM algorithms to use cameras with precalibrated extrinsic parameters. However, if the extrinsic parameters are not accurate, SLAM systems tends to fail to track the camera movements or large errors can be observed on the map. Conversely, using well calibrated camera parameters, the vehicle pose in the global coordinate system can be easily computed after tracking the camera pose.

- Mapping: the initial map is expanded by estimating the 3D structure of the environment where the camera observes. Adding all tracked features to the map leads to a considerable redundancy in the mapped landmarks which results in a quick expand in the size of the map. Therefore, modern SLAM algorithms are using the notion of "keyframes" [Klein 2007, Mur-Artal 2015]. This means that a frame is added to the map as a keyframe only if its pose is, at some level, different from the poses of already existing keyframes in the map. This keyframe representation allows to avoid mapping redundant information from successive frames, and helps to handle incremental drift by readjusting a more compact set of poses in real time using classical loop closure constraints [Meilland 2013]. During mapping, if a 2D point measurement is matched with a 3D point landmark, then the set of descriptors of the landmark is enlarged by adding the descriptor of the 2D point measurement.

In addition to the three modules mentioned above, the back-end part usually includes two additional modules depending on the application goals: re-initialization and global map optimization. For example in a visual SLAM system, the re-

initialization process is required when tracking fails due to a rapid camera movement or some other issues related to the robot or the environment. In such a case, the robot has to re-estimate the position of the camera in the mapped scene. Therefore, this process is called re-initialization. In a SLAM application where no re-initialization module is incorporated, the system will stop working after a single tracking failure. Such a system is not useful in most of autonomous driving applications.

The second additional module is the global map optimization. SLAM mapped scenes usually include cumulative estimate errors on the camera movement and consequently on the mapped landmarks. A global map optimization is required in such a case to reduce the error. The global map optimization process makes sure that the map is refined taking into account the consistency of all the mapped information. When a previously mapped region is revisited after some camera movement, the cumulative error from the old position to the current position can be calculated. After that, a loop closure constraint from these information is used to perform a whole map optimization. In the loop closure process, the system seeks for a loop by matching the current frame with the previously mapped keyframes. If the loop is validated, it means that the robot has returned to one of the already observed scenes. In this case, the cumulative error that occurred while the robot was moving in this loop can be estimated. We can note that the loop detection phase can be performed using some techniques that are similar to those of the re-initialization module. Basically, re-initialization is performed to recover only one robot pose in the map, while loop detection is performed to obtain a geometrically consistent map.

2.3 History of visual SLAM

The first works on SLAM in the robotics community are attributed to Smith and Cheeseman [Smith 1986]. In this work, the authors have used a Kalman filter to solve the localization and mapping problem. They have proposed to estimate the uncertainty and the correlation between the position of the sensor and the structure of the environment. Subsequently, a significant number of solutions were proposed to achieve real-time SLAM. For instance, EKF (extended Kalman filter) approaches [Newman 1999, Davison 2003] have become widely used solutions for real-time SLAM applications.

EKF-based approaches are characterized by a state vector composed of the landmarks locations. Uncertainty is represented by a probabilistic density function. The EKF is particularly sensitive to bad associations, an incorrect measurement can lead to the discrepancy of the entire filter. Moreover, the complexity of the EKF is quadratic based on the number of landmarks on the map. Its use is therefore difficult in large-scale environments. These limitations have led a large number of studies to move towards more scalable SLAM algorithms adapted to large-scale environments, in particular the works on local filtering of the map [Williams 2001, Tardós 2002], and the works on compressed filters [Guivant 2001, Folkesson 2003, Yoon 2006].

Monte-Carlo methods [Dellaert 1999, Fox 1999, Thrun 2001] were proposed as a solution to the localization problem. Compared to EKF methods, Monte-Carlo has brought much more scalable means to the problem of state representation. In this scope, some other approaches based on particle filters were also proposed to provide a solution to the scalability issues associated with EKF-based SLAM systems. One of the most famous approaches based on particle filters is the FastSLAM proposed by Montemerlo *et al.* [Montemerlo 2002]. The main idea in their system is the Rao-Blackwellized representation of the posterior. This design allowed to represent the posterior distribution as a particle filter on the robot's path, while allocating an independent Kalman filter to each primitive associated with each particle. The FastSlam algorithm can be summarized in three main steps:

1) Sampling: the proposal distribution is computed for each particle to sample a pose.

2) Importance Weighting: an importance weight is assigned to each particle k according to:

$$w_t^{[k]} = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{2.3}$$

3) Re-sampling: low importance weighted particles will be rejected and replaced by samples with higher rates.

The complexity of this algorithm is logarithmic, $O(M \log(K))$, where M is the number of particles used and K is the number of landmarks in the map. The major problem with this approach is that there is no way to determine the number of particles needed to accurately represent the state of the system. In other words, using multiple particles require a significant memory occupation and computing time, and using a few particles leads to inaccurate results. Although particle filter-based methods have proven to be successful for several applications, Julier and Uhlmann [Julier 2001] have shown that over very long trajectories, the state estimates provided by filter-based methods are inaccurate.

An alternative to particle filter-based methods are the pose graph methods. This type of approach has become very popular in the recent years, showing mapping capabilities at very large scales compared to filter-based methods. In the pose graph SLAM methods, the problem is represented as a set of nodes and edges, where the nodes represent the poses of the sensors and landmarks, and the edges represent the constraints that determine the spatial relationship between these poses. Once the graph has been reconstructed, the problem comes down to finding the configuration of the nodes which ensures maximum consistency with the constraints.

The formulation of the problem of SLAM by pose graph was initially introduced by Lu and Milios [Lu 1997]. The pose graph formulation of SLAM has taken several years to become popular due to the relatively high complexity of solving the error minimization problem. Advances in linear algebra and improvements in optimization methods have allowed this formulation to become one of the most widely used.

Dellaert and Kaess [Dellaert 2006] have introduced the square root smoothing and mapping (SAM), a graph-based visual SLAM technique that leverages the problem's sparse structure to calculate a solution. Thrun and Montemerlo [Thrun 2006] have proposed an offline pose graph SLAM solution whose goal is to produce a lower-dimensional problem using variable elimination techniques to reduce the graph. Subsequently, a large number of works on SLAM solutions based on pose graphs were published [Olson 2006, Grisetti 2007, Konolige 2010, Kümmerle 2011]. One of the most important qualities of graph-based approaches over filter-based approaches is the fact that the optimization, to solve the SLAM problem, keeps all the information about all the poses that have been recorded unlike filter-based approaches which give less importance to previous poses.

Kaess *et al.* [Kaess 2008] presented the iSAM system which is based on their previous work (SAM [Dellaert 2006]). The iSAM framework is an online solution for visual SLAM which performs smoothing using QR factorization of the square root information matrix. Later, Kaess *et al.* [Kaess 2012] proposed the iSAM2 which is formulated in the form of a factor graph as in Figure 2.5. The major contribution of iSAM2 is that it offers an efficient and precise method for the incremental update of the factorization of the system by recalculating only the elements of the matrix which have been changed. This means that it is no longer necessary to solve the entire system each time a pose is added to the graph.

A similar technique to visual SLAM called Structure from Motion (SfM) was firstly introduced in [Ullman 1979] and has seen tremendous evolution over the years [Tomasi 1992, Schaffalitzky 2002, Pollefeys 2004, Snavely 2006, Pollefeys 2008]. It has its origins in the fields of photogrammetry and computer vision. The SfM technique calculates the 3D structure of the camera position and the scene from a set of images. The main difference between SLAM and SfM is that SLAM requires a linear and ordered set of data to be processed in real-time; however, image-based SfM can be applied to an unordered set of data where it is also possible to input all images at once for processing. The standard procedure of SfM is to extract landmarks from the images, identify them through the matching process, and perform a nonlinear optimization (Bundle Adjustment) to minimize reprojection error.

Nistér *et al.* [Nistér 2004] introduced the Visual Odometry (VO) technique. In VO, there is no notion of long term memory, it consists in simultaneously determining the position of the camera for each received image and the position of landmarks in 3D, incrementally and in real-time. Mouragnon *et al.* [Mouragnon 2006] proposed a variant to the visual odometry approach proposed by Nistér *et al.* by adding a local bundle adjustment technique.

Klein and Murray [Klein 2007] proposed a monocular system called Parallel Tracking and Mapping (PTaM). PTaM is an image-based approach with two parallel threads as illustrated in Figure 2.6. The first thread performs a tracking and landmarks detection task, while the other task produces an optimized 3D map with BA. PTaM is a widely used framework for localization and augmented reality in small workspaces.

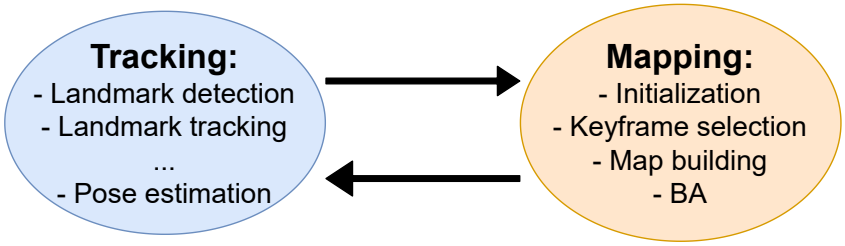


Figure 2.6: Overview of the main components of PTaM [Klein 2007].

More recently, a monocular SLAM approach with interesting properties was proposed under the name of ORB-SLAM [Mur-Artal 2015]. ORB-SLAM has been widely used for localization and mapping, with impressive mapping results. As shown in Figure 2.7, ORB-SLAM is based on three processing threads: the tracking thread, the local mapping thread and the loop closing thread. This architecture allowed ORB-SLAM to work in real time, in small/large, indoor/outdoor environments. The system generates a compact and traceable map. In addition, it offers the means to ensure the loop closure and the re-localization based on indexing approaches. However, despite the algorithmic quality of ORB-SLAM and its robustness in different navigation environment scenarios, it still suffers from time consuming and computational complexity. Moreover, the efficiency of this framework drops considerably in dynamic environments with moving objects and changing environmental conditions [Cui 2019].

In addition to these developments in visual SLAM which are mainly based on the detection and tracking of visual primitives (landmarks) which can be called sparse approaches, new so-called direct approaches or dense approaches have been adopted. These methods do not use visual primitives (featureless approaches), they rely on texture tracking. Engel *et al.* [Engel 2014] proposed LSD-SLAM which stands for Large-Scale Direct SLAM and it consists in estimating the pose of a monocular camera by image alignment. LSD-SLAM incorporates a multi-scale approach combined with a technique of estimation and statistic filtering of semi-dense depth maps where only the points with a high gradient value are taken into account. A global optimization is performed in a pose graph composed of keyframes as nodes and the relative transformations between these images as edges. LSD-SLAM has been extended to make it compatible with stereo cameras instead of only monocular cameras. In the stereo LSD-SLAM version proposed by Engel *et al.* [Engel 2015], several advantages can be noted: the initialization is instantaneous, the scale is fixed automatically and large rotations are better managed. In addition, stereo LSD-SLAM includes automatic exposure correction which makes the error function less sensitive to illumination changes.

In this thesis, we have used a keyframe-based SLAM framework [Lébraly 2011, Royer 2016] that is very similar to some widely used frameworks such as ORB-SLAM [Mur-Artal 2015]. The mapping phase of our system starts with interest points detection, matching and bundle adjustment. We extract a keyframe from the

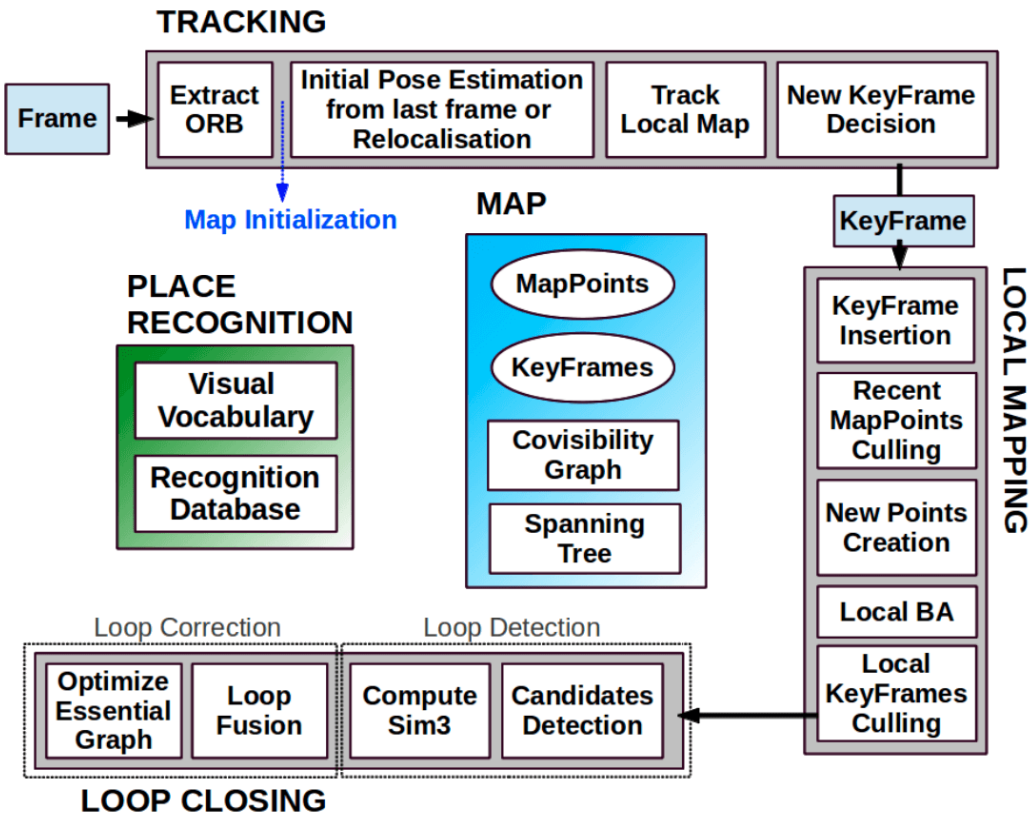


Figure 2.7: Overview of the ORB-SLAM system [Mur-Artal 2015].

video flow every $\sim$ 1 meter, and for each keyframe, we compute its 6DoF pose and the set of its related 3D points through triangulation. Therefore, a keyframe is added to the map by storing its extracted features and its corresponding 6DoF pose (similarly, a keyframe is retrieved from the map by retrieving its corresponding features and 6DoF pose). All the stored features will be linked with their corresponding 3D points in the map. To make sure that we build an experience-based map, all keyframes belonging to a same run are grouped in a same structure called *traversal*. This will make easier accessing to the features recorded in a particular run (benefits of experience-based mapping). Moreover, our mapping system also involves an online loop-closure operation which detects and closes loops in the map. In the localization phase, our system consists, in a first step, in predicting the current pose using the last computed pose and the wheel odometry input. Afterwards, the most relevant keyframe according to the predicted pose will be retrieved from the map to perform a 2D/3D matching with the detected interest points from the current input image. Finally, we use RANSAC and PnP (Perspective-n-Point) to compute the current pose of the vehicle which will be optimized in the last step. A diagram illustrating the localization and mapping process of our SLAM system is presented in Figure 2.8.

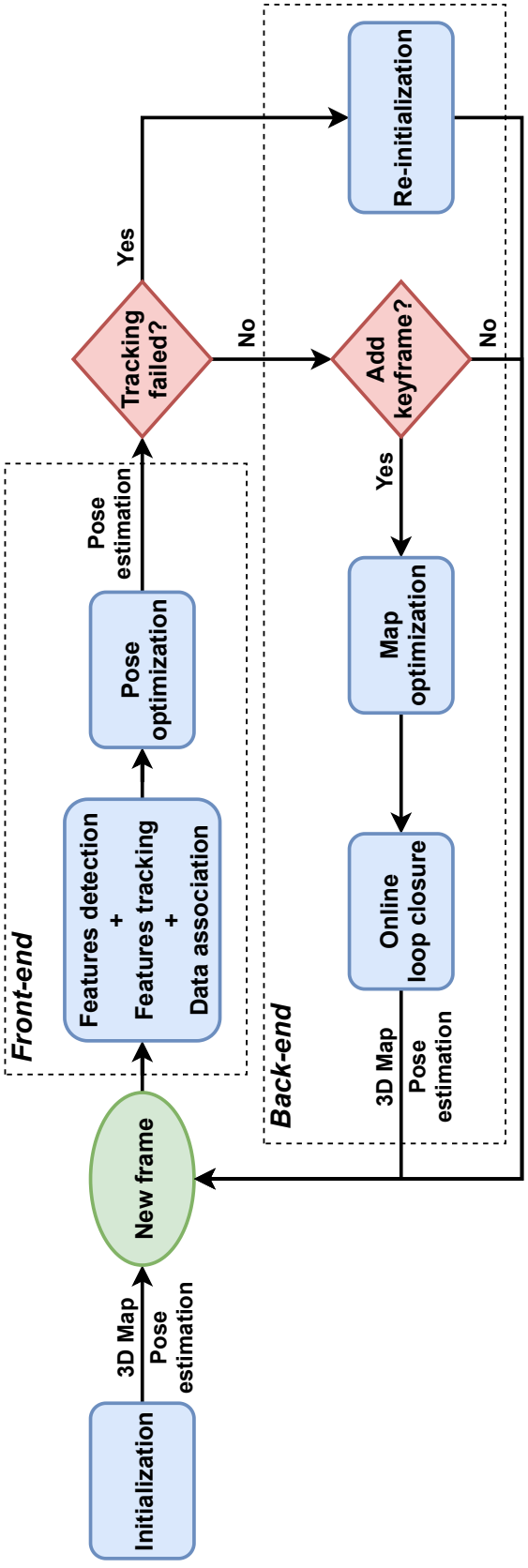


Figure 2.8: A flowchart diagram illustrating the operating process of the SLAM system used in this thesis.

In Figure 2.9, we present an example of a 3D map generated using our SLAM system. This map was created using a $\sim$ 200 meters long sequence from our own dataset [Bouaziz 2020] which was recorded in a parking lot in the Clermont Auvergne University campus.

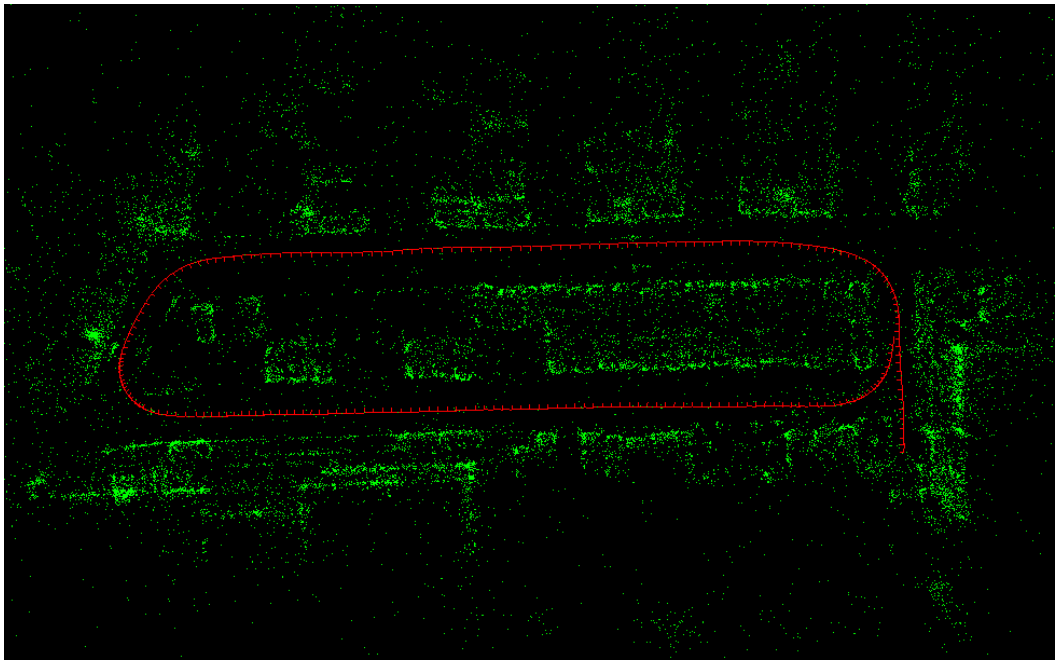


Figure 2.9: An aerial view of a 3D map generated with our SLAM system. The small red axes denote the 3D poses of the registered keyframes while the green points represent the mapped landmarks.

2.4 Lifelong navigation

In most real-world robotic scenarios, robots should be able to long-termly operate in dynamic and daily changing environments, and SLAM should be one of their most fundamental capabilities. However, most existing SLAM frameworks are evaluated in static environments or in scenes with only a few dynamic objects, such as moving people. It is only recently that attempts have been made to extend localization performance in dynamic environments.

A fundamental problem in long-term visual navigation is the presence of natural scene changes due to lighting, seasonal, and weather variations. Light Detection And Ranging (LiDAR) systems can overcome this limitation, but such sensors are still relatively expensive and require large payload capacities that are impractical in small robotic systems with limited mass. Although Radar systems are also promising in this regard [Cen 2018], they are still more expensive than vision-based systems.

In vision-based SLAM, traditional feature-based comparison techniques are deemed to be unsuitable for long-term operations due to their weakness to changing

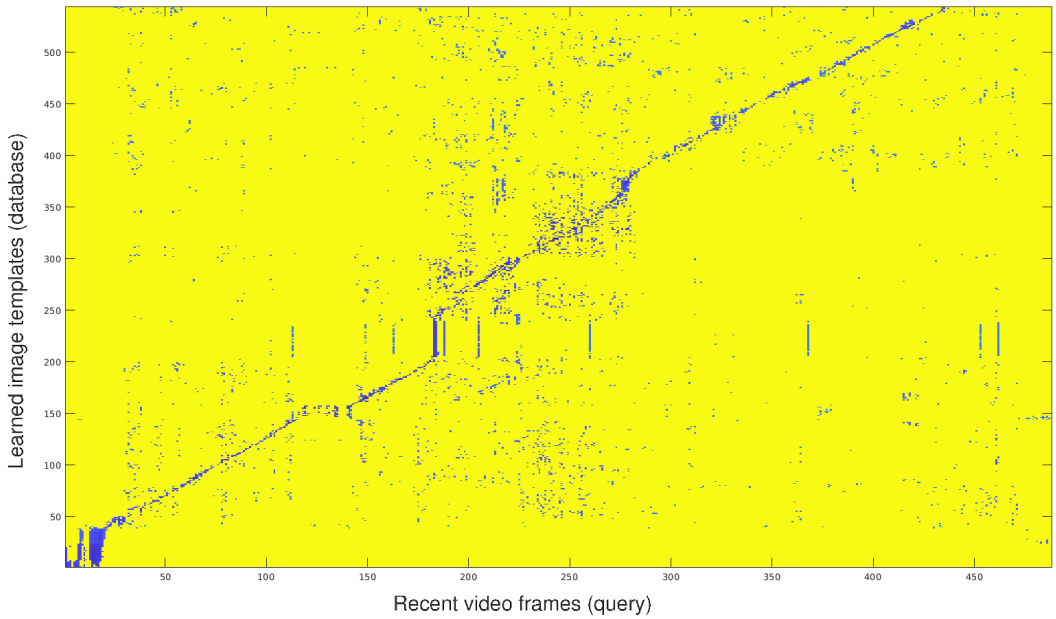
conditions. On this basis, an image-based approach was proposed by Murillo and Kosecka [Murillo 2009] to improve localization in dynamic environments using the Gist representation [Oliva 2001] of panoramic images. Unlike local features descriptors such as SIFT and SURF, Gist is a global descriptor calculated with the entire image. It corresponds to an abstract representation of the scene. In this approach, recognizing a place requires a deep search in the database to find the corresponding image, which is costly in large-scale environments. Milford and Wyeth [Milford 2012] proposed to enhance the performance of global image descriptors for places recognition by matching sequences (SeqSLAM) of images instead of unique images and they achieved impressive results on various seasonal datasets. SeqSLAM is based on calculating frame-to-frame dissimilarity scores between all query and database images and storing them in a so-called difference matrix, then computing a straight line path through the difference matrix to finally select the path with the smallest sum of the dissimilarity scores. An example of matched frames with SeqSLAM and the corresponding difference matrix is presented in Figure 2.10.

Although SeqSLAM has shown excellent performance in some situations, it remains sensitive to viewpoint changes. Pepperell *et al.* [Pepperell 2016] extended the classic SeqSLAM structure that uses linear image databases. They integrated an oriented graph structure to represent the roads. They also used panoramic images to minimize the variance of viewpoints. Despite all these improvements, the global image methods still remain sensitive to significant variations in viewpoint. Moreover, these methods can only be used to find a sequence-to-sequence correspondence between the query and the database datasets (as in Figure 2.10), this means that they cannot provide a 6DoF estimation of the vehicle pose.

While single-session SLAM has been widely addressed [Bosse 2004, Thrun 2006, Kaess 2008, Kaess 2012], multi-session mapping deals with persistence issue. This means that the robotic system must be able to remember multiple representations of the environment in which it operates. A multi-session map allows a robot to operate robustly over long periods of time. In a long-term scenario where a mobile robotic system repeatedly travels through previously visited areas, the robot cannot simply treat each mission as an entirely new experience that does not involve previously created maps. However, the robot also cannot treat its entire lifetime experiences as one large mission where all data is represented in a single pose graph and processed in a single batch optimization. Therefore, multi-session mapping implies that when the mobile robot starts a new experience, it tries to localize itself in a previously created map. This solution has the advantage of always using the same reference map. It also ensures that only one multi-session map is created through the multiple runs, instead of several independent ones.

In order to build a multi-session map, Churchill and Newman [Churchill 2012] proposed an approach in which a place can have different appearances. They developed a mapping system based on a "plastic" map representation (a compromise between adapting to new models and preserving old models). This map is represented by two structures:

(a) Example of query and database matching frames from University of Bonn dataset [Vysotska 2015].



(b) Difference matrix. Blue lines correspond to a match between a sequence of query frames with a sequence of database frames.

Figure 2.10: Example of a result obtained with SeqSLAM.

- *Short-Term Memory*: reacts quickly to changes, uses a high update rate (high plasticity).
- *Long-Term Memory*: Should not react to temporary variations, uses low update rate (low plasticity).

This structure allows to memorize different experiences for each place rather than trying to match different appearances between seasons and/or brightness changes. They have also proposed a similar approach in [Churchill 2013] in which they added new experiences whenever a localization failure occurred in some mapped area. However, in both [Churchill 2012, Churchill 2013], the proposed representations generate a direct dependence between the size of the map and the variations in the scene, which requires to match the query image with all experiences to find the best match.

Most of recent attempts that were made to improve localization performance in changing environments and to ensure a reliable lifelong navigation in dynamic environments can be categorized into one of the three following categories:

- Invariant features descriptors.
- Landmarks retrieval techniques.
- Map management techniques.

Therefore, in sub-section 2.4.1 we present some recent invariant features descriptors that are proposed to improve matching accuracy against illumination changes. In sub-section 2.4.2, we review some landmark retrieval techniques from the state of the art. And in sub-section 2.4.3 we provide a description of map management and we present some related state-of-the-art approaches.

2.4.1 Illumination invariant features descriptors for lifelong navigation

Local image descriptors can be applied in many areas in computer vision: pattern recognition, objects detection and tracking, 3D reconstruction of scenes, camera calibration, SfM, SLAM, etc. This type of approach is based on the pairing (or mapping) of points of interest characterized by a local descriptor. Depending on the application, certain invariance is necessary. In some contexts such as SfM and SLAM, it is essential to be invariant, at a certain level, to lighting transformations.

In the work of Maddern *et al.* [Maddern 2014], a lighting invariant approach is proposed to improve visual localization in different hours of the day. In this approach, the authors propose to compute an illumination invariant color space by converting an RGB image with three channels (I_R, I_G and I_B) into a corresponding illumination invariant image $I_{invariant}$, as follows:

$$I_{invariant} = 0.5 + \log(I_G) - \alpha \log(I_B) - (1 - \alpha) \log(I_R) \tag{2.4}$$

The optimal value of α was chosen with an exhaustive search. The resulting image $I_{invariant}$ has a less variance to sunlight and shadows than the initial RGB image as shown in Figure 2.11. However, this approach requires using a color camera since the equation (2.4) requires an RGB image as input. Moreover, the proposed illumination invariant color space is only able to reduce the appearance change due to sunlight, meaning that no other source of appearance change, including seasonal or day-to-night changes, can be tackled by this approach.

Pascoe *et al.* [Pascoe 2017] proposed a SLAM framework called NID-SLAM. This framework incorporates a whole-image metric, named Normalized Information Distance (NID), to estimate the camera motion. Unlike photometric error, the NID metric is not a function of the intensities of an image but rather a function of their entropies; therefore, images collected under very different lighting, weather and

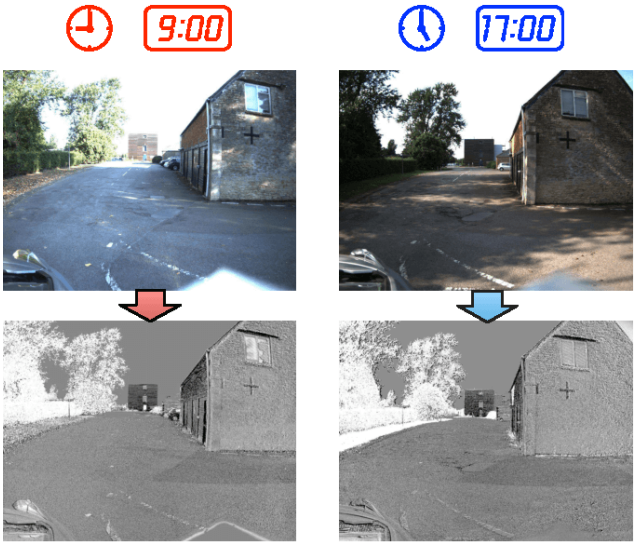


Figure 2.11: Improving image similarity at two different times of the day [Maddern 2014]. The RGB color model is converted to the illumination invariant color space using Equation (2.4).

seasonal conditions can be located relative to a common map and used to update depth maps despite changes in appearance.

In the same context, Diaz-Escobar *et al.* [Diaz-Escobar 2018] proposed a luminance invariant descriptor called LUIFT and stands for LUminance Invariant Feature Transform. As indicated in the name of this descriptor, LUIFT is used to extract the most significant local features in images degraded by nonuniform illumination, geometric distortions, and heavy scene noise. LUIFT utilizes image phase information rather than intensity variations to gain more robustness to nonuniform illuminations. The feature detector part of LUIFT is constructed using a modified Harris corner detector, while the feature descriptor part is constructed using a modified HOG-based method (Histogram of Oriented Gradients).

Many recent works replaced the descriptors generation methods with the neural networks like DeepDesc [Simo-Serra 2015], Matchnet [Han 2015], PN-Net [Balntas 2016], L2-Net [Tian 2017], LF-Net [Ono 2018], Superpoint [DeTone 2018], etc. For example, DeepDesc uses CNN — Convolutional Neural Networks — to learn the representation of different patches in the image. This means that DeepDesc is no longer based on hand-crafted features, but it is trained based on the correlations of the different patches. PN-NET proposes a triplet deep network architecture, in which, triplets of image patches are presented to the network, all these triplets are composed from two positive (matching) and one negative (non-matching) patches. The network then tries to learn a mapping that minimizes the distance between the two positives and maximizes the two distances between the positive and negative samples. Most of these descriptors are learned to build a good scale and viewpoint invariance while not paying much

attention to the illumination invariance.

Some other works like TILDE [Verdie 2015], LIFT [Yi 2016a], D2-Net [Dusmanu 2019] and SOSNet [Tian 2019] have paid more attention to illumination invariance. For instance, TILDE has introduced a learning-based method for feature point detection by training a regressor through supervised learning to work normally even if the lighting changes dramatically. The process consists firstly in identifying good keypoint candidates in multiple training images and secondly in training the regressor to predict a score map whose maxima are these points.

Unlike TILDE, which only performs feature detection, LIFT [Yi 2016a] — Learned Invariant Features Transform — is a novel architecture that can perform detection, orientation estimation, and description at the same time. As presented in Figure 2.12, LIFT combines a keypoint detector that is similar to TILDE [Verdie 2015], an orientation estimator like the one proposed in [Yi 2016b] and a similar descriptor to DeepDesc [Simo-Serra 2015] into a single unified pipeline and uses ground-truth that is generated using a SIFT-based SfM. The training process introduces the inverse training, which can minimize the influence of illumination on feature point detection.

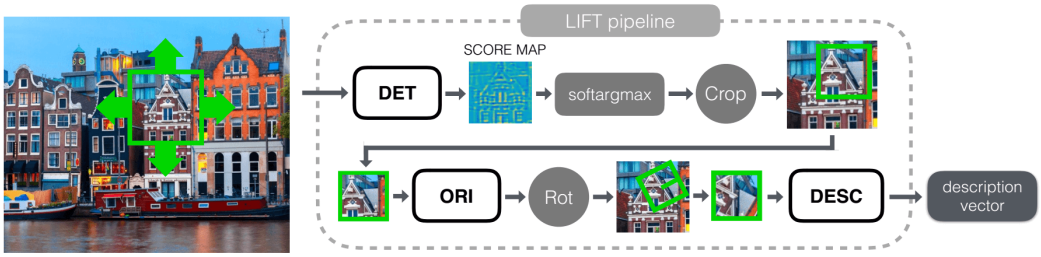


Figure 2.12: The feature extraction pipeline proposed in LIFT [Yi 2016a].

D2-Net [Dusmanu 2019] proposed an alternative formulation that solves the feature detection and description problem jointly by using a single network that plays a detect-and-describe role: dense feature description and feature detection are achieved simultaneously. The proposed network outputs a pixel-wise likelihood feature map and a joint descriptor. However, this method has difficulty in detecting accurate keypoints learned from low-resolution feature maps.

In order to learn more robust descriptors by minimizing the edge similarity between matching descriptors, SOSNet [Tian 2019] incorporated a second order similarity regularization R_{SOS}, that is usually used for graph matching and clustering tasks, and a triplet margin loss as part of their loss function to supervise the learning of real-valued descriptor. They have also designed a local descriptor evaluation method based on von Mises-Fischer distribution.

All these works can help improve localization against changes in environmental conditions, and can be combined with other map management approaches to further improve the robustness of localization under such conditions.

2.4.2 Landmarks retrieval

The problem of visual landmark retrieval/selection has been widely investigated in the scientific literature. Sutherland and Thompson [Sutherland 1993] were among the first researchers to study this problem. They demonstrated the existence of a strong dependence between the localization error and the configuration of the selected landmarks. In this regard, intensive effort has been devoted with the aim of selecting the optimal set of landmarks in order to reduce localization errors. For instance, Sinriech and Shoval [Sinriech 2000] specified a set of constraints on the number of landmarks and their distance from critical points in the environment, and formulated the positioning of landmarks as a nonlinear optimization problem. Sala *et al.* [Sala 2006] proposed a region-based decomposition of the environment while they made sure that at least k landmarks are observable on each region. They have used a graph-theoretical formulation to find such a decomposition that incorporates the minimum number of regions. This formulation allowed them to design a selection criteria that takes into account the region from which each landmark is visible to select a small set of landmarks that can be used for localization in a variety of regions. Similarly, Zhang *et al.* [Zhang 2005] proposed an entropy-based landmark selection method for SLAM. This method specifies a measure about which visible landmark is best in the sense of entropy reduction.

However, these works are addressing localization speed by reducing the set of selected landmarks. Moreover, they rely on pure geometrical reasoning for the selection. Some other recent works have addressed landmark selection to improve long-term localization. For instance, Mühlfellner *et al.* [Mühlfellner 2016] proposed a landmark selection approach, in which, a score is assigned to each landmark based on the number of times it has been observed. Landmarks with high scores are considered valuable for localization and consequently retrieved in the localization phase.

The works of Linegar *et al.* [Linegar 2015] were devoted to reduce computational costs of their previous work [Churchill 2013]. To do so, they exploited information about each past keyframe-to-keyframe localization attempt to recall and retrieve successful localization experiences under the current environmental condition. This approach is computationally inexpensive because it does not directly take into account the appearance information. However, it cannot effectively select future experiences unless there are enough past experiences stored in the history.

In the same regard, Bürki *et al.* [Bürki 2016] developed a distributed SLAM system with a cloud-based map for multi-vehicle cooperative scenarios. Their work focused on providing a real-time solution for bandwidth-constrained environments. They proposed an approach that reduces the remote data flow by extracting from the distributed map only relevant information for localization under actual environmental conditions. Their proposed approach consists, in a first step, in creating a kind of graph that represent the landmark co-observability in all past traversals (as in Figure 2.13). Then, they use a ranking function whose goal is to decide

how likely a candidate landmark is to be observed at the current environmental condition. However, this approach reaches its limits on maps containing multiple conditions at the same time.

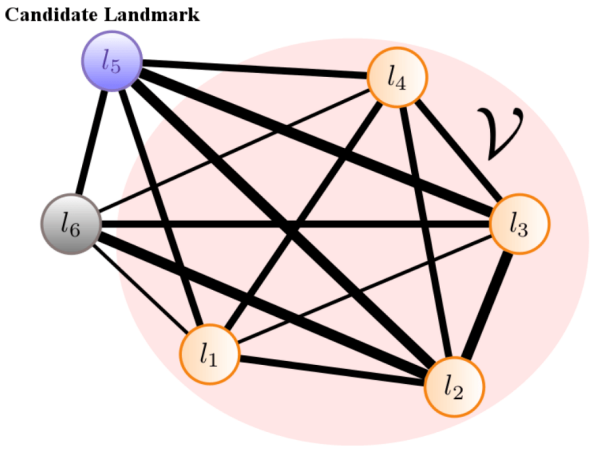


Figure 2.13: An example of a co-observability graph [Bürki 2016]. This graph represents which and how often landmarks (nodes) have been co-observed in the past (edges). The ranking function decides how likely a candidate landmark (blue nodes) is to be observed at the current environmental condition taking into account the recently observed landmarks along the current traversal (orange nodes).

More recently, Bürki *et al.* [Bürki 2018] improved their approach by proposing a new formulation of the ranking function for appearance-based landmark selection. This new formulation allowed to exploit frequently collected sensor data to significantly increase the landmark selection performance without the need to increase the size of the map.

The most recent work of Burki *et al.* [Bürki 2019] deals with the landmark retrieval problem in a similar way. They derived several appearance-based ranking functions from their earlier work and related them to popular ranking schemes from document retrieval techniques with text queries. They assumed that an environmental condition can be encoded in the session-observation relation of landmarks, i.e., a set of landmarks observed in the same sessions are assumed to have the same environmental condition. They used this encoding to select only the set of landmarks observed under the current environmental condition. However, this requires an exhaustive search to retrieve the set of landmarks that have the same session-observation encoding as the current landmarks. Furthermore, this encoding does not allow to accurately encode the environmental condition, e.g., we may observe a landmark taken in rain on both cloudy and sunny days. This means that a significant number of landmarks observed in rainy, cloudy, and sunny conditions can be mixed in the same session-observation encoding.

In the same respect, MacTavish *et al.* [MacTavish 2018] have derived a landmark retrieval technique from other communities. They have inspired their proposed landmark retrieval approach from recommender systems. The proposed recommender

system is based on a collaborative filtering approach (CF), which recommends experiences according to the current environmental conditions based on the landmark matching history. However, this technique suffers from the limitations of CF-based techniques, e.g., if a landmark is not seen during the training process, the system cannot create an embedding for it and consequently cannot query the model with that landmark. This issue is often referred to as the cold-start problem [Kim 2011] in the recommender systems community.

2.4.3 Map management

Although it is true that continuously adding landmarks from different environmental conditions into a single multi-session map can help improving visual long-term localization, the size of the resulting map rapidly increases and becomes impractical to maintain. Therefore, considerable efforts have been made to optimize the map representation by removing redundant information and maintaining a minimal set of landmarks that ensure a robust localization in different environmental appearances.

An early work proposed by Biber and Duckett [Biber 2005] was devoted to optimize map representation for long-term scenarios. They developed dynamic maps that manage environmental changes through the use of robust statistics and multiple local maps at different discrete time-scales, where the map that best represents the current sensor data is used for the localization. Short-term maps are updated online each time new sensory data are obtained while long-term ones are managed offline, typically once a day, based on long-term experience.

Mühlfellner *et al.* [Mühlfellner 2016] used their landmark scoring policy, which assigns scores to landmarks according to the number of different localization sessions in which they appear, to design a map management technique called *"Summary Maps"*. The proposed approach consists in using the landmark scores to summarize the map in an offline process carried out after each localization session. The main limitation of this approach is the fact that landmarks in rarely visited areas are assigned with low scores, which means that they are expected to be removed in the map summarization process. Dymczyk *et al.* [Dymczyk 2015] proposed a very similar approach that aims to solve the problem of bias towards regions which were visited more often. They designed a slightly different scoring policy that relates the actual number of trajectories to the expected number of trajectories to observe a landmark:

$$\text{score} = \frac{\text{actual number of observer trajectories}}{f(\text{expected number of observer trajectories})} \tag{2.5}$$

The expected number of landmark trajectories to observe a landmark is estimated from the landmark co-visibility graph.

In the same way, Bürki *et al.* [Bürki 2018] have also proposed a map management technique in addition to their proposed ranking function which is used for landmark retrieval as explained in the previous sub-section. They have introduced the two types of sessions: *"rich session"* and *"observation session"*. They assumed that

the environmental condition of a localization session, whose performance is worse than a predefined threshold, is not covered in the map. Accordingly, they add new landmarks to the map in order to cover this new encountered environmental condition. Landmarks added to the map in this fashion are referred to as a rich session. In the other case, if the localization has performed sufficiently well (the performance is higher than a predefined threshold), the map is then deemed to cover the encountered conditions. Therefore, they decide that no new landmarks need to be added to the map, but instead, they enrich the previously observed landmarks with new observations from the current localization session. Observations added to the map in this fashion are referred to as an observation session. A schematic illustration of the proposed map management process is presented in Figure 2.14. After a rich session, Bürki *et al.* [Bürki 2018] employ an offline map summarization process which aims to produce a reliable map with a fixed size that covers a high degree of variance in appearance as in [Mühlfellner 2016] and [Dymczyk 2015].

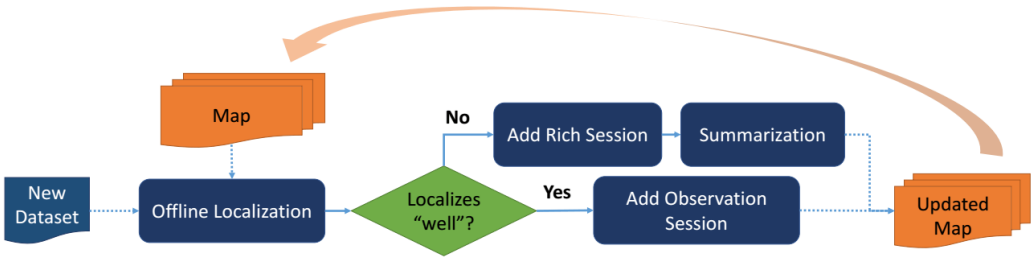


Figure 2.14: Schematic illustration of the map management process proposed in [Bürki 2018].

The work proposed by Krajník *et al.* [Krajník 2016] aims at predicting the current state of the environment based on previously observed and learned temporal patterns. Their proposed system builds a new independent map on each run, these maps are then integrated into a spatio-temporal occupancy grid where each cell contains a frequency-spectrum of its past states and allows the prediction of the cells' future states.

Halodová *et al.* [Halodová 2019] extended the work presented in [Krajník 2016]. They presented an adaptive map update scheme which removes or adds features based on their past influence on the localization quality. They proposed an adaptive scoring policy that increments or decrements each feature score based on whether it was matched correctly, incorrectly or not matched, and thereby they managed to removes the n worst scored features and replace them with n observed features in the current session. However, this map management strategy must be coupled with a very accurate landmark retrieval technique, otherwise the incorrectly matched features that were caused by inaccurate retrievals will be penalized by this scoring policy.

2.5 Conclusion

In this chapter, we have started with a general view on SLAM and its history. After that, we have discussed the influence of long-term changes of the environment on the localization performance and we have classified the state-of-the-art contributions in this regard into three main categories, and accordingly, presented some leading state-of-the-art approaches on each category.

In the work done in this thesis, we address long-term localization with respect to only two of the three categories presented in Section 2.4: landmark retrieval and map management. We did not approach the third category, illumination invariant feature descriptors, because this type of techniques cannot solve the problem of long-term navigation alone, but it can be used in conjunction with other approaches to further improve localization performance. On the other hand, landmark retrieval and map management are two complementary approaches that must be used together to properly address multi-session mapping. Landmark retrieval is used to extract a set of relevant features from a multi-session map to improve localization in long-term scenarios. This retrieval can be costly and even inaccurate in immense maps; therefore, the map management process aims to reduce the size of the multi-session map to ensure the efficiency of landmark retrieval.

In the next chapter, we will present and explain the operation process of our proposed keyframes retrieval approach that is designed to score and retrieve keyframes and their corresponding landmarks according to the environmental changes and the geometric distance. This approach belongs to the category of contributions stated in sub-section 2.4.2.

Chapter 3

Keyframes Retrieval

Contents

This chapter is organized as follows. Section 3.1 gives an introduction to our work. In Section 3.2, we exhibit our contribution in relation to the state-of-the-art. After that, a detailed description of the methodology of our approach is provided in Section 3.3. The experiments and results are presented in Sections 3.4 and 3.5 respectively. Finally, the chapter conclusion is given in Section 3.6.

3.1 Introduction

Estimating the probability of matching features between the current image and a keyframe in the map is a challenging task because, as illustrated in Figure 3.1, these two images may have been recorded at different viewpoints, under different environmental conditions, or even both at the same time.

Accordingly, in this chapter we address a keyframe retrieval solution with respect to the problem stated in Figure 3.1. We propose a localization approach able to take advantage of a visual landmark map composed of N sequences gathered at different times. This approach can be applied particularly to autonomous shuttles since using N sequences of the same trajectory to build a map is suitable for such usage. Generally, SLAM algorithms retrieve the keyframes that are used for localization based on their geometric distance to the vehicle pose. However, this technique has shown

(a) Different viewpoints.

(b) Different environmental conditions.

(c) Different viewpoints + different environmental conditions.

Figure 3.1: An example of different cases of matching the current image with a keyframe in the map. In (a), the current image (top) and the keyframe (bottom) have different viewpoints. In (b), the two images have a same viewpoint but different environmental conditions. In (c), the two images have a different viewpoint and different environmental conditions at the same time.

a weakness in long-term localization because it does not take into account environmental changes. The example presented in Figure 3.2 illustrates a bad localization case where the SLAM system has retrieved the closest keyframe (circled) to the vehicle estimate pose (purple). Such a case may produce a localization failure since the retrieved keyframe and the current image do not share the same environmental conditions (day against night matching).

This incents us to develop a new strategy for keyframe retrieval to adapt our SLAM algorithm [Lébraly 2011, Royer 2016] for long-term operation. During the localization process, we aim to maximize the number of matched points by retrieving relevant experiences from a map that integrates numerous environmental conditions. Our proposed approach takes advantage of statistics gathered in the first few meters of a traversal to compute a probabilistic ranking function. This function is used in the rest of the traversal to assign relevance scores to a set of keyframes in the map as illustrated in Figure 3.3. Those scores are computed by the ranking function taking into account the environmental conditions of each keyframe and its corresponding geometric distance to the vehicle estimate pose. Accordingly, the ranking function retrieves from the map the most relevant keyframes (i.e., the ones with the highest score). In order to ensure our ranking function consistency, we keep updating it regularly throughout the trajectory.

We evaluated our approach on the Oxford RobotCar dataset [Maddern 2017] and a new dataset recorded with our experimental vehicle that we make avail-

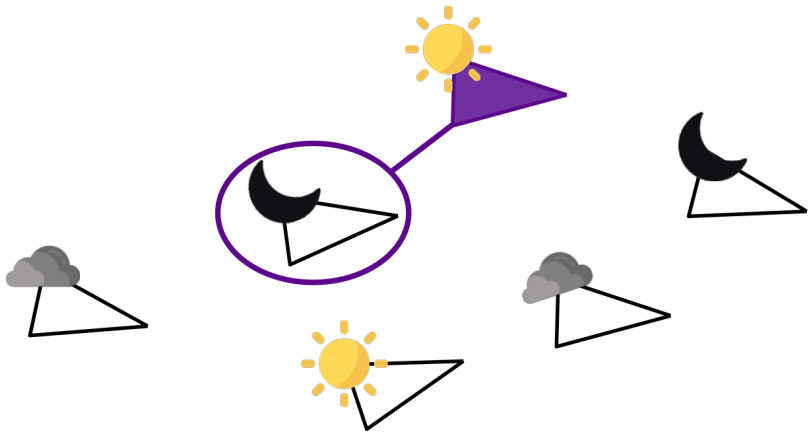


Figure 3.2: An example of a bad localization case where the SLAM system has retrieved the closest keyframe which was recorded under different environmental conditions.

able to the community. Our dataset is called IPLT[1](Institut Pascal Long-Term) dataset [Bouaziz 2020] and it contains, at the moment, 127 sequences recorded over two years in which the vehicle has followed the same path around a parking lot with slight lateral and angular deviations. This dataset contains various environmental conditions due to changes in luminance, weather, seasons and parked vehicles and each sequence is around 200 m in length. An aerial view of the parking lot where we have recorded our dataset is presented in Figure 3.4. Further details about the IPLT dataset are provided in Appendix A.

3.2 Contribution

The work presented in this chapter is addressing the landmark selection problem to improve long-term localization in dynamic environments. In contrast to other works [Mühlfellner 2016, Linegar 2015, Bürki 2016, Bürki 2018, Bürki 2019, MacTavish 2018] which have also addressed this issue by selecting a set of landmarks from the map, we approach the landmark selection in a keyframe-based way. This means that we do not consider retrieving landmarks directly from the map, but instead, we search for the most valuable keyframes for localization and we retrieve their associated landmarks. In fact, this keyframe formulation of the landmark selection problem allowed us to design a probabilistic ranking function that takes into account not only the environmental conditions of the keyframes (and their associated landmarks), but also their geometric distances to the vehicle pose. This also means that instead of computing a score for each landmark separately, which is costly and useless since all landmarks associated to a same keyframe share the

[1]To download our dataset please visit http://iplt.ip.uca.fr/datasets/ and enter the following username/password for a read-only access to our ftp server : ipltuser/iplt_ro

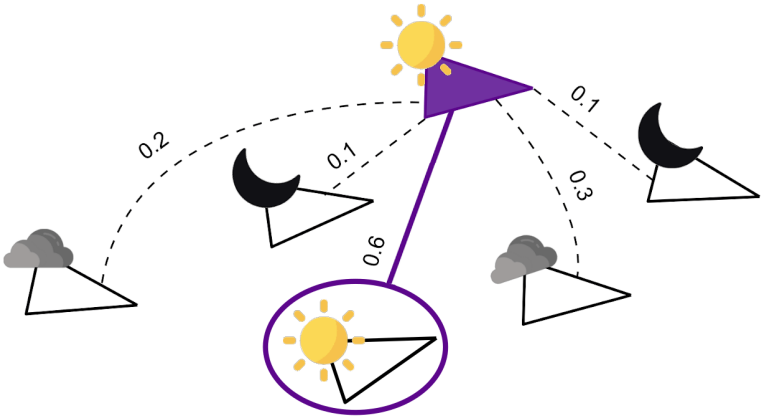


Figure 3.3: Scores assigned to keyframes by the ranking function. The one with the highest score (circled) will be retrieved and used for localization.

same environmental condition, the proposed ranking function computes and assigns a score to each keyframe. This formulation allows to minimize the computational time.

We note that comparing our approach's performance with most of the existing state-of-the-art techniques is very difficult. This is due to a major conceptual difference, where each state-of-the-art approach is proposed and applied in a specific mapping framework with particular features representation [Bürki 2019, Magnago 2019]. Such a difference in the representation makes it difficult to compare one approach applied to, say, a filter-based slam with another applied to a keyframe-based SLAM.

This chapter is based on a work [Bouaziz 2021] that was published at IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI 2021).

3.3 Methodology

In order to retrieve only relevant information for localization, we designed a ranking function that is partially built with offline data. The role of this function is to retrieve from the map the most relevant keyframes taking into account two main factors:

- The geometric distance between the vehicle pose estimation and the pose associated to the keyframe K_j which is used for localization.
- The environmental conditions of the keyframe K_j as well as the current environmental conditions (lighting, weather, season,...).

The ranking function is computed during the first few meters of the trajectory. After that, it will be used to estimate the probability that a point pt extracted from the current image I_i finds a match in the keyframe K_j retrieved from the map. The

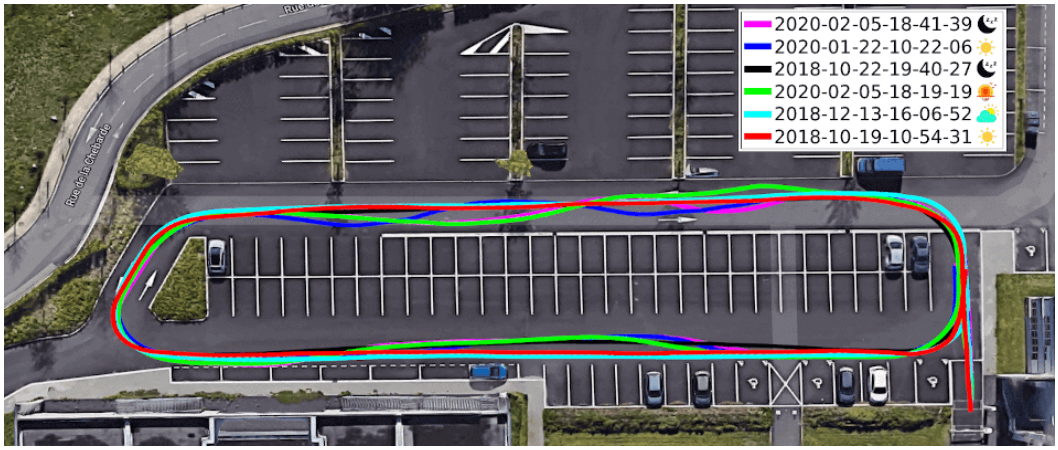


Figure 3.4: An aerial view of the parking lot where we have recorded the IPLT dataset. The colored curves are representing an example of the path traveled while recording 6 different sequences.

ranking function is described by the Equation (3.1):

$$P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot f_c(I_i, K_j) \tag{3.1}$$

$f_{dist}(I_i, K_j)$ is a function computing a score for matching the current image I_i with the keyframe K_j retrieved from the map by supposing that both of them were taken in similar conditions (weather, lighting...). This implies that computing this score depends only on the geometric distance between the two images. On the other hand, $f_c(I_i, K_j)$ supposes that the poses of the two images are identical; hence it takes into account only the appearance changes to compute the score of matching between the two images I_i and K_j. f_{dist} is computed offline with the use of some data collected specifically for this purpose. The calculation of $P(pt \in I_i, K_j)$ and the calculation/update of f_c are performed online during re-localization. We refer to re-localization the process of localizing the vehicle in a previously mapped place.

In Figure 3.5, we present a diagram that explains the re-localization process using the proposed ranking function. The ranking function ($P(pt \in I_i, K_j)$) takes as input the current image (and its corresponding predicted pose according to the odometry data) to retrieve from the multi-session map the keyframe (and its associated landmarks) that has the highest similarity score. A new pose will be computed by matching the current image and the retrieved keyframe. To ensure the consistency of f_c with the environmental condition changes, it will be updated along the traversal. This update task is performed in parallel with the retrieval of the most relevant keyframe for localization (according to the ranking function). Therefore, the ranking function retrieves another keyframe from the map (each time from a different traversal in a circular way) and uses it to update f_c (see Section 3.3.2.3 for more details). The following sections present details on both offline and online computations.

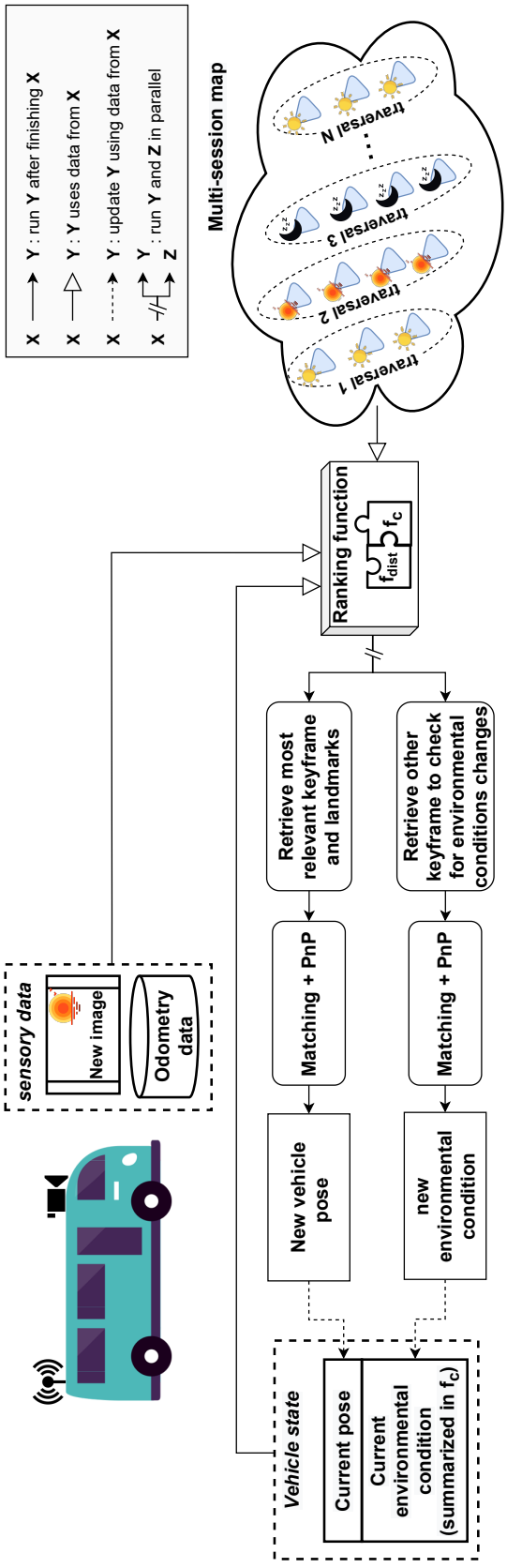


Figure 3.5: A diagram representing the operating mechanism of the re-localization part.

3.3.1 Offline computation of f_{dist}

We have recorded some sequences specifically for this step. These sequences were recorded successively in a short time period to avoid variation in lighting because $f_{dist}(I_i, K_j)$ assumes that I_i and K_j are taken under the same environmental conditions. The sequences were also taken at the same place and the vehicle has followed a slightly different path with some lateral and angular deviation between each pair of sequences.

We conducted three separate recording sessions at three different locations and times to evaluate the extent to which f_{dist} depends on the recording environment of the sequences. We used the sequences of each recording session to define a distinct f_{dist} function. This means that we have defined three different functions using the sequences of the three recording sessions. In the first session, we recorded 7 sequences, in the second session we recorded 3 sequences and in the third session we recorded 5 sequences. Figure 3.6 shows some examples of sequences recorded in the three sessions.

We compute f_{dist} in the same way for each recording session. For each pair of sequences <SeqA, SeqB> belonging to the same recording session, we use SeqA to generate a map using our SLAM algorithm, then we use SeqB to perform re-localization on the produced map. While performing re-localization with SeqB, for each current pose estimate $\hat{p}_i$ (estimated using the previous computed pose p_{i-1} + wheel odometry), we pick the 20 closest keyframes and their corresponding poses from the map built with SeqA: K^1, ..., K^{20} and we match each of them ($K^l, l \in [1, 20]$) to our current image I_i and calculate the current pose p_i in order to calculate the inlier rate: $inliers/(inliers + outliers)$. Then, we compute the longitudinal, lateral and angular gap between the current pose p_i and the pose p^l associated to the keyframe K^l. Therefore, for each pose p^l, we record a quadruplet consisting of the percentage of inliers, the longitudinal, lateral and angular distance.

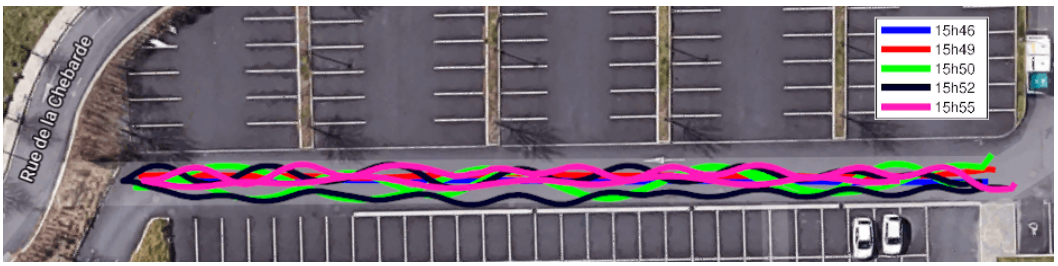
These collected data (the recorded quadruplets) are used to define f_{dist} as the function that computes the percentage of inliers giving the longitudinal, lateral and angular distance between the poses. In sub-section 3.3.1.1 and 3.3.1.2, we present two different ways to define the function f_{dist}. Sub-section 3.3.1.1 presents the earlier version of f_{dist}, it consists in defining and multiplying three independent 2D functions that model the camera motion along longitudinal, lateral, and angular displacements independently. The second definition of f_{dist} (sub-section 3.3.1.2) represents the newer version which is in use at the present time. It consists of a single 4D function represented as a linear combination of Gaussians that models the camera motion along longitudinal, lateral, and angular (yaw angle) displacements.

3.3.1.1 Defining f_{dist} as a multiplication of three independent functions

This definition of f_{dist} consists in multiplying three independent 2D functions modeling the camera motion along longitudinal, lateral, and angular displacements: f_{long}, f_{lat} and f_{ang}. Each one of these three functions is designed to compute a score for matching the current image I with the keyframe K while assuming that both

(a) Session 1: we present an example of 4 sequences recorded in this session. These sequences were recorded the same day at 15:16, 15:22, 15:23 and 15:24.

(b) Session 2: we present the 3 sequences recorded in this session. These sequences were recorded the same day at 15:37, 15:40 and 15:43.



(c) Session 3: we present the 5 sequences recorded in this session. These sequences were recorded the same day at 15:46, 15:49, 15:50, 15:52 and 15:55.

Figure 3.6: Example of sequences used for the calculation of f_{dist} from three different recording sessions. The two last sessions ((b) and (c)) were performed the same day while the first one was performed in a different day.

of them were taken in similar conditions. The computed scores represent the similarity level between I and K with respect to the longitudinal, lateral, and angular distances of their acquisition poses. Therefore f_{dist} can be represented as follows:

$$f_{dist}(I,K) = f_{long}(I,K) \cdot f_{lat}(I,K) \cdot f_{ang}(I,K) \tag{3.2}$$

In order to define these three functions, we proceed to a sampling phase of the collected data in which we assign the longitudinal and lateral distances into bins of $0.1m$ and the angular distances into bins of $0.1°$. After that, we calculate the median of each bin. Finally, we smooth these medians with the RLOESS smoothing function proposed in [Cleveland 1979]. Thus, we obtain three curves that define the functions f_{long}, f_{lat} and f_{ang}. In Figure 3.7, we present the form of the three functions computed with the data collected from the first recording session (Figure 3.6a).

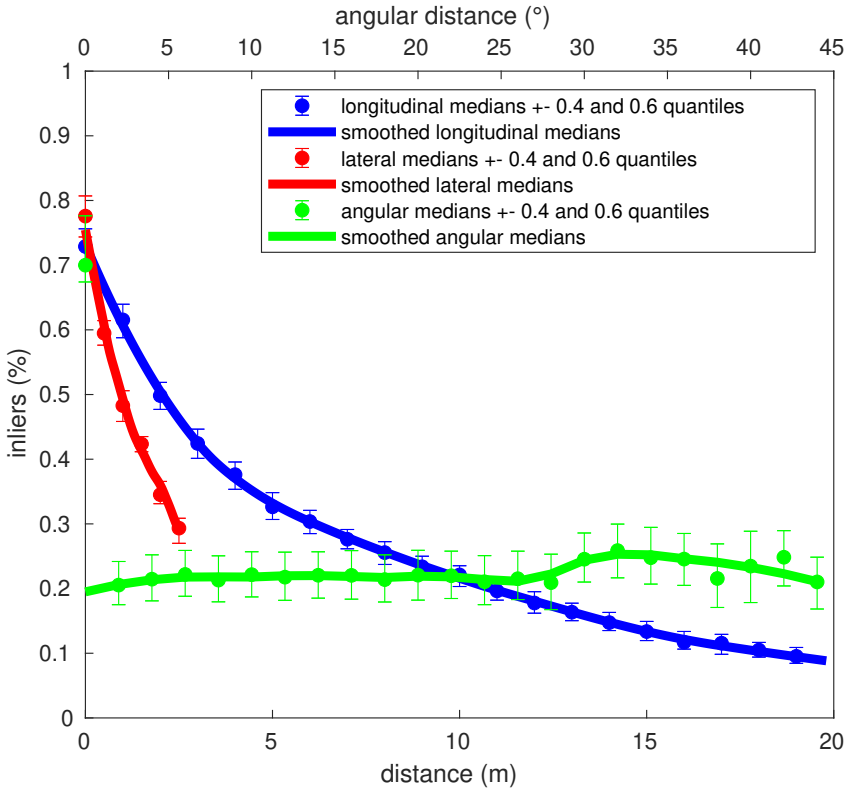


Figure 3.7: Form of the three functions f_{long}, f_{lat} and f_{ang}. We used the collected data to define each of them. For each curve, we compute the median of each bin along with the 0.4 and 0.6 quantiles (for clarity, we have not plotted all the medians). We also present the smoothed curves in the same color. Longitudinal and lateral distances are measured in meters (lower axis), while angular distances are measured in degrees (upper axis).

Visually, it is clear that f_{long} and f_{lat} are two decreasing functions. This is not the case for f_{ang} which is almost constant. In fact, this is not surprising since a pure rotation does not change much the appearance of the points in the image. Thus, we decided to remove it from equation (3.4). We also notice in the figure that the highest value of f_{long} and f_{lat} is around 0.75. Therefore, we normalized these two functions between $[\min(f_{long})..1]$ and $[\min(f_{lat})..1]$ (respectively for f_{long} and f_{lat}) using the following formula:

$$f^{norm} = (1 - \min(f)) \cdot \frac{f - \min(f)}{\max(f) - \min(f)} + \min(f) \tag{3.3}$$

Therefore, f_{dist} can be finally defined as follows:

$$f_{dist}(I,K) = f_{long}^{norm}(I,K) \cdot f_{lat}^{norm}(I,K) \tag{3.4}$$

The shape of f_{long}, f_{lat} and f_{ang} computed using the data collected from the other two recording sessions (Figures 3.6b and 3.6c) is very similar to that shown in Figure 3.7.

3.3.1.2 Defining f_{dist} as a linear combination of Gaussians

In this sub-section we introduce the novel form of f_{dist}. We have remodeled the previous form which omits the angular motion. Moreover, instead of considering the camera motion along the longitudinal, lateral, and angular displacements as three mutually independent functions, the novel form models the camera motion as a unified probability model defined as a Gaussian process. Therefore, this version of f_{dist} will be retained in further evaluations. We have chosen to define f_{dist} as a linear combination of Gaussians as in Equation (3.5):

$$\begin{aligned} f_{dist}(I,K) &= \sum_{h=1}^{n_g} a_h G_h, \text{ with:} \\ G_h &= e^{-\frac{d_x(I,K)^2}{b_h^2} - \frac{d_y(I,K)^2}{c_h^2} - \frac{d_r(I,K)^2}{d_h^2}} \end{aligned} \tag{3.5}$$

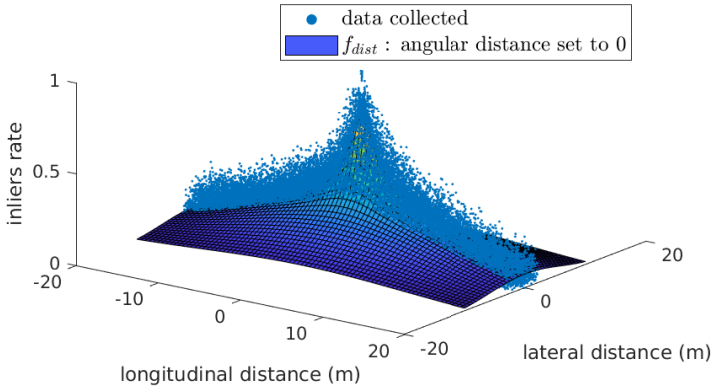
$d_x(I,K)$, $d_y(I,K)$, $d_r(I,K)$ are respectively the longitudinal, lateral and angular distances between the pose of the image I and the pose of the keyframe K. The parameters a_h, b_h, c_h and d_h are computed with a non linear Least-Squares minimization to fit f_{dist} to the data points. n_g is the number of Gaussians in the model and it was chosen to have the lowest residual error. In Table 3.1, we show the mean residual error with respect to n_g obtained while fitting the function f_{dist} to the data points computed with the sequences of the first recording session (Figure 3.6a).

Table 3.1: Mean of residual errors from fitting f_{dist}.

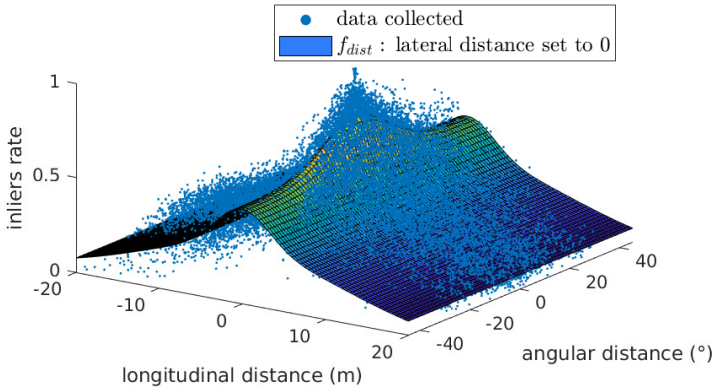
n	1	2	3	4
Mean	0.090	0.075	0.074	0.074

According to the table, we retained n_g=3 as the number of Gaussians in our model. Figure 3.8 presents the result of fitting the function f_{dist} to the collected data from the same recording session with $n_g = 3$.

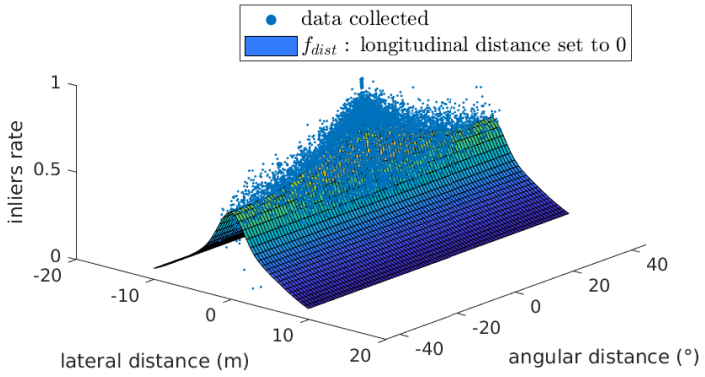
We performed the same experiment using the sequences of the second and the third recording sessions (Figures 3.6b and 3.6c). The shape of the data collected and the shape of the fitted function f_{dist} are visually identical to those computed



(a) f_{dist} with $d_r = 0$



(b) f_{dist} with $d_y = 0$



(c) f_{dist} with $d_x = 0$

Figure 3.8: Each sub-figure presents the surface of the 4D function f_{dist}. In (a), the inliers rate surface is plotted according to longitudinal and lateral distance while the angular distance d_r is set to 0. The same for (b) and (c), d_y and d_x are set to 0 respectively.

with the first recording session.

The shape of the data collected is dependant on the key-points detector and the features descriptor. In our experiments, we use Harris corner detector [Harris 1988] for extracting key-points which are matched with ZNCC — Zero-mean Normalized Cross-Correlation — computed on 11×11 pixel windows around each key-point. However, our method can still be applied in the same way using other descriptors.

3.3.2 Online computations

3.3.2.1 Calculating f_c

We compute the function f_c during the re-localization process. As explained previously, $f_c(I_i, K_j)$ does not depend on the acquisition location of the images I_i and K_j, but only takes into account the condition of the environment to estimate the score of matching between I_i and K_j. Accordingly, we deduce that f_c is traversal dependent, this means that the value of f_c between I_i and any image belonging to the same traversal as K_j will have the same value as $f_c(I_i, K_j)$ (in the normal case where there is no sudden change in the weather or in the lighting condition). Therefore, it is more suitable to design the function f_c as a 2D matrix F_c which has the shape of $[N \times N]$ (where N is the number of traversals existing in the map). The matrix F_c can be formalized as follows: $f_c(I_i, K_j) = F_c(trav(I_i), trav(K_j))$, where the function $trav(X)$ refers to the index of the traversal where the image X belongs (since I_i is the current image, $trav(I_i)$ refers to the current traversal). The value of $F_c(trav(I_i), trav(K_j))$ corresponds to the score of similarity (which depends on the environmental conditions) between the traversal which contains I_i and the traversal which contains K_j. Thus, Equation (3.1) becomes:

$$P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot F_c(trav(I_i), trav(K_j)) \tag{3.6}$$

In order to define and calculate the matrix F_c, we use our global map (which is composed of N sequences having different environmental conditions) as a reference map to perform a re-localization. During this re-localization phase, we devote the beginning of the trajectory to the calculation of the function. According to Equation (3.6), we have:

$$F_c(trav(I_i), trav(K_j)) = \frac{P(pt \in I_i, K_j)}{f_{dist}(I_i, K_j)} \tag{3.7}$$

Our idea consists in calculating the matrix F_c at the start of the trajectory according to Equation (3.7). Since the ranking function is not yet defined, we measured the inlier rate by matching the image I_i to the keyframe K_j and used it to replace $P(pt \in I_i, K_j)$ in the Equation (3.7). In the first few meters of the re-localization, we follow the steps described by Algorithm 1 to initialize the values of the matrix F_c.

Algorithm 1: Calculation of the F_c matrix in the first few meters

$c \leftarrow N + 1$ (the index of the current traversal)
Initialize the buffers: $\mathcal{P}[l] \leftarrow \varnothing, \forall l \in [1, N]$
Compute the initial pose p_0 by localizing the image I_0 within the map using bag-of-words.
for each current image I_i in the first few meters **do**
    Predict the pose $\hat{p}_i$ of I_i using the last computed pose p_{i-1} and the wheel odometry data.
    for each traversal l in the map **do**
        Pick from the traversal l the 3 closest keyframes to $\hat{p}_i$: $K_j^l, j \in [1, 3]$
        for each picked keyframe K_j^l **do**
            Match I_i to K_j^l and compute the inlier rate $P(pt \in I_i, K_j^l)$
            Calculate x such as: $x \leftarrow \dfrac{P(pt \in I_i, K_j^l)}{f_{dist}(I_i, K_j^l)}$
            $\mathcal{P}[l] \leftarrow \mathcal{P}[l] \cup \{x\}$
        end for
    end for
    Calculate pose p_i by matching I_i and the keyframe with the highest number of inliers among the keyframes $\{K_j^l\}, j \in [1, 3], l \in [1, N]$
end for
for each traversal l in the map **do**
    $F_c(c, l) \leftarrow mean(\mathcal{P}[l])$
end for

N is the number of traversals existing in the map.

3.3.2.2 Calculating the ranking function $P(pt \in I_i, K_j)$

The classic method for keyframes retrieval which consists in picking out the keyframes from the map only according to their geometric distance to the vehicle pose is not suitable for long-term operation. For example, if we perform a re-localization in the day while the closest keyframe to our current vehicle pose was taken during the night, the matching is extremely difficult; therefore, we have proposed to implement a ranking function which is able to consider other important criteria such as environmental conditions for the retrieval of keyframes.

We have calculated the matrix F_c in the beginning, the goal in the rest of the trajectory is to find the keyframe K^* that maximizes the probability of matching defined in Equation (3.6):

$$K^* = \underset{K}{\operatorname{argmax}}\, P(pt \in I_i, K) \tag{3.8}$$

K^* corresponds to the circled keyframe in the example presented in Figure 3.3. Searching for K^* in the whole map is costly. Therefore, we pick from each traversal the keyframe $K^l, l \in [1, N]$ which has the minimum geometric distance to the current vehicle pose estimation. For all keyframes $\{K^l\}, l \in [1, N]$, we calculate the probability of matching with Equation (3.6) to retrieve the one with the highest score (K^*) with Equation (3.8). K^* will be used for further matching and pose calculation and optimization.

In Section 3.5, we are comparing results obtained by the classic ranking function $\tau_1 = f_{dist}(I_i, K_j)$ (The score will be assigned based on the geometric distance between the two images as demonstrated in Figure 3.8) with results obtained by our proposed ranking function $\tau_2 = P(pt \in I_i, K)$ (Equation (3.6)).

3.3.2.3 Update of $F_c(trav(I_i), trav(K_j))$

It is interesting to update the values of the matrix F_c regularly after the first few meters of the trajectory. Indeed, this update can be useful when the state of the environment changes between the beginning and the end of the sequence. This update process is performed in parallel with the keyframes retrieval process. For this reason, we aim to avoid slowing down the re-localization by updating F_c for a single traversal at each iteration. Algorithm 2 describes the update process of the matrix F_c and the keyframe retrieval mechanism using our ranking function which has already been computed in the first few meters of the trajectory.

The value of the update rate α defines how fast the ranking function responds to sudden changes in the environment during the same re-localization session. In sub-section 3.5.1.3, we present an evaluation of different values of α with respect to a scenario where the environmental conditions change between the start and the end of the localization session.

Algorithm 2: Keyframe retrieval and update of F_c

Parameters: The update rate $\alpha \leftarrow 0.1$
Steps:
$c \leftarrow N + 1$ // the index of the current traversal
$l \leftarrow 1$ // the index of the first traversal
for each current image I_i **do**
    Predict the pose $\hat{p}_i$ of I_i using the last computed pose p_{i-1} and the wheel odometry data.
    // Update of F_c:
    Pick from traversal l the closest keyframe to $\hat{p}_i$: K^l
    Match I_i to K^l and compute the inlier rate: $P(pt \in I_i, K^l)$
    Calculate x such as: $x \leftarrow \dfrac{P(pt \in I_i, K^l)}{f_{dist}(I_i, K^l)}$
    Update F_c: $F_c(c, l) \leftarrow (1 - \alpha)F_c(c, l) + \alpha x$
    $l \leftarrow l + 1$
    if $l > N$ **then**
        $l \leftarrow 1$
    end if
    // Keyframe retrieval and pose calculation:
    Retrieve from the map the keyframe K^* which has the highest score assigned by the ranking function:
$$K^* = \underset{K}{\operatorname{argmax}}\, P(pt \in I_i, K)$$
    Calculate the pose p_i by matching I_i and K^*
end for

3.4 Experiments

We divide each dataset into mapping sequences and test sequences. We first proceeded to a mapping phase in which we built a global map consisting of N traversals from the mapping sequences with varying environmental conditions. We used the generated global maps as reference maps in our experiments in order to perform re-localization with the test sequences. This allowed us to evaluate the effectiveness of our algorithm in different conditions. We are also comparing results obtained while using $\tau_1 = f_{dist}(I_i, K_j)$ as a ranking function (which takes into account only the geometric distance as a criterion to assign scores to keyframes) with results obtained while using our proposed ranking function $\tau_2 = P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot F_c(trav(I_i), trav(K_j))$ (which takes into consideration the geometric distance and the environmental conditions of the keyframes).

3.4.1 Experimental setup

We present the average number of inliers observed in each sequence as well as the average number of localization failures per km as criteria for the comparison. Practically, we found that the localization can be considered as reliable when there are at least 30 points matched between the current image and the database, below this threshold, we consider a localization failure. This is a conservative threshold to ensure the security of our autonomous shuttle [Royer 2016].

The number of meters m required to initialize the matrix F_c (in the start of the trajectory) was chosen experimentally to minimise the distance between the value of F_c calculated after m meters and its mean $\bar{F}_c$ along the trajectory. In Table 3.2, we illustrate an example of this distance ($|F_c - \bar{F}_c|$) for different values of m.

Table 3.2: Value of $|F_c - \bar{F}_c|$ with respect to the choice of m.

m (meters)	5	10	20	30
$\lvert F_c - \bar{F}_c \rvert$	0.985	0.066	0.027	0.024

According to Table 3.2, we have chosen to fix the parameter m to 20 meters. Thus, we devote the first 20 meters of the trajectory for the calculation of the matrix F_c.

All calculations were performed on a Dell tower with an Intel Xeon W-2133 CPU with localization running in real-time.

3.4.2 Datasets

Intensive work on SLAM algorithms has produced a large number of related datasets such as KITTI [Geiger 2013], Ford Campus Dataset [Cordts 2016]. . . . The vast majority of these datasets are designed for localization in static environments with very small environmental changes. However, datasets with many environmental conditions are required for applications that aim for long-term localization in dynamic environments. Datasets like Oxford RobotCar [Maddern 2017], NCLT [Carlevaris-Bianco 2015a] and UTBM [Yan 2019] are widely used for long-term localization applications since they include different environmental conditions. Both of the last two mentioned datasets contain only a few number of different sequences, which makes it difficult for us to test our approach on them. For this reason, we are using only Oxford RobotCar and our own dataset (IPLT) on the test phase.

3.4.2.1 IPLT dataset

The IPLT dataset was created from recorded images of two gray-scale 100° FOV cameras mounted on our experimental vehicle (one front and one rear camera) and

wheel-odometry. From this dataset, we have selected 103 sequences at different hours during the day, with varying weather conditions (rain, sun, overcast) and some lateral and angular deviations, 10 of them were used for the construction of the first global map (see Figure 3.9).

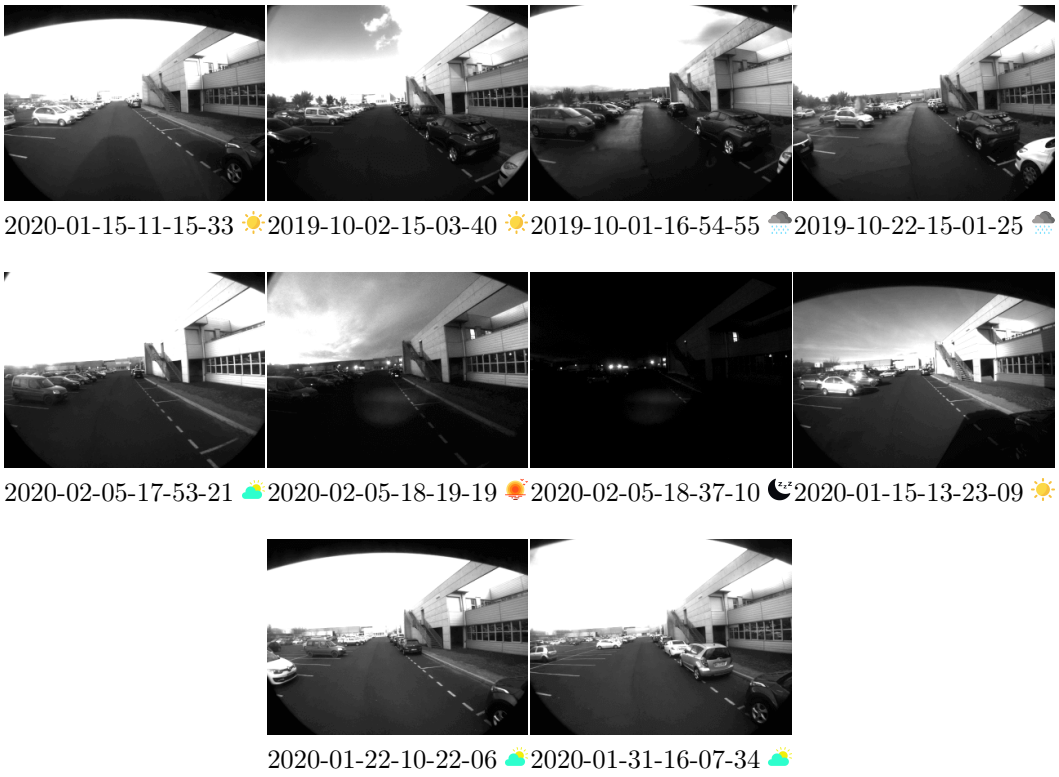


Figure 3.9: An overview of images from IPLT dataset used for the construction of the first global map. For each sequence we are indicating the acquisition date in the format "yyyy-mm-dd-HH-MM-SS" and symbolizing the environmental condition by a small icon. Please refer to Table 3.3 for more details about the designation of condition icons used in this document.

It is difficult to perform re-localization using night-time sequences on a map containing only day-time sequences, hence, we added a dusk sequence to the map to work as an intermediate between day and night sequences. This allowed the SLAM algorithm to find matches linking all the traversals so all the poses have been optimized in the same bundle adjustment. We verified manually the map to make sure that all the poses are geometrically coherent. This means that there is no differential drift between the traversals. This global map contains sequences having different environmental conditions, it also includes some lateral and angular deviations between the sequences as illustrated in Figure 3.4.

Table 3.3: Designation of condition icons.

Icon	Designation
	Day & sunny condition
	Dusk condition
	Night condition
	Cloudy weather
	Rainy weather
	Snow condition

3.4.2.2 Oxford RobotCar dataset

In the Oxford RobotCar dataset, the itinerary and the direction of traversal followed during individual recordings vary between the different sequences. Accordingly, we have identified a ∼1.6 km segment in which the vehicle has followed the same route and direction in 22 different sequences (as shown in Figure 3.10). All of

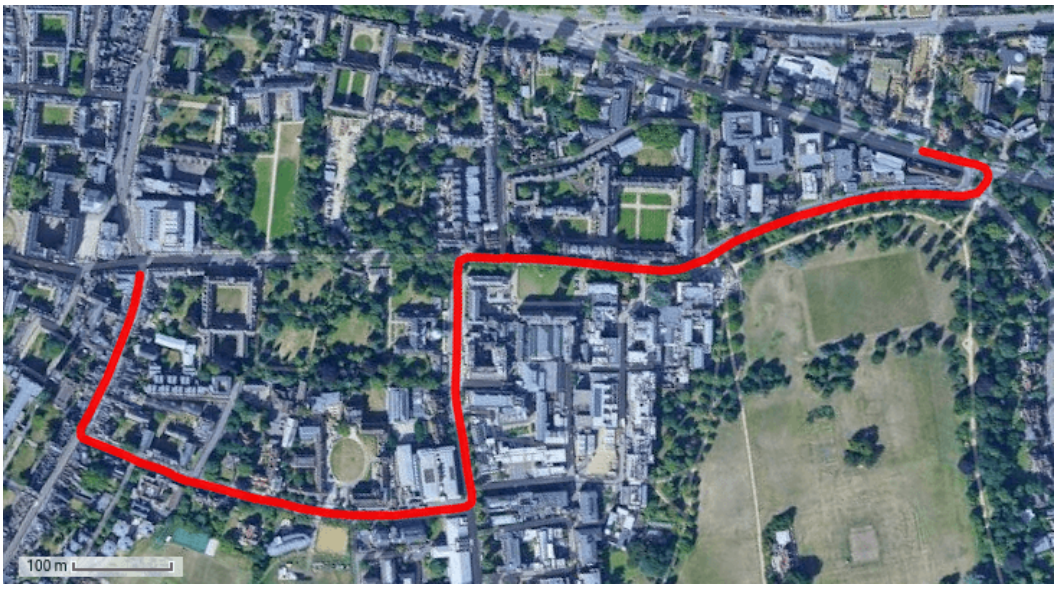


Figure 3.10: A ∼1.6 km segment in which the Oxford vehicle has followed the same route and direction in 22 different sequences.

these sequences were recorded during daytime, only one sequence was recorded at dusk and 2 sequences were recorded at night. The dusk sequence was recorded at the very beginning of the sunset time (16:07) and it was not possible for us to use it to properly localize the night sequences in a map created from daytime sequences. Therefore, we have used only daytime sequences (with varying weather conditions) due the lack of intermediate dusk sequences to match day-time and night-time sequences. We used the visual odometry together with both front left

and right cameras in our mapping framework.

We picked 8 sequences from this dataset to build our second global map (the Oxford map) while we used 12 other sequences for our tests. Figure 3.11 illustrates an overview of images taken from the sequences used for the construction of the Oxford map.

Figure 3.11: An overview of images from Oxford RobotCar dataset used for the construction of the Oxford map.

In our tests, we are also interested in testing the effect of lateral and angular variations between sequences on the localization performance. However, the Oxford RobotCar dataset does not provide sequences with such criteria. This is one of the main reasons that led us to record our own dataset (IPLT).

3.5 Results

In this section, we demonstrate the efficiency of our keyframes retrieval approach by analyzing results obtained after performing re-localization with different sequences. As we mentioned in the previous section, we performed our tests on the two global maps obtained from the two datasets.

3.5.1 IPLT dataset

We decompose this section into three sub-sections. In sub-section 3.5.1.1, we show the overall results of the comparison between τ_1 and τ_2 ($\tau_1 = f_{dist}(I_i, K_j)$ and $\tau_2 = P(pt \in I_i, K_j) = f_{dist}(I_i, K_j) \cdot F_c(trav(I_i), trav(K_j))$). In sub-section 3.5.1.2, we present a detailed study of the effect of lateral and angular deviations on the ranking function. Sub-section 3.5.1.3 presents a detailed study on the effect of update rate α on the update process on the ranking function.

3.5.1.1 Overall results

We are using 93 sequences having different environmental conditions from IPLT dataset for this test phase. In Figure 3.12 we present an illustration of images extracted from some of these sequences.

Figure 3.12: An overview of images recorded with the front camera for some of the sequences of IPLT dataset used in our tests.

In Figure 3.13, we present a comparison between localization performance on the IPLT map while using τ_1 and τ_2 as ranking functions. This figure was generated using all the 93 test sequences. These sequences were classified manually into 5 different classes according to their environmental condition. These 5 classes (sun, overcast, rain, dusk and night) are containing respectively 13, 39, 12, 17 and 11 sequences. The "global" class regroups all the 93 sequences. For each class, we show the average number of inliers observed on the sequences of this class. We also present the number of localization failures per km experienced on these sequences.

Overall, we note that our proposed keyframe retrieval approach has significantly increased the number of inliers observed during re-localization for all classes. We also notice that with our approach, we have successfully reduced the number of bad localizations per km from 61.24 to only 0.16 ($\sim 400\times$). Such a criterion (the number of localization errors) is very important in autonomous shuttle applications since a localization failure leads to the interruption of the driverless system and requires manual interventions. For example, if the shuttle travels 100 km per day, then our approach allows to reduce the daily manual interventions from more than 6,000 to only 16.

In Table 3.4, we monitor the computational cost of processing a single frame by the tracking thread of our system. The values shown in the table are representing the mean value computed on all the frames of the 93 test sequences when using

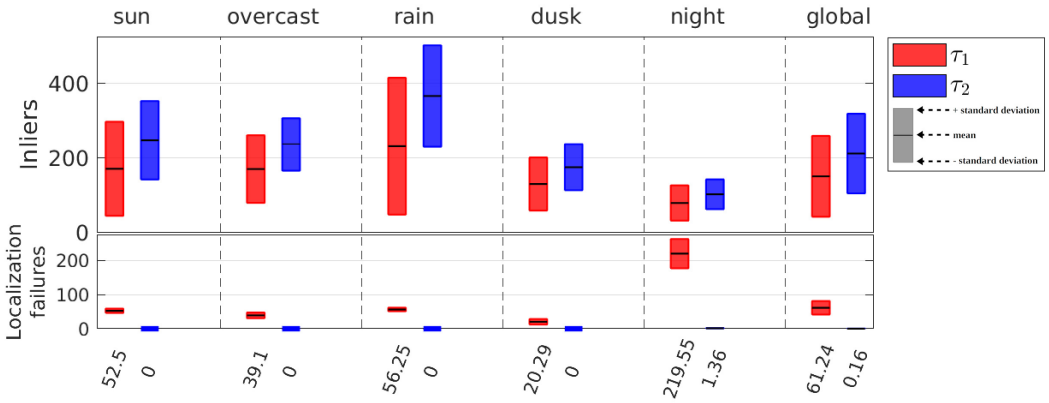


Figure 3.13: A comparison between the two ranking functions τ_1 and τ_2 on the IPLT map using the 93 test sequences from IPLT dataset. The comparison is done according to two criteria: the average number of inliers per image and the average number of localization failures per km. Each box represents the mean value + and - the standard deviation of inliers (or localization failures) recorded while performing re-localization using all the sequences of the corresponding class on the global map. The color of the boxes indicates which ranking function was used to record these values. For better readability of localization failures values, we plot the average number of localization failures in the bottom x-axis.

τ_1 and τ_2. Since we execute a particular processing in the first m meters of the trajectory for τ_2 (as described in Algorithm 1), we decomposed the computation time into two parts: computation time in the first m meters (computation time of Algorithm 1) and computation time in the rest of the trajectory (computation time of Algorithm 2).

Table 3.4: Computational cost of processing a frame.

	τ_1	τ_2
Computational time in the first m meters (ms)	7.5	58.7
Computational time in the rest of the trajectory (ms)	7.8	13.7

According to Table 3.4, τ_2 is significantly more costly than τ_1 in the first m meters of the trajectory (58.7ms versus 7.5ms). This is because we perform an exhaustive matching with the neighboring keyframes to compute the matrix F_c (as described in Algorithm 1, we match the current frame with the 3 closest keyframes from each traversal). It is possible to reduce this computation time by reducing the matching rate (i.e., instead of matching the current frame with the 3 nearest keyframes, we can match it with only 1 or 2 keyframes from each traversal). In our case, we were still able to localize in real-time on the first m meters since we use a camera that captures 10 frames per second (58.7ms * 10 $<$ 1s). The computation time in the rest of the trajectory is a little higher for τ_2 than for τ_1. This is not surprising since we update F_c along the trajectory as described in Algorithm 2.

3.5.1.2 Detailed study on the effects of deviations from the learned trajectory

To evaluate the effect of lateral and angular deviations on the ranking function, we have selected a mapping and a test sequence with some deviations. Figure 3.14 shows the path followed when a re-localization was performed on the IPLT map using the test sequence 2020-02-05-18-41-39 . The other path shown in the same figure is a sequence from the global map that was recorded a few minutes before (2020-02-05-18-37-10). Therefore, these two sequences have very similar environmental conditions. The only difference between these two sequences is that we have performed some lateral and angular deviations in the test sequence. These deviations represent what happens when we have to avoid an obstacle on the way.

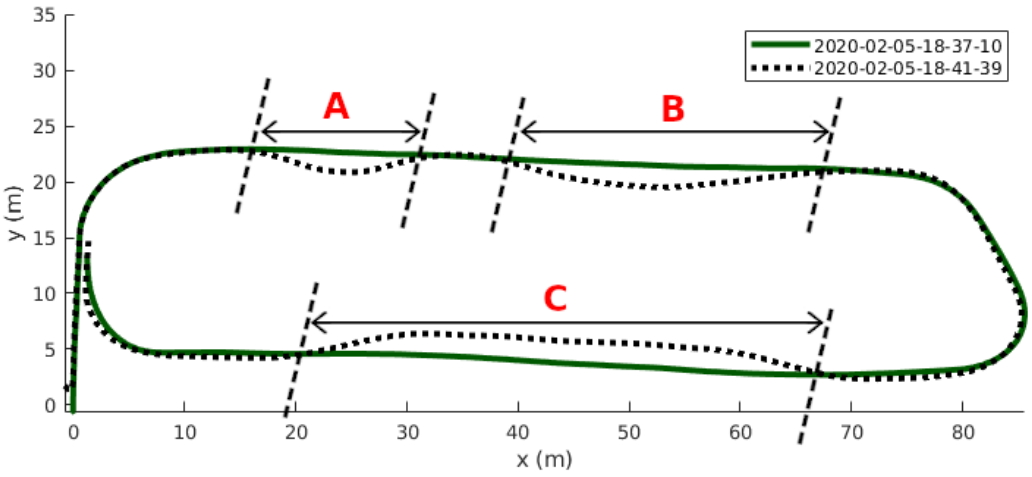
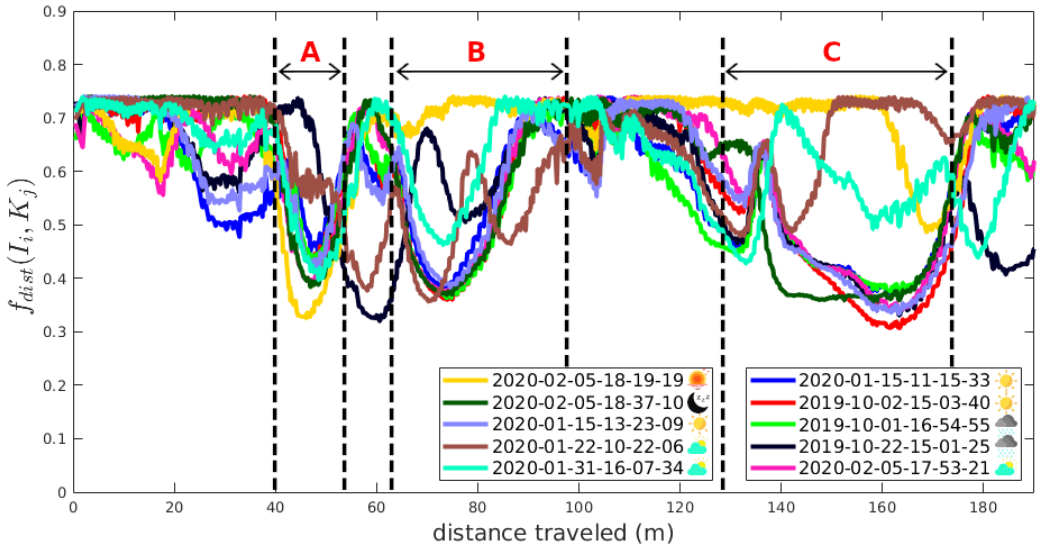
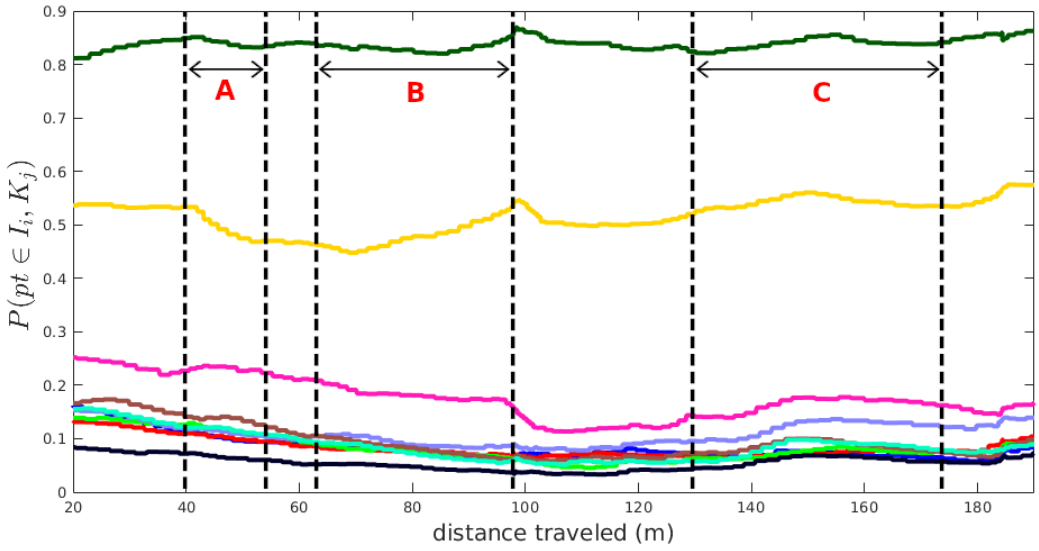


Figure 3.14: Path traveled for the sequence 2020-02-05-18-37-10 and the sequence 2020-02-05-18-41-39 . The dark green path represents the sequence 2020-02-05-18-37-10 from the map, while the dashed path represents the sequence 2020-02-05-18-41-39 from the test dataset. The two sequences have very similar environmental conditions, but with some deviations between them (segments A, B and C represent three zones with a deviation).

In Figure 3.15, we demonstrate that our ranking function τ_2 depends on both environmental conditions and geometric distance to assign scores to keyframes which helps to retrieve good keyframes for localization. In segments A, B and C, the two sequences have some lateral and angular deviations (Figure 3.14), thus, τ_1 was not able to retrieve keyframes from traversals with similar environmental condition in those zones (Sub-figure 3.15a), while our proposed ranking function τ_2 has successfully retrieved good keyframes even with the deviation (Sub-figure 3.15b). With τ_2, the night sequence is clearly privileged from the beginning to the end of the trajectory, which means that the day/night change is more important than a lateral deviation of 2 or 3m.



(a) Values assigned to keyframes by τ_1 while re-localizing with the sequence 2020-02-05-18-41-39.



(b) Values assigned to keyframes by τ_2 while re-localizing with the sequence 2020-02-05-18-41-39.

Figure 3.15: The effect of lateral and angular deviations on the two ranking functions τ_1 and τ_2. The choice of the ranking function is illustrated in sub-figures (a) and (b). Each traversal in the map is represented by a curve with a different color. For each traversal, the curve indicates the computed probability of matching the current image with the closest keyframe of that traversal. The colors of the curves in sub-figure (b) are the same as in sub-figure (a).

3.5.1.3 Detailed study on the update process of F_c

In Section 3.3.2.3, we have presented the update paradigm of the matrix F_c, however, the sequences used in the previous tests are quite short and they do not contain significant changes in the environmental condition and consequently they cannot prove the interest of this update process. For this reason, we recorded a long sequence (2019-12-05-16-43-56) which has multiple loops in the parking lot starting from 16:44 until 17:54 and it includes day, dusk and night conditions (the sequence was recorded in December and the time of sunset on that day was 17:05). In Figure 3.16, we inspect the update process of the matrix F_c. We plot the values located in the last row (or column) of the matrix F_c along a re-localization session using this long sequence on the IPLT map (built with the N =10 sequences presented in Figure 3.9). The last row (with index $N+1$) of this matrix is referring to the current traversal (the sequence 2019-12-05-16-43-56). Therefore, this row contains the similarity scores between the current images (images of the current traversal) and the closest keyframe from each one of the 10 traversals in the map. We observe that

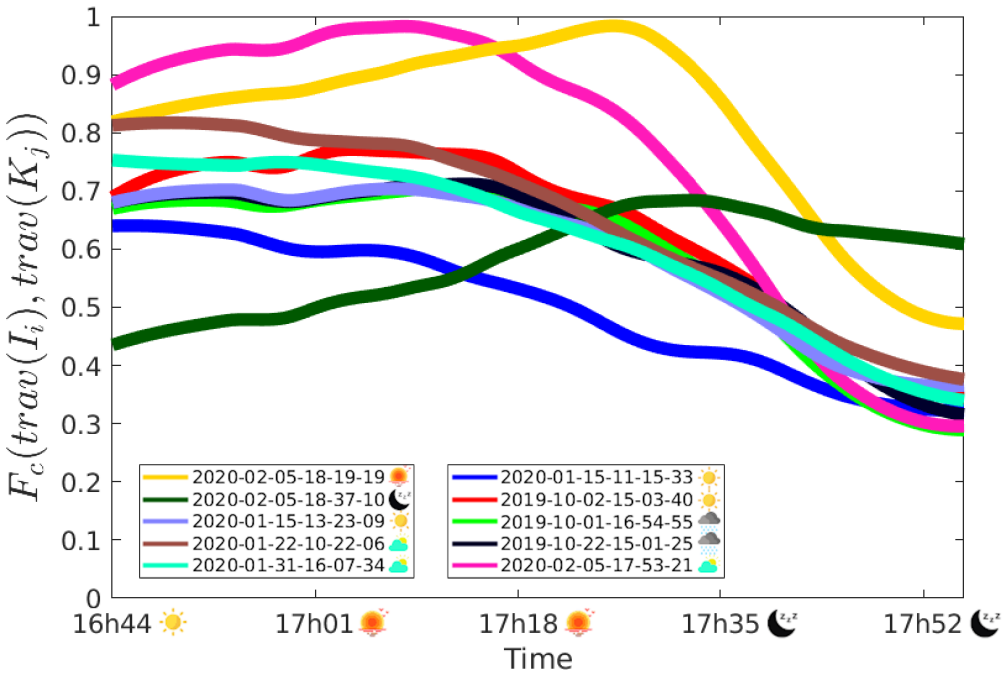
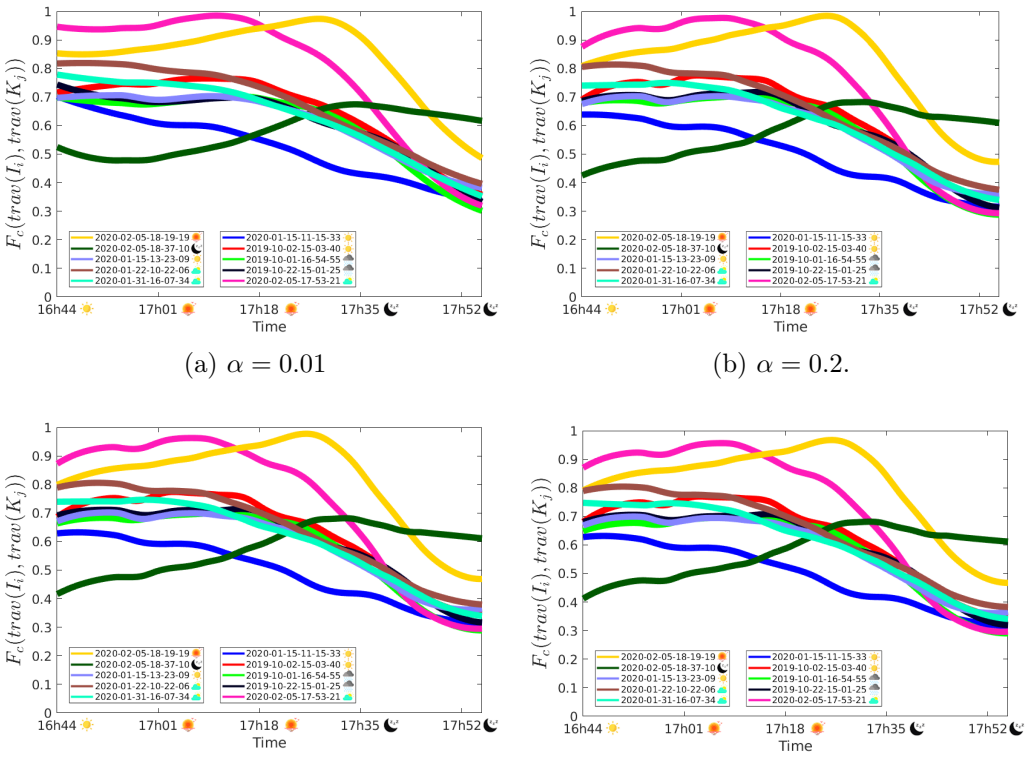


Figure 3.16: The values of F_c along a sequence of one hour and 10 minutes that includes day, dusk and night conditions. Each curve corresponds to a traversal in the map (N curves for N traversals), and the values in each curve indicate the similarity scores between the images of the current traversal and their closest keyframes belonging to the traversal represented by that curve. These curves have been smoothed for better readability.

in the start of the trajectory (before the dusk), the curve representing the traversal 2020-02-05-17-53-21 (the pink curve) is at the top. This means that this traversal has the highest similarity with the current environmental conditions. Therefore,

keyframes from this traversal will be used for localization[2]. In the dusk time, we notice a decrease in the values of the pink curve and an increase in the yellow one (2020-02-05-18-19-19). This means that F_c is indicating that there are more resemblance with the traversal represented by the yellow curve (which was recorded in the dusk). Finally, at the beginning of the night-time, we consider a notable decrease in all the curves except for the dark green one (2020-02-02-18-37-10). This is the only traversal recorded in the night, and accordingly, F_c is recommending retrieving keyframes from this traversal.

The value of the update rate α used to build Figure 3.16 is 0.1. In Figure 3.17, we present the same experience while using other values for α.



(a) $\alpha = 0.01$ (b) $\alpha = 0.2$.

(c) $\alpha = 0.5$ (d) $\alpha = 1$

Figure 3.17: Different values for the update rate α.

In (a), the update rate is very small and therefore we consider a lag in the curves, e.g., the yellow curve that corresponds to the dusk traversal (2020-02-05-18-19-19) has increased a bit late compared to the other sub-figures. This means that the ranking function is slow to respond to environmental changes with this reduced value of the update rate. Although we have significantly changed the value

[2]We would like to point out that $P(pt \in I_i, K_j)$ also takes into account the geometric distance. The fact that F_c is assigning high scores to the keyframes of a given traversal does not necessarily mean that they will be retrieved. If f_{dist} has assigned low scores to these keyframes, the ranking function may retrieve keyframes from other traversals.

of α in (b), (c) and (d), we see no noticeable difference between these three sub-figures. This similarity is due to the use of a smoothing filter before plotting the curves. Therefore, we present a raw version of these curves without any smoothing operation in Appendix B.

3.5.2 Oxford RobotCar dataset

To evaluate the performance of our approach on the Oxford RobotCar dataset, we used 12 test sequences with varying environmental conditions as illustrated in Figure 3.18.

Figure 3.18: An overview of images from the 12 test sequences of Oxford RobotCar dataset.

These 12 sequences were classified into 4 classes: sun, overcast, rain and snow. The sun class contains 5 sequences, the overcast contains 4 sequences, the rain contains 2 sequences and the snow class contains only 1 sequence.

In Figure 3.19, we are comparing the re-localization performance of τ_1 and τ_2 on the Oxford global map which incorporates 8 sequences (as presented in Figure 3.11). The comparison was performed with respect to the average number of inliers per image and localization failures per km as we did for IPLT dataset.

According to the figure, τ_2 has managed to increase the number of inliers and decrease the number of localization failures from 9 failures per km to only 0.5. In this dataset, we consider a less significant increase in terms of inliers compared to the increase observed on IPLT dataset (Figure 3.13). This is due to the fact that the IPLT dataset contains more sequences with more changes in environmental

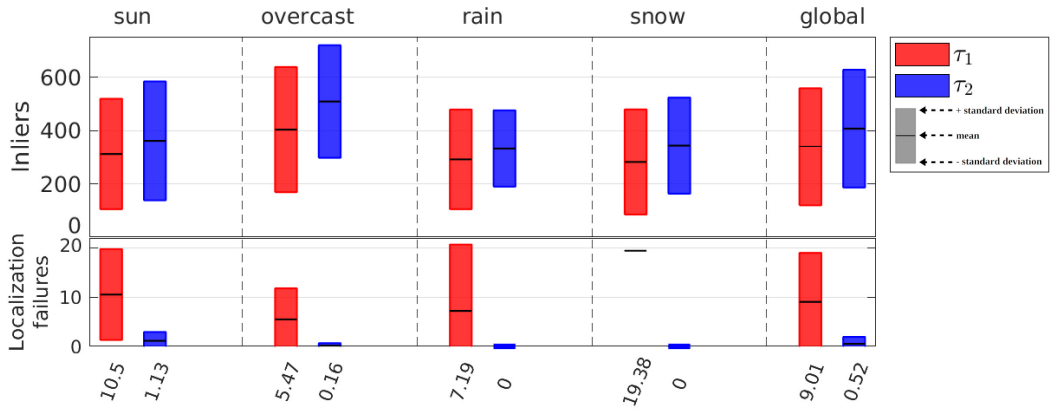


Figure 3.19: A comparison between the two ranking functions τ_1 and τ_2 on the Oxford global map using the 12 test sequences from Oxford RobotCar dataset.

conditions and it incorporates some sequences with lateral and angular deviations that seriously impact the performance of τ_1.

We also note that this dataset contains a significant number of overexposed images especially in sunny sequences [Jatzkowski 2018] (thus the 1.13 localization failures per km for τ_2 in the sun class). We present some examples of these images in Figure 3.20.

Figure 3.20: Example of extremely overexposed images that led to localization failures.

3.6 Conclusion

In this chapter, we have put forward an efficient keyframe retrieval technique that is used to retrieve keyframes based on their Euclidean distance from the current image and the environmental condition. We have proposed a probabilistic ranking function that exploits information collected during the first few meters of the trajectory to determine whether a keyframe is suitable for localization or not.

Until now, classical localization techniques did not sufficiently consider environmental changes (especially lighting changes produced by day/night transitions). This led to a serious problem in practice, especially in long-term localization applications where we consider a significant number of localization errors (such as in the case of localization with τ_1). Our experiments on two different datasets have

demonstrated that our ranking function τ_2 was able to retrieve good keyframes under different environmental conditions. This in turn helped to improve the localization performance and considerably reduce the number of localization errors (e.g., we reduced the number of localization failures in IPLT dataset by $\sim$400$\times$), which is a great improvement for all kind of applications that require an autonomous robot.

Although we did not present a comparative study with the state of the art, we proved that our approach provides efficient localization with very low computational cost after initializing the ranking function (after the first m meters of the trajectory). Our approach is designed on a keyframe-based structure, where a score is computed for each keyframe. Most state-of-the-art techniques propose landmark-based ranking functions. This means that they compute a score for each landmark in the map, which is costly in large-scale maps that incorporate a significant number of landmarks.

In this chapter, we have focused only on the re-localization process, but we can also improve further the performance by performing SLAM-LOC (re-localization + mapping) to take advantage of the landmarks of the last mapped poses. Our proposed keyframe retrieval technique is able to improve the SLAM-LOC by combining the 2D/3D constraints extracted from the retrieved keyframe with these extracted from the previously mapped poses. For feature detection and description, we have used Harris+ZNCC in our SLAM platform. As a perspective, we aim to use other more modern descriptors that can further improve the performance.

Despite this significant improvement in localization performance, landmark retrieval techniques, such as the approach presented in this chapter, are not sufficient to ensure lifelong navigation for autonomous robots. These kinds of approaches reach their limits when the size of the map increases considerably, leading to a significant increase in computation time. Therefore, we aim to solve this problem in the next chapter by proposing different map management techniques whose role is to avoid the continuous expansion of the map.

Chapter 4

Map Management

Contents

The structure of the chapter is organized as follows. Section 4.1 gives an introduction to this chapter. Section 4.2 highlights the different contributions presented in this chapter. A detailed description of our different proposed map management techniques is provided in sections 4.3, 4.4, and 4.5. Sections 4.6 and 4.7 describe the experimentation details and provide the experimental results. Finally, a conclusion to this chapter is given in Section 4.8.

4.1 Introduction

Significant efforts have been made in visual-based localization in static environments or with little change, but it is only recently that localization in dynamic environments with changing environmental conditions has been addressed. Achieving reliable lifelong navigation in such environments is one of the biggest challenges for visual SLAM. In this chapter, we are interested in real-time visual-based localization in outdoor environments for autonomous shuttles. In such applications, shuttles repeatedly traverse the same path at different times. This means that they

are very likely to be exposed to many different environmental conditions that can deteriorate the localization performance, even when they revisit familiar locations. In such a scenario, environmental changes can lead to major difficulties when associating data between the current image and the landmarks in the map. Autonomous shuttles must be able to deal with such environmental changes to ensure reliable long-term localization.

Building a map that covers all environmental conditions by continuously adding landmarks to this map can help improve localization performance under changing conditions. However, this also leads to an incessant growth of the map size, which is relative to the number of traversals performed by the shuttle (experiences). This means that localization after multiple traversals requires an immense amount of memory to store the map, as well as a high-end CPU to find matching points between the current image and the corresponding landmarks in a huge database. In other words, long-term real-time localization will be impossible after a certain number of localization sessions. Therefore, several map update strategies are developed in this chapter to prevent such cases and ensure reliable real-time long-term localization.

As in the previous chapter, the work presented in this chapter was evaluated on both IPLT and Oxford RobotCar datasets.

4.2 Contributions

Maps are a critical component in self-driving applications. High-precision maps with loads of reconstructed 3D points are created specifically for localization tasks using on-board cameras. However, due to the limited computational and memory resources of the mobile computing platform, these maps impose a significant burden on real-time processing. Therefore, map management is one of the most suitable techniques to save the processing power and memory while maintaining the accuracy of localization.

The goal of this chapter is to provide a real-time solution for long-term localization with respect to the map growth problem. For this reason, we present several map management strategies to reduce the size of the map in an offline fashion. The role of the proposed map management strategies is to remove superfluous data from the map, which leads to a significant reduction in the computational cost of the SLAM algorithm in long-term scenarios.

Other works have also addressed the map management issue in the last few years [Biber 2005], [Mühlfellner 2016], [Dymczyk 2015], [Bürki 2018], [Krajník 2016], [Halodová 2019], ... Some of these works reduce the size of the map according to a scoring policy that suggests to remove landmarks with the lowest observation rate [Mühlfellner 2016, Dymczyk 2015]. Some other works have proposed to eliminate the landmarks that were incorporated in localization failures [Bürki 2018] and have a high incorrect matching rate [Halodová 2019].

The work presented in this chapter is also addressing the same issue in different fashions. In order to ensure a long-term consistency, we have proposed three different

map management techniques with the aim of limiting the size of the map over the multiple runs of the autonomous shuttle while avoiding to degrade the localization performance:

1) We present an improvement for the Summary Maps approach proposed by Mühlfellner *et al.* [Mühlfellner 2016]. In this work, the authors are scoring landmarks according to the number of different localization sessions in which they appear, and they are removing the landmarks with the lowest scores in an offline process. In our improved version of Summary Maps, we impose a new constraint on the landmark removal, the purpose of which is to ensure the presence of a uniform number of landmarks at each traversal after performing the map summarization (i.e., compression).

2) We present a traversal-based map management technique whose role is to remove the less relevant traversals from the map. The traversal-based map management is used to limit the number of traversals in the map to a fixed number $\hat{N}$. For this reason, we use information related to the position of the sun to compute the correlation of the traversals to produce a map with $\hat{N}$ traversals with diverse environmental conditions.

3) We present another traversal-based map management technique that aims to limit the total number of traversals in the map to $\hat{N}$ traversals. Instead of using information related to the sun position, we propose to exploit the matrix F_c computed in the previous chapter, which defines the similarity values between the traversals, to determine which traversal has the highest similarity with the others to finally remove it.

In the three subsequent sections (4.3, 4.4, and 4.5), we present in more detail the methodology of the three proposed map management techniques mentioned above. In the results section, we present a comparison between these different methods, we also present a comparison with the Summary Maps approach proposed by Mühlfellner *et al.* [Mühlfellner 2016].

4.3 Map management with improved summary maps

In this section, we present our map management approach which is extending the works proposed by Mühlfellner *et al.* [Mühlfellner 2016] named Summary Maps (and denoted by SM) that we have re-implemented on our mapping framework. Mühlfellner *et al.* [Mühlfellner 2016] have defined a landmark scoring policy, which assigns scores to landmarks according to the number of different localization sessions in which they appear. Therefore, landmarks which were observed in multiple sessions are assigned with high scores by the scoring policy and deemed valuable for localization as a result. This means that landmarks with low scores are deemed useless and irrelevant for localization. Therefore, the Summary Maps approach consists in using these scores to summarize the map in an offline process carried out after each localization session to remove the most insignificant landmarks.

The strategy of the scoring function proposed by Mühlfellner *et al.* is illustrated in Figure 4.1. The figure presents an example of localization and shows the landmarks observed in two different sessions, i and j.

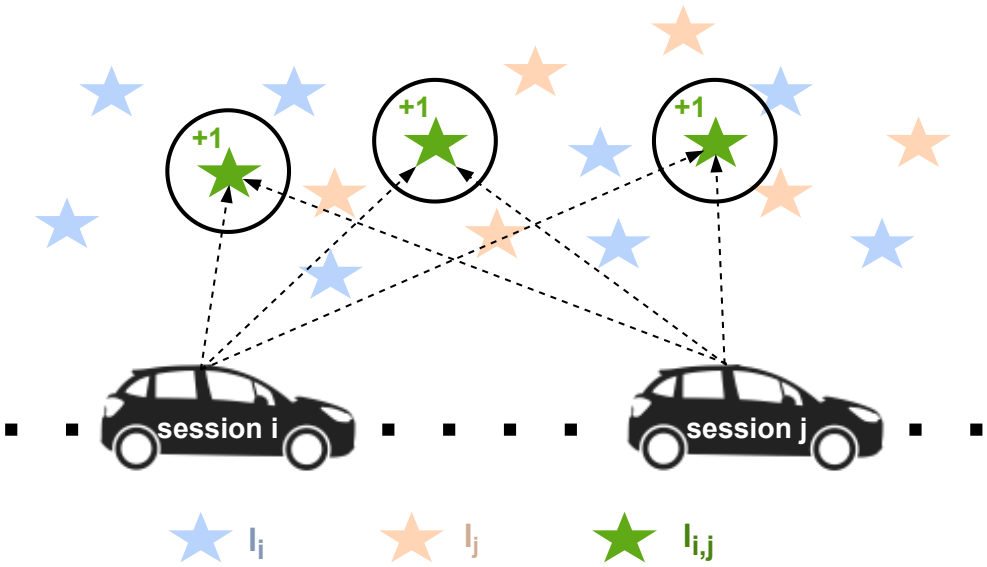


Figure 4.1: The landmark scoring policy proposed by Mühlfellner *et al.*. The figure illustrates a localization example where some landmarks ($l_{i,j}$) were observed in two different sessions (session i and session j) and accordingly, their scores were increased by the scoring policy.

A major limitation of this approach is the fact that landmarks in rarely visited areas are assigned with low scores since they are infrequently observed, which means that they are expected to be removed in the map summarization process. Dymczyk *et al.* [Dymczyk 2015] referred to this problem as the bias towards regions that were more frequently visited, and their works were devoted to solve this problem. They upgraded the scoring policy proposed in [Mühlfellner 2016] to take into account the expected number of trajectories to observe a landmark.

In our work, we are not interested with the improvements proposed by Dymczyk *et al.* since we evaluate our approach on two datasets where all the sequences are travels of the same path. Therefore, we are not concerned by the problem of bias towards regions that were more frequently visited addressed in [Dymczyk 2015]. However, we have identified another limitation of the Summary Maps approach which is somehow similar to the previously mentioned limitation and it can be defined as a problem of bias towards more experienced environmental conditions. This bias occurs when a set of sequences with resembling environmental conditions are mixed with a sequence with an odd environmental condition in the same map (e.g. a set of day-time sequences are included in the same map with one night-time sequence). In this case, landmarks observed under the odd environmental condition are assigned with low scores by the scoring policy since they were not observed as frequently as the landmarks from the other sequences, which means that

they will be filtered out in the map summarization step.

In Table 4.1, we present an example of bias towards more experienced environmental conditions that has been occurring when summarizing the IPLT map from the previous chapter with the re-implemented version of the Summary Maps. This map was built using the 10 sequences presented in Figure 3.9 and it incorporates only one night sequence (the odd sequence). In this table, we compress the map with different compression ratios (1, 1.5, 2, 3, 5 and 10) using the Summary Maps approach. Compressing the map with a compression ratio r means that we sort all the landmarks based on their scores assigned by the scoring policy, then we remove the $100 * (1 - 1/r)$ percent of the landmarks with the lowest scores (e.g., we remove the 50% lowest scored landmarks for $r = 2$).

Table 4.1: Bias towards more experienced environmental conditions in Summary Maps.

Traversal	Compression ratio r					
	1 (no compression)	1.5	2	3	5	10
1	140,524	113,922	100,371	75,087	50,304	20,817
2	127,687	93,539	72,205	48,527	28,803	14,071
3	149,065	83,821	52,025	34,912	18,913	9,324
4	140,900	72,955	37,769	26,157	15,868	8,946
5	122,122	97,989	86,896	55,751	29,495	14,022
6	124,643	89,807	76,360	44,106	18,436	9,495
7	72,044	41,333	28,777	16,310	5,092	2,709
8	116,091	67,204	44,438	32,717	24,424	12,825
9	127,972	96,797	78,262	51,610	34,322	16,692
10	143,640	85,758	55,241	36,385	27,280	17,567
total	1,264,688	843,125	632,344	421,562	252,937	126,469

The table shows the number of landmarks observed on each traversal after compressing the map with the Summary Maps approach using different compression ratios. The second column ($r = 1$) represent the initial map with no compression (100*(1-1/1) = 0%). The last row represents the total number of landmarks on each map.

According to the table, a considerable number of landmarks belonging to the 7^{th} traversal (which corresponds to the night sequence 2020-02-05-18-37-10) were removed after the map was compressed. This proves that the Summary Maps approach omits landmarks observed in rarely experienced environmental conditions. This can cause a serious problem when localizing with night sequences on the compressed map, as most of the night landmarks have been filtered.

Therefore, we propose an improvement for the Summary Maps in this section. Our proposed technique consists in imposing a constraint that ensures the distribution of a uniform number of landmarks across all traversals after the map summarization (compression). Therefore, we call our technique the Uniform Summary Maps (and denote it by USM). The procedure followed to ensure a uniform distribution of landmarks over all traversals is explained in Algorithm 3.

Algorithm 3: Uniform Summary Maps

Parameters:
The total number of landmarks to remove: n_{tot}
The number of traversals in the map: N
Steps:
repeat
    Compute the number of landmarks n^l_{land} observed on each traversal l, with $l \in [1, N]$
    Sort the n^l_{land} landmarks of each traversal l according to the scoring policy
    Compute the highest number of landmarks: $n_{\text{max}} \leftarrow \max(\{n^1_{land}, \ldots, n^N_{land}\})$
    Find the set of traversals $\mathcal{S} = \{L^1_{\text{max}}, \ldots, L^s_{\text{max}}\}$ having n_{max} landmarks
    Find the traversal L_{smax} having the second-highest number of landmarks:
    $$n_{\text{smax}} \leftarrow \max(\{n^1_{land}, \ldots, n^N_{land}\} \setminus \{n_{\text{max}}\})$$
    $n_{\text{diff}} \leftarrow n_{\text{max}} - n_{\text{smax}}$
    Compute the number of landmarks to remove in the current iteration:
    $n_{\text{rem}} \leftarrow \min(n_{\text{diff}} * |\mathcal{S}|, n_{\text{tot}})$ /* the function min is used to make sure that we do not remove more than n_{tot} landmarks */
    for each traversal $L_{\text{max}} \in \mathcal{S}$ **do**
        Remove the lowest scored $\frac{n_{\text{rem}}}{|\mathcal{S}|}$ landmarks from traversal L_{max}
    end for
    $n_{\text{tot}} \leftarrow n_{\text{tot}} - n_{\text{rem}}$
until $n_{\text{tot}} \leq 0$

To better understand the pseudo-code presented in Algorithm 3, we present an execution example in Figure 4.2. In the example, a map with $N = 4$ traversals is compressed using our algorithm. n^l_{land} denotes the number of landmarks belonging to the traversal l. We run our algorithm to remove $n_{\text{tot}} = 260$ landmarks from the map while maintaining a uniform distribution of landmarks across the N traversals. In the first iteration, the algorithm has computed the number of landmarks n_{rem} to remove from the 4^{th} traversal (the traversal with the highest number of landmarks). In the second iteration, we note that there are two traversals with the highest number of landmarks: $\mathcal{S} = \{2, 4\}$. Accordingly, the algorithm has computed the number of landmarks to remove n_{rem} and removed $n_{\text{rem}}/|\mathcal{S}|$ landmarks from each traversal in $\mathcal{S}$. In the third iteration, the set $\mathcal{S}$ contains 3 traversals. The algorithm has computed the number of landmarks to be removed in this iteration $n_{\text{diff}} * |\mathcal{S}| = 150$. However, this number is larger than the total number of landmarks to be removed $n_{\text{tot}} = 120$. Therefore, the statement $\min(n_{\text{diff}} * |\mathcal{S}|, n_{\text{tot}})$ was placed to ensure that our algorithm does not remove more than n_{tot} landmarks.

As we presented in Table 4.1 the results of compressing the map with the Summary Maps approach, Table 4.2 shows the result of compressing the map with our improved version of Summary Maps. The table shows that after compressing the

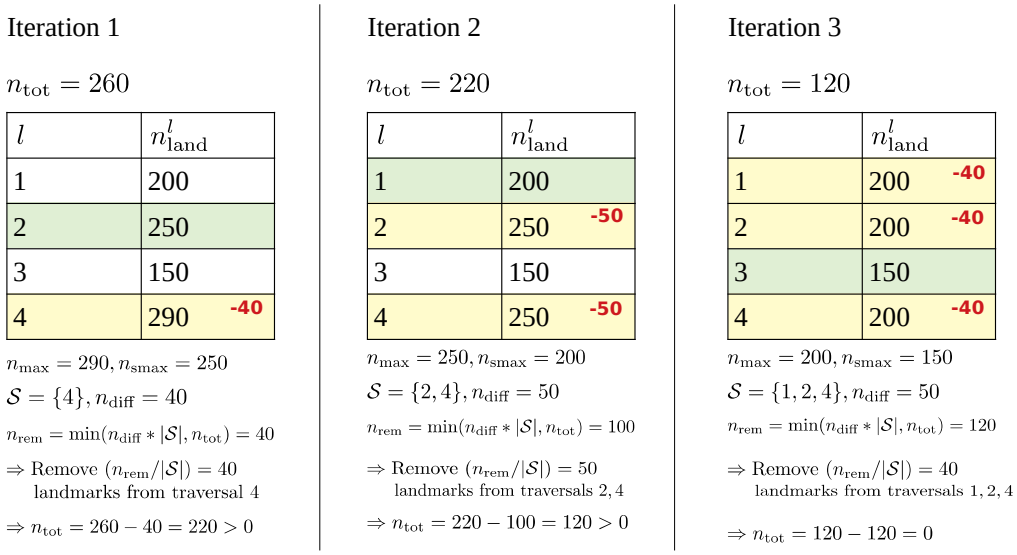


Figure 4.2: An example illustrating the execution mechanism of our proposed algorithm.

map with different compression ratios, the number of landmarks stays uniform across the different traversals.

Table 4.2: Compressing the map with the Uniform Summary Maps.

Traversal	Compression ratio r					
	1 (no compression)	1.5	2	3	5	10
1	140,524	85,676	63,235	42,157	25,294	12,647
2	127,687	85,676	63,235	42,157	25,294	12,647
3	149,065	85,676	63,235	42,157	25,294	12,647
4	140,900	85,676	63,235	42,157	25,294	12,647
5	122,122	85,676	63,235	42,157	25,294	12,647
6	124,643	85,676	63,235	42,157	25,294	12,647
7	72,044	72,044	63,235	42,157	25,294	12,647
8	116,091	85,676	63,235	42,157	25,294	12,647
9	127,972	85,676	63,235	42,157	25,294	12,647
10	143,640	85,676	63,235	42,157	25,294	12,647
total	1,264,688	843,128	632,350	421,570	252,940	126,470

The Table 4.2 demonstrates that the imposed constraint helped to avoid the omission of the night traversal in this example. However, this technique suffers when a significant number of traversals are included in the map. Therefore, in the following two sections, we present other map management techniques that are able to overcome this limitation.

4.4 Map management based on solar information

The work presented in this section is based on a contribution that was submitted to the 14th International Conference on Intelligent Robotics and Applications (ICIRA 2021) and which addresses our proposed map update strategy using solar information. In this approach we aim to maintain a reliable map with a fixed size throughout the frequent traversals.

In a previous work done at Institut Pascal, Royer *et al.* [Royer 2016] employed an autonomous shuttle for three months, totaling nearly 1500 km of autonomous travel on an industrial site. During this operation, the authors have experienced some difficulties for long-term navigation. One of the most challenging difficulties identified in this work is lighting changes over the day, where the tests demonstrated that variations in lighting (caused by changes in the direction of the sunlight and the sun elevation) have more impact than long-term changes of the environment on the localization performance. This has motivated us to exploit information related to the sun position to design a map management approach with the purpose of preventing the continuous growth of the map. The sun position information are used to determine which traversals to keep and which ones to remove from the map. Traversals with correlated sun position coordinates are considered to have similar environmental conditions. Thus, our approach consists, in a first time, in computing the sun positions related to all the traversals in the map, then, in exploiting all those computed values to classify this map into relevant and irrelevant traversals to finally produce a compressed map that incorporates a minimum number of traversals with diverse environmental conditions.

Therefore, the core idea of our approach is to produce a map that incorporates a minimum number of traversals ($\hat{N}$) and able to operate in different environmental conditions. $\hat{N}$ is the number of traversals to be maintained in the map after carrying out the map management. Different values of $\hat{N}$ were chosen in the experiments section to evaluate the efficiency of our approach in different cases.

The methodology of our approach is described in the Figure 4.3. After each localization session, we test whether the number of traversals in the map (N) is greater than the predefined number $\hat{N}$. If so, our map management algorithm will be executed offline to reduce the size of the map. $N > \hat{N}$ means that the map contains an extra traversal ($N = \hat{N} + 1$), which implies that it occupies additional memory space. In this case, our approach has to act offline to bound the size of the map, i.e., it has to decide which traversal has more similarity to the others to eventually remove it.

As explained in Figure 4.3, this mechanism incorporates two main parts, similarity computation and classification. In the similarity computation part, we exploit information related to the sun position at each acquisition time and associate the traversals with the corresponding environmental conditions. This finally allows us to calculate the similarity between the traversals with respect to the associated environmental conditions. In the classification part, we use the previously computed similarity information to classify the traversals according to their importance and

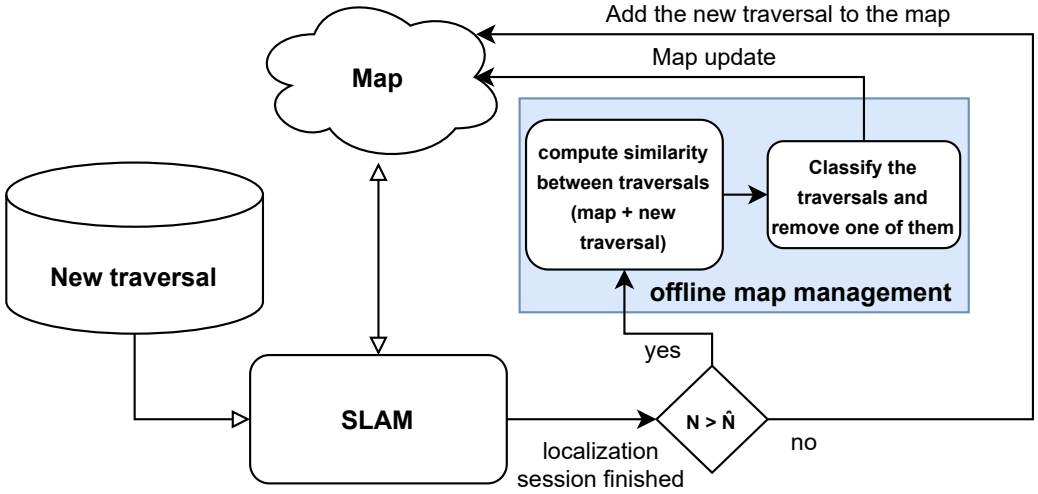


Figure 4.3: A diagram illustrating the map management process proposed in this section. A new traversal is localized and added to the existing map. After completing the re-localization and mapping with SLAM, the map management algorithm is run offline to check whether the map needs to be compressed or not (if $N > \hat{N}$). If the answer is yes, the algorithm must act in two parts. First, it must compute the similarities between the N traversals in the map ($N = \hat{N} + 1$). Second, it uses these similarity information to classify all the traversals and select the one that needs to be removed.

remove the less important ones.

In sub-sections 4.4.1 and 4.4.2, we present in details the methodology followed in both similarity computation and classification parts.

4.4.1 Similarity computation

In this part, we take advantage of the sun position to compare resemblance between the traversals in the map in order to determine which ones to keep in the map and which ones to remove. For each traversal in the map, we use the Astronomical Almanac's algorithm [Michalsky 1988] to compute the corresponding sun spherical coordinates: the Solar Elevation Angle (el) and the Solar Azimuth Angle (az) (see Figure 4.4). This algorithm takes as inputs the acquisition date/time and GPS latitude/longitude coordinates of each corresponding traversal. These inputs (date/time + GPS coordinates) are captured at the first frame of each traversal. This can be valid only when using short sequences like the case of the sequences used in evaluation part of this chapter, where the IPLT dataset incorporates $\sim$ 200 m length sequences, each one of them was recorded over $\sim$ 2 minutes and the Oxford RobotCar dataset incorporates $\sim$ 1.6 km sequences recorded over $\sim$ 5 minutes. This means that there will be no significant gap in the sun position between the start and the end of the traversal. We note that our technique can still be applied using longer sequences by rerunning the Astronomical Almanac's algorithm with updated inputs every few minutes or few kilometers.

After computing all the sun positions associated to the N traversals in the map,

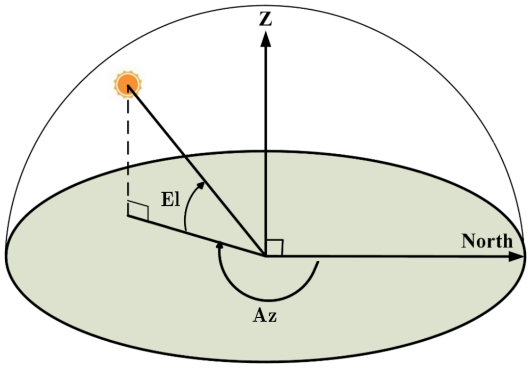


Figure 4.4: Solar Elevation Angle and Solar Azimuth Angle [Cheng 2019].

we generate a similarity matrix D with the shape of $[N \times N]$. D is a 2D symmetric distance matrix containing the distances, taken pairwise, between the computed sun positions of the N traversals.

In this section, we tested two variants of the similarity matrix D. The first variant, D_{el}, is built by computing the distances between the elevation angles of the traversals:

$$D_{el}(i,j) = dist(el_i, el_j), \quad \forall i,j \in [1,N] \tag{4.1}$$

Where the function *dist* computes the angular distance between two solar elevation angles. The second variant, D_{az_el}, is built using both the elevation and azimuth angles. For each traversal i, we compute the Cartesian vector from the spherical coordinates:

$$\mathbf{u_i} = \begin{pmatrix} \cos az_i \cos el_i \\ \sin az_i \cos el_i \\ \sin el_i \end{pmatrix}, \quad \forall i \in [1,N] \tag{4.2}$$

After that, we compute the distance matrix D_{az_el} by calculating the angle between each pair of the N Cartesian vectors:

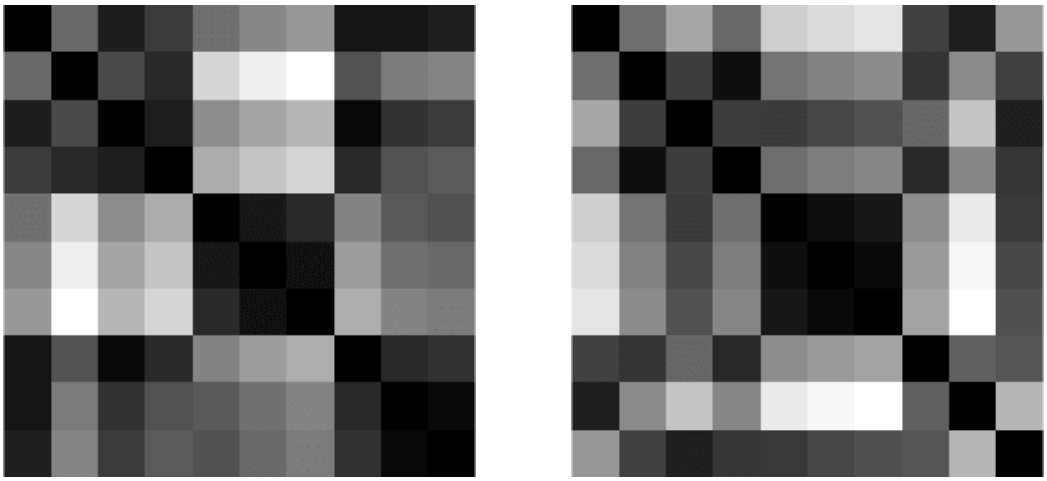
$$D_{az\_el}(i,j) = |\arccos(\mathbf{u_i} \cdot \mathbf{u_j})|, \quad \forall i,j \in [1,N] \tag{4.3}$$

Figure 4.5 illustrates an example of the similarity matrices D_{el} and D_{az_el}. Both matrices were built using the same map that contains 10 traversals (the IPLT map from the previous chapter).

4.4.2 Classification and traversal removal

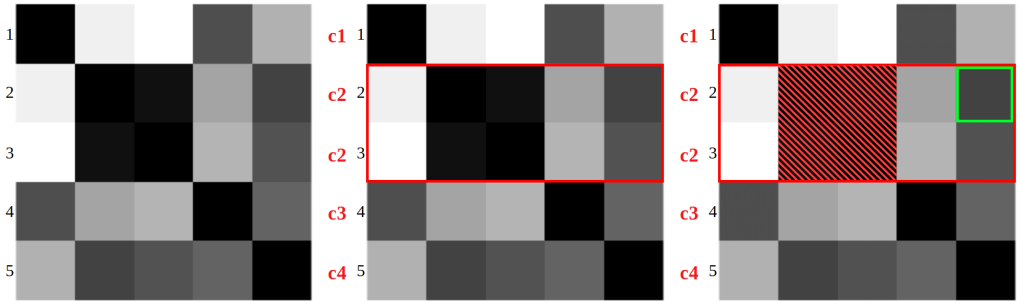
In this part, we exploit the matrix D (D_{el} or D_{az_el}) built in the previous step to find out which traversal has to be removed. A hierarchical clustering [Szekely 2005] algorithm is used to classify the matrix D in order to select the traversal to remove as explained in Figure 4.6.

Figure 4.6 shows the different steps of the classification and the traversal removal:



(a) D_{el} (b) D_{az_el}

Figure 4.5: Example of matrices D_{el} and D_{az_el} built with a map containing $N = 10$ traversals. The values of the matrices were normalized between 0 and 1, black color refers to 0 (high similarity) and white color refers to 1 (low similarity).



(a) D (b) Clustering D (c) searching for min

Figure 4.6: Steps for classification and the selection of the traversal to remove.

(a) D is the distance matrix obtained from the previous sub-section, we apply this method the same way on D_{el} and on D_{az_el}. In this example, D refers to matrix D_{az_el} obtained with a map containing $N = 5$ traversals.

(b) We classify D into $\hat{N}$ (4 in this example) classes using the hierarchical clustering algorithm. This results in the two traversals i and j that have the greatest similarity being placed in the same class ($i = 2$ and $j = 3$ in this example).

(c) In this step, we want to remove a traversal from the map while keeping the map as diverse as possible. To do this, we need to remove either the traversal i or the traversal j. Therefore, we search which of them has more similarity to the other traversals by looking for the minimum value in the i^{th} and j^{th} rows of the matrix while ignoring the diagonal and the elements with coordinates

(i, j) and (j, i) (the shaded area in the figure). After finding the minimum, we remove its associated traversal (traversal 2 in the example).

Lighting changes produced by day-night transitions can result in an enormous visual gap between images recorded in the same place but at different times of the day [Shoman 2020]. Localization under night condition can be considered as a particular case since there are no lights provided by the sun after astronomical twilight. Therefore, streetlights are almost the only source of light, which means that all sequences recorded during the night are visually similar. This means that it is pointless to keep more than one night-time traversal in the map, and on the other hand, removing all night-time traversals can lead to a serious degradation of the localization performance for night-time sequences. This is demonstrated in Figure 4.7 where we have evaluated the performance of localization with the sequences of the "night" class from IPLT dataset on 7 different maps (the "night" class incorporates 11 sequences as detailed in the previous chapter). Each of these maps contains a single traversal corresponding to a night traversal, as indicated in the legend of the figure. According to this figure, the localization performance is almost equivalent on all these maps[1]. This means that no matter which night traversal is included in the map, the localization performance on night sequences will be reliable. This can be explained by the fact that urban lighting comes from fixed sources during the night, unlike natural sunlight, whose direction and characteristics changes throughout the day.

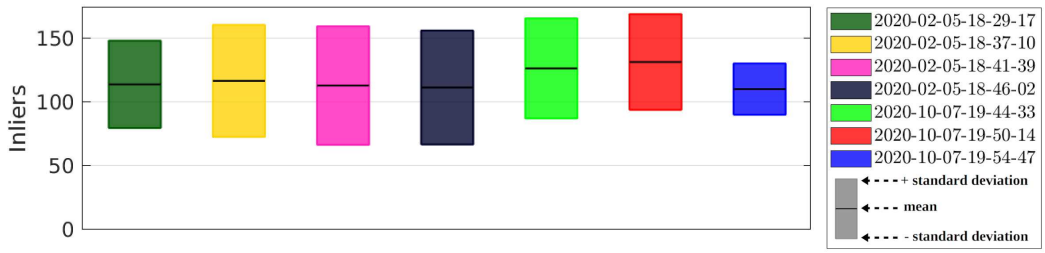


Figure 4.7: Localization performance on 7 different maps containing night traversals. Each map was built using only one sequence (a night sequence) which is indicated on the legend of the figure. We have not experienced any localization failure when localizing on these maps.

For this reason, we impose a new constraint on our algorithm: the traversal with the lowest sun elevation angle is not nominated as the traversal to be removed (the night traversals have a negative sun elevation angle), any other night traversal will be removed. This ensures that the produced map will incorporate only one night traversal.

In the next section, we will present another map management technique that is able to take into account other factors, such as rain or overcast, to remove irrelevant

[1]In the sequence 2020-10-07-19-54-47, the shuttle was driving on the left lane which is not the case for the other sequences. This explains why the localization on the map generated by this sequence has produced slightly less inliers than the other maps in the figure.

traversals from the map. Moreover, it does not require any imposed constraint to avoid removing night sequences.

4.5 Map management based on environmental conditions similarity (using F_c matrix)

The work presented in this section is based on a contribution that was submitted to a special issue on Machine Vision Theory and Applications for Cyber Physical Systems of the Multimedia Tools and Applications journal (MTAP). In this work, we propose a map management technique that is based on the F_c matrix defined in the previous chapter. We extended the mapping system of the aforementioned framework to adapt its maps to achieve long-term operations. Like in the previous section, the idea is to reduce the size of the map while keeping it as diverse as possible to cover the maximum number of different environmental conditions.

The principle of this approach is very similar to that proposed in the previous section. The main difference between them is the data used to compute the similarity between the traversals. Therefore, the map management procedure proposed in this section consists in limiting the total number of traversals in the map (N) to a predefined number of traversals $\hat{N}$. When the number of traversals in the map N exceeds the predefined number $\hat{N}$ ($N = \hat{N} + 1$), our algorithm must choose a traversal to remove from the map. The choice depends mainly on the matrix F_c. However, as explained in Section 3.3.2.3 from the previous chapter, F_c is regularly updated along the traversal. Therefore, we compute its average value $\bar{F}_c$ at the end of each SLAM session:

$$\bar{F}_c = \frac{1}{n}\sum_{i=1}^{n} F_c^i \tag{4.4}$$

where n is the total number of images in the trajectory and F_c^i is the value of the matrix F_c at iteration i.

Our approach consists in using this matrix $\bar{F}_c$ to select the traversal which has the most resemblance to the others and then removing it from the map.

In Figure 4.8, we present an example of the matrix $\bar{F}_c$ computed on a map that contains 10 traversals (the IPLT map from previous chapter). The values of the matrix are between 0 and $1/\max(f_{dist})$. The value $1/\max(f_{dist})$ corresponds to the score assigned by F_c to two identical images. This can be theoretically deduced from the Equation (4.5) that was defined in Chapter 3:

$$F_c(trav(I_i), trav(K_j)) = \frac{P(pt \in I_i, K_j)}{f_{dist}(I_i, K_j)} \tag{4.5}$$

Assuming that the two images I_i and K_j are identical, the value of $P(pt \in I_i, K_j)$ is 1 and the value of $f_{dist}(I_i, K_j)$ is $\max(f_{dist})$, hence the value $1/\max(f_{dist})$.

To perform hierarchical clustering on the matrix $\bar{F}_c$, we first need to convert it into a distance matrix. Since $\bar{F}_c$ is symmetric and its values are included between 0

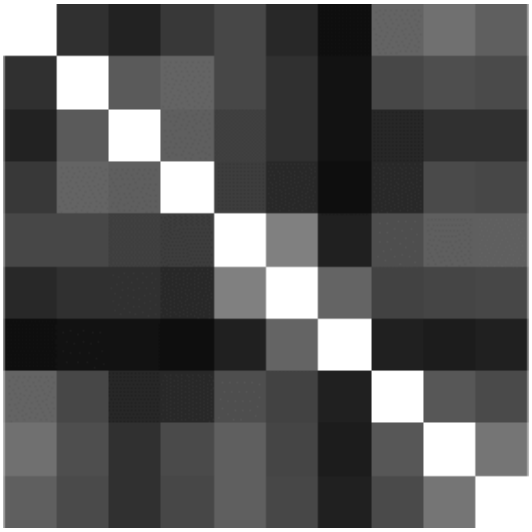


Figure 4.8: The matrix $\bar{F}_c$ computed on the IPLT map. The values of the matrix are between 0 and $1/\max(f_{dist})$, black color refers to 0 (low similarity) and white color refers to $1/\max(f_{dist})$ (high similarity).

and $1/max(f_{dist})$, we can get a distance matrix by normalizing it as follows:

$$D = J - \bar{F}_c \cdot max(f_{dist}) \tag{4.6}$$

where J is a matrix of all ones with the same dimension as $\bar{F}_c$ ($[N \times N]$). The resulting matrix D is a distance matrix with zero diagonal and all its values belong to $[0, 1]$.

We used the hierarchical clustering algorithm to classify the normalized matrix D exactly the same way as we did with D_{el} and D_{az_el} in the previous section.

4.6 Experiments

We have tested our map management techniques on the same datasets used in the previous chapter. We also have used the same division for the mapping sequences and test sequences (10 mapping sequences and 93 test sequences for IPLT dataset, and 8 mapping sequences and 12 test sequences for Oxford RobotCar dataset).

We evaluate the work done in this chapter in two different scenarios. In both scenarios, we aim to limit the number of traversals used to build the global map to a predefined number $\hat{N}$. We call an $\hat{N}$-session map a map constituted from $\hat{N}$ traversals. In other words, we aim to choose the best $\hat{N}$-session map from a total of N traversals.

To evaluate the efficiency of the maps generated with our approaches, we first proceed to build all possible maps constituted of $\hat{N}$ traversals. This will result in

obtaining n different $\hat{N}$-session maps:

$$n = {}_{N}C_{\hat{N}} = \binom{N}{\hat{N}} = \frac{N!}{(N-\hat{N})!\,\hat{N}!} \tag{4.7}$$

However, this paradigm is only feasible with the map management with solar information (denoted by MMSE when using only solar elevation and MMSAE when using both solar azimuth and elevation as described in Section 4.4) and the map management with the similarity matrix F_c (denoted by MMFc) since both of them are traversal-based map management techniques, where less relevant traversals are removed. The concept of the Summary Maps (SM) and the Uniform Summary Maps (USM) is quite different since in both approaches, the map summarization is performed in a landmark-based way, where less relevant landmarks are removed. Therefore, to fairly compare the SM and the USM with the other proposed approaches (MMSE, MMSAE and MMFc), we search for the compression ratio $\hat{r}$ that generates a map with approximately the same number of landmarks incorporated in a map containing $\hat{N}$ traversals:

$$\hat{r} = \frac{n_{\text{land}}(M_N)}{n_{\text{land}}(M_{\hat{N}})} \tag{4.8}$$

where M_X is a map containing X traversals and the term $n_{\text{land}}(M)$ denotes the number of landmarks contained in the map M.

This way, we make sure that the maps generated with the SM, USM, MMSE, MMSAE and MMFc techniques are equivalent in terms of number of landmarks (and respectively in terms of memory occupancy), which guarantees a fair comparison between the localization performance after compressing the map.

4.6.1 Evaluation scenario 1

This scenario is similar to the evaluation scenario from the previous chapter. We use the mapping sequences to generate a compressed map for each map management technique. Then we use the test sequences to evaluate the localization performance on these compressed maps.

The goal is to determine if there exists a small map that can be used to localize every test sequence without significantly increasing the failure rate of the localization compared to the map which contains all the traversals. It will also allow to rank all the possible $\hat{N}$-session maps, which means that we can position the result of our approach among all the n possibilities. Thus we will be able to compare the result of our incremental approach with the global optimal $\hat{N}$-session map. Of course, this is possible only for an evaluation purpose, the exhaustive search of the global optimum is intractable for real use cases with hundreds of traversals like the case in the second evaluation scenario. Therefore, we perform this exhaustive search only in the first evaluation scenario.

In order to compute localization error, the n generated $\hat{N}$-session maps are

evaluated by performing re-localization with a selection of sequences from the test dataset. We have selected a set of sequences with different environmental conditions from the test sequences (30 sequences were selected from the IPLT test dataset and all the 12 test sequences from the Oxford RobotCar dataset). Since we do not possess the ground-truth poses to compute the error in the IPLT dataset, for each test sequence, we create the corresponding ground-truth poses by performing re-localization on the global map while retrieving from the map the 4 keyframes which have the highest probability of matching $P(pt \in I_i, K_j)$. The landmarks of these 4 retrieved keyframes are matched to the current image to calculate the ground-truth pose using PnP+RANSAC. The Oxford RobotCar dataset is providing the RTK ground-truth poses [Maddern 2020]. However, we deplore a lack of ground-truth poses for some of the sequences used in this work (either a total or a partial lack, e.g., 2014-11-18-13-20-12, 2014-12-16-09-14-09, 2015-02-03-08-45-10, etc.). Also, the ground-truth poses provided by this dataset comprehend a large error margin (∼15 cm in latitude and longitude) which is an order of magnitude higher than the local accuracy usually given by visual localization in such applications. Therefore, we decided to proceed in the same way for ground-truth calculation as we did for the IPLT dataset.

To compute the localization error, we perform re-localization with the test sequences on all the n $\hat{N}$-session maps. For each $\hat{N}$-session map, we compute the average of localization errors of all the test sequences. The localization errors correspond to the Euclidean distance between the ground-truth poses calculated on the global map and the poses computed while re-localizing on the $\hat{N}$-session maps.

Afterwards, we proceed to an incremental search step. In this step, we employ each of our map management approaches to build a map from $\hat{N}$ traversals among N traversals (feasible only with the traversal-based techniques: MMSE, MMSAE and MMFc). Considering the fact that the order in which the traversals are added to the map can influence the resulting map, we test our approach with $100,000$ different permutations in the order of the N traversals. This results in generating several $\hat{N}$-session maps when we run each map management technique (MMSE, MMSAE or MMFc) $100,000$ times with different permutations in the order of the N traversals. Thereby, we pick the most reproduced $\hat{N}$-session map (denoted by M^*) and use it to evaluate the localization performance with the test sequences.

Since both of SM and USM are not traversal-based techniques, we just proceed to a map compression after the end of each mapping session with a compression ratio $\hat{r}$ computed with Equation (4.8).

4.6.2 Evaluation scenario 2

This scenario is modeled on the real use case of autonomous shuttles. This means that we aim to find the best representation of the environment among all the encountered sequences (mapping and test sequences). The map is built in an incremental way where we perform SLAM-LOC (re-localization + mapping) with each sequence of the mapping and test datasets. After each localization session, the map manage-

ment is performed to ensure that the size of the map remains limited. The quality of localization is measured along all the sequences (unlike in scenario 1 where the quality of localization is measured when re-localizing with the test sequences).

We have evaluated this scenario only with the IPLT dataset since it incorporates a significant number of sequences (mapping + test sequences = 103 sequences), allowing to properly simulate a real use case of an autonomous shuttle. This is not the case for Oxford RobotCar dataset which includes only 20 sequences.

During each mapping session, we use our keyframe retrieval technique (introduced in the previous chapter) to extract landmarks from the two most relevant keyframes in the map. These retrieved landmarks are first combined with the landmarks extracted from the most recently mapped keyframe (from the current mapping session), and then matched with the current image to compute the current vehicle pose. We retrieve two keyframes instead of only one, as in evaluation scenario 1, to obtain a more accurate pose and reduce the cumulative mapping drift as much as possible. The mapping drift accumulates progressively and becomes significant after a large number of mapping sessions (described as a photocopy of a photocopy in [MacTavish 2017]). Therefore, data association in this evaluation scenario is performed between the current image and the two retrieved keyframes according to τ_2 and the most recently mapped keyframe.

4.7 Results

In this section, we show the efficiency of the different proposed map management approaches in the two evaluation scenarios described above. As in the keyframes retrieval approach, we test our map management with two datasets.

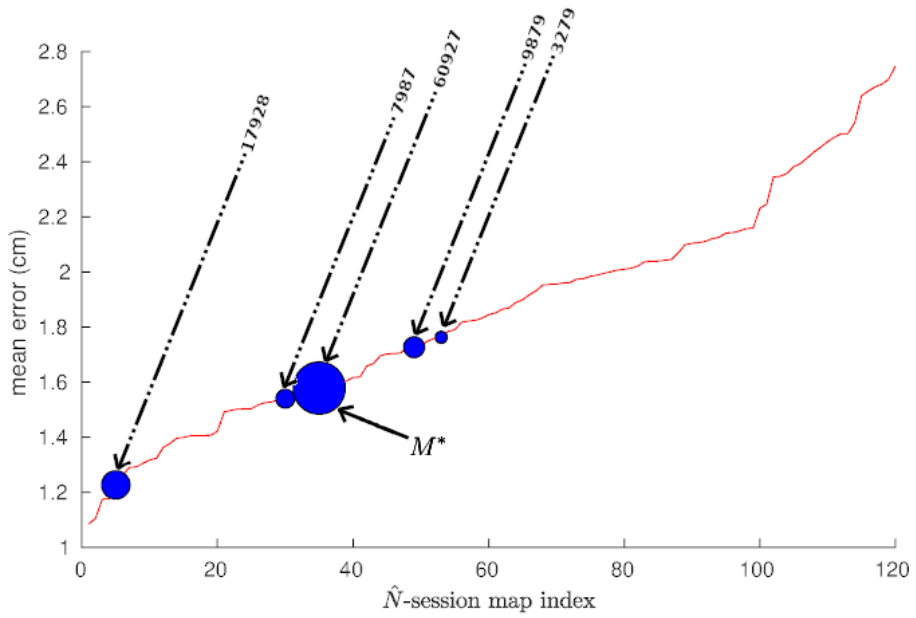
4.7.1 Evaluation scenario 1

4.7.1.1 IPLT dataset

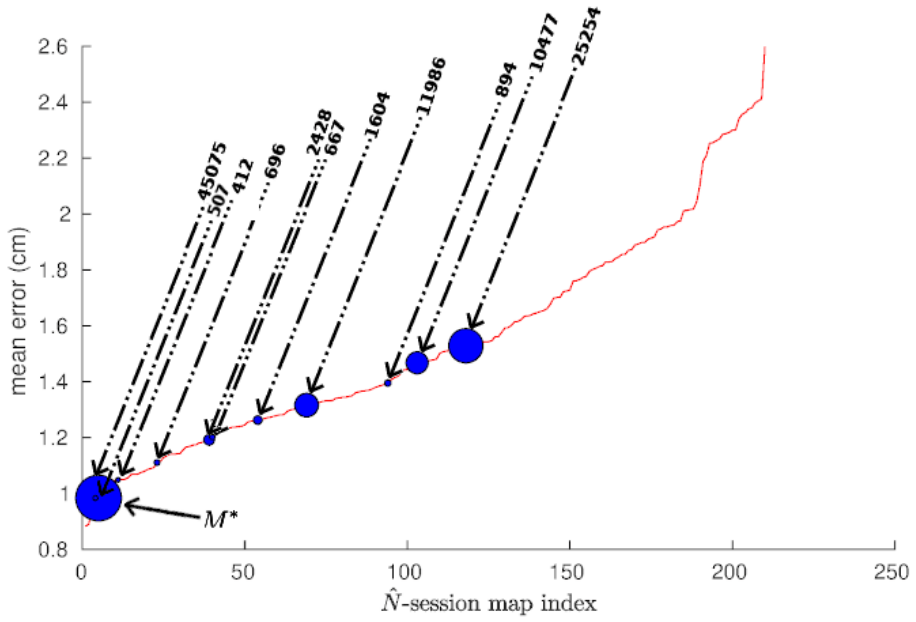
We evaluated our different approaches on the IPLT dataset using the global map from the previous chapter (presented in Figure 3.9). We have used three different values of $\hat{N}$ to demonstrate its efficiency on different setups: $\hat{N} = \{3, 4, 5\}$.

In Figure 4.9, we show the result of the exhaustive search performed with $\hat{N} = 3$ and 4. The exhaustive search allowed us to rank all the possible $\hat{N}$-session maps according to the localization error recorded on each of them. We also present on this figure all the $\hat{N}$-session maps that were generated with the MMFc approach.

This figure shows that the localization error has decreased globally when we passed from $\hat{N} = 3$ to $\hat{N} = 4$ (which is clear by the red curve in the two sub-figures). In the first sub-figure (4.9a), $\hat{N}$ is equal to 3. According to Equation (4.7), the number of all possible $\hat{N}$-session maps composed from 3 traversals out of $N = 10$ traversals is $n = 120$. In the second sub-figure (4.9b), $\hat{N}$ is equal to 4, accordingly $n = 210$. As visible in both sub-figures, our approach has produced multiple results as consequence of using $100,000$ different permutations of the N traversals as input.



(a) $\hat{N} = 3$

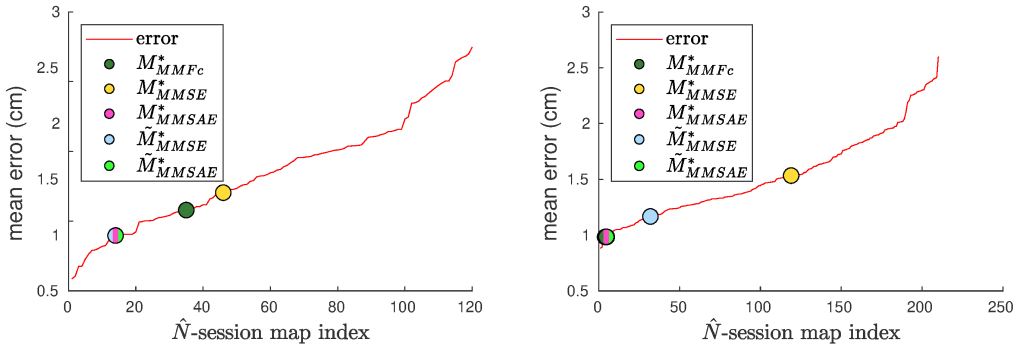


(b) $\hat{N} = 4$

Figure 4.9: Average localization errors for the IPLT dataset global map. The red curve designates the average of localization errors obtained while performing re-localization with a selection of test sequences on the n $\hat{N}$-session maps. All the $\hat{N}$-session maps are sorted in ascending order according to their corresponding errors. The blue dots are indicating the results of the MMFc approach on $100,000$ different permutations of the N traversals. We specify the number of times each result has been reproduced by the MMFc approach with the size of its corresponding dot and with the annotations on the figures.

Therefore, we take into consideration only the $\hat{N}$-session map M^*, which denotes the most reproduced $\hat{N}$-session map by the corresponding map management approach. M^* will be used for further tests and comparisons. It is also noticeable from the figure that all $\hat{N}$-session maps generated by our approach are placed in the lower bound of the curves for both values of $\hat{N}$. This means that for all the 100,000 permutations of the N traversals, our approach has efficiently avoided producing maps whose localization errors are significant.

In Figure 4.10, we present a comparison between the maps M^*_{MMFc}, M^*_{MMSE} and M^*_{MMSAE} which refer to the most reproduced $\hat{N}$-session maps using the MMFc, MMSE and MMSAE approaches, respectively. We also include in the comparison two other $\hat{N}$-session maps, denoted $\tilde{M}^*_{MMSE}$ and $\tilde{M}^*_{MMSAE}$, which were reproduced with the MMSE and MMSAE approaches while imposing the constraint that ensures keeping a night sequence on the map as described in sub-section 4.4.2.



(a) $\hat{N} = 3$ (b) $\hat{N} = 4$

Figure 4.10: A comparison between the $\hat{N}$-session maps produced with the different approaches using $\hat{N} = \{3, 4\}$. In (a), the three maps M^*_{MMSAE}, $\tilde{M}^*_{MMSE}$ and $\tilde{M}^*_{MMSAE}$ are pointing to the same $\hat{N}$-session map. In (b), M^*_{MMSAE} and $\tilde{M}^*_{MMSAE}$ are pointing to the same $\hat{N}$-session map, while M^*_{MMFc} is pointing to an adjacent $\hat{N}$-session map (adjacent index).

We want to remind that for each choice of $\hat{N}$, the size of the maps generated by our different approaches are very close since they incorporate the same number of traversals (for M^*_{MMFc}, M^*_{MMSE}, M^*_{MMSAE}, $\tilde{M}^*_{MMSE}$ and $\tilde{M}^*_{MMSAE}$). For both of SM and USM approaches, we compress the maps M_{SM} and M_{USM} with the compression ratio $\hat{r}$ computed according to Equation (4.8). This guarantees a fair comparison between the different approaches.

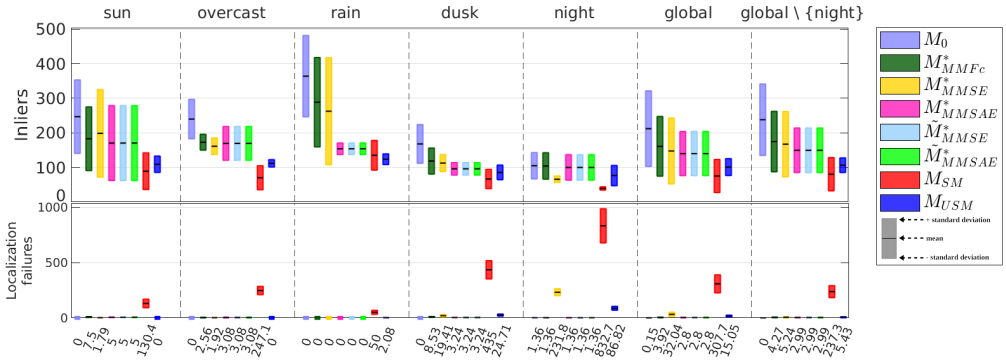
In order to evaluate the influence of the map compression in the localization performance, we compare the performance of localization on the initial global map, M_0, which is composed of N traversals with performance of localization on the most reproduced $\hat{N}$-session map for each map management technique (the maps compared in Figure 4.10). In Figure 4.11 we present the average number of inliers per image and the average number of localization failures per km recorded while

re-localizing on the global map M_0 and on the maps produced with our different map management techniques (with $\hat{N} = \{3, 4, 5\}$). We also present a comparison with the state-of-the-art Summary Maps (SM) approach proposed by Mühlfellner *et al.* [Mühlfellner 2016]. The test sequences were classified manually into 5 different classes as in Chapter 3 (13 sequences in "sun" class, 39 in "overcast", 12 in "rain", 17 in "dusk" and 11 in "night"). The "global" class contains all 93 sequences while the "global \ {night}" class contains all the sequences excluding the night sequences (82 sequences).

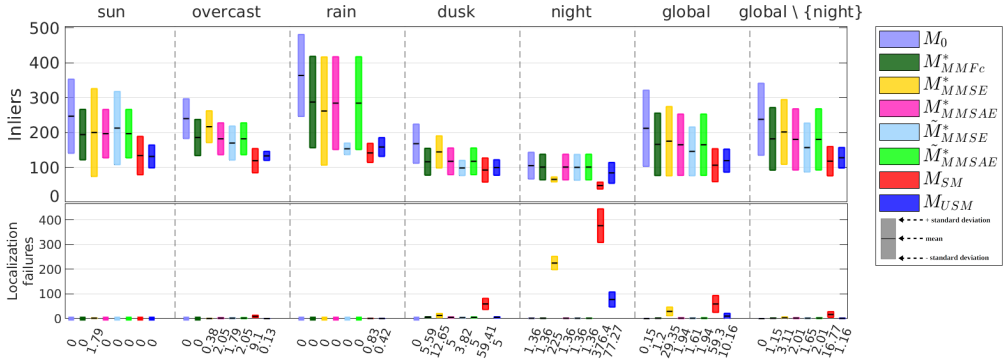
For a better understanding of the results, we present in Table 4.3 the traversals included in each map produced by our different approaches (excluding M_{SM} and M_{USM} since both of them include all the N traversals with a reduced number of inliers).

Table 4.3: Traversals included in each map produced by our approaches on IPLT dataset.

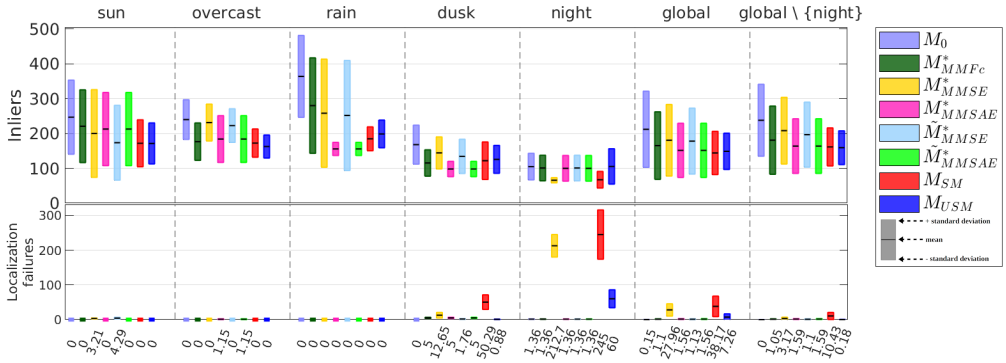
map	$\hat{N} = 3$	$\hat{N} = 4$	$\hat{N} = 5$
M^*_{MMFc}	2020-01-15-11-15-33	2019-10-01-16-54-55	2019-10-02-15-03-40
	2019-10-01-16-54-55	2020-02-05-18-37-10	2019-10-01-16-54-55
	2020-02-05-18-37-10	2020-01-15-13-23-09	2020-02-05-18-37-10
	-	2020-01-31-16-07-34	2020-01-15-13-23-09
	-	-	2020-01-31-16-07-34
M^*_{MMSE}	2020-01-15-11-15-33	2020-01-15-11-15-33	2020-01-15-11-15-33
	2019-10-02-15-03-40	2019-10-02-15-03-40	2019-10-02-15-03-40
	2019-10-22-15-01-25	2019-10-22-15-01-25	2019-10-22-15-01-25
	-	2020-02-05-17-53-21	2020-02-05-17-53-21
	-	-	2020-01-31-16-07-34
M^*_{MMSAE}	2019-10-02-15-03-40	2019-10-01-16-54-55	2019-10-02-15-03-40
	2020-02-05-18-37-10	2020-02-05-18-37-10	2020-02-05-18-37-10
	2020-01-22-10-22-06	2020-01-15-13-23-09	2020-01-15-13-23-09
	-	2020-01-22-10-22-06	2020-01-22-10-22-06
	-	-	2020-01-31-16-07-34
$\tilde{M}^*_{MMSE}$	2019-10-02-15-03-40	2019-10-02-15-03-40	2019-10-02-15-03-40
	2020-02-05-18-37-10	2020-02-05-18-37-10	2019-10-22-15-01-25
	2020-01-22-10-22-06	2020-01-15-13-23-09	2020-02-05-17-53-21
	-	2020-01-22-10-22-06	2020-02-05-18-37-10
	-	-	2020-01-22-10-22-06
$\tilde{M}^*_{MMSAE}$	2019-10-02-15-03-40	2019-10-01-16-54-55	2019-10-02-15-03-40
	2020-02-05-18-37-10	2020-02-05-18-37-10	2020-02-05-18-37-10
	2020-01-22-10-22-06	2020-01-15-13-23-09	2020-01-15-13-23-09
	-	2020-01-22-10-22-06	2020-01-22-10-22-06
	-	-	2020-01-31-16-07-34



(a) $\hat{N} = 3$



(b) $\hat{N} = 4$



(c) $\hat{N} = 5$

Figure 4.11: Comparison of localization performance between M_0 and each of the maps produced by the different approaches using the 93 test sequences from IPLT dataset. Subfigures (a), (b) and (c) show the localization performance when choosing 3, 4 and 5 as values for the parameter $\hat{N}$, respectively. Each box represents the mean value + and - the standard deviation of inliers (or localization failures) recorded while performing re-localization using all the sequences of the corresponding class. The colors of the boxes indicate which map management technique was used to produce the map where the localization was performed. For better readability of the localization failures values, we plot the average number of localization failures per km in the lower x-axis.

It is clear from Figure 4.11 that the localization performance is globally degraded after limiting the map to $\hat{N} = 5$ traversals and is degraded more for $\hat{N} = 4$ and 3 traversals. For all the choices of $\hat{N}$, the localization performance on the map M^*_{MMSE} with night sequences was very poor since this map does not include any night traversal (see Table 4.3). This is not the case for the $\hat{N}$-session maps M^*_{MMSAE} and M^*_{MMFc} which incorporate a night traversal. Both $\widetilde{M}^*_{MMSE}$ and $\widetilde{M}^*_{MMSAE}$ also include a night traversal due to the imposed constraint explained in sub-section 4.4.2. Therefore, the localization performance with night sequences on those maps is reliable (1.36 localization failures per km, which is the same value recorded on M_0).

According to the same figure, localization can be reliable in overcast and rainy conditions even if the map does not include traversals of the same class. For example with $\hat{N} = 3$, both M^*_{MMFc} and M^*_{MMSE} include a traversal from "rain" class, while it is not the case for the other maps, which is distinguishable by the inliers average. However, all of them have not experienced any localization failure (except for M_{SM} and M_{USM}).

On the other hand, the map M_{SM} has shown a major weakness in localization, especially for night sequences. This confirms that this approach is not well suited when the landmarks are not uniformly distributed over the different environmental conditions as it is the case of the global map M_0, which contains only one night sequence (Figure 3.9). This means that landmarks observed in the night are assigned with a low score since they have been observed only in one session (in Section 4.3, we have called this phenomenon as the bias towards more experienced environmental conditions). As expected, the localization performance has been improved especially for night sequences when using the map M_{USM} which incorporates a uniform number of landmarks over the different traversals (as shown in Table 4.2). However, even with the improvement, the USM approach was not able to provide a very good localization performance under the night condition. This means that the scoring policy used in both approaches (SM and USM), which removes landmarks according to their observation rate in different sessions, does not allow to efficiently compress the map to ensure lifelong navigation.

Overall, the localization performance has been degraded when compressing the map with different values of $\hat{N}$. However, even with this degradation, the localization can still be considered as reliable with the use of some of our proposed map management techniques especially MMFc, which is visibly providing slightly better results than the other methods. Using MMFc with $\hat{N} = 5$, we consider 1.1 localization failures per km over all the test sequences ("global" class). This means that only 20 images among all the 93 sequences (which incorporate a total of 159,074 images) where matched with less than 30 inliers (i.e., $\sim 1.26e^{-4}$ failures per image).

Figure 4.12 presents a curve inspecting the memory occupation and landmarks count of the different maps used in our tests with respect of the choice of $\hat{N}$. According to this figure, the choice of $\hat{N} = 5$ has resulted in compressing the map M_0 with a compression ratio $r \approx 2$. Despite this compression, we note that all of M^*_{MMFc}, M^*_{MMSAE}, $\widetilde{M}^*_{MMSE}$ and $\widetilde{M}^*_{MMSAE}$ were able to achieve a very compet-

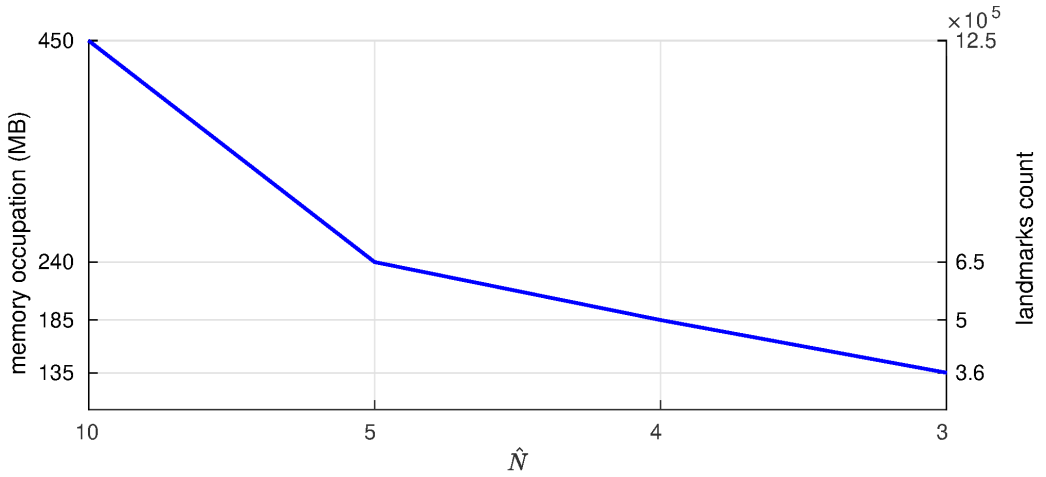


Figure 4.12: A curve in which we inspect the size of the maps with the different approaches. For each value of $\hat{N}$, the curve is presenting the corresponding memory occupation (Megabytes) and the number of landmarks in each one of these maps (all the maps have approximately the same size). For $\hat{N} = 10$, we provide the size of the map M_0.

itive level of performance with the uncompressed map M_0. We also note that all our approaches have remarkably outperformed a map with a same compression rate that was generated with another approach (M_{SM}).

4.7.1.2 Oxford RobotCar dataset

As as we did with the IPLT dataset, we evaluate our approach on the Oxford RobotCar dataset. The global map used here is the same as in Chapter 3 (Figure 3.11). In Figure 4.13, we present the average of localization error of all the n $\hat{N}$-session maps composed of $\hat{N} = 3$ traversals (according to Equation (4.7), $n = 56$). We also indicate the solutions produced with the MMFc approach. Since this global map is built only by $N = 8$ traversals, we exhibit the result of our approach on all the possible permutations of 8 traversals ($8! = 40320$ permutations).

In the figure, there is a sudden increase in the curve, as the last 10 indexed $\hat{N}$-session maps have a remarkable higher localization error. This is caused by the fact that these 10 $\hat{N}$-session maps do not incorporate any rainy sequence (the map M_0 contains 3 rainy sequences as presented in figure 3.11). This means that these $\hat{N}$-session maps are composed of only sunny and cloudy sequences. M_0 also incorporates 1 sunny sequence and 4 cloudy sequences (i.e. 5 sequences), which means that the combination of sequences that do not contain any rainy sequence is ${}_5C_3 = 10$.

Figure 4.14 presents a comparison between the most reproduced $\hat{N}$-session maps with the MMFc, MMSE and MMSAE techniques. Unlike the IPLT dataset, the sequences used for the mapping and the test datasets in Oxford RobotCar do not incorporate any night sequences. Therefore, we do not include the maps $\widetilde{M}^*_{MMSE}$ and $\widetilde{M}^*_{MMSAE}$ in the comparison (the maps that were obtained by imposing the

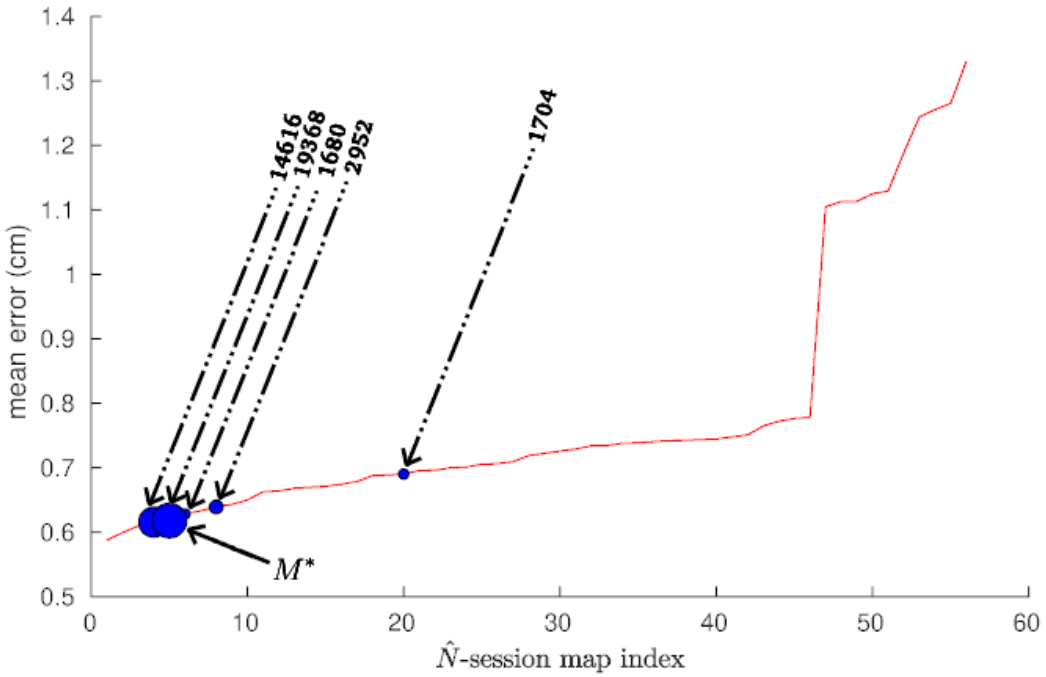


Figure 4.13: Average localization errors of the Oxford RobotCar dataset global map with $\hat{N} = 3$. The blue dots are indicating the results of the MMFc approach on all the possible permutations of the N traversals.

constraint that ensures keeping a night sequence on the map as described in subsection 4.4.2).

As we did with the IPLT dataset, we evaluate the influence of the map compression in the localization performance with the Oxford RobotCar dataset. In Figure 4.15, we compare the performance of localization on the initial global map M_0, which is composed of $N = 8$ traversals, with the performance of localization on the most reproduced $\hat{N}$-session map for each map management technique. We also compare these techniques with the SM and USM approaches. The comparison is performed according to the inliers average and the number of localization failures per km.

This figure shows that the performance of localization on the Oxford RobotCar dataset has slightly decreased with $\hat{N} = 3$ when using our different approaches. This kind of degradation is expected after compressing the map from 8 traversals to only 3. According to the figure, there is a significant failure rate for the sequences under the "rain" class. This can be explained by the fact that the test sequences include a sequence (2014-11-21-16-07-03 🌧️🌅) that was recorded at the beginning of dusk and during a heavy rain (see Figure 3.18). On the other hand, localization on the "snow" class (the snow class contains only one sequence 2015-02-03-08-45-10 🌨️) has not led to any localization failures with M^*_{MMFc} and M^*_{MMSAE}. Both of these $\hat{N}$-session maps contain the sequence 2015-02-20-16-34-06 🌧️ (see Table 4.4) which

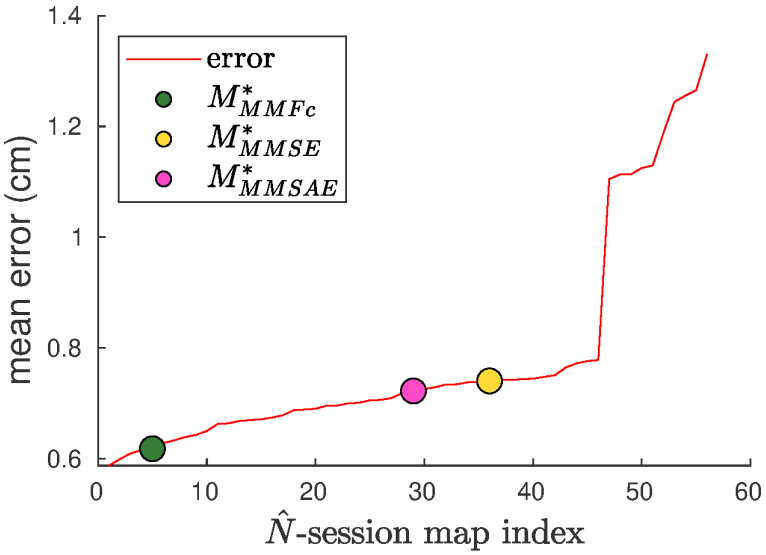


Figure 4.14: A comparison between the $\hat{N}$-session maps produced with the different approaches using $\hat{N} = 3$ with Oxford RobotCar dataset.

is visually similar to the snow sequence 2015-02-03-08-45-10 that was recorded on a soft snow.

Table 4.4: Traversals included in each map produced by our approaches on Oxford RobotCar dataset.

Map	M^*_{MMFc}	M^*_{MMSE}	M^*_{MMSAE}
$\hat{N} = 3$	2014-06-26-09-53-12	2014-06-24-14-47-45	2014-06-26-09-53-12
	2015-02-20-16-34-06	2015-07-08-13-37-17	2015-02-20-16-34-06
	2014-12-05-11-09-10	2014-11-25-09-18-32	2015-07-08-13-37-17

In Table 4.5, we present the memory occupancy and the number of landmarks included in each map used to evaluate our approach on the Oxford RobotCar dataset.

Table 4.5: Memory occupancy and number of landmarks in each map built using Oxford RobotCar dataset.

Map ($\hat{N} = 3$)	M_0	M^*_{MMFc}	M^*_{MMSE}	M^*_{MMSAE}	M_{SM}	M_{USM}
Memory occupancy (MB)	~ 970	~ 330	~ 330	~ 330	~ 430	~ 430
Number of landmarks ($\times 10^6$)	~ 14	~ 5	~ 5	~ 5	~ 5	~ 5

We notice from the table that both M_{SM} and M_{USM} occupy slightly more mem-

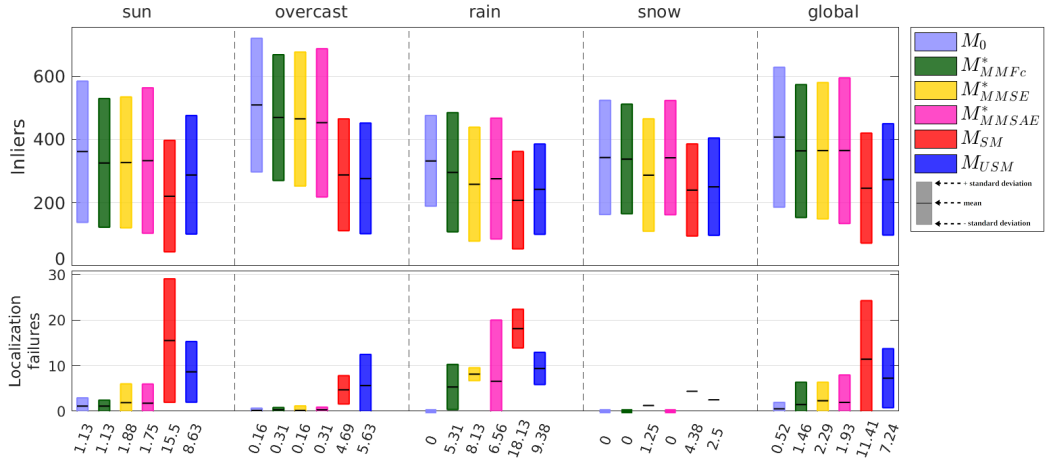


Figure 4.15: Comparison of localization performance between M_0 and each of the maps produced by the different approaches using the 12 test sequences from Oxford RobotCar dataset. $\tilde{M}^*_{MMSE}$ and $\tilde{M}^*_{MMSAE}$ are excluded since this dataset does not include night sequences.

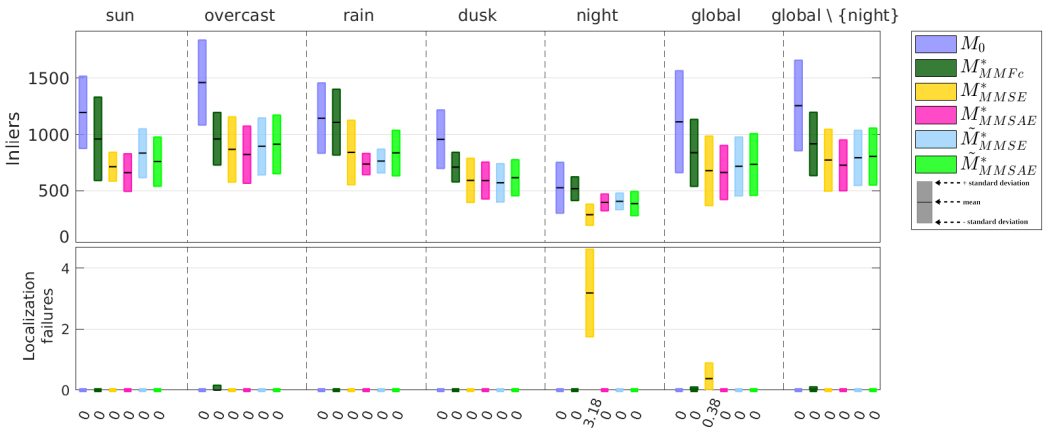
ory space than the other techniques, although the number of landmarks is approximately the same. This is due to the fact that these two maps contain information about all the N traversals, such as the poses of the vehicle and the cameras. This is not the case for M^*_{MMFc}, M^*_{MMSE} and M^*_{MMSAE}, which only contain information about the $\hat{N}$ maintained traversals (all other traversals are completely removed).

4.7.2 Evaluation scenario 2

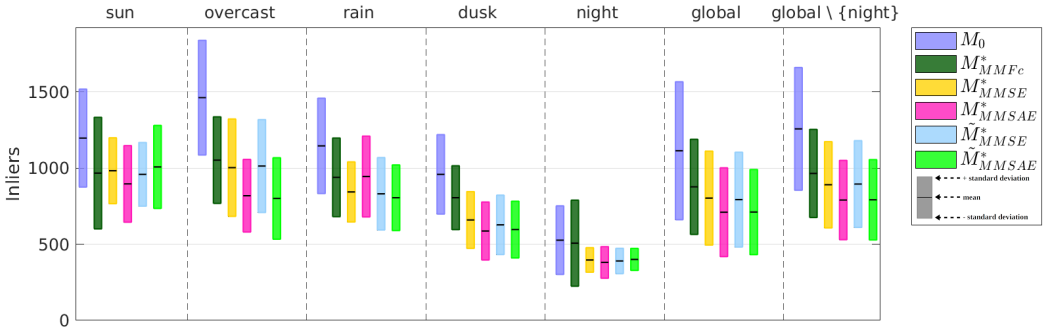
This scenario is modeled on the real use case of autonomous shuttles. Unlike in the previous scenario where we build a map from only N traversals ($N = 10$ for IPLT dataset and 8 for Oxford RobotCar dataset), in this scenario we use all the sequences to perform mapping and incorporate them one by one like in the case where an autonomous shuttle is regularly visiting a same location in different times to build a long-term map. Therefore, the localization performance is evaluated while performing SLAM-LOC to build a long-term map. This means that the evaluation of different permutations of the mapping sequences in this scenario is intractable. For this reason, we performed this evaluation scenario with a random permutation of all the sequences. We have performed this evaluation scenario only on the IPLT dataset since it incorporates a sufficient number of sequences (103 sequences) allowing to properly evaluate the localization performance. This is not the case for the Oxford RobotCar dataset where we consider only 20 sequences in total.

In Figure 4.16, we present a comparison between the different proposed map management techniques with respect to the average number of inliers per image and localization failures per km.

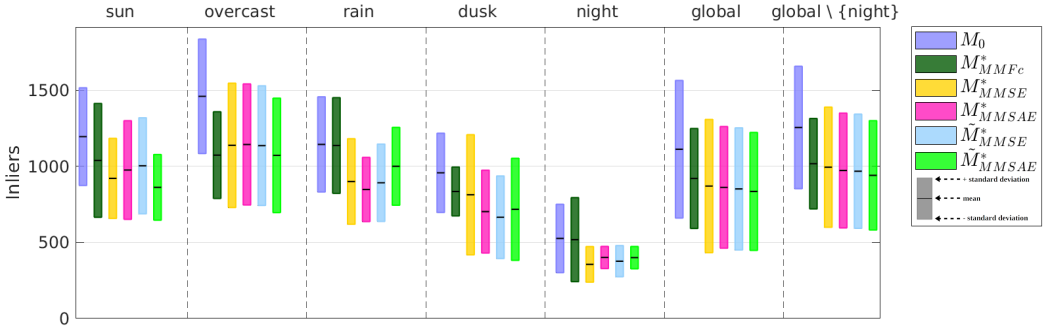
The figure shows that all presented techniques have successfully provided a re-



(a) $\hat{N} = 3$



(b) $\hat{N} = 4$



(c) $\hat{N} = 5$

Figure 4.16: Comparison of localization performance between M_0 and each of the maps produced by the different approaches with respect to evaluation scenario 2. We have used all the 103 test+mapping sequences from IPLT dataset in this comparison. Sub-figures (a), (b) and (c) show the localization performance when choosing 3, 4 and 5 as values for the parameter $\hat{N}$, respectively. In both (b) and (c), we have not experienced any single localization failure for all the presented techniques.

liable lifelong navigation under different environmental conditions without localization failures (except for MMSE with $\hat{N} = 3$). For all values of $\hat{N}$, it is noticeable that the MMFc method has achieved the best performance among all proposed techniques. This is due to the fact that the similarity scores included in the matrix F_c take into account all environmental conditions, which is not the case for the scores provided by the two methods MMSE and MMSAE, which only consider illumination changes.

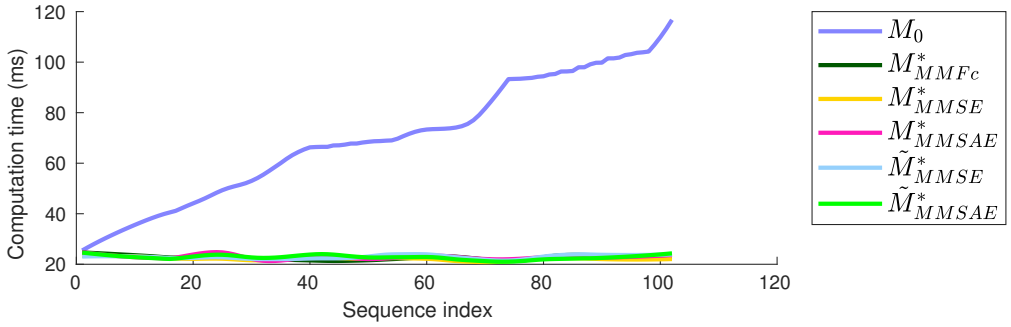
According to the figure, there is a notable increase in the number of inliers compared to the comparison performed in the evaluation scenario 1 (Figure 4.11). This is due to the use of the SLAM-LOC instead of re-localization in this scenario. This means that the previously mapped landmarks are bringing a gain and helping localizing the current image. This noticeable increase in the inliers is also due to the retrieval of two keyframes from the map instead of only one. This helped increasing the number of 2D/3D constraints which produced a large number of inliers.

We have not presented M_{SM} and M_{USM} in the figure because in both approaches the SLAM-LOC was not possible after a certain number of mapping sessions (50 sessions for M_{SM} and 55 sessions for M_{USM}). After inspecting both maps, we found that the number of 3D points was significantly reduced by the two approaches ($\sim$700 3D points). This means that each of these 3D points was enriched with a large amount of landmarks observed in different traversals and at different viewpoints. This problem is due to the fact that the scoring policy used in both approaches recommends removing the landmarks with the lowest observation rates. In other words, after each mapping session, both approaches filter out the 3D points associated with a low number of landmarks observed in different sessions and keep the 3D points enriched with a high number of landmarks. Therefore, the localization is no longer possible with a such reduced number of 3D points in the map.

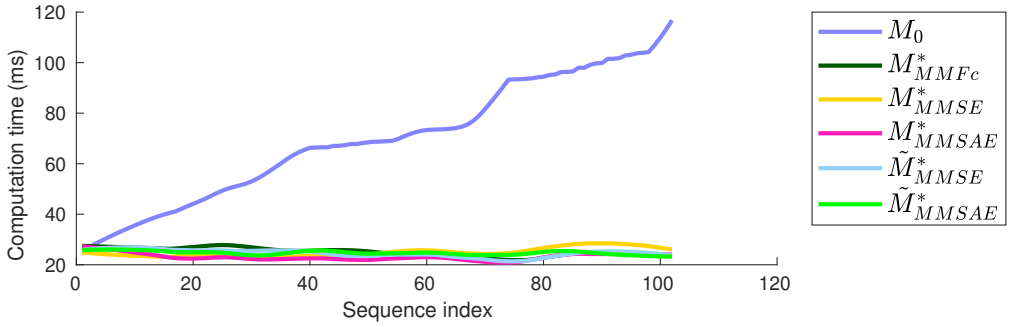
We notice from the figure that all the presented techniques have provided a reliable localization performance without any matching failure in different environmental conditions (except for MMSE with $\hat{N} = 3$). It is also noticeable that the technique MMFc has produced the $\hat{N}$-session map with the best long-term localization performance.

The map M_0 that incorporates all the 103 sequences has shown the best performance according to the figure. However, we note that it also has the highest computational complexity. Since M_0 is ceaselessly growing in this evaluation scenario, the computation time is also increasing. In Figure 4.17, we present a comparison between the different map management techniques with respect to the computation time required to process a single frame by the tracking thread of our system. This figure shows the average computational cost (time required to process a frame) recorded while performing the comparison presented in Figure 4.16.

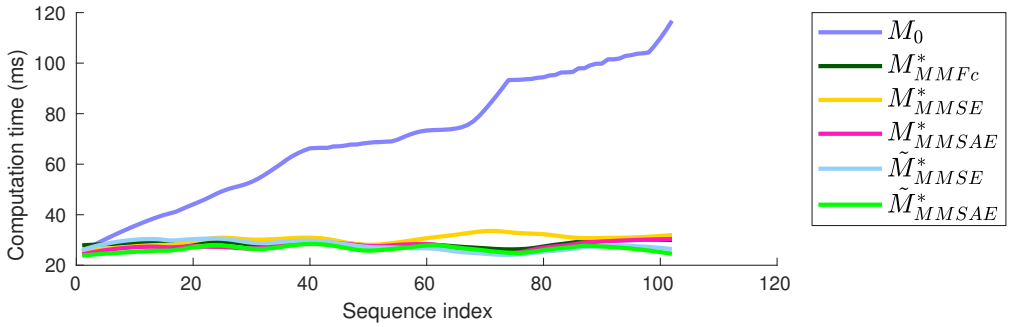
According to the figure, the computational complexity of processing frames on M_0 increases with respect to the number of sequences contained in the map. This means that without performing map management, real-time performance will not be possible at some point of the autonomous shuttle's operational routine. For each value of $\hat{N}$, the computational complexity of processing frames on all other maps



(a) $\hat{N} = 3$



(b) $\hat{N} = 4$



(c) $\hat{N} = 5$

Figure 4.17: Comparison of computational cost between M_0 and each of the maps produced by the different approaches with respect to evaluation scenario 2. We have used all the 103 test+mapping sequences from IPLT dataset in this comparison. Each sub-figure shows the average computation time required to process a single frame on each of the 103 used sequences for each value of $\hat{N}$. The sequences are sorted according to the same order adopted while performing the evaluation scenario 2.

shown in the figure is approximately equivalent, which is expected since they all incorporate the same amount of data (same number of traversals). Increasing the value of $\hat{N}$ leads to an increase in the computation time. This means that the value of $\hat{N}$ should be chosen with caution as it has a direct impact on the localization performance and computational complexity.

4.8 Conclusion

In this chapter, we have presented three different algorithms which can be used for long-term localization process. These three algorithms have in common their capability to bound the size of the map to avoid its continued inflation and each one of them is designed to use a different kind of information to remove superfluous data from the map. In the first algorithm, we have proposed an improvement for the state-of-the-art Summary Maps technique to avoid omitting landmarks observed under rarely experienced environmental conditions. In the second algorithm, we have used the sun position information as a guideline to filter out the traversals and their associated landmarks which were recorded under resembling lighting conditions. The last algorithm is based on some fundamental concepts proposed in the keyframe retrieval approach explained in the previous chapter. It uses the similarity matrix F_c as information to classify the traversals in the map and remove unimportant ones based on their resemblance.

To test the efficiency of our proposed approaches, we have evaluated the localization performance in two different scenarios. Except for the improved Summary Maps technique (USM) which has failed in the second evaluation scenario, our approaches have been demonstrated to be able to achieve a reliable lifelong navigation on two different datasets while limiting the size of the map. Although the localization performance has been slightly reduced after compressing the map, our proposed approaches have successfully restrained the computational complexity. We have also demonstrated that our different techniques have noticeably outperformed localization performance of a state-of-the-art approach (Summary Maps).

As perspectives, we aim to further improve our different techniques in several aspects. We would like to add a spatial constraint to the USM technique that ensures a uniform spatial distribution of 3D points over the compressed map, and ensures the maintenance of a sufficient number of 3D points that guarantee successful navigation under different environmental conditions to avoid running into similar cases as in evaluation scenario 2. We also aim to combine other weather information with the solar coordinates to further improve both the MMSE and MMSAE approaches. Moreover, we are interested in improving our traversal-based map management techniques to dynamically adjust the value of $\hat{N}$ in such a way that the best localization performance can be achieved while preserving the real-time capability.

Chapter 5

Conclusion and perspectives

Conclusion

In this PhD thesis, we have been interested in lifelong navigation in dynamic environments. Particularly, we have been focused on both localization and mapping aspects for autonomous shuttles applications. The work detailed in this manuscript is structured in two main contributions.

First, we present a contribution to landmark retrieval, which is employed to extract relevant landmarks for the current environmental conditions. Unlike most state-of-the-art approaches, our proposed landmark retrieval is performed in a keyframe-based aspect, i.e., instead of searching directly on the map for relevant landmarks for localization, our approach searches for the keyframes that were taken under similar environmental conditions as the current frame and extracts their associated landmarks. Given that classic keyframe retrieval techniques, which retrieve keyframes based only on their geometric distances from the vehicle pose, fail in long-term scenarios where the autonomous shuttle revisits the same location under different environmental conditions. We designed a ranking function that serves as a heuristic for keyframe retrieval taking into account both geometric distance and environmental conditions.

The suggested ranking function creates at the beginning of each localization session a similarity matrix, called F_c, which contains similarity scores between the different traversals included in the map. The values in this matrix define the similarity scores between the environmental conditions of all traversals in the maps, taken pairwise. This ranking function is used as a guideline during the rest of the trajectory to retrieve the keyframes which are deemed to provide the best localization performance. Given a map that incorporates multiple traversals taken under different environmental conditions, our ranking function takes into consideration the similarity scores in the matrix F_c and the geometric distances between the vehicle pose and the 3D poses of all the mapped keyframes to extract the most relevant keyframes and their associated landmarks.

By performing data association between the landmarks of the current frame and those retrieved by our designed ranking function, we have demonstrated, in two different datasets, that our approach was able to retrieve good keyframes under different environmental conditions allowing to remarkably improve the localization performance and significantly reduce the number of localization errors compared to a classic technique that retrieves the closest keyframes to the vehicle pose (e.g., we have reduced the number of localization failures in IPLT dataset by $\sim$400$\times$).

The second contribution of this thesis is focused on finding efficient ways to limit the amount of data to be stored in the map for the purpose of reducing the computational complexity of localization over long-term scenarios. In this respect, we have proposed three different map management techniques which have in common their capability to bound the size of the map to avoid its continued inflation. Each one of the proposed techniques uses a different kind of information to remove superfluous data from the map and all of them are carried out offline after each localization session. The first technique is based on a state-of-the-art approach, called Summary Maps, which uses inter-session landmark co-observability information to remove less important data from the map. Our contribution to the Summary Maps consists in adding a constraint that ensures a uniform distribution of landmarks over all the traversals. The second technique uses information related to the sun position to compute similarity between the traversals in the map and filter out landmarks observed in similar brightness levels. The third technique consists in using the matrix F_c which was defined by our keyframe retrieval technique and which contains similarity scores between the different traversals in the map. This matrix is used as a criterion to detect and remove redundant data from the map.

The three proposed map management techniques were evaluated in two different scenarios. The first scenario consists in using a set of mapping sequences to build a map with the different techniques and evaluating the re-localization performance on each of them with another set of test sequences. The evaluations performed in this scenario were validated on both the Oxford RobotCar and IPLT datasets. The second scenario was inspired by the autonomous shuttles application scenario, where we perform incremental localization and mapping with all available sequences. The evaluations performed in this scenario were validated on the IPLT dataset.

The evaluations in both scenarios and on both datasets have demonstrated that the re-localization performance on compact maps generated with our different techniques was reliable and achieved a very competitive level of performance with the uncompressed map and has remarkably outperformed the state-of-the-art SM technique (except for the USM technique that has suffered the same problem as SM in the second evaluation scenario). The evaluations have also demonstrated that computational complexity has been drastically reduced when re-localizing on a compressed map.

Another contribution provided in this thesis is the IPLT dataset, which we have made available to the community in the hope that it will be useful to other researchers working in the field of long-term localization. This dataset was recorded over a two-year period in a parking lot and to date includes 127 sequences recorded under a variety of environmental conditions (luminance, weather, seasonal changes...).

Perspectives

Based on the obtained results, the proposed keyframe retrieval method allows improving the retrieval under various challenging conditions, leading to an increase in the number of matching points and a decrease in the localization failure rate. Moreover, the proposed map management techniques enable efficient reduction of the map size while preserving localization performance, allowing long-term real-time localization in dynamic environments. However, there are still several perspectives to further improve both keyframe retrieval and map management techniques.

As mentioned in Chapter 3, during the first few meters of the trajectory, the proposed ranking function is learned using Equation (3.7) to model an abstract representation of the environmental conditions of the different traversals in the map. Accordingly, our first perspective consists in using other learning-based techniques, especially Convolutional Neural Networks (CNN) methods, to learn our ranking function.

As a second perspective, we will take a deeper look at the keyframe retrieval technique, in particular the algorithm executed in the first meters of the trajectory, and investigate the best configuration to reduce the computation time and to maintain a good performance.

In Chapter 4, we demonstrated that both localization performance and computational complexity have a strong dependency upon the value of $\hat{N}$. Therefore, another interesting perspective is envisaged to dynamically adjust this value for a best trade-off between localization performance and computational complexity.

As a fourth perspective, we aim to extend our work to other more modern feature descriptors, especially illumination invariant ones, which can further improve both keyframe retrieval and map management performance.

Another interesting perspective is to integrate semantic information into our system to further improve the two proposed approaches. This kind of information can be used to retrieve only the landmarks that belong to non-dynamic elements (e.g., we can exclude retrieving landmarks that belong to clouds, parked cars, pedestrians, etc.). This information can also be considered in the map management process to filter out landmarks that belong to dynamic elements which are irrelevant for localization. This can help reduce the number of erroneous matches and improve the overall performance of our navigation system. Figure 5.1 illustrates the result of a semantic segmentation task performed on Cityscapes dataset [Cordts 2016].

As a sixth perspective, we will explore the advantages of developing a hybrid technique that combines different map management techniques. For example, combining the USM and MMFc techniques to design a method that consists of removing irrelevant traversals in a first step, and then removing irrelevant landmarks from the remaining traversals in a second step.

The technique proposed in Chapter 3 makes it possible to learn and store an abstract representation of the environmental conditions of different sequences in the matrix F_c. An interesting perspective is to exploit this matrix in other domains. As demonstrated in Figure 3.16, the values of F_c change dynamically depending

(a) Input (b) Output

Figure 5.1: Illustration of a semantic segmentation task. (a) depicts the input image from the Cityscapes dataset and (b) is the output generated by an instance segmentation algorithm [Zhou 2020].

on the changes in the environmental conditions. This means that this matrix can be used to detect changes in environmental conditions. Therefore, it is conceivable to use this matrix in other domains that can benefit from this property, such as autonomous irrigation control systems in agriculture, where irrigation can be automatically disabled when it rains.

As a final applicative perspective, we intend to employ the work presented in this thesis on a daily operational autonomous shuttle that is deployed over long periods to further evaluate the efficiency of our approaches.

Appendix A

IPLT dataset

Autonomous driving applications are very critical and should be taken with absolute caution before deployment on public roads. Therefore, real-world data are needed in development, testing and validation phases.

Intensive work on SLAM algorithms has produced a large number of related datasets such as Ford Campus Dataset [Pandey 2011], Málaga Urban Dataset [Blanco-Claraco 2014], Waymo Open Dataset [Sun 2020],.... Some of these datasets were recorded in static environments with very little environmental changes, while some others are not revisiting a same location when recording different sequences. KiTTi [Geiger 2013] is widely used dataset in SLAM applications, unfortunately, this dataset is not incorporating many environmental conditions since it was collected over one week (from 2011-09-26 to 2011-10-03). Later, a new dataset with a novel labeling scheme and data for 2D and 3D semantic segmentation was proposed in KiTTi 360 [Xie 2016]. However, this dataset consists of only 11 individual sequences and there is little overlap in trajectories between them.

Applications destined for autonomous driving and aiming for long-term localization uses must be evaluated on real-life scenarios where environment is changing over time. The VPRiCE challenge [Suenderhauf 2015] is a dataset that offers some challenging cases for localization. Unfortunately, this dataset is offering only few sequences of some places that were revisited twice on different times. Similarly, the CMU Seasons dataset [Bansal 2014] was acquired in urban and suburban environments totaling over 8.5 km of travel and contains 7,159 reference images and 75,335 query images acquired in different seasons. Sattler *et al.* [Sattler 2018] have also presented a challenging dataset, called Aachen Day-Night, which incorporates 4,328 daytime images and 98 night-time queries. The NCLT [Carlevaris-Bianco 2015b], Oxford RobotCar [Maddern 2017] and UTBM RobotCar [Yan 2019] datasets are three widely used datasets for long-term tracking applications as they include different environmental conditions. The UTBM RobotCar dataset is including only few sequences (11 sequences in total) while in the two others, the traversed path is varied on each recording session.

In addition to environmental conditions, we are also interested in evaluating the effect of the lateral and angular deviation between sequences on the localization performance. However, the previously mentioned datasets do not provide sequences with such characteristics. This is the main reason that led us to record our own dataset and make it available to the community.

This appendix presents our own dataset, called IPLT (Institut Pascal Long-Term) dataset, which mainly addresses localization under challenging conditions

issues (snow, rain, change of season. . .). Before explaining in details the composition of our dataset, it is important to explore the structure of an autonomous robot first. Figure A.1 represents the operating mechanism of a general autonomous navigation platform in See-Think-Act cycle as explained in [Siegwart 2011].

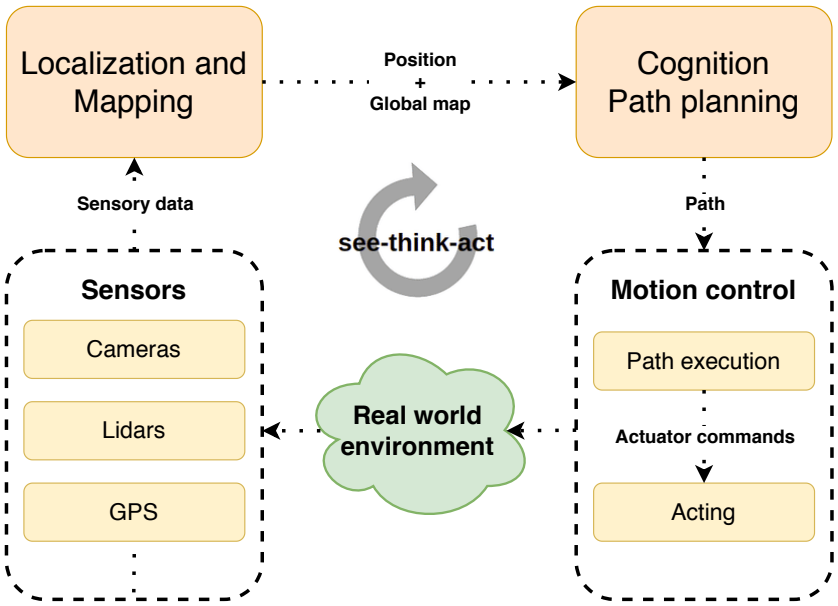


Figure A.1: Autonomous driving platform represented in See-Think-Act Cycle [Siegwart 2011].

According to Figure A.1, we can identify the four main modules interfering in this See-Think-Act mechanism:

- Perception of the environment and the state of the robot thanks to the different equipped sensors.
- Robot localization and mapping in the environment.
- Obstacle avoidance and trajectory planning.
- Processing and executing mission orders.

In our case, we are interested only in the first two modules which are directly dependant to the dataset presented in this appendix. Our experimental vehicle acquires external environmental data through different equipped sensors (camera images, laser scans, GPS data, odometry data,...). Then, these sensory information are received by the localization and mapping (SLAM) module to allow the vehicle to interpret the environment so it can localize and update the map.

In our dataset, we repeatedly traverse the same parking lot, therefore, we managed to record many dynamic elements such as weather and lighting changes, seasonal changes, parking lot state changes (parked cars changes, empty parking lot, full

parking lot, ...), moving cars, moving pedestrians, ... In Figure A.2, we present an overview of images showing some types of environmental conditions included in our dataset.

Figure A.2: An overview of images recorded with the front camera for some sequences of our dataset.

This dataset is composed of 127 sequences in total and they are distributed as follows:

- 22 sequences with sunny condition
- 43 sequences with cloudy weather
- 19 sequences with rainy weather
- 19 sequences with dusk condition
- 14 sequences with night condition
- 4 sequences with fog condition
- 5 sequences with snow condition
- 1 long sequence (`2019-12-05-16-43-56.bag`) recorded over one hour and 10 minutes with multiple loops in the parking lot starting from 16:44 until 17:54 and it incorporates day, dusk and night conditions.

We made our dataset public online in the hope of facilitating evaluations for researchers focusing on long-term autonomous navigation in dynamic environments. Our dataset can be downloaded through the link: `http://iplt.ip.uca.fr/datasets/`. Please enter the following username/password for a read-only access to our ftp server: ipltuser/iplt_ro.

In all the sequences, the vehicle has followed the same path, while in some of them, we made some slight lateral and angular deviations as specified in the Figure 3.4. Each sequence is about 200 m length.

All the sequences in our dataset were recorded with our experimental vehicle presented in Figure A.3. It consists of an electric shuttle that is equipped with two cameras (front and rear), four LiDAR systems (two front and two rear), a consumer grade global positioning system (GPS).... Each camera is recording gray-scale images with 10Hz frequency and both of them are having 100° FoV (Field of View).

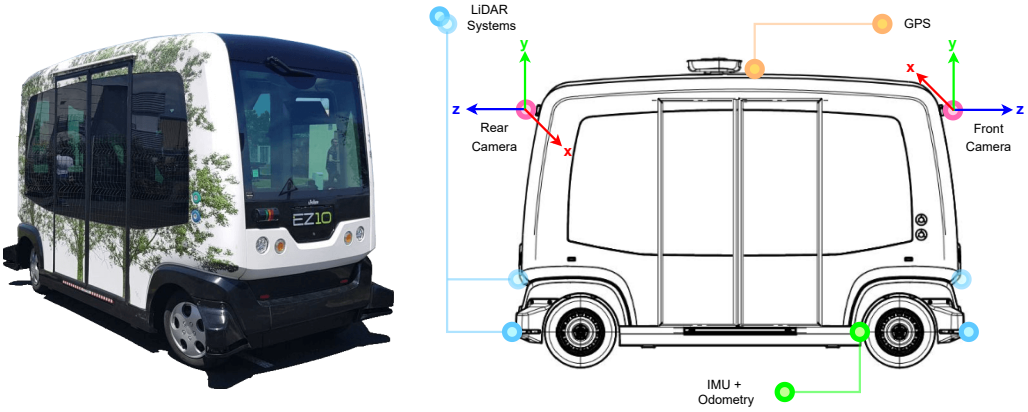


Figure A.3: The EasyMile EZ10 electric shuttle used to record our dataset.

The cameras were slightly moved in April 2019, so we have two different calibration settings, one for sequences recorded before April 2019 and one for more recent sequences. All the sequences are saved in rosbag files format and can be read by the ROS middleware [Quigley 2009]. The rosbag files contain the following rostopics:

- `/cameras/front/image`: front camera images.
- `/cameras/back/image`: rear camera images.
- `/robot/odom`: absolute poses calculated by wheel odometry.
- `/lidars/front_left/scan`: front-left lidar data.
- `/lidars/front_right/scan`: front-right lidar data.
- `/lidars/back_left/scan`: back-left lidar data.
- `/lidars/back_right/scan`: back-right lidar data.

- `/gps_planar`: GPS data.

- `/tf_static`: contains the extrinsic parameters of all sensors (cameras, lidars, GPS, ...).

In Table A.1, we present the intrinsic parameters of our two cameras which are expressed in the unified camera model [Barreto 2006]. The unified camera model has five parameters: $[f_x, f_y, u_0, v_0, \xi]$ and they are used to project a 3D point $P(X_s, Y_s, Z_s)$ expressed in the Spherical coordinates into a 2D Point p_c expressed in the image plane as explained in Equation (A.1) and Figure A.4.

Table A.1: Intrinsic parameters of the cameras.

from_2018-10-19_to_2019-03-08

	f_x	f_y	u_0	v_0	ξ
front	766.3141	769.5469	324.2513	239.7592	1.4513
back	763.5804	766.0006	326.2222	250.7755	1.4523

from_2019-10-01

	f_x	f_y	u_0	v_0	ξ
front	770.0887	768.9841	330.3834	222.0791	1.4666
back	764.4637	763.1171	322.6882	247.8716	1.4565

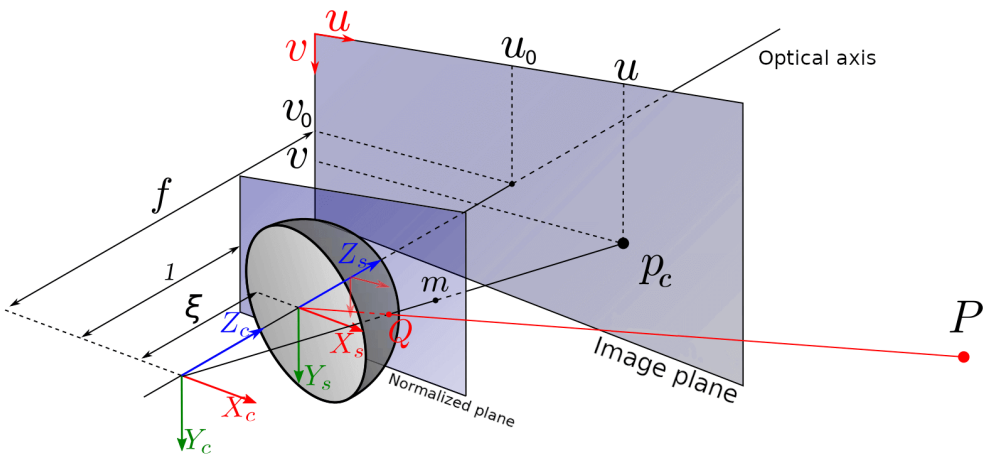


Figure A.4: Unified camera model. A 3D point P is projected in the image plane of the camera into a distorted point p_c [Lébraly 2012].

$$p_c = K m_c$$

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } m_c = \begin{bmatrix} \frac{X_s}{\rho} \\ \frac{Y_s}{\rho} \\ \frac{Z_s}{\rho} + \xi \end{bmatrix} \tag{A.1}$$

$$\text{with } \rho = \sqrt{X_s^2 + Y_s^2 + Z_s^2} \text{ and } \xi = Z_c \geq 0$$

Table A.2 shows the extrinsic parameters of the cameras which are already integrated in the rosbag files. We have expressed the extrinsic parameters of the front camera in the coordinate system of the rear camera, this means that we present the translation and the rotation of the front camera with respect to the axis of the rear camera (see Figure A.3). The rotations are presented in quaternions.

Table A.2: Extrinsic parameters of the cameras.

from_2018-10-19_to_2019-03-08

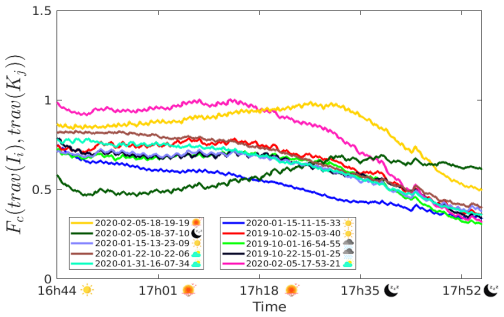
Rotation				Translation		
q_x	q_y	q_z	q_w	t_x	t_y	t_z
0.0030	-0.9998	0.01479	0.0123	-0.0304	-0.0698	-3.4635

from_2019-10-01

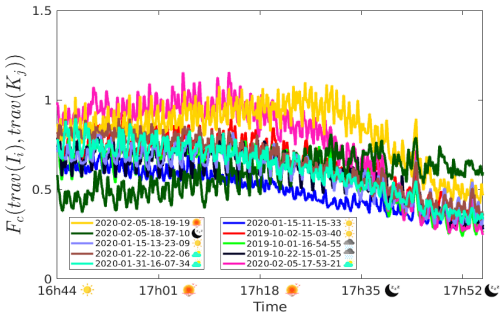
Rotation				Translation		
q_x	q_y	q_z	q_w	t_x	t_y	t_z
0.0002	-0.9998	0.0200	0.0089	0.0600	-0.0321	-3.4637

Appendix B

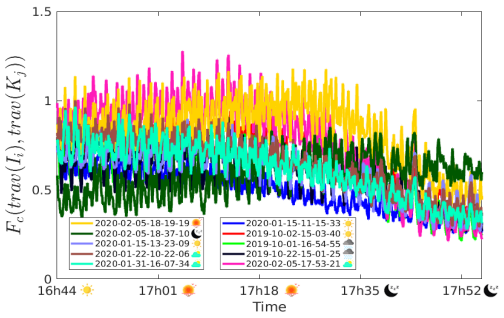
Update rate evaluation



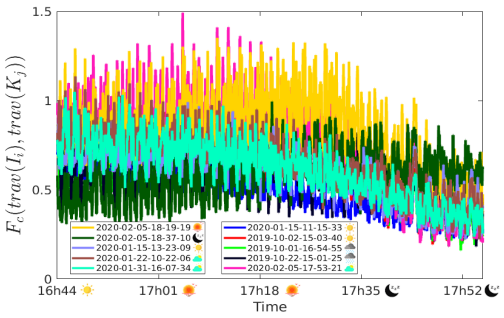
(a) $\alpha = 0.01$



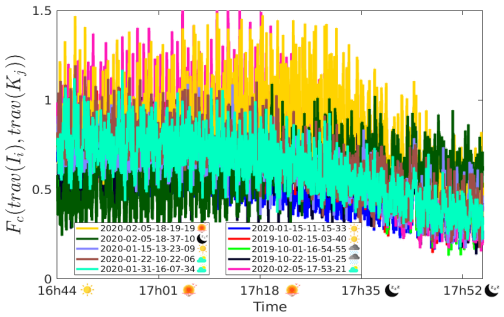
(b) $\alpha = 0.1$



(c) $\alpha = 0.2$



(d) $\alpha = 0.5$



(e) $\alpha = 1$

Figure B.1: Different values for the update rate α without smoothing. Increasing the value of α leads to an increase in the noise of the values of F_c while decreasing α leads to a slowdown in the response to environmental changes.

Bibliography

[Alberto 2009] Sergio Alberto, Rodriguez Florez, Vincent Fremont and Philippe Bonnifait. *An experiment of a 3d real-time robust visual odometry for intelligent vehicles.* In 2009 12th International IEEE Conference on Intelligent Transportation Systems, pages 1–6. IEEE, 2009.

[Antonelli 2001] Gianluca Antonelli, Stefano Chiaverini, Nilanjan Sarkar and Michael West. *Adaptive control of an autonomous underwater vehicle: experimental results on ODIN.* IEEE Transactions on Control Systems Technology, vol. 9, no. 5, pages 756–765, 2001.

[Appapogu 2019] Rahul Dev Appapogu. *Autonomous navigation in GPS denied environments using MPC and LQR with potential field based obstacle avoidance.* PhD thesis, Colorado School of Mines, 2019.

[Arora 2016] BS Arora, J Morgan, SM Ord, SJ Tingay, M Bell, JR Callingham, KS Dwarakanath, P Hancock, L Hindson, N Hurley-Walker*et al. Ionospheric Modelling using GPS to Calibrate the MWA. II: Regional ionospheric modelling using GPS and GLONASS to estimate ionospheric gradients.* Publications of the Astronomical Society of Australia, vol. 33, 2016.

[Bai 2018] Dongdong Bai, Chaoqun Wang, Bo Zhang, Xiaodong Yi and Xuejun Yang. *Sequence searching with CNN features for robust and fast visual place recognition.* Computers & Graphics, vol. 70, pages 270–280, 2018.

[Balntas 2016] Vassileios Balntas, Edward Johns, Lilian Tang and Krystian Mikolajczyk. *PN-Net: Conjoined triple deep network for learning local image descriptors.* arXiv preprint arXiv:1601.05030, 2016.

[Bansal 2014] Aayush Bansal, Hernán Badino and Daniel Huber. *Understanding how camera configuration and environmental conditions affect appearance-based localization.* In 2014 IEEE Intelligent Vehicles Symposium Proceedings, pages 800–807. IEEE, 2014.

[Barreto 2006] Joao P Barreto. *Unifying image plane liftings for central catadioptric and dioptric cameras.* In Imaging Beyond the Pinhole Camera, pages 21–38. Springer, 2006.

[Bay 2006] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. *Surf: Speeded up robust features.* In European conference on computer vision, pages 404–417. Springer, 2006.

[Biber 2005] Peter Biber, Tom Duckett*et al. Dynamic maps for long-term operation of mobile service robots.* In Robotics: science and systems, pages 17–24, 2005.

[Blanco-Claraco 2014] José-Luis Blanco-Claraco, Francisco-Angel Moreno-Duenas and Javier González-Jiménez. *The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario.* The International Journal of Robotics Research, vol. 33, no. 2, pages 207–214, 2014.

[Bokovoy 2017] Andrey Bokovoy and Konstantin Yakovlev. *Original loop-closure detection algorithm for monocular vslam.* In International Conference on Analysis of Images, Social Networks and Texts, pages 210–220. Springer, 2017.

[Bosse 2004] Michael Bosse, Paul Newman, John Leonard and Seth Teller. *Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework.* The International Journal of Robotics Research, vol. 23, no. 12, pages 1113–1139, 2004.

[Bouaziz 2020] Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome. *Institut Pascal Long-Term dataset.* In RFIA, 2020.

[Bouaziz 2021] Youssef Bouaziz, Eric Royer, Guillaume Bresson and Michel Dhome. *Keyframes retrieval for robust long-term visual localization in changing conditions.* In 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI), pages 000093–000100. IEEE, 2021.

[Bürki 2016] Mathias Bürki, Igor Gilitschenski, Elena Stumm, Roland Siegwart and Juan Nieto. *Appearance-based landmark selection for efficient long-term visual localization.* In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4137–4143. IEEE, 2016.

[Bürki 2018] Mathias Bürki, Marcin Dymczyk, Igor Gilitschenski, Cesar Cadena, Roland Siegwart and Juan Nieto. *Map management for efficient long-term visual localization in outdoor environments.* In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 682–688. IEEE, 2018.

[Bürki 2019] Mathias Bürki, Cesar Cadena, Igor Gilitschenski, Roland Siegwart and Juan Nieto. *Appearance-based landmark selection for visual localization.* Journal of Field Robotics, vol. 36, no. 6, pages 1041–1073, 2019.

[Cadena 2016] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid and John J Leonard. *Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age.* IEEE Transactions on robotics, vol. 32, no. 6, pages 1309–1332, 2016.

[Caltabiano 2005] Daniele Caltabiano and Giovanni Muscato. A robotic system for volcano exploration. INTECH Open Access Publisher, 2005.

[Carlevaris-Bianco 2015a] Nicholas Carlevaris-Bianco, Arash K. Ushani and Ryan M. Eustice. *University of Michigan North Campus long-term vision*

and lidar dataset. International Journal of Robotics Research, vol. 35, no. 9, pages 1023–1035, 2015.

[Carlevaris-Bianco 2015b] Nicholas Carlevaris-Bianco, Arash K. Ushani and Ryan M. Eustice. *University of Michigan North Campus long-term vision and lidar dataset.* International Journal of Robotics Research, vol. 35, no. 9, pages 1023–1035, 2015.

[Cen 2018] Sarah H Cen and Paul Newman. *Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions.* In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6045–6052. IEEE, 2018.

[Cheng 2019] Hsu-Yung Cheng, Chih-Chang Yu, Kuo-Chang Hsu, Chi-Chang Chan, Mei-Hui Tseng and Chih-Lung Lin. *Estimating Solar Irradiance on Tilted Surface with Arbitrary Orientations and Tilt Angles.* Energies, vol. 12, no. 8, page 1427, 2019.

[Churchill 2012] Winston Churchill and Paul Newman. *Practice makes perfect? managing and leveraging visual experiences for lifelong navigation.* In 2012 IEEE International Conference on Robotics and Automation, pages 4525–4532. IEEE, 2012.

[Churchill 2013] Winston Churchill and Paul Newman. *Experience-based navigation for long-term localisation.* The International Journal of Robotics Research, vol. 32, no. 14, pages 1645–1661, 2013.

[Cleveland 1979] William S Cleveland. *Robust locally weighted regression and smoothing scatterplots.* Journal of the American statistical association, vol. 74, no. 368, pages 829–836, 1979.

[Cordts 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. *The cityscapes dataset for semantic urban scene understanding.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3213–3223, 2016.

[Cui 2019] Linyan Cui and Fei Wen. *A monocular ORB-SLAM in dynamic environments.* In Journal of Physics: Conference Series, volume 1168, page 052037. IOP Publishing, 2019.

[Davison 2002] Andrew J Davison and David W. Murray. *Simultaneous localization and map-building using active vision.* IEEE transactions on pattern analysis and machine intelligence, vol. 24, no. 7, pages 865–880, 2002.

[Davison 2003] Andrew J Davison. *Real-time simultaneous localisation and mapping with a single camera.* In Computer Vision, IEEE International Conference on, volume 3, pages 1403–1403. IEEE Computer Society, 2003.

[Davison 2007] Andrew J Davison, Ian D Reid, Nicholas D Molton and Olivier Stasse. *MonoSLAM: Real-time single camera SLAM.* IEEE transactions on pattern analysis and machine intelligence, vol. 29, no. 6, pages 1052–1067, 2007.

[Dellaert 1999] Frank Dellaert, Dieter Fox, Wolfram Burgard and Sebastian Thrun. *Monte carlo localization for mobile robots.* In Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), volume 2, pages 1322–1328. IEEE, 1999.

[Dellaert 2006] Frank Dellaert and Michael Kaess. *Square Root SAM: Simultaneous localization and mapping via square root information smoothing.* The International Journal of Robotics Research, vol. 25, no. 12, pages 1181–1203, 2006.

[DeTone 2018] Daniel DeTone, Tomasz Malisiewicz and Andrew Rabinovich. *Superpoint: Self-supervised interest point detection and description.* In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 224–236, 2018.

[Di 2008] Kaichang Di, Fengliang Xu, Jue Wang, Sanchit Agarwal, Evgenia Brodyagina, Rongxing Li and Larry Matthies. *Photogrammetric processing of rover imagery of the 2003 Mars Exploration Rover mission.* ISPRS Journal of Photogrammetry and Remote Sensing, vol. 63, no. 2, pages 181–201, 2008.

[Diaz-Escobar 2018] Julia Diaz-Escobar, Vitaly Kober and Jose A Gonzalez-Fraga. *LUIFT: LUminance Invariant Feature Transform.* Mathematical Problems in Engineering, vol. 2018, pages 1–17, 2018.

[Dusmanu 2019] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii and Torsten Sattler. *D2-net: A trainable cnn for joint description and detection of local features.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8092–8101, 2019.

[Dymczyk 2015] Marcin Dymczyk, Simon Lynen, Titus Cieslewski, Michael Bosse, Roland Siegwart and Paul Furgale. *The gist of maps-summarizing experience for lifelong localization.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 2767–2773. IEEE, 2015.

[Eade 2006] Ethan Eade and Tom Drummond. *Scalable monocular SLAM.* In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 1, pages 469–476. IEEE, 2006.

[Engel 2014] Jakob Engel, Thomas Schöps and Daniel Cremers. *LSD-SLAM: Large-scale direct monocular SLAM.* In European conference on computer vision, pages 834–849. Springer, 2014.

[Engel 2015] Jakob Engel, Jörg Stückler and Daniel Cremers. *Large-scale direct SLAM with stereo cameras.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1935–1942. IEEE, 2015.

[Engelhard 2011] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm and Wolfram Burgard. *Real-time 3D visual SLAM with a hand-held RGB-D camera.* In Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden, volume 180, pages 1–15, 2011.

[Folkesson 2003] John Folkesson and Henrik Christensen. *Outdoor exploration and slam using a compressed filter.* In ICRA, pages 419–426, 2003.

[Fox 1999] Dieter Fox, Wolfram Burgard, Frank Dellaert and Sebastian Thrun. *Monte carlo localization: Efficient position estimation for mobile robots.* AAAI/IAAI, vol. 1999, no. 343-349, pages 2–2, 1999.

[Gallant 2016] Marc J Gallant and Joshua A Marshall. *Automated rapid mapping of joint orientations with mobile LiDAR.* International Journal of Rock Mechanics and Mining Sciences, vol. 90, pages 1–14, 2016.

[Geiger 2013] Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun. *Vision meets robotics: The kitti dataset.* The International Journal of Robotics Research, vol. 32, no. 11, pages 1231–1237, 2013.

[Grisetti 2007] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka and Wolfram Burgard. *A tree parameterization for efficiently computing maximum likelihood maps using gradient descent.* In Robotics: Science and Systems, volume 3, page 9, 2007.

[Guivant 2001] Jose Guivant and Eduardo Nebot. *Compressed filter for real time implementation of simultaneous localization and map building.* In FSR 2001 International Conference on Field and Service Robots, volume 1, pages 309–314, 2001.

[Gutmann 1999] J-S Gutmann and Kurt Konolige. *Incremental mapping of large cyclic environments.* In Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No. 99EX375), pages 318–325. IEEE, 1999.

[Halodová 2019] Lucie Halodová, Eliska Dvoráková, Filip Majer, Tomás Vintr, Oscar Martinez Mozos, Feras Dayoub and Tomás Krajník. *Predictive and adaptive maps for long-term visual navigation in changing environments.* In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7033–7039. IEEE, 2019.

[Han 2015] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar and Alexander C Berg. *Matchnet: Unifying feature and metric learning for patch-based matching.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3279–3286, 2015.

[Harris 1988] Christopher G Harris, Mike Stephens*et al.* *A combined corner and edge detector.* In Alvey vision conference, volume 15, pages 10–5244. Citeseer, 1988.

[Hasan 2014] Kazi Mahmud Hasan, Khondker Jahid Reza*et al.* *Path planning algorithm development for autonomous vacuum cleaner robots.* In 2014 International Conference on Informatics, Electronics & Vision (ICIEV), pages 1–6. IEEE, 2014.

[Holmquist 2017] Karl Holmquist. *SLAMIt A Sub-Map Based SLAM System: Online creation of multi-leveled map*, 2017.

[Jatzkowski 2018] Inga Jatzkowski, Daniel Wilke and Markus Maurer. *A Deep-Learning Approach for the Detection of Overexposure in Automotive Camera Images.* In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 2030–2035. IEEE, 2018.

[Julier 2001] Simon J Julier and Jeffrey K Uhlmann. *A counter example to the theory of simultaneous localization and map building.* In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), volume 4, pages 4238–4243. IEEE, 2001.

[Kaess 2008] Michael Kaess, Ananth Ranganathan and Frank Dellaert. *iSAM: Incremental smoothing and mapping.* IEEE Transactions on Robotics, vol. 24, no. 6, pages 1365–1378, 2008.

[Kaess 2012] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard and Frank Dellaert. *iSAM2: Incremental smoothing and mapping using the Bayes tree.* The International Journal of Robotics Research, vol. 31, no. 2, pages 216–235, 2012.

[Kim 2011] Heung-Nam Kim, Abdulmotaleb El-Saddik and Geun-Sik Jo. *Collaborative error-reflected models for cold-start recommender systems.* Decision Support Systems, vol. 51, no. 3, pages 519–531, 2011.

[Klein 2007] Georg Klein and David Murray. *Parallel tracking and mapping for small AR workspaces.* In 2007 6th IEEE and ACM international symposium on mixed and augmented reality, pages 225–234. IEEE, 2007.

[Konolige 2010] Kurt Konolige, James Bowman, JD Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit and Pascal Fua. *View-based maps.* The International Journal of Robotics Research, vol. 29, no. 8, pages 941–957, 2010.

[Krajník 2016] Tomás Krajník, Jaime Pulido Fentanes, Marc Hanheide and Tom Duckett. *Persistent localization and life-long mapping in changing environments using the frequency map enhancement.* In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4558–4563. IEEE, 2016.

[Kschischang 2001] Frank R Kschischang, Brendan J Frey and H-A Loeliger. *Factor graphs and the sum-product algorithm.* IEEE Transactions on information theory, vol. 47, no. 2, pages 498–519, 2001.

[Kümmerle 2011] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige and Wolfram Burgard. *g 2 o: A general framework for graph optimization.* In 2011 IEEE International Conference on Robotics and Automation, pages 3607–3613. IEEE, 2011.

[Lébraly 2011] Pierre Lébraly, Eric Royer, Omar Ait-Aider, Clément Deymier and Michel Dhome. *Fast calibration of embedded non-overlapping cameras.* In 2011 IEEE international conference on robotics and automation, pages 221–227. IEEE, 2011.

[Lébraly 2012] Pierre Lébraly. *Etalonnage de caméras à champs disjoints et reconstruction 3D: Application à un robot mobile.* PhD thesis, 2012.

[Linegar 2015] Chris Linegar, Winston Churchill and Paul Newman. *Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 90–97. IEEE, 2015.

[Liu 2020] Zhaoqin Liu, Kaichang Di, Jian Li, Jianfeng Xie, Xiaofeng Cui, Luhua Xi, Wenhui Wan, Man Peng, Bin Liu, Yexin Wang*et al. Landing site topographic mapping and rover localization for Changâe-4 mission.* Science China Information Sciences, vol. 63, no. 4, pages 1–12, 2020.

[Lowe 2004] David G Lowe. *Distinctive image features from scale-invariant keypoints.* International journal of computer vision, vol. 60, no. 2, pages 91–110, 2004.

[Lu 1997] Feng Lu and Evangelos Milios. *Globally consistent range scan alignment for environment mapping.* Autonomous robots, vol. 4, no. 4, pages 333–349, 1997.

[MacTavish 2017] Kirk MacTavish, Michael Paton and Timothy D Barfoot. *Visual triage: A bag-of-words experience selector for long-term visual route following.* In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 2065–2072. IEEE, 2017.

[MacTavish 2018] Kirk MacTavish, Michael Paton and Timothy D Barfoot. *Selective memory: Recalling relevant experience for long-term visual localization.* Journal of Field Robotics, vol. 35, no. 8, pages 1265–1292, 2018.

[Maddern 2014] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill and Paul Newman. *Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles.* In Proceedings of the Visual Place Recognition in Chang-

ing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, volume 2, page 3, 2014.

[Maddern 2017] Will Maddern, Geoff Pascoe, Chris Linegar and Paul Newman. *1 Year, 1000km: The Oxford RobotCar Dataset.* The International Journal of Robotics Research (IJRR), vol. 36, no. 1, pages 3–15, 2017.

[Maddern 2020] Will Maddern, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans and Paul Newman. *Real-time Kinematic Ground Truth for the Oxford RobotCar Dataset.* arXiv preprint arXiv: 2002.10152, 2020.

[Magnago 2019] Valerio Magnago, Luigi Palopoli, Roberto Passerone, Daniele Fontanelli and David Macii. *Effective landmark placement for robot indoor localization with position uncertainty constraints.* IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pages 4443–4455, 2019.

[Meilland 2013] Maxime Meilland and Andrew I Comport. *On unifying key-frame and voxel-based dense visual SLAM at large scales.* In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3677–3683. IEEE, 2013.

[Michalsky 1988] Joseph J Michalsky. *The astronomical almanac's algorithm for approximate solar position (1950–2050).* Solar energy, vol. 40, no. 3, pages 227–235, 1988.

[Milford 2012] Michael J Milford and Gordon F Wyeth. *SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights.* In 2012 IEEE International Conference on Robotics and Automation, pages 1643–1649. IEEE, 2012.

[Montemerlo 2002] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit*et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem.* Aaai/iaai, vol. 593598, 2002.

[Mouragnon 2006] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser and Patrick Sayd. *Monocular vision based SLAM for mobile robots.* In 18th International Conference on Pattern Recognition (ICPR'06), volume 3, pages 1027–1031. IEEE, 2006.

[Mühlfellner 2016] Peter Mühlfellner, Mathias Bürki, Michael Bosse, Wojciech Derendarz, Roland Philippsen and Paul Furgale. *Summary maps for lifelong visual localization.* Journal of Field Robotics, vol. 33, no. 5, pages 561–590, 2016.

[Mur-Artal 2015] Raul Mur-Artal, Jose Maria Martinez Montiel and Juan D Tardos. *ORB-SLAM: a versatile and accurate monocular SLAM system.* IEEE transactions on robotics, vol. 31, no. 5, pages 1147–1163, 2015.

[Murillo 2007] Ana Cris Murillo, José Jesús Guerrero and C Sagues. *Surf features for efficient robot localization with omnidirectional images.* In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 3901–3907. IEEE, 2007.

[Murillo 2009] Ana C Murillo and Jana Kosecka. *Experiments in place recognition using gist panoramas.* In 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pages 2196–2203. IEEE, 2009.

[Naseer 2014] Tayyab Naseer, Luciano Spinello, Wolfram Burgard and Cyrill Stachniss. *Robust visual robot localization across seasons using network flows.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

[Newman 1999] Paul Newman. *On the structure and solution of the simultaneous localisation and map building problem.* 1999.

[Nistér 2004] David Nistér, Oleg Naroditsky and James Bergen. *Visual odometry.* In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., volume 1, pages I–I. Ieee, 2004.

[Oliva 2001] Aude Oliva and Antonio Torralba. *Modeling the shape of the scene: A holistic representation of the spatial envelope.* International journal of computer vision, vol. 42, no. 3, pages 145–175, 2001.

[Olson 2006] Edwin Olson, John Leonard and Seth Teller. *Fast iterative alignment of pose graphs with poor initial estimates.* In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 2262–2269. IEEE, 2006.

[Ono 2018] Yuki Ono, Eduard Trulls, Pascal Fua and Kwang Moo Yi. *LF-Net: Learning local features from images.* arXiv preprint arXiv:1805.09662, 2018.

[Pandey 2011] Gaurav Pandey, James R McBride and Ryan M Eustice. *Ford Campus vision and lidar data set.* The International Journal of Robotics Research, vol. 30, no. 13, pages 1543–1552, 2011.

[Pascoe 2017] Geoffrey Pascoe, Will Maddern, Michael Tanner, Pedro Piniés and Paul Newman. *Nid-slam: Robust monocular slam using normalised information distance.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1435–1444, 2017.

[Pepperell 2016] Edward Pepperell, Peter Corke and Michael Milford. *Routed roads: Probabilistic vision-based place recognition for changing conditions, split streets and varied viewpoints.* The International Journal of Robotics Research, vol. 35, no. 9, pages 1057–1179, 2016.

[Petry 2013] Marcelo R Petry, António Paulo Moreira and Luís Paulo Reisinst. *Increasing illumination invariance of SURF feature detector through color constancy.* In Portuguese Conference on Artificial Intelligence, pages 259–270. Springer, 2013.

[Pollefeys 2004] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops and Reinhard Koch. *Visual modeling with a hand-held camera.* International Journal of Computer Vision, vol. 59, no. 3, pages 207–232, 2004.

[Pollefeys 2008] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell*et al. Detailed real-time urban 3d reconstruction from video.* International Journal of Computer Vision, vol. 78, no. 2, pages 143–167, 2008.

[Quigley 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler and Andrew Y Ng. *ROS: an open-source Robot Operating System.* In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[Raghu 2014] N Raghu, KN Manjunatha and B Kiran. *Tracking of satellites by using Phased Array Antenna.* In 2014 International Conference on Electronics and Communication Systems (ICECS), pages 1–6. IEEE, 2014.

[Royer 2016] Eric Royer, François Marmoiton, Serge Alizon, Datta Ramadasan, Morgan Slade, Ange Nizard, Michel Dhome, Benoit Thuilot and Florent Bonjean. *Lessons learned after more than 1000 km in an autonomous shuttle guided by vision.* In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pages 2248–2253. IEEE, 2016.

[Sala 2006] Pablo Sala, Robert Sim, Ali Shokoufandeh and Sven Dickinson. *Landmark selection for vision-based navigation.* IEEE Transactions on robotics, vol. 22, no. 2, pages 334–349, 2006.

[Sattler 2018] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic*et al. Benchmarking 6dof outdoor visual localization in changing conditions.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8601–8610, 2018.

[Schaffalitzky 2002] Frederik Schaffalitzky and Andrew Zisserman. *Multi-view matching for unordered image sets, or âHow do I organize my holiday snaps?â.* In European conference on computer vision, pages 414–431. Springer, 2002.

[Sheena 2016] S Sheena and M Sheena. *A comparison of SIFT and SURF algorithm for the recognition of an efficient iris biometric system.* International Journal

of Advanced Research in Computer and Communication Engineering, vol. 5, 2016.

[Shi 1994] Jianbo Shi*et al.* *Good features to track.* In 1994 Proceedings of IEEE conference on computer vision and pattern recognition, pages 593–600. IEEE, 1994.

[Shoman 2020] Sota Shoman, Tomohiro Mashita, Alexander Plopski, Photchara Ratsamee and Yuki Uranishi. *Real-to-Synthetic Feature Transform for Illumination Invariant Camera Localization.* IEEE Computer Graphics and Applications, 2020.

[Siegwart 2011] Roland Siegwart, Illah Reza Nourbakhsh and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

[Simo-Serra 2015] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua and Francesc Moreno-Noguer. *Discriminative learning of deep convolutional feature point descriptors.* In Proceedings of the IEEE International Conference on Computer Vision, pages 118–126, 2015.

[Sinriech 2000] David Sinriech and Shraga Shoval. *Landmark configuration for absolute positioning of autonomous vehicles.* IIE Transactions, vol. 32, no. 7, pages 613–624, 2000.

[Smith 1986] Randall C Smith and Peter Cheeseman. *On the representation and estimation of spatial uncertainty.* The international journal of Robotics Research, vol. 5, no. 4, pages 56–68, 1986.

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

[Suenderhauf 2015] Niko Suenderhauf. *The VPRiCE Challenge 2015 â Visual Place Recognition in Changing Environments*, 2015.

[Sun 2020] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine*et al.* *Scalability in perception for autonomous driving: Waymo open dataset.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2446–2454, 2020.

[Sutherland 1993] Karen T Sutherland and William B Thompson. *Inexact navigation.* In [1993] Proceedings IEEE International Conference on Robotics and Automation, pages 1–7. IEEE, 1993.

[Szekely 2005] Gabor J Szekely, Maria L Rizzo*et al.* *Hierarchical clustering via joint between-within distances: Extending Ward's minimum variance method.* Journal of classification, vol. 22, no. 2, pages 151–184, 2005.

[Tardós 2002] Juan D Tardós, José Neira, Paul M Newman and John J Leonard. *Robust mapping and localization in indoor environments using sonar data.* The International Journal of Robotics Research, vol. 21, no. 4, pages 311–330, 2002.

[Thrun 2001] Sebastian Thrun, Dieter Fox, Wolfram Burgard and Frank Dellaert. *Robust Monte Carlo localization for mobile robots.* Artificial intelligence, vol. 128, no. 1-2, pages 99–141, 2001.

[Thrun 2006] Sebastian Thrun and Michael Montemerlo. *The graph SLAM algorithm with applications to large-scale mapping of urban structures.* The International Journal of Robotics Research, vol. 25, no. 5-6, pages 403–429, 2006.

[Tian 2017] Yurun Tian, Bin Fan and Fuchao Wu. *L2-net: Deep learning of discriminative patch descriptor in euclidean space.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 661–669, 2017.

[Tian 2019] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen and Vassileios Balntas. *Sosnet: Second order similarity regularization for local descriptor learning.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 11016–11025, 2019.

[Tomasi 1992] Carlo Tomasi and Takeo Kanade. *Shape and motion from image streams under orthography: a factorization method.* International journal of computer vision, vol. 9, no. 2, pages 137–154, 1992.

[Triggs 1999] Bill Triggs, Philip F McLauchlan, Richard I Hartley and Andrew W Fitzgibbon. *Bundle adjustmentâa modern synthesis.* In International workshop on vision algorithms, pages 298–372. Springer, 1999.

[Ullman 1979] Shimon Ullman. *The interpretation of structure from motion.* Proceedings of the Royal Society of London. Series B. Biological Sciences, vol. 203, no. 1153, pages 405–426, 1979.

[Vanicek 2018] P Vanicek and L Beran. *Navigation of robotics platform in unknown spaces using LIDAR, Raspberry PI and hector slam.* Journal of Fundamental and Applied Sciences, vol. 10, no. 3S, pages 494–506, 2018.

[Verdie 2015] Yannick Verdie, Kwang Yi, Pascal Fua and Vincent Lepetit. *Tilde: A temporally invariant learned detector.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5279–5288, 2015.

[Vysotska 2015] Olga Vysotska, Tayyab Naseer, Luciano Spinello, Wolfram Burgard and Cyrill Stachniss. *Efficient and effective matching of image sequences under substantial appearance changes exploiting GPS priors.* In 2015 IEEE

International Conference on Robotics and Automation (ICRA), pages 2774–2779. IEEE, 2015.

[WANG 2014] BaoFeng WANG, JianLiang ZHOU, GeShi TANG, KaiChang DI, WenHui WAN, ChuanKai LIU and Jia WANG. *Research on visual localization method of lunar rover.* Scientia Sinica Informationis, vol. 44, no. 4, pages 452–460, 2014.

[Weidner 2017] Nick Weidner, Sharmin Rahman, Alberto Quattrini Li and Ioannis Rekleitis. *Underwater cave mapping using stereo vision.* In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 5709–5715. IEEE, 2017.

[Williams 2001] Stefan Bernard Williams. *Efficient solutions to autonomous mapping and navigation problems.* 2001.

[Xiang 2020] Zejun Xiang, Ronghua Yang, Chang Deng, Mingxing Teng, Mengkun She and Degui Teng. *An Illumination Insensitive Descriptor Combining the CSLBP Features for Street View Images in Augmented Reality: Experimental Studies.* ISPRS International Journal of Geo-Information, vol. 9, no. 6, page 362, 2020.

[Xie 2016] Jun Xie, Martin Kiefel, Ming-Ting Sun and Andreas Geiger. *Semantic instance annotation of street scenes by 3d to 2d label transfer.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3688–3697, 2016.

[Yan 2019] Zhi Yan, Li Sun, Tomas Krajnik and Yassine Ruichek. *EU long-term dataset with multiple sensors for autonomous driving.* arXiv preprint arXiv:1909.03330, 2019.

[Yi 2016a] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit and Pascal Fua. *Lift: Learned invariant feature transform.* In European Conference on Computer Vision, pages 467–483. Springer, 2016.

[Yi 2016b] Kwang Moo Yi, Yannick Verdie, Pascal Fua and Vincent Lepetit. *Learning to assign orientations to feature points.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 107–116, 2016.

[Yoon 2006] Suk-June Yoon, Hyun-Do Choi, Sung-Kee Park, Soo-Hyun Kim and Yoon-Keun Kwak. *Simultaneous Localization & Map-building of Mobile Robot in the Outdoor Environments by Vision-based Compressed Extended Kalman Filter.* Journal of Institute of Control, Robotics and Systems, vol. 12, no. 6, pages 585–593, 2006.

[Zhang 2005] Sen Zhang, Lihua Xie and Martin David Adams. *Entropy based feature selection scheme for real time simultaneous localization and map building.* In

2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1175–1180. IEEE, 2005.

[Zhou 2020] Qianyu Zhou, Zhengyang Feng, Guangliang Cheng, Xin Tan, Jianping Shi and Lizhuang Ma. *Uncertainty-aware consistency regularization for cross-domain semantic segmentation.* arXiv preprint arXiv:2004.08878, 2020.