Localisation an mapping based on laser rangefinder and vision coupling for the autonomous navigation of a mobile robot

Yassine Ahmine

Thèse de Doctorat en Cotutelle

Mention Sciences pour l'Ingénieur
Spécialité Robotique

présentée à *l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)* **de**
l'Université de Picardie Jules Verne
et
à la Faculté de Technologie – Département d'Electronique
de l'Université Amar Telidji de Laghouat

par

Yassine Ahmine

pour obtenir le grade de Docteur de l'Université de Picardie Jules Verne

Localisation et cartographie basées sur le couplage Télémètre Laser et Vision pour la navigation autonome d'un robot mobile

Soutenue le 09/06/2021, après avis des rapporteurs, devant le jury d'examen :

M. Mohammed Belkheiri, Professeur, UATL	**Président**
M. Ezio Malis, Chargée de Recherches HDR, INRIA Sophia Antipolis	**Rapporteur**
M. José M. Martinez Montiel, Professeur, Universidad de Zaragoza	**Rapporteur**
M. Noureddine Ouadah, Maitre de Recherche A, CDTA	**Examinateur**
M^me^ Fatima Chouireb, Professeur, UATL	**Directeur de thèse**
M. El Mustapha Mouaddib, Professeur, UPJV	**Co-encadrant**
M. Guillaume Caron, Maître de Conférences HDR, UPJV / CNRS	**Co-encadrant**

Acknowledgements

Before diving into the research details, I would like, first of all, to particularly thank my supervisors: Fatima Chouireb, Guillaume Caron, and El Mustapha Mouaddib. Their commitment, help, and demand made me grow and permitted me to become better.

I also have special thanks to Mohammed Belkheiri and Abdelhamid Rabhi for their kindness and support.

Hocine, Khalil, Farouk, Noureddine, Tawsif, Eder, Monika, Jordan, Thibault, Clément, Julien, Audrey and all the other colleges from the MIS and LTSS thank you for all the good times we had. It was great!

My friends: Sarah, Walid, Hamza, Walid, Chawki, Sarah, Mohamed(s), Cherif, Hame ... Thank you for your support. Well, thank you for trying to be supportive and for making fun of me of course!

I would like of course to thank my parents and my sister without whom none of this would have been possible. It's all thanks to you if I made this.

Finally, thanks to anyone who helped me from near or far finish this thesis. It was a day to day work that I think particularly suits these words of Winston Churchill:

"**Success Is Going from Failure to Failure Without Losing Your Enthusiasm**".

Contents

List of Figures

List of Acronyms

SLAM:	Simultaneous Localization And Mapping.
VO:	Visual Odometry.
V-SLAM:	Visual Simultaneous Localization And Mapping.
EKF:	Extended Kalman Filter.
LRF:	Laser Range Finder.
DTAM:	Dense Tracking And Mapping.
SVO:	Semi direct monocular Visual Odometry.
LSD-SLAM:	Large-Scale Direct monocular Simultaneous Localization And Mapping.
ORB-SLAM:	Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping.
DSO:	Direct Sparse Odometry.
IMU:	Inertial Measurement Units.
RGB-D:	Red, Green, Blue, and Depth.
LK:	Lucas Kanade.
VIO:	Visual Inertial Odometry.
Mono-VO:	Monocular Visual Odometry.
RIEKF:	Right Invariant Extended Kalman Filter.
MAV:	Micro Aerial Vehicles.
GMM:	Gaussian Mixture Model.
VOCUS:	Visual Object detection with a CompUtational attention System.
CIE:	Commission Internationale de l'Eclairage.
DoG:	Difference of Gaussians.
AIC:	Akaike Information Criteria.
ATE:	Absolute Trajectory Error.
RMSE:	Root Mean Squared Error.
UKF-LG:	Unscented Kalman Filter on Lie Groups.
L-UKF-LG:	Left Unscented Kalman Filter on Lie Groups.
R-UKF-LG:	Right Unscented Kalman Filter on Lie Groups.
SE(3)-UKF:	Special Euclidean group Unscented Kalman Filter.
SURF:	Speeded Up Robust Features.
AFA-LK:	Adaptive Forwards Additive Lucas Kanade.
FA-LK:	Forwards Additive Lucas Kanade.
RANSAC:	RANdom SAmple Consensus.

SIFT:	Scale-Invariant Feature Transform.
ESM:	Efficient Second order Minimization.
Conditional-LK:	Conditional Lucas Kanade.
MS-COCO:	MicroSoft Comon Objects in COntext.
AFAMIS:	Adaptive Forwards Additive "Modélisation, Information, et Systèmes".
ECD:	Error Cumulative Distribution.
IC-LK:	Inverse Compositional Lucas Kanade.
FC-LK:	Forwards Compositional Lucas Kanade.
SDM:	Supervised Descent Method.
CVO:	Continuous direct sparse Visual Odometry.
OP-B:	Optimized Pyramid-Based.
OP-B:	Photometric Pyramid-Based.
3D:	Three Dimensions.
IRLS-ICP:	Iteratively Re-weighted Least Squares Iterative Closest Point.
TUM:	Technical University of Munich.
RPE:	Relative Pose Error.
CNN:	Convolutional Neural Networks.

Research Articles of the Author

The contributions presented in this dissertation were published or are submitted for publication through the following articles:

- Y. Ahmine, F. Chouireb and A. Bencherif, "Monocular SLAM based on salient regions detection," 2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B), Boumerdes, 2017, pp. 1-6. Related to Chapter 4.
- Y. Ahmine, F. Chouireb, G. Caron, "Feature Points Distribution Based on GMM and Salient Regions for Monocular SLAM", to be submitted to Signal, Image and Video Processing journal. Related to Chapter 4.
- Y. Ahmine, G. Caron, E. Mouaddib, F. Chouireb, "Adaptive Lucas-Kanade tracking," Image and Vision Computing, Volume 88, 2019, Pages 1-8. https://hal.archives-ouvertes.fr/hal-02193928. Related to Chapter 5
- Y. Ahmine, G. Caron, F. Chouireb, E. Mouaddib, "Continuous Scale-Space Direct Image Alignment for Visual Odometry from RGB-D Images". IEEE Robotics and Automation Letters. IEEE in press, 2021. https://hal.archives-ouvertes.fr/hal-03130945. (Accepted for presentation in ICRA 2021) Related to Chapter 6.

Chapitre 1

Résumé de la thése

1.1 Introduction

Le travail de recherche présenté dans cette thèse porte sur les problématiques liées aux deux domaines que sont *l'odométrie visuelle* (dénotée VO) ainsi que *la localisation et la cartographie simultanées visuelles* (dénotées V-SLAM). Deux domaines qui sont équivalents dans le sens où tous deux cherchent à estimer la trajectoire d'une caméra se déplaçant dans l'espace ainsi que la carte de son environnement. La différence entre eux étant que les approches de type VO produisent des cartes locales alors que celles de type V-SLAM permettent de générer des cartes globales. La cohérence de l'estimation représente elle aussi un point de différenciation entre ces deux types d'approches. Les algorithmes de VO ont ainsi pour objectif d'assurer la cohérence locale des estimations, alors que ceux de V-SLAM cherchent à assurer la cohérence globale de ces dernières (Scaramuzza and Fraundorfer, 2011).

1.2 Problématiques de la thèse

C'est dans ce contexte que s'inscrit cette thèse qui a pour objectif de proposer des solutions aux problématiques liées à *l'odométrie visuelle* ainsi que *la localisation et la cartographie simultanées visuelles*. En premier lieu, comment initialiser des points d'intérêts adaptés pour l'estimation ? Ce premier point est traité dans le Chapitre 4 de la thèse, en se basant sur le filtrage de Kalman. En second lieu, comment élargir le domaine de convergence d'algorithmes directs d'alignement d'images ? Ce second point est l'objet des chapitres 5 et 6 de la thèse. L'ensemble des chapitres 4, 5 et 6 est décrit dans les sections 1.3, 1.4 et 1.5.

1.3 Détection de régions saillantes pour l'odométrie visuelle et l'odométrie visuelle inertielle

Dans le contexte de l'odométrie visuelle basée amers visuels, la détection de points d'intérêts "fiables" joue un rôle clé dans le succès de l'algorithme. Un point d'intérêt est ainsi dit "fiable" s'il est facilement re-détecté (suivi) dans d'autres images. La stratégie d'initialisation de points d'intérêts basée saillance proposée dans le Chapitre 4 a en effet pour objectif de permettre d'initialiser de tels points. Pour ce faire, la stratégie proposée est intégrée à deux systèmes d'odométrie visuelle et d'odométrie visuelle inertielle basées filtrage de Kalman ; l'objectif étant de vérifier l'intérêt de cette stratégie au regard de l'amélioration des performances des algorithmes considérés.

D'autres travaux de la littérature considèrent des primitives additionnelles telles que les lignes Pumarola et al. (2017); Zhou et al. (2019), afin d'améliorer les résultats d'estimation. Notre approche, en revanche, ne considère que des points, et n'utilise que l'information des images afin de faciliter leur détection et leur suivi.

1.3.1 Stratégie d'initialisation de points d'intérêts

Durant l'exploration de l'environnement, l'algorithme d'odométrie visuelle basé amers visuels a besoin d'initialiser de nouveaux points d'intérêts à chaque fois que de nouvelles régions sont explorées. Ce procédé consiste en l'application d'un détecteur de points d'intérêts sur des zones spécifiques de l'image, appelées *zones de recherche* (de forme rectangulaire), réparties selon une densité de probabilité uniforme (Civera et al., 2009). Les points d'intérêts détectés sont par conséquent répartis uniformément dans l'image. L'un des inconvénients de cette stratégie cependant, est qu'elle n'exploite pas toute l'information présente dans l'image durant le processus d'initialisation, dans le sens où les régions saillantes ne sont pas favorisées. La proposition de ce travail est d'utiliser l'information de saillance des images dans le processus de détection des points d'intérêts, afin de répartir plus d'amers dans les parties saillantes de l'image tout en gardant une couverture complète de cette dernière.

Les différentes étapes impliquées dans la méthode proposée, illustrées dans la figure 1.1, sont résumées comme suit : L'image d'entrée est d'abord utilisée pour calculer une carte de saillance ; celle-ci est ensuite convertie en une image binaire selon un seuil défini par l'utilisateur ; après cela, les pixels blancs (supérieurs au seuil) sont utilisés pour entraîner un modèle de mélanges Gaussiens (dénoté GMM), qui

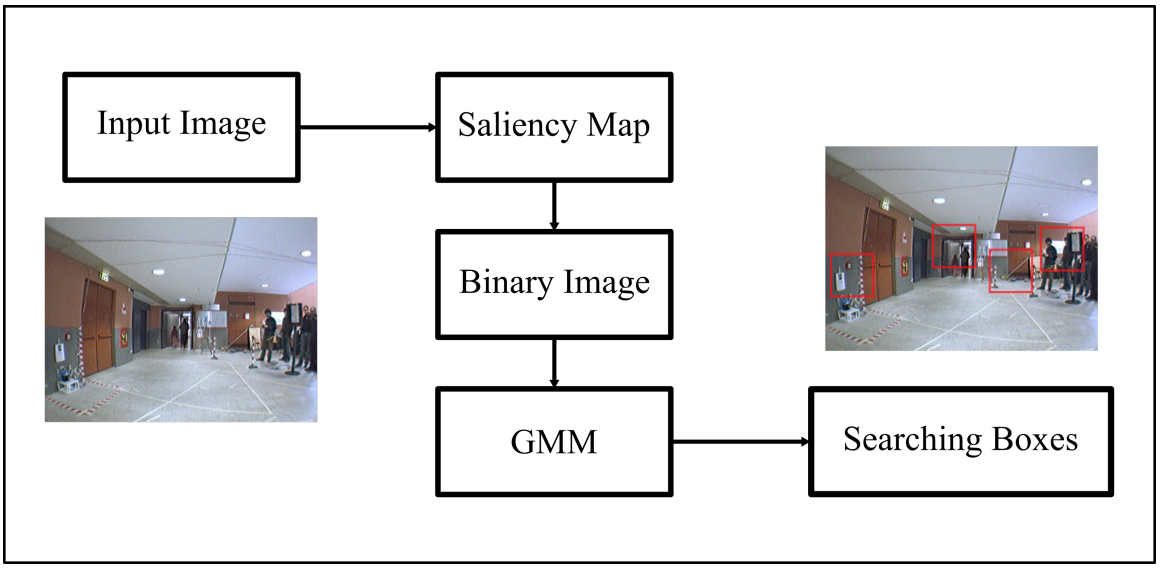


FIGURE *1.1: Présentation de la méthode d'initialisation de points d'intérêts proposée. Les zones de recherche sont illustrées par des rectangles rouges.*

permet de générer les centres des *zones de recherche*. Ces étapes sont décrites plus en détail dans ce qui suit.

Carte de saillance La carte de saillance est calculée à partir de l'image d'entrée en utilisant la partie ascendante ("bottom-up") de l'approche proposée par (Frintrop and Jensfelt, 2008) appelée VOCUS. Cette approche est en fait un système d'attention visuelle d'inspiration humaine qui utilise les variations d'intensité, de couleur et d'orientation de l'image pour générer une carte de saillance. Trois cartes de "visibilité" sont d'abord calculées pour cela :

- **Carte d'intensité :** Obtenues premièrement en calculant la pyramide d'images en niveaux de gris à partir de l'image d'entrée. Ensuite, en appliquant des filtres centre-actifs et centre-inactifs [1] à chaque niveau de la pyramide.
- **Carte de couleur :** Calculée au travers de la transformation de l'espace de couleur de l'image vers le CIE 1976 L*a*b (Schwiegerling, 2004). Des filtres centre-actifs et centre-inactifs sont ensuite appliqués aux quatre pyramides calculées à partir des chaines de couleur (vert, rouge, bleu et jaune) obtenues à partir de l'image transformée.

[1] Un filtrage centre-actif se fait en calculant la différence entre les valeurs des pixels de l'image et leurs voisins. Le filtrage centre-inactif pour sa part se fait en calculant la différence entre les les voisins et le pixel de l'image, ce qui est l'inverse du filtrage centre-actif. Les valeurs négatives sont mises à 0.

- **Carte d'orientation :** Basée sur l'application d'un filtre de Gabor à la pyramide de différence de Gaussiennes calculée à partir de l'image en niveaux de gris.

La carte de saillance est ensuite calculée à partir de la moyenne de ces trois cartes. C'est une image en niveaux de gris dont la valeur d'intensité des pixels correspond à leurs degrés de saillance.

Modèles de mélanges Gaussiens Les modèles de mélanges Gaussiens $\mathcal{G}$, définis dans l'équation suivante, sont des modèles probabilistes qui permettent de modéliser des densités de probabilités multimodales.

$$\mathcal{G}(\boldsymbol{\iota}) = \sum_{j=1}^{k} w_j \, \mathcal{N}(\boldsymbol{\iota}\,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1.1}$$

ou w_j et $\boldsymbol{\iota}$ sont respectivement le poids de la $j$ème Gaussienne et les coordonnées $(u, v)^T$ dans l'image. $\boldsymbol{\mu}$ et $\boldsymbol{\Sigma}$ représentent pour leur part la moyenne et la matrice de covariance de la Gaussienne. Les modèles de mélanges Gaussiens sont utilisés dans ce travail afin de modéliser la distribution spatiale de l'information de saillance ; ceci en convertissant la carte de saillance générée en une image binaire afin de ne garder que les pixels les plus saillants, et utiliser ensuite les coordonnées de ces pixels pour entrainer un GMM. Ce dernier, une fois entrainé, permet de générer les coordonnées u et v des centres des *zones de recherche*. En dernier lieu, un algorithme de détection de points d'intérêts est appliqué dans ces zones de recherche.

Il est important de noter que l'image est divisée en 8 sous-images auxquelles l'approche décrite précédemment est appliquée (figure 1.2), et ce afin de limiter la complexité de l'algorithme.

1.4 Alignement d'images basé espace d'échelles

Cette partie vise à proposer un algorithme qui permet d'utiliser la représentation des images dans l'espace d'échelles dans le contexte de l'alignement **direct** d'images. Ceci pouvant être considéré comme le problème de maximisation de la similarité entre deux images selon un modèle géométrique. En notant que **direct** dans ce contexte veut dire que les valeurs d'intensité des pixels sont directement utilisées par l'algorithme. L'utilisation de la représentation dans l'espace d'échelles permet de contrôler le degré

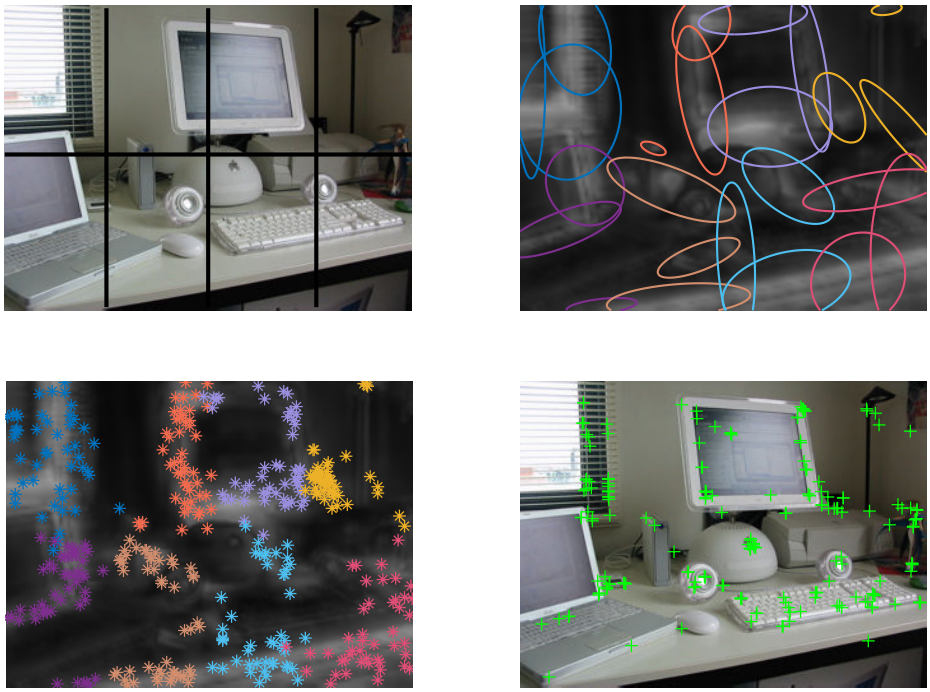


FIGURE *1.2: Carte de saillance ainsi que les centres des zones de recherche en utilisant un GMM pour chaque sous-image. Différentes couleurs correspondent à différents GMMs.*

---

de lissage des images durant le processus d'optimisation, permettant ainsi d'éliminer les minimas locaux qui peuvent amener ce type d'algorithmes à échouer.

1.4.1 Description de la méthode

L'objectif est d'aligner des paires d'images $\{\boldsymbol{I}_1, \boldsymbol{I}_0\}$ obtenues à partir d'une caméra en mouvement dans une scène statique (ou majoritairement statique) selon un modèle géométrique prédéfini. Ce qui équivaut à chercher le vecteur de paramètres $\boldsymbol{\xi}$ d'une fonction de transformation $\boldsymbol{\mathcal{W}}(\mathbf{x}, \boldsymbol{\xi})$ qui permet de lier les coordonnées de la paire $\{\boldsymbol{I}_1, \boldsymbol{I}_0\}$, où $\mathbf{x} = (u, v)^T$ représente les coordonnées du pixel. Ceci est équivalent au problème d'optimisation suivant (Lucas and Kanade, 1981) :

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}} \left[\boldsymbol{I}_1(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi})) - \boldsymbol{I}_0(\mathbf{x})\right]^2 . \tag{1.2}$$

Nous proposons d'utiliser $G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}); \lambda)$ et $G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref})$, les représentations des images $\boldsymbol{I}_1$ et $\boldsymbol{I}_0$ dans l'espace d'échelles (Lindeberg, 2014) selon les paramètres λ et

λ_{ref} respectivement :

$$G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi});\lambda) = \boldsymbol{I}_1(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})) * \mathbf{g}(\mathbf{x};\lambda), \tag{1.3}$$

et

$$G_{\boldsymbol{I}_0}(\mathbf{x};\lambda_{ref}) = \boldsymbol{I}_0(\mathbf{x}) * \mathbf{g}(\mathbf{x};\lambda_{ref}), \tag{1.4}$$

où $*$ est l'opérateur de convolution et $\mathbf{g}(\mathbf{x};\lambda)$ est un noyau Gaussien autour de la moyenne $\boldsymbol{\mu}$:

$$\mathbf{g}(\mathbf{x};\lambda) = \frac{1}{2\pi\lambda^2}\exp(-\frac{(\mathbf{x}-\boldsymbol{\mu})^2}{2\lambda^2}). \tag{1.5}$$

La représentation dans l'espace d'échelles d'une image est illustrée dans la figure 1.3. L'équation (1.2) peut ainsi être reformulée de la manière suivante :

$$\tilde{\boldsymbol{\xi}}^* = \underset{\tilde{\boldsymbol{\xi}}}{\operatorname{argmin}} \frac{1}{2}\sum_{\mathbf{x}} \left[G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi});\lambda) - G_{\boldsymbol{I}_0}(\mathbf{x};\lambda_{ref})\right]^2, \tag{1.6}$$

où $\tilde{\boldsymbol{\xi}} = (\boldsymbol{\xi},\lambda)^T$ est le vecteur augmenté contenant les paramètres géométriques et d'échelle. Ce vecteur de paramètres peut être optimisé de manière itérative en adoptant une approche additive directe, comme montré dans l'expression qui suit :

$$\tilde{\boldsymbol{\xi}} \leftarrow \tilde{\boldsymbol{\xi}} + \alpha\Delta\tilde{\boldsymbol{\xi}}, \tag{1.7}$$

où α est un gain. L'équation (1.6) est un problème de moindre carrés non-linéaire qui peut être résolu en considérant une approche de type Gauss-Newton (Baker and Matthews, 2004). L'expression de l'incrément est ensuite la suivante :

$$\Delta\tilde{\boldsymbol{\xi}} = \mathbf{H}^{-1}\sum_{\mathbf{x}}\left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial\tilde{\boldsymbol{\xi}}}\right]^T \left[\mathbf{G}_{\boldsymbol{I}_0}(\mathbf{x};\lambda_{ref}) - G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi});\lambda)\right], \tag{1.8}$$

$\mathbf{H}$ représente la matrice Hessienne :

$$\mathbf{H} = \sum_{\mathbf{x}}\left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial\tilde{\boldsymbol{\xi}}}\right]^T \left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial\tilde{\boldsymbol{\xi}}}\right]. \tag{1.9}$$

Les paramètres $\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})$ et λ de $G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi});\lambda)$ ont été omis afin de rendre la ligne de *Jacobienne* $\frac{\partial G_{\boldsymbol{I}_1}}{\partial\tilde{\boldsymbol{\xi}}} = \left[\nabla_{u,v}G_{\boldsymbol{I}_1}\frac{\partial\boldsymbol{\mathcal{W}}}{\partial\boldsymbol{\xi}}, \nabla_\lambda G_{\boldsymbol{I}_1}\right]$ d'un pixel $\mathbf{x}$ plus lisible.

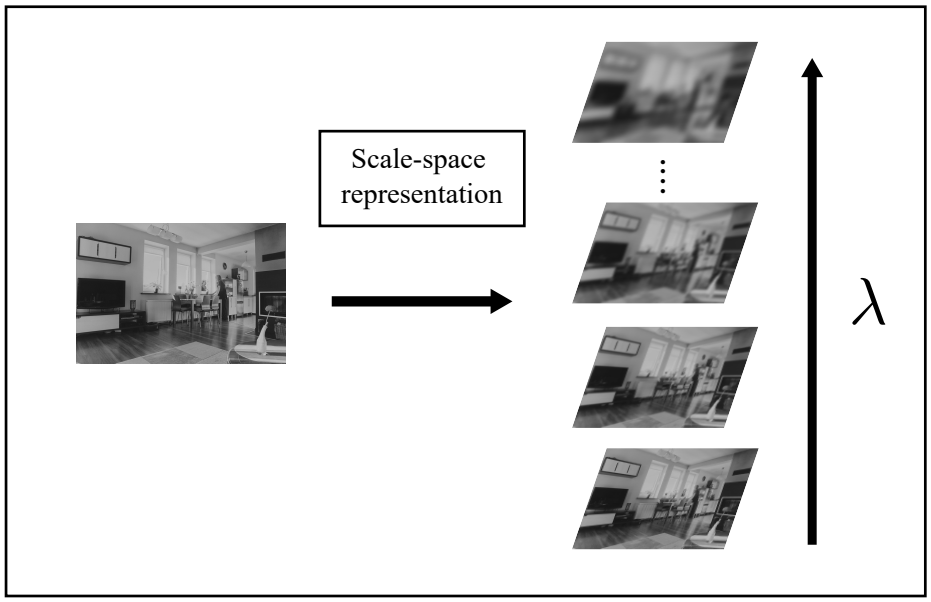


FIGURE *1.3: Représentation d'une image dans l'espace d'échelles. La valeur du paramètre d'échelle contrôle le degré de lissage de l'image.*

Les gradients de $G_{\boldsymbol{I}_1}$ dans l'espace et l'échelle sont évalués à $\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})$ en utilisant les différences finies :

$$\nabla_u G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left.\frac{G_{\boldsymbol{I}_1}(u+l_u,v;\lambda) - G_{\boldsymbol{I}_1}(u-l_u,v;\lambda)}{2l_u}\right|_{\mathbf{x}=\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})} \tag{1.10}$$

$$\nabla_v G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left.\frac{G_{\boldsymbol{I}_1}(u,v+l_v;\lambda) - G_{\boldsymbol{I}_1}(u,v-l_v;\lambda)}{2l_v}\right|_{\mathbf{x}=\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})} \tag{1.11}$$

$$\nabla_\lambda G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left.\frac{\boldsymbol{I}_1 * (\mathbf{g}(\mathbf{x};\lambda+l_\lambda) - \mathbf{g}(\mathbf{x};\lambda-l_\lambda))}{2l_\lambda}\right|_{\mathbf{x}=\mathcal{W}(\mathbf{x};\boldsymbol{\xi})}, \tag{1.12}$$

où l_u, l_v et l_λ sont les pas des différences finies. La *Jacobienne* nécessaire pour le calcul des incréments dépend des gradients $G_{\boldsymbol{I}_1}$ ainsi que de la dérivée de la fonction de transformation par rapport aux paramètres $\boldsymbol{\xi}$.

1.5 Combiner vision et information de profondeur pour VO et SLAM

Dans le Chapitre 6, nous considérons l'intensité des pixels ainsi que l'information de profondeur pour estimer de le déplacement d'un agent se déplaçant dans l'espace 3D. L'algorithme résultant représente le bloc de base d'un système SLAM présenté dans le même chapitre.

1.5.1 Alignement d'images RGB-D

Cette partie présente l'algorithme d'alignement d'images RGB-D proposé, qui utilise, à chaque instant t, les dernières images non-distordues ($\boldsymbol{I}_t$, $\boldsymbol{I}_{t-1}$) ainsi que les cartes de profondeur correspondantes ($\boldsymbol{D}_t$, $\boldsymbol{D}_{t-1}$), afin d'estimer le déplacement relatif de la caméra en 3D. L'approche proposée utilise la représentation des images dans l'espace d'échelles dans le but d'augmenter la robustesse aux déplacements importants de la caméra. La différence principale avec l'AFA-LK, présenté dans le Chapitre 5, réside dans le fait que l'estimation du déplacement de la caméra se fait au travers de l'optimisation jointe des paramètres géométriques et d'échelle dans chaque niveau d'une pyramide d'images. Les pyramides d'images sont aussi utilisées par la seconde approche considérée dans le Chapitre 6, l'algorithme proposé par (Steinbruecker et al., 2011). Ce dernier utilise directement les valeurs d'intensité des pixels dans le processus d'estimation. La différence principale entre les deux approches étant que celle proposée adapte l'échelle de l'image de manière automatique dans chaque niveau de la pyramide d'image durant l'optimisation. L'approche proposée est ainsi nommée OP-B, alors que celle de (Steinbruecker et al., 2011) qui n'adapte pas l'échelle de l'image est nommée PP-B. En effet, PP-B utilise une échelle constante liée à la résolution de l'image dans la pyramide d'images.

L'approche proposée représente les images dans l'espace d'échelles. L'expression de l'erreur résultante peut ainsi être exprimée de la manière suivante :

$$G_{r_i}(\boldsymbol{\xi}) = G_{I_1}(\mathcal{W}(\mathbf{x}, \boldsymbol{\xi}); \lambda) - G_{I_0}(\mathbf{x}; \lambda_{ref}). \tag{1.13}$$

En empilant le vecteur de paramètres géométriques et le paramètre d'échelle $\tilde{\boldsymbol{\xi}} = (\boldsymbol{\xi}^T, \lambda)^T$, et après la linéarisation de l'équation (1.13), nous obtenons la fonction de coût suivante :

$$\tilde{\boldsymbol{\xi}}^* = \underset{\tilde{\boldsymbol{\xi}}}{\operatorname{argmin}} \sum_i \left[G_{r_i}(0) + \tilde{\mathbf{J}}_i \, \Delta\tilde{\boldsymbol{\xi}} \right]^2, \tag{1.14}$$

où $G_{r_i}(0) = G_{I_1}(\mathcal{W}(\mathbf{x}, 0); \lambda) - G_{I_0}(\mathbf{x}; \lambda_{ref})$ et la ligne de jacobienne correspondant au $i$ème pixel est calculée comme suit :

$$\tilde{\mathbf{J}}_i = \left[\frac{\partial G_{I_1}}{\partial \boldsymbol{\xi}}, \frac{\partial G_{I_1}}{\partial \lambda} \right]. \tag{1.15}$$

où $\frac{\partial G_{I_1}}{\partial \lambda}$ est calculé en utilisant les différences finies, comme dans l'équation (1.12). Ceci résulte dans l'expression suivante :

$$\Delta\tilde{\boldsymbol{\xi}} = \begin{pmatrix} \Delta\boldsymbol{\xi} \\ \Delta\lambda \end{pmatrix} = -\tilde{\mathbf{H}}^{-1} \sum_i \tilde{\mathbf{J}}_i^T G_{r_i}, \tag{1.16}$$

où $\tilde{\mathbf{H}} = \sum_i \tilde{\mathbf{J}}_i^T \tilde{\mathbf{J}}_i$. L'équation de mise à jour des paramètres peut ainsi être calculée de la manière suivante :

$$\lambda \leftarrow \lambda + \Delta\lambda. \tag{1.17}$$

La matrice de transformation est mise à jour selon l'équation (1.18). Ce calcul itératif de l'incrément (équation (1.16)) suivi de la mise à jour des paramètres (équations (1.18) et (1.17)) est répété à chaque niveau de la pyramide d'images jusqu'à convergence de l'algorithme. Le résultat est ensuite propagé vers le prochain niveau, du niveau le plus grossier au plus fin.

$$\mathbf{T} \leftarrow \Gamma(\alpha\,\boldsymbol{\xi})\,.\,\mathbf{T}, \tag{1.18}$$

avec

$$\Gamma(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}}), \tag{1.19}$$

où exp(.) est la matrice exponentielle et $\hat{\boldsymbol{\xi}}$ est la matrice antisymétrique de $\boldsymbol{\xi}$.

1.5.2 SLAM basé vision et information de profondeur

Cette partie décrit comment l'algorithme d'alignement d'images est intégré à un système SLAM qui permet la fusion des informations fournies par un télémètre laser (dénoté LRF) et une caméra. Ce système SLAM est constitué d'une partie frontale (intégrant l'algorithme OP-B) permettant d'estimer le mouvement relatif de la caméra de $t-1$ à t à chaque nouvelle acquisition d'une image. La seconde partie, le "back-end", permet d'assurer la cohérence globale de l'estimation en construisant/optimisant un graphe de poses d'images-clés. Les images-clés mentionnées ici sont l'ensemble des images commençant par la première image acquise et présentant un déplacement supérieur à un seuil prédéfini entre elles. Une image est par conséquent considérée comme image-clé si son déplacement par rapport à l'image-clé précédente

est supérieur au seuil. Les nœuds du graphe de poses sont donc l'ensemble des poses absolues d'images-clés, tandis que ses arêtes sont des contraintes entre les nœuds représentés sous forme de transformations relatives. De plus, des contraintes supplémentaires peuvent être ajoutées au graphe de poses lorsque l'agent passe par des lieux précédemment visités (détection de fermeture de boucle), ce qui permet de réduire la dérive de l'estimation et d'assurer sa cohérence globale (Grisetti et al., 2010).

Le processus d'estimation de l'algorithme peut être résumé de la manière suivante. Le graphe de poses est d'abord initialisé à l'aide de la première acquisition, dont la pose est considérée comme l'origine du référentiel global $\{\mathcal{W}\}$. Après cela, au fur et à mesure que l'agent acquiert les données des capteurs, l'algorithme estime le mouvement relatif entre des paires d'images et de balayages lasers consécutifs à travers le dispositif de suivi ("tracker") frontal ; l'estimation résultante permettant de vérifier si la dernière image est une nouvelle image-clé. À ce stade, lorsqu'une nouvelle image-clé est détectée, l'algorithme l'intègre en tant que nœud supplémentaire dans le graphe de poses. La transformation relative entre la dernière et la nouvelle image-clé est, quant à elle, ajoutée en tant que contrainte. De plus, l'image-clé ajoutée est utilisée pour la détection de fermeture de boucle. Ce processus d'estimation ainsi que les principaux éléments impliqués dans le système SLAM proposé sont illustrés dans la figure 1.4.

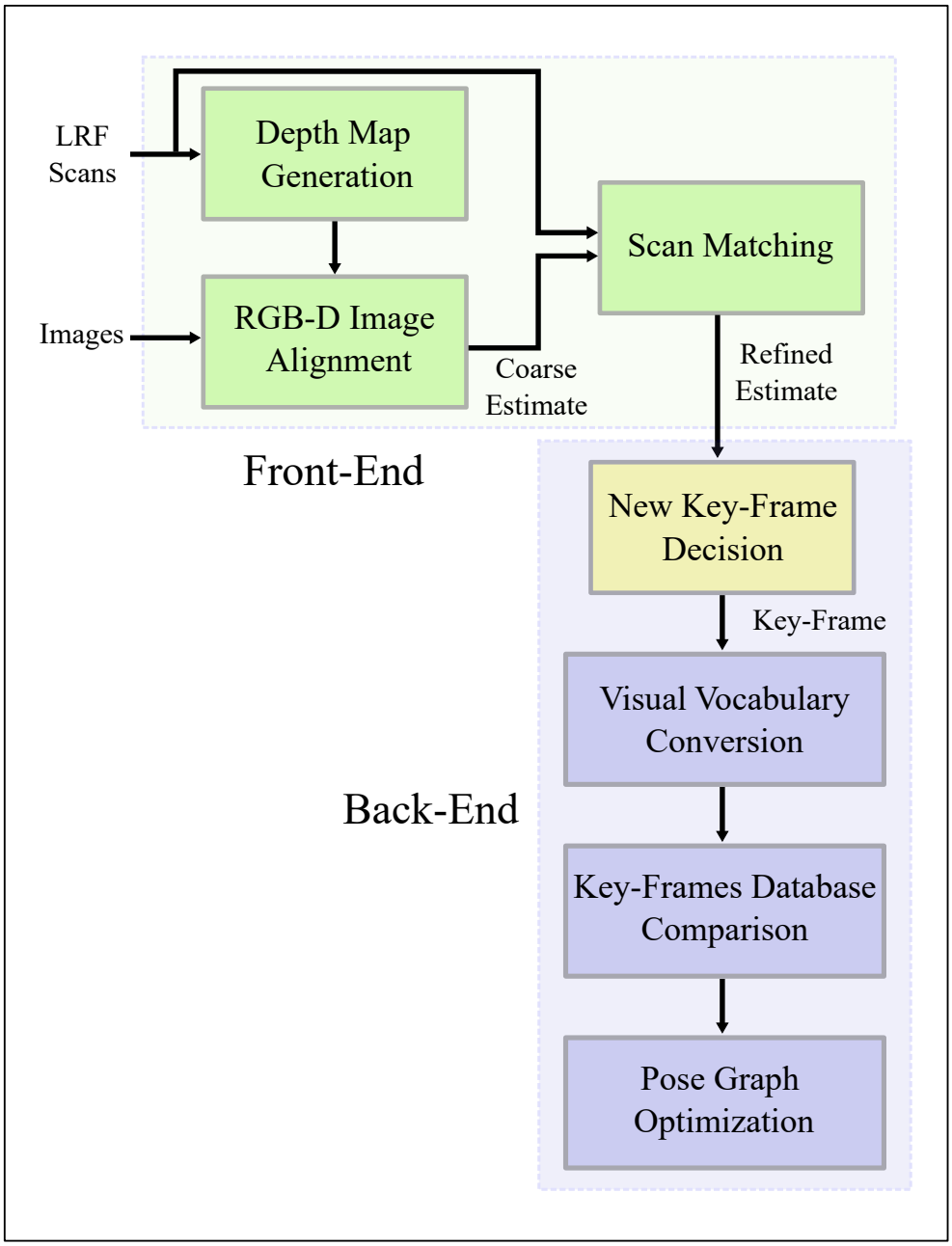


FIGURE *1.4: Vue d'ensemble du système SLAM proposé, montrant toutes les étapes impliquées dans le processus d'estimation.*

Chapitre 2

Introduction

During an exploration task, a moving agent has to know where it is to be autonomous. This agent needs, in fact, a representation (map) of its environment for this purpose. While, at the same time, this representation can only be acquired through navigation as the explored environment is a priori unknown. This typical chicken and egg problem can be solved by the concurrent estimation of the agent location and the environment map, which is known in the robotics community as SLAM (Simultaneous Localization And Mapping) and can be related to VO (Visual Odometry) in the context of computer vision. That is the object of this thesis, which aims at proposing a contribution to solving this problem.

2.1 A Brief History of Visual SLAM and VO

The first works related to the localization and mapping problems can be dated back to the 80s, at a period when probabilistic approaches began to be used by the robotics research community. For instance, (Moravec, 1980) was able to estimate the motion of a Mars rover using the information provided by a camera, which represents the first example of a VO system. Numerous works that provided solutions and theoretical bases to the localization and mapping problems followed up, including the research of (Matthies and Shafer, 1987) related to the error modeling in stereo navigation and the one of (Smith et al., 1988), which stands out for its implications concerning the uncertainty of the map elements (landmarks). It was, indeed, highly assumed that this uncertainty would grow unbounded when adopting a joint estimation of the robot pose and the landmark locations ; consequently, from then, researchers considered the localization and mapping problems as two separate problems.

Later, during the 90s, with works such as the one of (Csorba, 1997), it became clear that a simultaneous approach to the localization and mapping problems was convergent. Furthermore, the correlation between landmarks estimates, which researchers tried to minimize, was the key to the convergence of the estimation solution (Durrant-Whyte and Bailey, 2006).

The 2000s saw an increased interest of researchers towards the SLAM problem, leading to diverse research in this field. Alike, the study of the convergence and the consistency properties of the Extended Kalman Filter-based SLAM (EKF-SLAM) made by (Dissanayake et al., 2001) and the work of (Montemerlo et al., 2002) that used particle filters, permitting to consider non-Gaussian uncertainties. This approach later shown to exhibit good consistency results by (Bailey et al., 2006). (Thrun et al., 2004), for their part, tackled the problem of computational efficiency by using information filters, which have the advantage of providing a sparse structure. The computational cost is, indeed, one of the main concerns in SLAM ; for example, in the case of a naive EKF implementation, there is a quadratic complexity $O(n^2)$ in the number n of features (Durrant-Whyte and Bailey, 2006). Leading to important efforts from the researcher community to develop efficient SLAM systems. For instance, one can consider (Leonard and Feder, 2000) and (Guivant and Nebot, 2001), who worked on optimizing and increasing the computational efficiency of the EKF-SLAM. Another aspect from which SLAM approaches varied concerns the kind of used sensors. Like (Williams et al., 2000), who worked on simultaneously localizing an underwater robot and mapping its environment using sonar, and (Hahnel et al., 2003), who considered a laser range finder for the mapping of large scale environments.

A breakthrough happened, in 2003, when (Davison, 2003) developed a real-time EKF-based SLAM system using a monocular camera. It has been a long since researchers worked on developing algorithms for the estimation of the pose/structure using cameras. Notably, in the domain of Structure From Motion (SFM) (Brown, 1976; Sturm and Triggs, 1996; Triggs et al., 2000). The built systems were, however, limited to offline applications. It is worth noting that optimization techniques, mostly based on least-squares minimization, were the standard for these approaches. Moreover, it was at the same period that (Nister et al., 2004), who first coined the term VO, provided an online algorithm for the estimation of the ego-motion of a robot equipped with a camera. The work of (Davison, 2003) paved the way to numerous works on Visual SLAM (V-SLAM), such as (Pupilli and Calway, 2005) that used particle

filters to estimate the pose of a camera, or (Mouragnon et al., 2006) that proposed an optimization-based real-time V-SLAM system.

The second half of the 2000s witnessed efforts of the researchers to build systems for long-term applications ; consequently, scalability and global consistency became the leitmotivs of the V-SLAM and VO communities. In this context, two approaches, based on two different paradigms, were proposed. The first, developed by (Eade and Drummond, 2007), considered a graph of coalesced maps built upon an Extended Kalman Filter. The second, based on the innovative idea of decoupling the motion estimation from the mapping, was proposed by (Klein and Murray, 2007). These two approaches allowed the development of two streams in V-SLAM ; one considering filtering, where the EKF was the standard ; the other preferring optimization techniques mostly based on the Gauss-Newton and Levenberg-Marquardt algorithms.

It was later, in 2010, that (Strasdat et al., 2010) showed that optimization-based approaches were more efficient in the case of pure vision-based SLAM. These conclusions induced an increased interest towards the development of optimization-based approaches (Pirchheim and Reitmayr, 2011; Kaess et al., 2012; Herrera et al., 2014) and many systems, such as the DTAM of (Newcombe et al., 2011), were proposed. The increase of the computational capabilities during the 2010s permitted the rise of dense (and semi-dense) methods that used all (a dense subset of) the pixels of the image for the estimation, as the Dense V-SLAM proposed by (Kerl et al., 2013), the fast Semi-direct monocular Visual Odometry (SVO) of (Forster et al., 2014), in addition to the Large-Scale Direct monocular SLAM (LSD-SLAM) (Engel et al., 2014). Large scale consistency became also achievable during that period and V-SLAM algorithms were able to provide estimates for trajectories of many kilometers (about 2km), mostly because of the development of highly efficient backend tools such as graph-based optimization algorithms (Kummerle et al., 2011), and methods for the recognition of previously visited places (loop closure detection) (Newman and Kin Ho, 2005), (Stachniss et al., 2004), (Williams et al., 2009).

One of the systems that most efficiently used these tools for large-scale estimation was the ORB-SLAM proposed by (Mur-Artal et al., 2015). While the Direct Sparse Odometry (DSO) developed by (Engel et al., 2018) considered the problem differently by directly using the pixel brightness information in the optimization scheme.

Moreover, the V-SLAM and VO communities worked on developing more and more varied approaches. Each one considering different sensors and aspects of the

problem (Brossard et al., 2018; Schöps et al., 2019). For instance, the OpenVSLAM of (Sumikura et al., 2019) provides an easily usable software for different types of cameras. Other works used non-conventional vision sensors as the event-based cameras (Gallego et al., 2018; Bryner et al., 2019), or considered the integration of machine learning methods (Gao and Zhang, 2017; Zhang et al., 2017).

2.2 Taxonomy of the V-SLAM and VO

We consider here V-SLAM and VO to be equivalent in the sense that they both aim at estimating some state vector x (position of map points and pose of the camera). The difference between them is in the kind of built map, which is local in the case of VO and global in the case of V-SLAM. Another difference is that V-SLAM seeks global consistency of the estimate, whereas VO is only concerned with its local consistency (Scaramuzza and Fraundorfer, 2011). Both approaches can be distinguished according to several aspects, as shown in figure 2.1. On the one hand, it is possible to consider the type of used quantities for the estimation, typically direct pixel brightness or geometric primitives (such as features and lines), resulting in a **Direct Versus Indirect** classification. The consideration of the pixel density used by the SLAM or VO algorithm permits the distinction between **Sparse Versus Dense** methods. Alternatively, the used map representation enables us to characterize **Metric Versus Topological** maps. The navigation strategy, for its part, allows the categorization of **Passive Versus Active** strategies. On the other hand, considering the number of agents involved in the estimation leads to **Single-robot Versus Multi-robot** categories. In the remainder of this section, we look at these categories in more detail.

2.2.1 Direct Versus Indirect

V-SLAM and VO algorithms typically consider a probabilistic model that takes as input noisy measurements $\mathbf{z}$ (image stream) and outputs an estimation of the model parameters $\mathbf{x}$ (position of map points and pose of the camera). The objective is hence to find the model parameters that maximize the probability $p(\mathbf{z}|\mathbf{x})$ of observing the measurements. Direct methods (Silveira et al., 2008; Kerl et al., 2013) use directly the pixel brightness for this estimation problem, most commonly using a non-linear optimization scheme, whereas indirect methods (Mur-Artal et al., 2015; Zhu et al., 2012) estimate the model parameters in a two-step scheme. The first step consists in features detection/matching, which provides information about the locations of the

features (geometric information) ; the second consists in using these locations for the estimation.

2.2.2 Sparse Versus Dense

Sparse approaches were historically the first to be developed (Davison, 2003). They consider a subset of image pixels for the estimation (Forster et al., 2014) and have the advantage of requiring a limited amount of computation. Dense (semi-dense) approaches, for their part, use the entire set (an important part) of image pixels (Engel et al., 2014, 2018) for the estimation. These approaches have the advantage of being more precise than the sparse ones but at the cost of increased computations.

2.2.3 Metric Versus Topological

The first category of algorithms, referred to as metric, generates map elements related by metric information like distances (Pire et al., 2017; Schöps et al., 2019). The second category concerns the algorithms that compute maps containing elements linked by adjacency relationships. Such algorithms are known as topological (Angeli et al., 2009; Blochliger et al., 2018).

2.2.4 Passive Versus Active

During navigation, it is possible to consider two types of strategies. In the first type, the V-SLAM/VO algorithm estimates the state vector, while another entity is in charge of navigation control. Such strategies are known as passive (Kaess et al., 2012; DeTone et al., 2018). In the second type, the V-SLAM/VO algorithm controls the motion of the robot to build the map. Such strategies are known as active (Carlone et al., 2014; Mu et al., 2016). Active methods tend to provide more accurate maps in less time but at the cost of constraining the robot motion.

2.2.5 Single-Robot Versus Multi-Robot

Most V-SLAM and VO algorithms are developed for single-robots (Wang et al., 2017; Shin et al., 2020), where the algorithm is only concerned with using the information provided by its platform. However, recently, systems that jointly use the information provided by multiple robots gained popularity (Brossard et al., 2019a). Such systems give rise to additional issues related to communication and synchronization between robots.

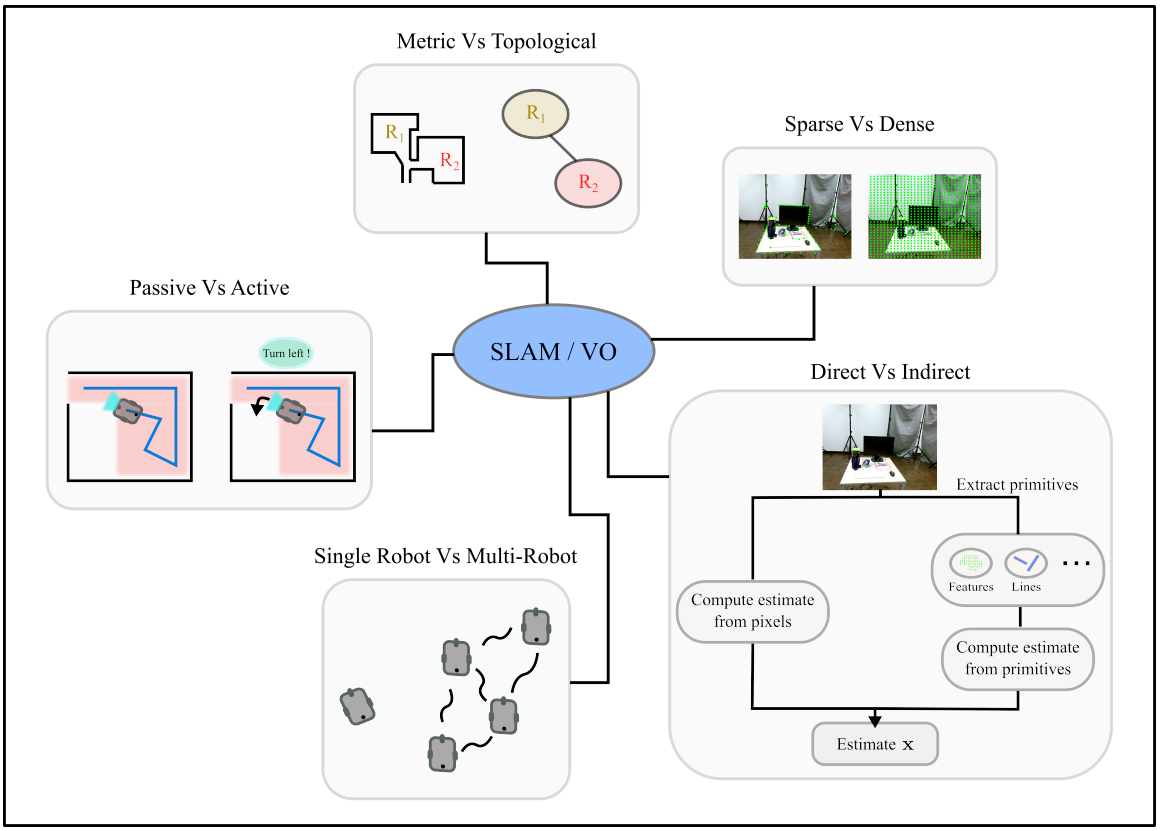


FIGURE *2.1: Illustration of the taxonomy of SLAM and VO.*

2.3 SLAM Paradigms

As mentioned before, SLAM systems aim at finding the estimates $\mathbf{x}$ that maximize the probability $p(\mathbf{z}|\mathbf{x})$ of observing some measurements $\mathbf{z}$, according to some motion model $p(\mathbf{x}|\mathbf{u})$ and control data $\mathbf{u}$, when available. This estimation is done based on three main paradigms : EKF, particle filters, and non-linear optimization.

2.3.1 Extended Kalman Filter

The first formulation to be considered in the context of SLAM is the EKF, introduced by (Smith et al., 1988). It intends to estimate a state vector $\mathbf{x}$, containing the pose of a mobile platform (robot) and landmark locations, as well as the corresponding covariance matrix modeling the estimation uncertainty. This estimation task is done through a recursive process based on two steps : prediction and correction. The former consists in the use of a motion model in order to predict the state of the robot (and the covariance matrix), based on the control input $\mathbf{u}$. The latter, also known as the update step, uses a measurement model to correct the predicted state vector (and the covariance matrix) based on the acquired measurements $\mathbf{z}$. EKF was popular among the SLAM community during the last decades, notably because of its simplicity and easy implementation (online). It was shown to be consistent and convergent when the Jacobians (used for linearization) are evaluated at ground-truth (Huang and Dissanayake, 2016), which is not the case in practical configurations. Consequently, issues related to linearization may arise, which represents one of the main drawbacks of EKF-based SLAM in addition to its poor scalability to large configurations.

2.3.2 Particle Filters

Introduced by (Montemerlo et al., 2002) in the Fast-SLAM, particle filters represent the second principal paradigm in SLAM. One of the key aspects of these filters is their ability to model multimodal distributions, which is generally more realistic than the Gaussian assumption of the EKF. During navigation, the algorithm maintains N particles, each one representing a guess on the robot path with its corresponding map. The latter is modeled in the form of a set of M multivariate Gaussians, where M is the number of landmarks, resulting in a total of $N.M$ Gaussians. When control information is available, a proposal distribution for the particles (next robot pose) is *stochastically* drawn according to the motion model. The following steps are then *resampling* and correction. The *resampling* step takes place based on a factor called

"importance" weight, which is the (Gaussian) probability of the measurement according to its predicted value (the mean of the Gaussian). The closer the measured value is to the predicted one, the higher the "importance" weight. When all the factors are computed, they are normalized so that they sum up to 1. Then the algorithm draws from the set of existing particles, a set of new ones according to the probability represented by the normalized "importance" weight. This *resampling* step aims at keeping the most plausible guesses (particles). After that, the landmarks of the surviving particles are updated, according to the standard EKF rules, using the measurement model.

2.3.3 Non-Linear Optimization

An alternative formulation to the filter-based formulation is to consider SLAM as a parameter estimation problem. The objective, in this case, is to find the best state vector $\mathbf{x}$ (in the least-squares sense) given sensory information $\mathbf{z}$ and the control input information $\mathbf{u}$. Intuitively, this can be represented in the form of a graph, as proposed by (Lu and Milios, 1997), where the nodes represent the robot trajectory and the map elements. The edges, for their part, represent the relative transformations (between robot poses) and the measurements. These quantities can be obtained from motion $\mathbf{g}$ and measurement $\mathbf{h}$ models, which are assumed to be affected by an additive Gaussian noise with covariance matrices $\mathbf{R}$ and $\mathbf{Q}$ respectively. The solution to the SLAM problem, in this case, can be shown to be equivalent to finding the optimum of the following equation :

$$\mathbf{x}^*, \mathbf{m}^* = \underset{\mathbf{x},\mathbf{m}}{\operatorname{argmin}} \left[\sum_t [\mathbf{x}_t - \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t)]^T \mathbf{R}^{-1} [\mathbf{x}_t - \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t)] + \sum_t [\mathbf{z}_t - \mathbf{h}(\mathbf{x}_t, M)]^T \mathbf{Q}^{-1} [\mathbf{z}_t - \mathbf{h}(\mathbf{x}_t, M)] \right]. \tag{2.1}$$

We use the subscript t to denote that the whole trajectory until the time step t is considered for the optimization. Most of the SLAM solutions consider linearized forms of g and h to solve this optimization problem, where gradient descent methods (Gauss-Newton, Levenberg-Marquardt) are the gold standard. It is worth noting that additional constraints (loop closure, data association) can be added to equation (2.1). Optimization-based SLAM scales well with the number of landmarks because of its sparse structure, resulting in large maps. In the case of V-SLAM, researchers frequently use keyframe-based strategies, i.e., methods where only a subset of selected

frames is used for the optimization to limit their computational cost. Despite this advantage, issues may arise when the initial estimates are far from their true values. Furthermore, long-term operations arise issues that are still not addressed, such as the exploration of a continuously growing environment (Kaess et al., 2012), (Johannsson et al., 2013), (Siciliano and Khatib, 2016).

2.3.4 Relation Between Paradigms

Most SLAM systems are built upon the previously mentioned paradigms, each one handling the involved probabilistic quantities in a specific manner. Figure 2.2 shows the relationships between these quantities in the form of Markov random fields[1]. The EKF approaches consider the problem from a filter-based perspective by marginalizing the past robot pose at each time step. The marginalization, in this context, means that the dependencies between the map elements and the previous robot pose are propagated to the former, i.e., landmarks that were previously independent become dependent. This action results in a denser filter covariance matrix at each time step, which hinders the scalability of the EKF. Particle filter methods, on the other hand, seek at preserving the independence between landmarks by maintaining a set of trajectories (particles) at each time step. In this case, the trajectory update is based on particle filtering, while the map update is done similarly to the EKF. This strategy has the advantage of being computationally more efficient than the EKF approach regarding the number of landmarks. However, the number of needed particles can become very substantial, especially in trajectories that contain a lot of nested loops. Optimization-based approaches usually consider a subset of selected poses that summarize the robot trajectory, along with the map elements, and all the involved probabilistic quantities are jointly optimized. Such strategies take advantage of the fact that the problem becomes very sparse when the whole trajectory is considered and permit to obtain very accurate estimates when good initialization is provided.

2.4 Related Works

Chapter 3 During the past years, interest in the development of algorithms related to SLAM and VO has grown importantly, notably because of the emergence of robotic platforms equipped with various sensors and with increasing computation capabilities.

[1]Markov random fields, also known as Markov networks, are undirected and connected graphs. Each node of them represents a random variable ; while, the graph edges represent the stochastic dependencies between these nodes.

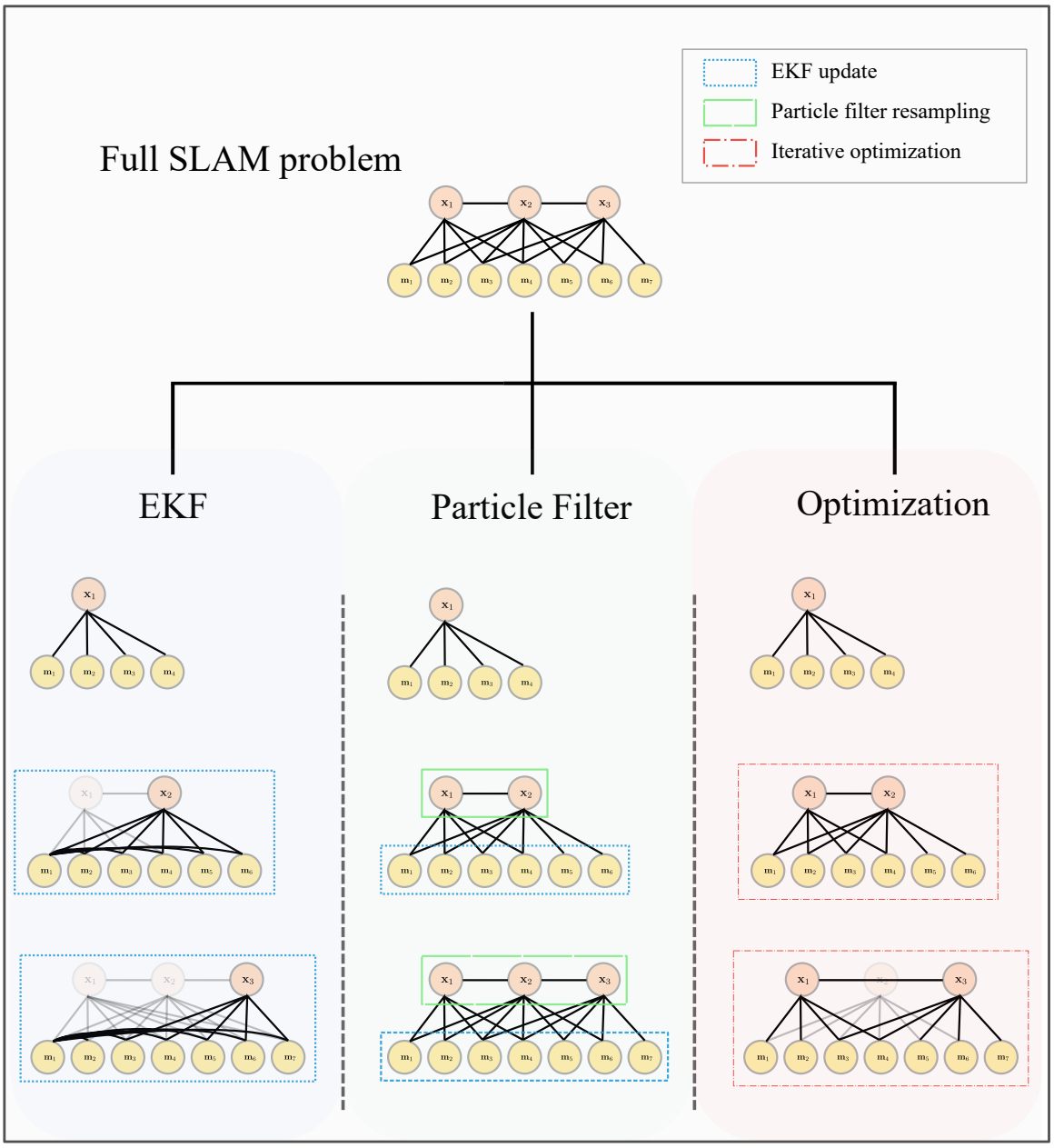


FIGURE *2.2: Representation of the different paradigms commonly used in SLAM in the form of Markov random fields (the measurements $\boldsymbol{z}_i$ are implicit). The way the dependencies between the robot poses and landmark locations is handled differs according to the paradigm.*

The research community consequently worked on developing methods and algorithms to solve SLAM and VO problems using the sensory information provided by these platforms. One of the paradigms used for that is the Extended Kalman Filter (EKF), which permits to deal with heterogeneous uncertain quantities, represented by the different sensors measurements, in an efficient manner.

Many EKF-based approaches were proposed, incorporating various sensors in different configurations, such as (Mallios et al., 2010), who considered the problem of underwater SLAM using a sonar sensor. Other works moreover used LRFs (Laser Range Finders), which provide very accurate range and bearing measurements, as (Eyice and Çulha, 2018). Vision sensors were also importantly used for SLAM, notably because of their advantage in terms of price and compactness. One can consider for instance (Eade and Drummond, 2007), who developed an algorithm that builds a set of coalesced maps for large scale SLAM, or the work of (Bresson et al., 2015). The EKF also offers a framework for information fusion and many recent researchers worked on leveraging this aspect. For example, (Barrau and Bonnabel, 2015; Geneva et al., 2019b) developed algorithms that fuse the visual information with IMU measurements. Other approaches furthermore focused on implementing consistent and efficient variants to the EKF (Brossard et al., 2019b; Geneva et al., 2019a).

In the context of feature-based V-SLAM and VO, one of the key steps to ensure initialization of "good" features that will be tracked over frames, is the feature extraction step. Many feature detectors can be found in the literature, such as the ones proposed by (Lowe, 2004; Dalal and Triggs, 2005). Moreover, researchers such as (Frintrop and Jensfelt, 2008) proposed a strategy inspired by the human visual intention system (Itti et al., 1998) for features extraction. This strategy was later used in combination with a learned objects database by (Yu et al., 2011) to select key features. Other works considered the issues related to linearization errors and proposed strategies to alleviate it, as (Bresson et al., 2011). Alternative initialization strategies considered not only point features but also line segments to enhance the estimation accuracy like the works of (Pumarola et al., 2017; Zhou et al., 2019).

Chapter 4 & 5 The estimation of the parametric geometric transformation between two images, i.e. image alignment, is a key part of various computer vision and robotics applications such as : optical flow estimation (Bruhn et al., 2005), VO (Forster et al., 2014), V-SLAM (Engel et al., 2014), image mosaicing (Capel, 2004), and

image stitching (Brown and Lowe, 2007). The techniques used for this estimation task typically consider a feature-based or a dense strategy.

Most commonly, feature-based strategies use a set of distinct features for the image alignment, through a feature extraction/matching process followed by parameters estimation from points correspondences. This kind of image registration algorithms has the advantage of being robust to changes in the scale, orientation, and lighting because features descriptors used by these algorithms are relatively invariant to such changes (Lowe, 2004; Dalal and Triggs, 2005). It is worth noting that the "scale" here refers to a geometric quantity in contrast to the "scale" used in the rest of the chapter, which refers to the degree of smoothness (blurring) of the considered image. Their main drawback is that features have to be evenly distributed and precisely located in each image in order to achieve sub-pixel accuracy, which can be challenging in low-textured environments. Examples of such algorithms include the trackers used by (Klein and Murray, 2007) and (Mur-Artal et al., 2015).

Conversely, dense methods try to make use of all image pixels for the estimation. This is typically done using the brightness of the pixel in an optimization scheme (Comport et al., 2010). This type of method permits to achieve better accuracy than feature-based methods and is more suitable for low-textured scenes. They are, however, more sensitive to large displacements. The algorithm proposed by (Kerl et al., 2013) for RBG-D cameras is based on a dense direct image alignment approach, for instance. Other methods try to take the best of both worlds by using, in the optimization process, a subset of image pixels with high gradients (Engel et al., 2018), or a descriptor vector instead of pixels (Antonakos et al., 2015).

One of the most used algorithms for direct image alignment is the Lucas-Kanade (LK) algorithm (Lucas and Kanade, 1981), which was developed in the early eighties. Subsequently, numerous extensions and variants to this method were developed and can be found in the literature (Benhimane and Malis, 2004; Oron et al., 2014). For instance, (Baker and Matthews, 2004) proposed a unifying framework for the variants of the LK algorithm, showing their equivalence. Recently, interest grew for combining feature descriptors with the LK approach (Crivellaro and Lepetit, 2014; Bristow and Lucey, 2016). Including the work of (Alismail et al., 2016), who proposed a binary feature descriptor used with the LK algorithm, for dense image alignment under drastic illumination changes. Other approaches considered the use of learning methods combined with the LK algorithm. Such as Chang et al. (2017), who proposed the combination of deep learning with the LK algorithm, and achieved subpixel accuracy

with large displacements and color variations. Another example is the approach developed by Lin et al. (2016), which learns a linear model to predict displacement from image appearance.

As stated before, the LK algorithm suffers from sensitivity to large displacements, and the image alignment optimization scheme is highly susceptible to falling into local minima in such conditions. In order to overcome this drawback a coarse-to-fine approach is generally adopted (Bouguet, 2000; Sharmin and Brad, 2012). This approach permits the first order Taylor's expansion, used by the LK algorithm, to better approximate the cost function and to enlarge the convergence domain (basin of convergence). Alternatively, one can blur the images with an isotropic Gaussian kernel to make the higher order terms of Taylor's expansion negligible (Mobahi et al., 2012). Such a blurring can be referred to as scale-space smoothing because smoothing an image with an isotropic Gaussian kernel permits to build the scale-space representation of the considered image (Lindeberg, 2014). (Mobahi et al., 2012), in their work, proposed to blur the objective function instead of the input images to remove local minima. (Sharmin and Brad, 2012) studied the effect of pre-filtering the input images on the LK optical flow and concluded that Gaussian filtering provides the best results. However, they only provided empirical results for the choice of the standard deviation of the Gaussian kernel.

2.5 Thesis Problematics and Objectives

This thesis focuses on finding solutions to two main problematics related to VO and V-SLAM. First, how to initialize interest points suited for estimation ? That is done based on the EKF paradigm in Chapter 4. Second, how to enlarge the basin of convergence when considering an optimization-based approach ? This point concerning Chapters 5 and 6.

2.6 Organization of the Thesis

The rest of this dissertation is organized in the following way. The mathematical prerequisites needed for the different works related to this thesis are first shortly presented in Chapter 3. Second, Chapter 4 presents the first contribution of this thesis related to features initialization. Next, the second contribution tackling image alignment based on the representation of images in the scale-space domain is proposed in

Chapter 5, considering notably the estimation of homography transformations between planar objects. After that, Chapter 6 shows the results of the consideration of depth information in addition to images for SLAM. Finally, conclusions about the works considered in this dissertation are presented and discussed in Chapter 7.

Chapitre 3

Prerequisites

This chapter focuses on the main mathematical tools that are used in the subsequent chapters, in an attempt to make this thesis self-contained. We therefore first present a reminder on some calculus concepts, i.e. the first and second order derivatives of a function and the Taylor series expansion. The Kalman filter and its non-linear counterpart, the EKF (Extended Kalman Filter), is then described with emphasis on its underlying Gaussian assumptions. Thereafter, we consider some gradient descent optimization approaches, with a special interest in the Gauss-Newton algorithm, which is viewed from a probabilistic point of view. The last part of this chapter is about *Lie groups* and their corresponding algebra (tangent space), which is considered as a means by which the increments (corrections) of an optimization (estimation) algorithm can be computed.

3.1 Reminder on Calculus

3.1.1 Multivariate Differentiation

A *scalar field* $F : \mathbb{R}^n \to \mathbb{R}$ is a function that maps vectors onto scalars. In addition to that, we can define a *vector field* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ as a function that maps vectors onto vectors.
The *gradient*, i.e. the first derivative, of a scalar field F is a vector field $\nabla F : \mathbb{R}^n \to \mathbb{R}^n$

$$\nabla F = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix}. \tag{3.1}$$

The *Hessian*, i.e. the second derivative, of a scalar field F is a function $\mathbf{H}_F$:

$\mathbb{R}^n \to \mathbb{R}^{n\times n}$ that maps an n-vector onto an $n \times n$ matrix :

$$\mathbf{H}_F = \begin{pmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{pmatrix}. \tag{3.2}$$

The *Jacobian*, i.e. the first derivative, of a vector field $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is a function $\mathbf{J} : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ that maps an n-vector onto an $m \times n$ matrix :

$$\mathbf{J_f} = \begin{pmatrix} \frac{\partial \mathbf{f}_1}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_m}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}_m}{\partial x_n} \end{pmatrix}. \tag{3.3}$$

It is interesting to note that the transpose of the gradient vector can be seen as a special case of the Jacobian matrix (where $m = 1$).
The second derivative of a vector field $\mathbf{f}$ can be defined as the function $\mathbf{H_f} : \mathbb{R}^n \to \mathbb{R}^{n\times m\times n}$ that maps an n-vector onto a third order tensor. It is called the *Hessian* (tensor) of f (Strasdat, 2012).

3.1.2 Taylor Series Expansion

Consider a real valued function $f : \mathbb{R} \to \mathbb{R}$ infinitely differentiable. If f is analytic around a point x_0, then it can be written in the series :

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ... = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}\ (x - x_0)^k. \tag{3.4}$$

In practical applications, the Taylor series expansion of a function f in the neighborhood of a point x_0 is considered until an n-th order :

$$f(x) \approx f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + ... + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^{(n)} = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}\ (x - x_0)^k. \tag{3.5}$$

The Taylor series expansion can be generalized to the scalar fields F. For example, the second-order Taylor series approximation of F around $\mathbf{x}_0$ can be expressed as follows :

$$F(\mathbf{x}) \approx F(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^T\ \nabla F(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T\ \mathbf{H}_F(\mathbf{x}_0)\ (\mathbf{x} - \mathbf{x}_0). \tag{3.6}$$

3.2 Filter-Based State Estimation

3.2.1 The Kalman Filter

In robotics, filter-based state estimation arises from the need of inferring state variables $\mathbf{x}$ (containing the robot pose and the map elements), that cannot be observed directly, from the available sensor and control data (denoted $\mathbf{z}$ and $\mathbf{u}$ respectively), at a certain time t. The derivation of the equations shown in this part can be found in (Thrun et al., 2005). This inference can be expressed in terms of conditional probability as follows :

$$p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}), \tag{3.7}$$

where all the available information until time t is considered. These filters are established on the basis of the Bayes theorem, while making the assumption that the process underlying the evolution of the states is Markovian [1] (Thrun et al., 2005). Figure 3.1 shows this process as a dynamic Bayesian network. Consequently, it is possible to express the previous probability in the following manner :

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)}{p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t)}, \tag{3.8}$$

where the term $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t)$ can be considered constant since the current measurements are assumed to be independent from the previous ones and from the controls :

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \propto\ p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{3.9}$$

The last expression, which represents the basis of Bayesian recursive filters, contains what is called a state transition distribution $p(\mathbf{z}_t|\mathbf{x}_t)$ and a state update distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$.

One of the earliest tractable implementations of this kind of filters is the Kalman Filter (KF), proposed by (Kalman, 1960), which makes the assumption that the different distributions can be represented by multivariate normal densities :

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = det(2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}}\ exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{3.10}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\mu} \in \mathbb{R}^n$ are, respectively, the covariance and the mean of the distribution and $det(.)$ is the determinant.

[1] The Markov assumption states that the future and past states (x_{t+1} and x_{t-1} respectively) are independent if the current state x_t is known.

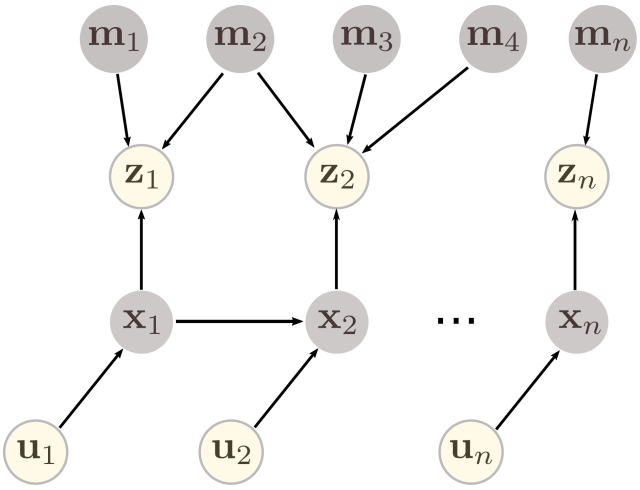


FIGURE *3.1: The process considered in the estimation as dynamic Bayesian network. Dark nodes represent the unobservable variables (robot pose and map elements) that we seek to infer from the observable variables (measurements and controls).*

In addition to that, this filter considers systems where the transition and update distributions are linear functions in their arguments :

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}_t \, \mathbf{x}_{t-1} + \mathbf{B}_t \, \mathbf{u}_t + \boldsymbol{\epsilon}_t \\ \mathbf{z}_t &= \mathbf{C}_t \, \mathbf{x}_t + \boldsymbol{\gamma}_t, \end{aligned} \tag{3.11}$$

where $\mathbf{A}_t$, $\mathbf{B}_t$, and $\mathbf{C}_t$ are the state transition, input, and output matrices respectively ; $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ and $\boldsymbol{\gamma}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ represent additive noises modeling the uncertainties in the process and measurements respectively. Based on that, the Kalman filter recursively estimates the mean and the covariance (the first and second order moments necessary to define a Gaussian distribution) of the states in two steps : prediction and correction. The prediction step consists in predicting the two moments of the normal probability modeling the states distribution (Thrun et al., 2005) :

$$\begin{aligned} \bar{\boldsymbol{\mu}}_t &= \mathbf{A}_t \, \boldsymbol{\mu}_{t-1} + \mathbf{B}_t \, \mathbf{u}_t \\ \bar{\boldsymbol{\Sigma}}_t &= \mathbf{A}_t \, \boldsymbol{\Sigma}_{t-1} \, \mathbf{A}_t^T + \mathbf{R}_t. \end{aligned} \tag{3.12}$$

The $\bar{(.)}$ represents the prediction on the state mean and covariance. Thereafter, the correction step follows with the integration of the measurements to update the estimates (Thrun et al., 2005) :

$$\begin{aligned} \mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t \, \mathbf{C}_t^T \, (\mathbf{C}_t \, \bar{\boldsymbol{\Sigma}}_t \, \mathbf{C}_t^T + \mathbf{Q}_t)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t \, (\mathbf{z}_t - \mathbf{C}_t \, \bar{\boldsymbol{\mu}}_t) \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t \, \mathbf{C}_t) \, \bar{\boldsymbol{\Sigma}}_t. \end{aligned} \tag{3.13}$$

In the previous equation, the term $\mathbf{K}_t$, called the Kalman gain, controls the innovation influence on the filter correction. Figure 3.2 shows how the Kalman filter handles the state variables belief according to control and measurement information in a 1D toy example.

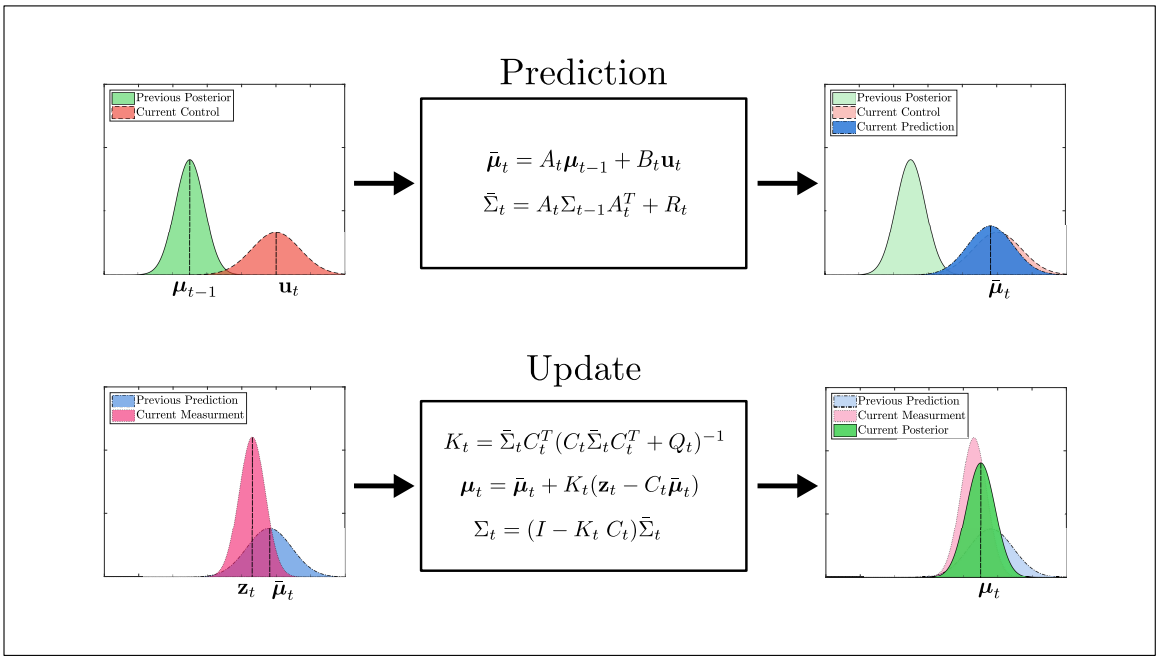


FIGURE *3.2: Illustration showing how probability densities are handled by the Kalman Filter in the 1D case. Top : presents the prediction step of the KF, the predicted belief is computed from the previous posterior belief and the control information. Down : shows the correction step of the KF, the previously predicted belief is corrected according to the measurement information to compute the current posterior belief.*

3.2.2 The Extended Kalman Filter

The Kalman filter has several advantages, especially its simplicity of implementation. However, in many applications, the involved state variables are governed by non-linear transition g and measurement h functions, as in equation (3.14). Its linearity assumption consequently does not hold anymore, which makes this kind of filter unusable for such applications.

$$\begin{aligned} \mathbf{x}_t &= \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \\ \mathbf{z}_t &= \mathbf{h}(\mathbf{x}_t) + \boldsymbol{\gamma}_t. \end{aligned} \tag{3.14}$$

It is however possible to overcome this limitation by using linearized forms of $\mathbf{g}$ and $\mathbf{h}$, resulting in what is called the EKF (Extended Kalman Filter). This filter

considers indeed a first order Taylor approximation (equation (3.6)) for the estimation. The transition function can thus be linearized around the previous state mean $\boldsymbol{\mu}_{t-1}$ and the current control $\mathbf{u}_t$:

$$\mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx \mathbf{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)\ (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}), \tag{3.15}$$

where $\mathbf{G}_t = [\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\ \frac{\partial \mathbf{g}}{\partial \mathbf{u}}]$ is the Jacobian of the transition function evaluated at $\boldsymbol{\mu}_{t-1}$ and $\mathbf{u}_t$. Concerning the measurement function, it is possible to consider the expansion around the current state prediction :

$$\mathbf{h}(\mathbf{x}_{t-1}) \approx \mathbf{h}(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\bar{\boldsymbol{\mu}}_t)\ (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t), \tag{3.16}$$

where $\mathbf{H}_t = [\frac{\partial \mathbf{h}}{\partial \mathbf{x}}]$ is the Jacobian of the measurement function evaluated at $\bar{\boldsymbol{\mu}}_t$. Hence, the expressions for the prediction and correction steps can be developed similarly to the ones of the Kalman filter (Thrun et al., 2005) :

$$\text{prediction} \begin{cases} \bar{\boldsymbol{\mu}}_t &= \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t) \\ \bar{\boldsymbol{\Sigma}}_t &= G_t\ \boldsymbol{\Sigma}_{t-1}\ G_t^T + R_t \end{cases} \tag{3.17}$$

$$\text{correction} \begin{cases} \mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T\ (\mathbf{H}_t\ \bar{\boldsymbol{\Sigma}}_t\ \mathbf{H}_t^T + \mathbf{Q}_t)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t\ (\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t)) \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t\ \mathbf{H}_t)\ \bar{\boldsymbol{\Sigma}}_t, \end{cases}$$

3.3 Introduction to Least-Squares Optimization

This section treats the problem of finding the minimum of a cost function of the form :

$$F(\mathbf{x}) = a\ \mathbf{r}(\mathbf{x})^T \mathbf{W} \mathbf{r}(\mathbf{x}), \tag{3.18}$$

where $a > 0$, $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector field, and $\mathbf{W} \in \mathbb{R}^{m \times m}$ is a symmetric, *positive semi-definite*[1] matrix. Since multiplying a function by a constant scalar does not change the location of its extrema, we will consider, without loss of generality, $a = \frac{1}{2}$. In general, when considering a gradient-based approach (which is the case here), finding the global minimum of a non-linear function is not always guaranteed and instead one would seek to find the local minimum in the neighborhood of an initial guess $\mathbf{x}^0$. If a point $\mathbf{x}^*$ is a minimum of F, then its first derivative at $\mathbf{x}^*$ is null $\nabla F(\mathbf{x}^*) = 0$, which is called the *necessary condition*. In addition to that the second derivative $\mathbf{H}_F(x^*)$ evaluated at this point is positive definite, which is the *sufficient*

[1] A matrix $\boldsymbol{\Lambda}$ is called positive semi-definite if $\{\forall \boldsymbol{\alpha} \in \mathbb{R}^n \mid \boldsymbol{\alpha} \neq 0\}, \quad \boldsymbol{\alpha}^T \boldsymbol{\Lambda} \boldsymbol{\alpha} \geq 0$.

condition. The subsequent parts will consider algorithms that take advantage of these derivatives to build methods for iteratively walking from an initial guess towards the minimum.

3.3.1 Gradient Descent Algorithm

As said before, the gradient descent (sometimes called the steepest descent) algorithm is an iterative method that updates the value of the point $\mathbf{x}$ at each iteration in the direction of the negative gradient (steepest descent) $(-\nabla F)$ as shown in Figure 3.3(a). Therefore, its update rule can be expressed as follows :

$$\begin{aligned} \boldsymbol{\delta} &= -\alpha \, \nabla F(\mathbf{x}^k) \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \boldsymbol{\delta}. \end{aligned} \tag{3.19}$$

Here $\alpha > 0$ is a step length selected so that the cost function decreases $F(\mathbf{x}^{k+1}) < F(\mathbf{x}^k)$. Detailed line search strategies for computing α can be found in the literature (Nocedal and Wright, 2006). Regarding the class functions considered in this section, the increment expression in equation (3.19) results in :

$$\boldsymbol{\delta} = -\alpha \, \mathbf{J_r}(\mathbf{x}^k)^T \mathbf{W} \, \mathbf{r}(\mathbf{x}^k), \tag{3.20}$$

where $\mathbf{J_r}(\mathbf{x}^k)$ is the Jacobian of $\mathbf{r}$ evaluated at $\mathbf{x}^k$. In the rest of this section we will omit the variables of the functions and the derivatives for readability.

3.3.2 Newton Algorithm

The Newton algorithm is an optimization approach that can be applied to any twice differentiable function. It results from the consideration of the Newton method, for finding the roots of a function, with the optimum necessary condition, which states that the first derivative must be equal to zero. The recursive expression resulting from that is the following :

$$\begin{aligned} \mathbf{H}_F \, \boldsymbol{\delta} &= -\nabla F \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \boldsymbol{\delta}, \end{aligned} \tag{3.21}$$

where $\mathbf{H}_F$ and ∇F are respectively the Hessian and gradient of F evaluated at $\mathbf{x}^k$. This results in the following update rule, in the context of least-squares optimization :

$$\left(\mathbf{J_r}^T \mathbf{W} \mathbf{J_r} + \mathbf{H_r} \mathbf{W} \, \mathbf{r}\right) \, \boldsymbol{\delta} = -\mathbf{J_r}^T \mathbf{W} \, \mathbf{r}. \tag{3.22}$$

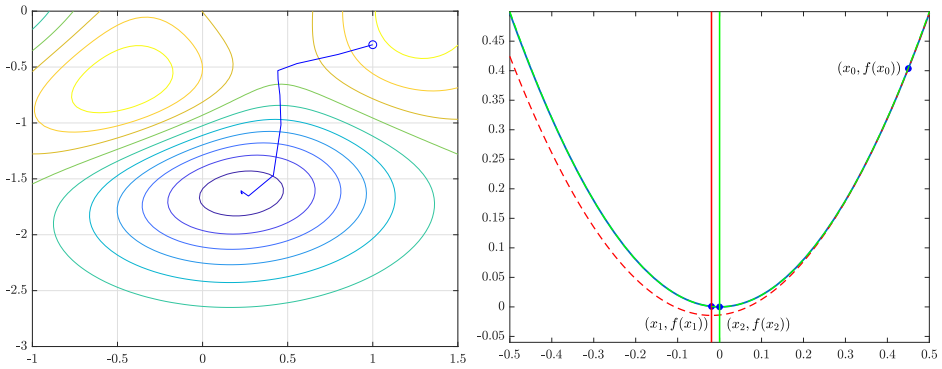


FIGURE *3.3: Illustration of the optimization process for gradient-based algorithms. (a) The steepest descent algorithm is applied to a 2D non-linear function, exhibiting a "zigzag" behavior because the approach iteratively walks towards the direction of the steepest descent. This results in a potential slow convergence of the method. (b) The Newton algorithm is considered with a 1D non-linear function (solid blue curve). In the first iteration the algorithm seeks the minimum (vertical red line) of the second order approximation around* $\mathbf{x}_0$ *(red dashed curve) leading to* $\mathbf{x}_1$*. A second iteration of the process (green dashed curve) lead to* $\mathbf{x}_2$*, which is very close to the minimum. Figure inspired by (Strasdat, 2012).*

An illustration of the behavior of the Newton algorithm, when applied to a one dimensional quadratic function is shown in Figure 3.3.

3.3.3 Gauss-Newton Algorithm

The computation of the second order derivative of r can be costly. Thus, the Gauss-Newton approach uses an approximation of the Hessian $H_r \approx J_r^T W J_r$ to overcome this drawback, which holds if the Hessian is small (close to the minimum) or if the value of $r(\mathbf{x}^k)$ is small. This leads to an alternative update rule :

$$\mathbf{J_r^T W J_r}\ \boldsymbol{\delta} = -\mathbf{J_r^T W}\ \mathbf{r}. \tag{3.23}$$

It is worth noting that the Gauss-Newton approach inherits the behavior of the Newton algorithm close to the extremum and approaches quadratic convergence in the vicinity of the minimum.

3.3.4 Levenberg–Marquardt Algorithm

The algorithms that we have seen so far exhibit two different behaviors. On the one hand, the gradient descent approach is guaranteed to converge towards the closest local minimum but may be slow to converge near the minimum. On the other hand, the

Newton and Gauss-Newton methods are fast to converge close to the extremum ; yet, they may be attracted by local maxima or saddle points. The Levenberg-Marquardt algorithm tries to take the best of both worlds by combining the Gauss-Newton and steepest descent approaches :

$$\left(\mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r} + \lambda\mathbf{I}\right)\ \boldsymbol{\delta} = -\mathbf{J}_\mathbf{r}^T\mathbf{W}\ \mathbf{r}, \tag{3.24}$$

where $\lambda > 0$ is a scalar controlling the behavior of the Levenberg-Marquardt algorithm, when it tends towards 0 the algorithm approaches the Gauss-Newton algorithm. On the contrary, if $\lambda \to \infty$, the Levenberg-Marquardt algorithm behaves like the gradient descent algorithm. Typically, the Levenberg-Marquardt algorithm works by fixing λ to an initial value λ_0. Then, if the increment $\boldsymbol{\delta}$ reduces the value of the cost function, it is assumed that the algorithm approaches the minimum and λ is reduced to augment the influence of the Gauss-Newton algorithm. Alternatively, if the increment increases the cost function, it is rejected and another try is considered with a greater λ.

3.3.5 Summary

To recapitulate, the optimization algorithms presented so far have in common that they iteratively search for the minimum of a function (of quadratic form in our case) departing from an initial guess $\mathbf{x}^0$. The difference between them lies in the used update rule for the increment, as depicted in table 3.1.

Gradient descent	$\mathbf{I}\ \boldsymbol{\delta} = -\mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{r}$
Newton	$\left(\mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r} + \mathbf{H}_\mathbf{r}\mathbf{W}\ \mathbf{r}\right)\ \boldsymbol{\delta} = -\mathbf{J}_\mathbf{r}^T\mathbf{W}\ \mathbf{r}$
Gauss-Newton	$\mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r}\ \boldsymbol{\delta} = -\mathbf{J}_\mathbf{r}^T\mathbf{W}\ \mathbf{r}$
Levenberg-Marquardt	$\left(\mathbf{J}_\mathbf{r}^T\mathbf{W}\mathbf{J}_\mathbf{r} + \lambda\mathbf{I}\right)\ \boldsymbol{\delta} = -\mathbf{J}_\mathbf{r}^T\mathbf{W}\ \mathbf{r}$

TABLE *3.1: Update rules for several least-squares optimization methods*

Each one of these approaches solves a linear equation (the normal equation) of the form $\mathbf{A}\boldsymbol{\delta} = \mathbf{b}$ to compute the increment $\boldsymbol{\delta}$. This type of equations can be solved in four ways :

- Using the Moore-Penrose pseudo-inverse if $\mathbf{A}^T\mathbf{A}$ is invertible. The resulting increment expression is then $\boldsymbol{\delta} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$.
- An alternative would be to use the QR decomposition of $\mathbf{A} = \mathbf{Q}\mathbf{R}$, where $\mathbf{Q}$ and $\mathbf{R}$ are orthogonal and upper triangular matrices respectively. The increment,

which is the solution of the normal equation would be in this case $\boldsymbol{\delta} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{b}$. This method has the advantage of preventing the calculation of $\mathbf{A}^T\mathbf{A}$ and its inversion, which can be more stable numerically.

- Another possible decomposition would be the Cholesky factorization of $\mathbf{A}^T\mathbf{A}$ into $\mathbf{L}^T\mathbf{L}$ where $\mathbf{L}$ is a lower triangular matrix. Consequently, $\boldsymbol{\delta}$ can be found in two steps, by first solving $\mathbf{L}\mathbf{y} = \mathbf{A}^T\mathbf{b}$ for $\mathbf{y}$, then solving $\mathbf{L}^T\boldsymbol{\delta} = \mathbf{y}$ for $\boldsymbol{\delta}$. This strategy is faster than the QR decomposition but less stable numerically, because of the need to compute $\mathbf{A}^T\mathbf{A}$.
- It is also possible to consider the SVD decomposition of $\mathbf{A}$ to compute the increment $\boldsymbol{\delta}$. This method was shown to be convenient to solve this kind of problems by (Hartley and Zisserman, 2004).

3.3.6 Least-Squares From a Probabilistic Point of View

As seen before, in robotic state estimation, one would seek to estimate state variables (robot pose, map elements) from exteroceptive measurements (images, laser scans, etc.) in addition to proprioceptive ones (odometry). This corresponds to maximizing the likelihood (3.7) or equivalently minimizing its negative logarithm.

$$\underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \underset{\mathbf{x}}{\operatorname{argmin}}(-\log\ p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t})) = \underset{\mathbf{x}}{\operatorname{argmin}}\, \boldsymbol{\chi}^2. \tag{3.25}$$

As seen in section 3.2, it is possible to express this likelihood in terms of measurement and state transition probabilities using the Bayes theorem. However, instead of considering only the previous robot position, as done for the Kalman filter, we would take into account the full robot trajectory. Such a consideration permits to solve the estimation problem in batch without the need of marginalizing the previous positions, which is done by recursive filters. Additionally, assuming that the measurements are independent and that the process involving the evolution of the states is Markovian, one expresses the likelihood (3.25) as the following joint probability (Kaess et al., 2008) :

$$\begin{aligned}\boldsymbol{\chi}^2 &= -\eta\, \log\left(p(\mathbf{x}_0)\ \prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)\ p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)\right) \\ &= -\eta\left(\log\ p(\mathbf{x}_0) + \sum_{i=1}^{t} \log\ p(\mathbf{z}_i|\mathbf{x}_i) + \sum_{i=1}^{t} \log\ p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)\right).\end{aligned} \tag{3.26}$$

Moreover, if we assume that the different distributions are Gaussian :

$$p(\mathbf{z}_i|\mathbf{x}_i) \propto \exp(-(\mathbf{h}(\mathbf{x}_i) - \mathbf{z}_i)^T \mathbf{\Sigma}_{\mathbf{h}}^{-1} (\mathbf{h}(\mathbf{x}_i) - \mathbf{z}_i)), \tag{3.27}$$

and

$$p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \propto \exp(-(\mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i)^T \mathbf{\Sigma}_{\mathbf{g}}^{-1} (\mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i)). \tag{3.28}$$

Then our optimization problem becomes of the following quadratic form :

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, \boldsymbol{\chi}^2 \propto \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{t} \|\mathbf{h}(\mathbf{x}_i) - \mathbf{z}_i\|^2_{\mathbf{\Sigma}_{\mathbf{h}}} + \|\mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\mathbf{\Sigma}_{\mathbf{g}}}, \tag{3.29}$$

where $\|\mathbf{e}\|^2_{\mathbf{\Sigma}} = \mathbf{e}^T \mathbf{\Sigma}^{-1} \mathbf{e}$ represents the Mahalanobis distance with covariance $\mathbf{\Sigma}$. The prior $-\log\ p(\mathbf{x}_0)$, being constant, was dropped for simplicity. Equation (3.29) being quadratic, it is suitable for least-squares optimization such as Gauss-Newton or Levenberg-Marquardt. It is worth noting that the vector $\mathbf{x}$ may contain the robot pose in addition to the map elements in the SLAM context. Conversely, only the robot pose may be considered during the optimization in the context of Visual Odometry (VO).

3.4 Introduction to Lie Groups and Lie Algebras

The concepts of Lie groups and Lie algebras as a way to represent transformations in the 3D space is introduced in this section. A Lie group is a topological group that is also a smooth manifold, while the corresponding Lie algebra can be seen as its tangent vector space at the identity. The objective of the present section is to show how these concepts can be used in robotics and computer vision contexts, without providing a rigorous introduction to the theory of Lie groups. The reader interested in a thorough and rigorous introduction to this topic may consider (Gallier, 2011), (Varadarajan, 1984), and (Hall, 2015).

3.4.1 Generalities

Lie Groups

The set of Euclidean transformations, describing the motion of rigid bodies in space forms the Special Euclidean group SE(n), which is a Lie group. Such transformations include a translational part that describes the displacement of the rigid body, represented by an n-vector $\mathbf{t} \in \mathbb{R}^n$, in addition to a rotational part that describes the

orientation of the rigid body, represented by an $n \times n$ matrix $\mathbf{R} \in \mathrm{SO}(n)$. This can be expressed more formally in the following manner :

$$T = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathrm{SE}(n) \quad / \quad \mathbf{R} \in \mathrm{SO}(n) \quad \text{and} \quad \mathbf{t} \in \mathbb{R}^n, \tag{3.30}$$

with $\mathrm{SO}(n)$ being the Special Orthogonal group :

$$\mathbf{R} \in \mathrm{SO}(n) \quad / \quad \mathbf{R}\mathbf{R}^T = \mathbf{I} \quad \text{and} \quad det(\mathbf{R}) = 1. \tag{3.31}$$

Lie Algebras

Lie algebra can be viewed as the differential space formed of the "infinitesimal transformations" around the identity element of the Lie group. This differential vector space is called the *tangent space* of the Lie group, which represents its linearization.

Let us consider a Lie group represented by matrices $\mathbf{G} \in \mathbb{R}^{n \times n}$ of k degrees of freedom. The corresponding algebra $\mathfrak{g} \in \mathbb{R}^{n \times n}$ is represented on the basis of k generators $\{\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_k\}$ with a k-vector of parameters $\boldsymbol{\xi}$. The mapping from the parameter vector to the Lie algebra is then done through the *hat*-operator $\hat{\cdot} : \mathbb{R}^k \to \mathbb{R}^{n \times n}$ expressed as follows :

$$\hat{\boldsymbol{\xi}} = \sum_{i=0}^{k} \xi_i \mathbf{G}_i. \tag{3.32}$$

We then consider the groups of 3D rotation transformation and rigid body motion to derive the relationships that permit to map *exactly* the elements of the tangent space with the transformations in the corresponding group. Such mappings are called the *exponential* and *logarithm* maps. We will also present the *adjoint* map $\mathrm{Adj}_{\mathbf{G}} : \mathfrak{g} \to \mathfrak{g}$ that offers a way to linearly transform tangent vectors from one tangent space to another.

3.4.2 SO(3)

The rotations in 3D can be expressed by the elements of the Special Orthogonal group SO(3) (Equation (3.31)). Furthermore, the composition and inversion in SO(3) are expressed by matrix multiplication and inversion. Because the rotation matrices are orthogonal, their inverse is equivalent to their transpose :

$$\mathbf{R}^{-1} = \mathbf{R}^T. \tag{3.33}$$

The underlying Lie algebra $\mathfrak{so}(3)$ is represented by the set of 3×3 skew symmetric matrices. Its three generators are obtained by deriving the rotation about each axis and evaluating them at the identity :

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \ \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \ \mathbf{G}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.34}$$

The elements of the tangent space, spanned by the generators, are then expressed using the parameters vector $\boldsymbol{\omega} \in \mathbb{R}^3$:

$$\hat{\boldsymbol{\omega}} = \omega_1\mathbf{G}_1 + \omega_2\mathbf{G}_2 + \omega_3\mathbf{G}_3 \in \mathfrak{so}(3). \tag{3.35}$$

As said before, there is a mapping that maps the elements $\hat{\boldsymbol{\omega}}$ of the Lie algebra into the transformation R in the Lie group, called the exponential map :

$$\exp(\boldsymbol{\omega}_\times) = \exp \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & -\omega_1 & 0 \end{pmatrix} \in \mathrm{SO}(3), \tag{3.36}$$

where $\boldsymbol{\omega}_\times = \hat{\boldsymbol{\omega}}$ is the skew symmetric matrix of $\boldsymbol{\omega}$. The closed form solution of the exponential map is the well known Rodrigues formula :

$$\exp(\boldsymbol{\omega}_\times) = \begin{cases} \mathbf{I} + \ \boldsymbol{\omega}_\times + \frac{1}{2}\,\boldsymbol{\omega}_\times^2 = \mathbf{I} & \text{for } \theta \to 0 \\ \mathbf{I} + \left(\frac{sin\,\theta}{\theta}\right)\ \boldsymbol{\omega}_\times + \left(\frac{1-cos\,\theta}{\theta^2}\right)\ \boldsymbol{\omega}_\times^2 & \text{otherwise,} \end{cases} \tag{3.37}$$

with $\theta = \sqrt{\boldsymbol{\omega}^T\boldsymbol{\omega}}$. This formula shows that the exponential map induces a rotation of θ radians around the axis defined by $\boldsymbol{\omega}$.

The logarithm map permits to have the inverse mapping so that the transformations are converted from SO(3) to $\mathfrak{so}(3)$:

$$\ln(R) = \begin{cases} \frac{1}{2}\,(\mathbf{R} - \mathbf{R}^T) = 0 & \text{for } d \to 1 \\ \frac{arccos\,d}{2\,\sqrt{1-d^2}}\ (\mathbf{R} - \mathbf{R}^T) & \text{for } d \in (-1, 1), \end{cases} \tag{3.38}$$

where $d = \frac{1}{2}(\mathrm{trace}(\mathbf{R}) - 1)$. The parameter vector $\boldsymbol{\omega}$ is then retrieved from the off diagonal elements of $\ln(\mathbf{R})$.

The *adjoint* map $\text{Adjoint}_{\mathbf{R}} : \mathfrak{so}(3) \to \mathfrak{so}(3)$ allows to move the tangent vector $\boldsymbol{\omega}$ from the right hand-side to the left hand-side of the group element $\mathbf{R}$:

$$\begin{aligned} \mathbf{R}\exp(\boldsymbol{\omega}_\times) &= \exp\left(\text{Adjoint}_{\mathbf{R}}(\boldsymbol{\omega}_\times)\right)\ \mathbf{R}, \\ \mathbf{R}\exp(\boldsymbol{\omega}_\times) &= \exp(\text{Ad}_{\mathbf{R}}\ .\ \boldsymbol{\omega}_\times)\ \mathbf{R}, \\ \mathbf{R}\exp(\boldsymbol{\omega}_\times) &= \exp(\mathbf{R}\ .\ \boldsymbol{\omega}_\times)\ \mathbf{R}. \end{aligned} \tag{3.39}$$

The adjoint transformation for SO(3), $\text{Ad}_{\mathbf{R}} = \mathbf{R}$, is the same rotation matrix used to represent the group element.

3.4.3 SE(3)

The Special Euclidean group SE(3) is used to represent the motion of rigid bodies in the 3D space. It is well suited to linearly represent transformations using homogeneous representation (Equation (3.30)). Furthermore, the composition of two elements $\mathbf{T}_1, \mathbf{T}_2 \in \text{SE}(3)$ is done through matrix multiplication :

$$\mathbf{T}_1\ \mathbf{T}_2 = \begin{pmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{t}_2 + \mathbf{t}_1 \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.40}$$

The transformation inversion is equivalent to matrix inversion :

$$\mathbf{T} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.41}$$

Since the special Euclidean group has 6 degrees of freedom, the corresponding Lie algebra is spanned by a set of 6 generators, obtained through the differentiation of the transformations (translation and rotation) at the identity :

$$\mathbf{G}_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ \mathbf{G}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ \mathbf{G}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.42}$$

$$\mathbf{G}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ \mathbf{G}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},\ \mathbf{G}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

As a result, the elements of the Lie algebra are represented using the hat-operator on the twist vector $\boldsymbol{\xi} = (\boldsymbol{\upsilon}\ \boldsymbol{\omega})^T \in \mathbb{R}^6$:

$$\hat{\boldsymbol{\xi}} = \upsilon_1 \mathbf{G}_1 + \upsilon_2 \mathbf{G}_2 + \upsilon_3 \mathbf{G}_3 + \omega_1 \mathbf{G}_4 + \omega_2 \mathbf{G}_5 + \omega_3 \mathbf{G}_6 \in \mathfrak{se}(3). \tag{3.43}$$

Moreover, the conversion from the tangent space to the group SE(3) is done using the matrix exponential $\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$:

$$\exp(\hat{\boldsymbol{\xi}}) = \begin{pmatrix} \exp(\boldsymbol{\omega}_\times) & \boldsymbol{\nu}\,\boldsymbol{\upsilon} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{3.44}$$

where $\exp(\boldsymbol{\omega}_\times)$ represents the solution of the Rodrigues formula (3.37) and $\boldsymbol{\nu}\,\boldsymbol{\upsilon}$ is obtained through the following expression :

$$\boldsymbol{\nu} = \begin{cases} \mathbf{I} + \frac{1}{2}\,\boldsymbol{\omega}_\times + \frac{1}{6}\,\boldsymbol{\omega}_\times^2 = \mathbf{I} & \text{for } \theta \to 0 \\ \mathbf{I} + \frac{1-cos\,\theta}{\theta^2}\,\boldsymbol{\omega}_\times + \frac{\theta - sin\,\theta}{\theta^3}\,\boldsymbol{\omega}_\times^2 & \text{otherwise,} \end{cases} \tag{3.45}$$

with $\theta = \sqrt{\boldsymbol{\omega}^T \boldsymbol{\omega}}$.

The logarithm mapping $\ln : \mathrm{SE}(3) \to \mathfrak{se}(3)$ can be obtained by computing $\ln(\mathbf{R})$, then the translation can be found :

$$\boldsymbol{\upsilon} = \boldsymbol{\nu}^{-1}\mathbf{t}, \tag{3.46}$$

where

$$\boldsymbol{\nu}^{-1} = \mathbf{I} - \frac{1}{2}\boldsymbol{\omega}_\times + \frac{1}{\theta^2}\left(1 - \frac{\theta^2\ sin\,\theta}{2\theta(1 - cos\,\theta)}\right)\boldsymbol{\omega}_\times^2 \tag{3.47}$$

The adjoint map for SE(3) is expressed in the following manner :

$$Ad_{\mathbf{T}} = \begin{pmatrix} \mathbf{R} & \mathbf{t}_\times\mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}. \tag{3.48}$$

3.5 Conclusion

This chapter showed the essential mathematical tools and theoretical background needed for the remainder of this thesis. The EKF, which is the basis of the algorithms considered in Chapter 4, has been first presented. After that, the least-squares optimization and Lie groups theories, used in Chapters 5 and 6, were introduced.

Chapitre 4

Salient Regions Detection for Visual Odometry and Visual Inertial Odometry

When considering a feature-based VO system, the detection of "reliable" features (key points) plays a key role in the success of the algorithm. What is meant by a "reliable" feature here is one that can be easily matched (or tracked) in other images. In this chapter, we present a features initialization strategy based on image saliency information, which permits to initialize "reliable" features in the context of VO. This strategy is integrated into two EKF-based VO and Visual Inertial Odometry (VIO) systems to give insights about its benefits regarding their performances.

Other works related to features initialization strategies use additional primitives such as lines (Pumarola et al., 2017; Zhou et al., 2019) or alleviate the errors resulting from the EKF linearization process (Bresson et al., 2011). Alternatively, the proposed initialization strategy only considers the image information to facilitate the detection and tracking of reliable features.

The remainder of the chapter is henceforth organized in the following manner. The EKF-based VO systems considered for the integration of our feature initialization procedure are first presented in section (4.1). Second, the several elements included in the proposed strategy are described in section (4.2), while showing how they are integrated into the considered VO and VIO systems. Third, the results of the conducted experiments in addition to the discussion of their implications are shown in section (4.3). Finally, a conclusion and some perspectives about the proposed work are provided in section (4.4).

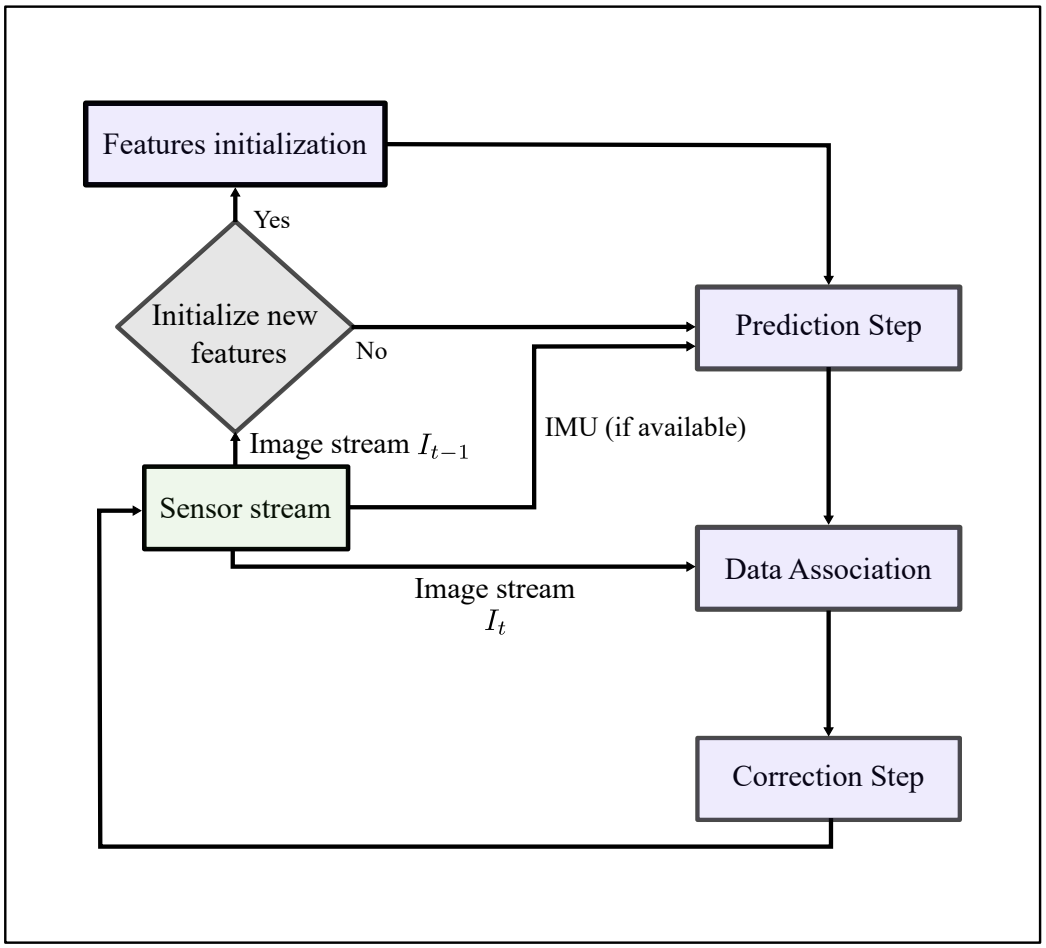


FIGURE *4.1: The principle steps involved in the considered EKF-based VO methods.*

4.1 EKF-Based VO

The two VO systems described in this part are based on two sensor configurations. The first encompasses the purely monocular configuration and was proposed by (Civera et al., 2009). The second, which was proposed by (Wu et al., 2017), permits the fusion of the information provided by an IMU (Inertial Measurement Unit) and a monocular camera. The diagram of figure 4.1 shows the principle steps involved in the two approaches.

In addition to the used sensors, the two systems differ in their implementation. (Civera et al., 2009) use a standard EKF, while (Wu et al., 2017) consider a Right Invariant EKF (RIEKF) for the estimation. This is more thoroughly described in the next part, which presents the equations that govern the prediction and correction steps of the two approaches.

4.1.1 Monocular-Based VO

State Variables

This system aims at estimating a state vector $\mathbf{x}$ composed of the camera pose $\mathbf{x}^{\mathcal{WC}}$ and its velocity vector $\mathbf{v}^{\mathcal{WC}}$, along with the locations of a set of landmarks $\mathbf{x}^{\mathcal{WL}}$. All these quantities are represented in the 3D space according to the world coordinate frame $\{\mathcal{W}\}$:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}^{\mathcal{WC}} \\ \mathbf{v}^{\mathcal{WC}} \\ \mathbf{x}^{\mathcal{WL}} \end{pmatrix}. \tag{4.1}$$

The camera pose $\mathbf{x}^{\mathcal{WC}}$ is represented by a translation vector $\mathbf{t}^{\mathcal{WC}} \in \mathbb{R}^3$ and a unit quaternion $\mathbf{q}^{\mathcal{WC}}$ encoding its orientation thus $\mathbf{x}^{\mathcal{WC}} = (\mathbf{t}^{\mathcal{WC}^T}, \mathbf{q}^{\mathcal{WC}^T})^T$ – noting that the elements of $\mathbf{q}^{\mathcal{WC}}$ are elements of $\mathbb{R}^4$. Its velocity $\mathbf{v}^{\mathcal{WC}}$ comprises a linear part $\boldsymbol{\nu}^{\mathcal{WC}} \in \mathbb{R}^3$ in addition to an angular part $\boldsymbol{\omega}^{\mathcal{WC}} \in \mathbb{R}^3$ thus $\mathbf{v}^{\mathcal{WC}} = (\boldsymbol{\nu}^{\mathcal{WC}^T}, \boldsymbol{\omega}^{\mathcal{WC}^T})^T$. The landmark locations are represented using the inverse depth parametrization proposed by (Civera et al., 2008), shown in figure 4.2. The ith landmark is hence represented by a 6-vector as follows :

$$\mathbf{x}_i^{\mathcal{WL}} = (x_i,\ y_i,\ z_i,\ \rho_i,\ \theta_i,\ \phi_i)^T, \tag{4.2}$$

where the (x_i, y_i, z_i) component represents the camera coordinates at landmark initialization time and the vector $(\rho_i, \theta_i, \phi_i)$ represents the landmark inverse depth, azimuth, and elevation respectively. All these quantities are expressed in $\{\mathcal{W}\}$ and the upper-script was omitted for readability.

Prediction Step

At each time t, an image is available and the algorithm makes a prediction on the states based on a constant velocity motion model. The only variables affected by this model are the ones related to the camera since the surrounding environment, i.e. landmarks, is assumed to be static. The prediction is then made according to the following expressions :

$$\bar{\mathbf{x}}_t = \mathbf{g} \begin{pmatrix} \mathbf{x}_{t-1}^{\mathcal{WC}} \\ \mathbf{v}_{t-1}^{\mathcal{WC}} \\ \mathbf{x}_{t-1}^{\mathcal{WL}} \end{pmatrix}, \tag{4.3}$$

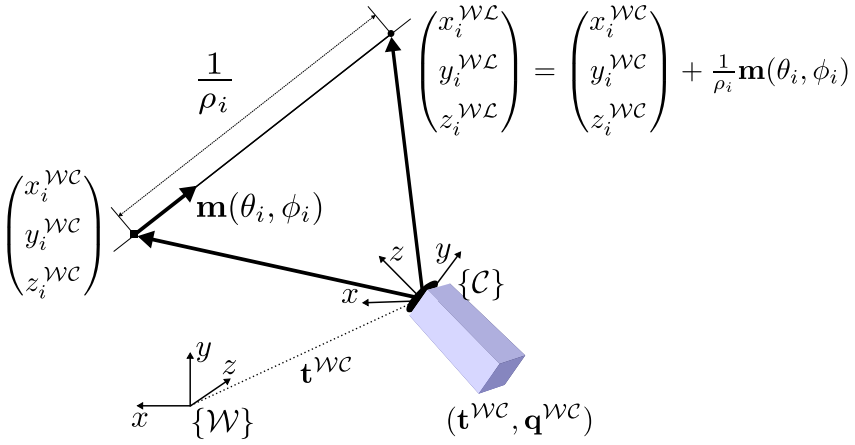


FIGURE *4.2: Inverse depth parametrization. Figure inspired by (Civera et al., 2008).*

---

where

$$\mathbf{g}\begin{pmatrix} \mathbf{x}_{t-1}^{\mathcal{WC}} \\ \mathbf{v}_{t-1}^{\mathcal{WC}} \\ \mathbf{x}_{t-1}^{\mathcal{WL}} \end{pmatrix} = \begin{pmatrix} \mathbf{t}_{t-1}^{\mathcal{WC}} + (\boldsymbol{\nu}_{t-1}^{\mathcal{WC}} + \boldsymbol{\epsilon}_{\boldsymbol{\nu}})\,\delta t \\ \mathbf{q}_{t-1}^{\mathcal{WC}} \times \mathbf{q}\left((\boldsymbol{\omega}_{t-1}^{\mathcal{WC}} + \boldsymbol{\epsilon}_{\boldsymbol{\omega}})\,\delta t\right) \\ \boldsymbol{\nu}_{t-1}^{\mathcal{WC}} + \boldsymbol{\epsilon}_{\boldsymbol{\nu}} \\ \boldsymbol{\omega}_{t-1}^{\mathcal{WC}} + \boldsymbol{\epsilon}_{\boldsymbol{\omega}} \\ \mathbf{x}_{t-1}^{\mathcal{WL}} \end{pmatrix}. \tag{4.4}$$

The term δt represents the time difference between two successive image acquisitions, which can be considered as unitary. $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\omega}}$ are zero mean Gaussian noises that affect the linear and angular velocities respectively. In addition to that, the operator $\times$ represents here the quaternion product between the unit quaternion encoding the previous orientation $\mathbf{q}_{t-1}^{\mathcal{WC}}$ and the one encoding the predicted change in orientation $\mathbf{q}\left((\boldsymbol{\omega}_{t-1}^{\mathcal{WC}} + \boldsymbol{\epsilon}_{\boldsymbol{\omega}})\,\delta t\right)$. These expressions permit the prediction of the states mean values ; the corresponding covariance matrix is, for its part, predicted in the following manner :

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{G}_t\,\boldsymbol{\Sigma}_{t-1}\,\mathbf{G}_t^T + \boldsymbol{\mathcal{E}}_t\,\mathbf{R}\,\boldsymbol{\mathcal{E}}_t^T, \tag{4.5}$$

where $\mathbf{G}_t$ and $\boldsymbol{\mathcal{E}}_t$ are the Jacobian matrices of the motion model with respect to the states and noises respectively. $\mathbf{R}$ represents the process noise covariance matrix as in Chapter 3.

Correction Step

The prediction made on the states is corrected during the correction step according to the available measurements. This is done by first transforming the landmarks locations into the camera coordinate frame $\mathcal{C}$, then projecting the transformed points into the image plane. The change in coordinates of the ith landmark is thus expressed as follows :

$$\begin{pmatrix} x_i^{\mathcal{CL}} \\ y_i^{\mathcal{CL}} \\ z_i^{\mathcal{CL}} \end{pmatrix} = \bar{\mathbf{R}}^{\mathcal{CW}} \left(\left(\begin{pmatrix} x_i^{\mathcal{WC}} \\ y_i^{\mathcal{WC}} \\ z_i^{\mathcal{WC}} \end{pmatrix} - \bar{\mathbf{t}}_t^{\mathcal{WC}} \right) + \frac{1}{\rho_i} \mathbf{m}(\theta_i, \phi_i) \right), \tag{4.6}$$

here $\bar{\mathbf{R}}^{\mathcal{CW}}$ is the predicted rotation matrix, which transforms a vector from $\mathcal{W}$ to $\mathcal{C}$ and $\bar{\mathbf{t}}_t^{\mathcal{WC}}$ is the predicted position of the camera. $\mathbf{m}(\theta_i, \phi_i)$ represents a unit directional vector describing the direction towards the ith landmark :

$$\mathbf{m}(\theta_i, \phi_i) = (cos\,\phi_i\, sin\,\theta_i,\ -sin\,\phi_i,\ cos\,\phi_i\, cos\,\theta_i)^T. \tag{4.7}$$

The resulting point can accordingly be projected into the image plane using the pinhole camera model :

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \mathbf{\Pi} \begin{pmatrix} x_i^{\mathcal{CL}} \\ y_i^{\mathcal{CL}} \\ z_i^{\mathcal{CL}} \end{pmatrix} = \begin{pmatrix} c_u + f_u \dfrac{x_i^{\mathcal{CL}}}{z_i^{\mathcal{CL}}} \\ \\ c_v + f_v \dfrac{y_i^{\mathcal{CL}}}{z_i^{\mathcal{CL}}} \end{pmatrix}, \tag{4.8}$$

where (f_u, f_v) represent the focal lengths and (c_u, c_v) the coordinates of the camera principal point – all described in pixel units. It is worth noting that when the uncertainty on a landmark location is below a certain threshold, the algorithm switches to the standard Cartesian representation for this landmark. The previous expressions permit to define a measurement model h :

$$\boldsymbol{\iota}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \mathbf{h}(\mathbf{x}_i^{\mathcal{WL}}), \tag{4.9}$$

where $\boldsymbol{\iota}_i$ is the location of the ith pixel in the image coordinate frame. As the prediction $\mathbf{h}(\mathbf{x}_i^{\mathcal{WL}})$ is computed, the measurement $\mathbf{z}_i$ is found using cross-correlation. The patch describing the feature is hence first warped according to the predicted motion and $\mathbf{z}_i$ is then searched for inside a region defined by the covariance of the prediction on the feature location. This permits the update of the states according to the

following expressions :

$$\begin{aligned}\mathbf{K}_t &= \bar{\mathbf{\Sigma}}_t\, \mathbf{H}_t^T\, (\mathbf{H}_t\, \bar{\mathbf{\Sigma}}_t\, \mathbf{H}_t^T + \mathbf{Q})^{-1} \\ \mathbf{x}_t &= \bar{\mathbf{x}}_t + \mathbf{K}_t\, (\mathbf{z}_t - \mathbf{h}(\bar{\mathbf{x}}_t)) \\ \mathbf{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t\, \mathbf{H}_t)\, \bar{\mathbf{\Sigma}}_t,\end{aligned} \tag{4.10}$$

where $\mathbf{H}_t$ is the Jacobian matrix of the measurement model. $\mathbf{Q}$ represents the covariance matrix of the measurements noise as in Chapter 3 (section (3.2)).

4.1.2 RIEKF-VIO

State Variables

RIEKF-VIO was designed to estimate the state variables related to a Micro Aerial Vehicle (MAV) equipped with an IMU and a monocular camera. These variables consist in the body position $\mathbf{t}^{\mathcal{WI}} \in \mathbb{R}^3$, velocity $\mathbf{v}^{\mathcal{WI}} \in \mathbb{R}^3$, and orientation $R^{\mathcal{WI}} \in SO(3)$, in addition to the IMU biases $\mathbf{b}_{\boldsymbol{\omega}}$ and $\mathbf{b}_{\mathbf{a}}$, as well as landmark positions $\mathbf{x}^{\mathcal{WL}}$. Following (Brossard et al., 2018), the state variables associated with positions and orientations are expressed in the $SE_{2+p}(3)$ Lie group :

$$\chi = \begin{pmatrix} \mathbf{R}^{\mathcal{WI}} & \mathbf{v}^{\mathcal{WI}} & \mathbf{t}^{\mathcal{WI}} & \mathbf{x}^{\mathcal{WL}} \\ & \mathbf{0}_{2+p\times 3} & \mathbf{I}_{2+p\times 2+p} & \end{pmatrix}, \tag{4.11}$$

where p is the number of landmarks. This representation used in the design of RIEKFs permits to reduce the linearization errors related to the standard EKF, and uncertainty can be accurately expressed in Lie algebra :

$$\boldsymbol{\chi} = \exp(\hat{\boldsymbol{\xi}}).\chi = \exp\begin{pmatrix} (\boldsymbol{\xi}_R)_\times & \boldsymbol{\xi}_{\mathbf{v}} & \boldsymbol{\xi}_{\mathbf{t}} & \boldsymbol{\xi}_{\mathbf{L}} \\ & \mathbf{0}_{2+p\times 5+p} & & \end{pmatrix}.\chi. \tag{4.12}$$

The random variable describing the states uncertainty is described using a bold symbol $\boldsymbol{\chi}$, while its mean is described using a light symbol χ. $\boldsymbol{\xi}_R, \boldsymbol{\xi}_{\mathbf{v}}$, and $\boldsymbol{\xi}_{\mathbf{t}}$ are the uncertainty associated with the body orientation, velocity, and position respectively. $\boldsymbol{\xi}_{\mathbf{L}}$ represents the landmarks uncertainty. The exponential map exp(.) and the skew symmetric matrix operator $(.)_\times$ were defined in Chapter 3 (section 3.4).

Prediction Step

Each time an IMU measurement is available, a prediction on the MAV states is made by integrating the following dynamic model :

$$System\ dynamics \begin{cases} \dot{\mathbf{R}}^{\mathcal{WI}} & = \mathbf{R}^{\mathcal{WI}} \left(\boldsymbol{\omega}^{\mathcal{I}} - \mathbf{b}_{\boldsymbol{\omega}} + \boldsymbol{\epsilon}_{\boldsymbol{\omega}}\right)_{\times} \\ \dot{\mathbf{v}}^{\mathcal{WI}} & = \mathbf{R}^{\mathcal{WI}} \left(\mathbf{a}^{\mathcal{I}} - \mathbf{b}_{\mathbf{a}} + \boldsymbol{\epsilon}_{\mathbf{a}}\right) + g \\ \dot{\mathbf{x}}^{\mathcal{WI}} & = \mathbf{v}^{\mathcal{WI}} \\ \dot{\mathbf{b}}_{\boldsymbol{\omega}} & = \boldsymbol{\epsilon}_{\mathbf{b}_{\boldsymbol{\omega}}} \\ \dot{\mathbf{b}}_{\mathbf{a}} & = \boldsymbol{\epsilon}_{\mathbf{b}_{\mathbf{a}}} \end{cases}, \tag{4.13}$$

where g is the gravity acceleration and $(\boldsymbol{\omega}^{\mathcal{I}}, \mathbf{a}^{\mathcal{I}})$ are the IMU angular velocity and linear acceleration expressed in the IMU coordinate frame $\{\mathcal{I}\}$. As for the Mono-VO (4.1.1), the scene is assumed to be static and the landmarks dynamics is therefore null. Moreover, the prediction $\bar{\boldsymbol{\Sigma}}_t$ on the covariance matrix $\boldsymbol{\Sigma}_t$ is expressed using $\mathbf{G}_t$ and $\boldsymbol{\mathcal{E}}_t$, the Jacobian matrices of the motion model with respect to the states and noises respectively, as :

$$\bar{\boldsymbol{\Sigma}}_t = \boldsymbol{\Phi}_{t-1}\, \boldsymbol{\Sigma}_{t-1} \boldsymbol{\Phi}_{t-1}^T + \mathbf{Q}_{t-1}, \tag{4.14}$$

where

$$\boldsymbol{\Phi}_{t-1} = \boldsymbol{\Phi}(t, t-1) = \exp\left(\int_{\mathrm{t-1}}^{\mathrm{t}} \mathbf{G}_\tau \, \mathrm{d}\tau\right), \tag{4.15}$$

and

$$\mathbf{Q}_{t-1} = \int_{t-1}^{t} \boldsymbol{\Phi}(t,\tau)\, \boldsymbol{\mathcal{E}}_t\, \mathbf{R}\, \boldsymbol{\mathcal{E}}_t^T\, \boldsymbol{\Phi}^T(t,\tau)\, d\tau. \tag{4.16}$$

Correction Step

The correction step permits the update of the state variables according to the image measurements. This is done accordingly by projecting the landmarks into the image plane using the pinhole camera model as for the Mono-VO. The difference however is in the transformation from the world coordinate system towards the camera coordinate system, which is done as follows :

$$\begin{pmatrix} x_i^{\mathcal{CL}} \\ y_i^{\mathcal{CL}} \\ z_i^{\mathcal{CL}} \end{pmatrix} = \mathbf{R}^{\mathcal{CI}} \left(\bar{\mathbf{R}}^{\mathcal{IW}} \left(\begin{pmatrix} x_i^{\mathcal{WL}} \\ y_i^{\mathcal{WL}} \\ z_i^{\mathcal{WL}} \end{pmatrix} - \bar{\mathbf{t}}_t^{\mathcal{WI}} \right) - \mathbf{t}^{\mathcal{CI}} \right), \tag{4.17}$$

where $\mathbf{R}^{\mathcal{CI}}$ and $\mathbf{t}^{\mathcal{CI}}$ are the IMU orientation and translation expressed in $\{\mathcal{C}\}$. The update rule is similar to the Mono-VO except for the mean update, which is done in

the following manner :

$$\begin{pmatrix} \delta\chi \\ \delta\mathbf{b}_{\boldsymbol{\omega}} \\ \delta\mathbf{b}_{\mathbf{a}} \end{pmatrix} = \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{h}(\bar{\chi}_t)\right) \tag{4.18}$$

and

$$\begin{aligned} \chi_t &= \exp(\delta\chi).\bar{\chi}_t \\ \mathbf{b}_{\boldsymbol{\omega}} &= \bar{\mathbf{b}}_{\boldsymbol{\omega}} + \delta\mathbf{b}_{\boldsymbol{\omega}} \\ \mathbf{b}_{\mathbf{a}} &= \bar{\mathbf{b}}_{\mathbf{a}} + \delta\mathbf{b}_{\mathbf{a}}. \end{aligned} \tag{4.19}$$

It is worth noting that the data association is done using a Kanade-Lucas-Tomasi (KLT) feature tracking algorithm (Lucas and Kanade, 1981; Tomasi and Kanade, 1991). Each feature in the previous image is hence tracked in the current image using the KLT tracker. Such a strategy is different from the one of the Mono-VO that uses a correlation-based data association strategy.

4.2 Features Initialization Strategy

Every time new areas are explored, the described algorithms need to initialize new features. This process consists in applying an interest point detector (a corner detector) on specific image regions called *searching boxes*, which are distributed on the image according to a uniform probability density (Civera et al., 2009). Features are consequently evenly detected in the image. However, one of the disadvantages of this strategy is that it does not exploit all of the image information during the initialization process in the sense that easily distinguishable image regions are not favored. The proposition of this work is to use the image saliency information to detect more interest points on salient regions (easily discriminated), while keeping a feature coverage of the whole image.

The different steps involved in the proposed method, shown in figure 4.3, are summarized as follows : The input image is first used to compute a saliency map ; the latter is then converted into a binary image according to a user-defined threshold ; after that, the white pixels (greater than the threshold) are used to train a Gaussian Mixture Model (GMM), which permits to generate *searching boxes* centers. These steps are described in more details in the subsequent parts.

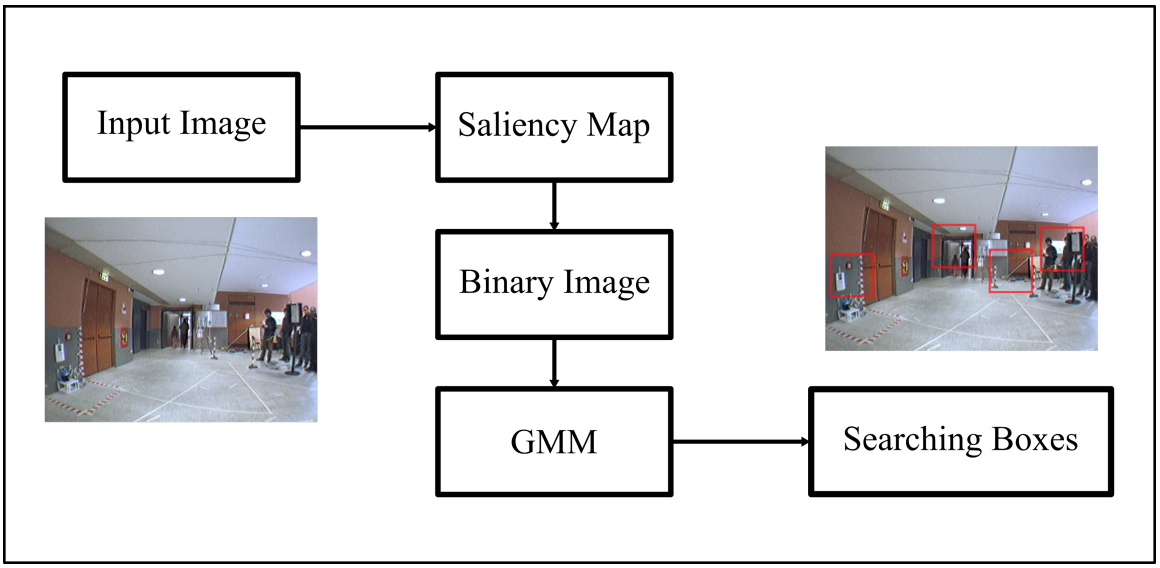


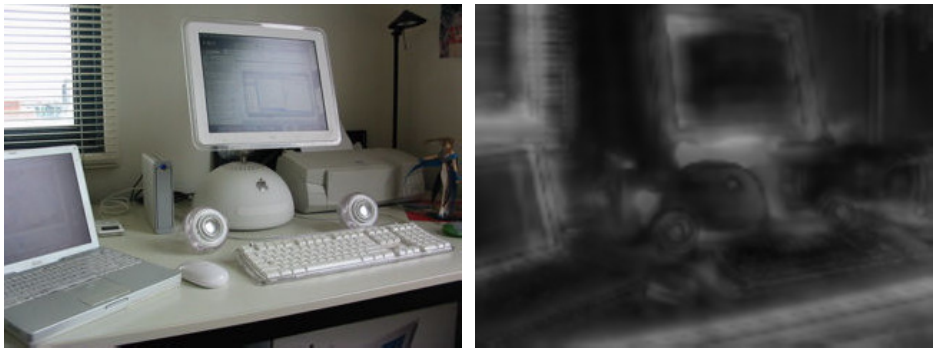
FIGURE *4.3: Presentation of the proposed features initialization strategy. The searching boxes are illustrated.*

4.2.1 Saliency Map

The saliency map is computed from the input image using the bottom-up part of the approach proposed by (Frintrop and Jensfelt, 2008) called VOCUS (Visual Object detection with a CompUtational attention System). This approach is in fact a human inspired visual attention system that uses the variations on image intensity, color, and orientation to generate a saliency map. Three "conspicuity" maps are at first computed for this :

- **The intensity map** obtained by first computing the grayscale image pyramid corresponding to the input image, and applying on-center and off-center filters[1] to each level of this pyramid.
- **The color map** computed through the transformation of the image color space towards CIE 1976 L*a*b (Schwiegerling, 2004). On-center and off-center filters are then applied to a four channels pyramid (green, red, blue, and yellow) obtained from the transformed image.
- **The orientation map** based on the application of a Gabor filter on the DoG (Difference of Gaussians) grayscale image pyramid.

[1]On-center filtering consists in computing the difference between the values of the image pixels and their surrounding. The off-center filtering for its part consists in computing the difference between the surrounding and the image pixels, which is the inverse of the on-center filtering. Negative values are set to 0 in both cases.



(a) *(b)*

FIGURE *4.4: Example image (a) along with its corresponding saliency map (b).*

The resulting saliency map is the mean of these "conspicuity" maps and a gray level image with intensity value corresponding to the degree of saliency. Figure 4.4 shows an example image along with its corresponding saliency map.

4.2.2 Gaussian Mixture Models as Generative Models

Gaussian Mixture Models, denoted $\mathcal{G}$ and defined in the following equation, are probabilistic models that permit to represent multimodal probability densities.

$$\mathcal{G}(\boldsymbol{\iota}) = \sum_{j=1}^{k} w_j \, \mathcal{N}(\boldsymbol{\iota}\,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{4.20}$$

where w_j is the weight of the jth component. They are used in this work to model the spatial distribution of the image saliency information ; by first converting the generated saliency map into a binary image (to keep the most salient pixels), then using the white pixels locations to train a GMM. An expectation maximization algorithm is used for this training process. After that, the resulting GMM is used to generate the (u, v) coordinates of a *searching box* center ; and finally, a feature detector is applied in the *searching box* to initialize interest points. This approach permits to control the spatial distribution of features according to image saliency (advantageous for the estimation precision) ; while, for instance, keeping all the image features without selection does not permit to have this kind of control. Figure 4.5 shows the saliency map with the corresponding generated *searching boxes* centers for the example image. As can be seen, the generated centers are located on salient regions, however they are not distributed on the whole image and some image regions are not covered. The approach permits indeed to model the spatial distribution of the saliency information but in a coarse manner, and fine salient regions are not encompassed by the model.

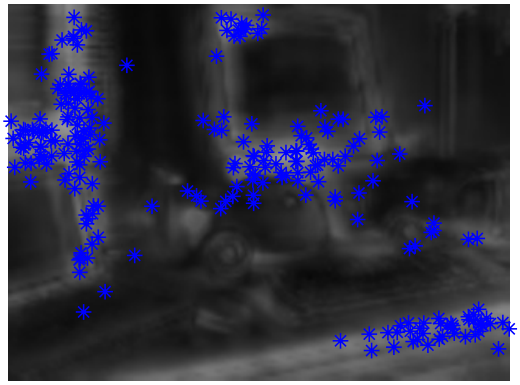


FIGURE *4.5: Saliency map and generated searching boxes centers using one GMM for the whole image.*

One solution to alleviate this drawback is to augment the number of the Gaussians used by the model. This would however result in more important computations, in addition to the increased model complexity that may cause the training process to fail. Another possibility is to divide the image into n_I sub-images and train one GMM for each sub-image, resulting in n_I models. This would permit the model to better encompass all the saliency information, since the area would be smaller. In addition to that, the computational cost of the training step would also be limited in such a case. Figure 4.6 shows the generated centers using this method, where $n_I = 8$. The reason behind this choice is based on the area covered by each sub-image, which should not be too small to have enough data for training GMM and not too large to avoid the drawback mentioned previously. We therefore considered a value $n_I = 8$, which we found in our tests to be a good compromise. The considered sub-images result in a minimum resolution of 120x80 pixels, permitting for complete scene elements to be present in a sub-image.

Another important point is the one concerning the number of used Gaussians in each sub-image. In the proposed strategy, the maximum number of Gaussians per GMM is set to $n_{\mathcal{G}} = 3$, in order to limit the computational cost of the method. This number being sufficient to model the image saliency, as shown in figure 4.6. The training is hence done using $1, 2$, and 3 Gaussians. After that, the GMM with the best quality according the Akaike Information Criteria (AIC) (Akaike et al., 1973), defined in the following equation, is selected :

$$AIC(\mathcal{L}_{max}) = 2\,n_p - ln(\mathcal{L}_{max}), \tag{4.21}$$

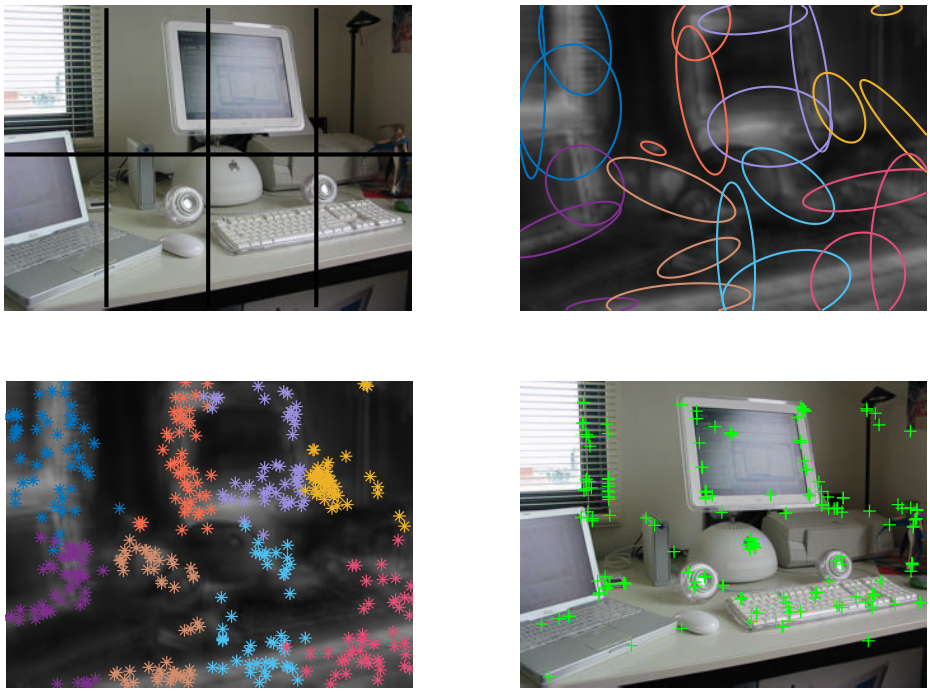


FIGURE *4.6: Saliency map and generated searching boxes centers using one GMM for each sub-image. Different colors correspond to different GMMs.*

---

where n_p is the number of estimated parameters and $\mathcal{L}_{max}$ is the maximum value of the saliency likelihood computed by the expectation maximization algorithm.

The n_I trained GMMs are then used to generate *searching boxes* centers to detect new features, as shown in Figure 4.6. The following section shows the results of the evaluation of the proposed features initialization strategy, when integrated to the two presented systems.

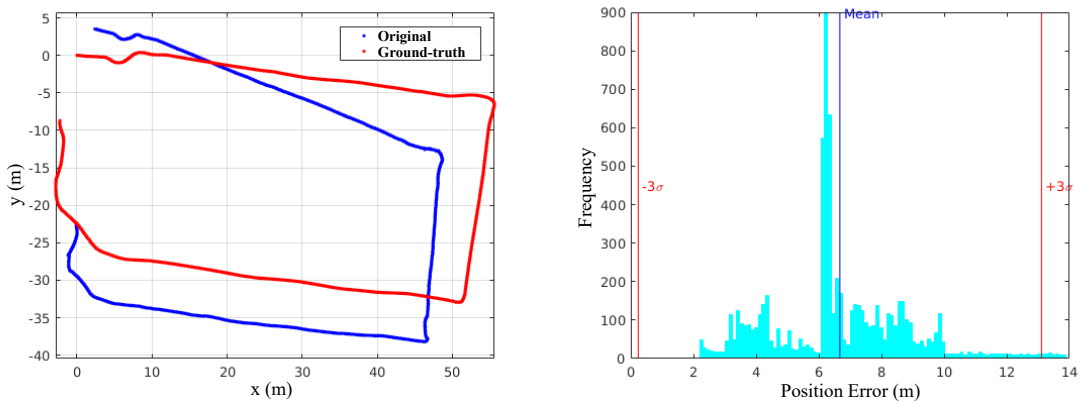
4.3 Results

This section presents the results of the performed experiments on the proposed features initialization strategy ; it is divided into two parts. The first one is related to the Mono-VO (Civera et al., 2009), while the second is related to the RIEKF-VIO (Wu et al., 2017). Furthermore, the RAWSEEDS (Fontana et al., 2014) and the EuroC (Burri et al., 2016) datasets are considered for this evaluation.

FIGURE *4.7: Example images of the Bicocca 26a sequence. Note that the considered algorithms use gray-scale and not color images.*

4.3.1 Mono-VO

In this part, the proposed method is compared to the original features initialization strategy used by (Civera et al., 2009). This strategy considers a uniform probability density to initialize features. Both approaches are tested on the Bicocca 26a sequence of the RAWSEEDS dataset, which consists in various sensor streams gathered from a wheeled robot moving through the Bicocca campus in Milan. To be noted that the only sensor stream considered here is the one from a low resolution monocular camera (240x320 pixels). Example images of this sequence are shown in figure 4.7. Furthermore, the corner detector used in these experiments is the FAST corner detector (Rosten and Drummond, 2005, 2006). The estimated trajectories along with the ATE (Absolute Trajectory Errors) occurrences are shown in figures 4.8 and 4.9.

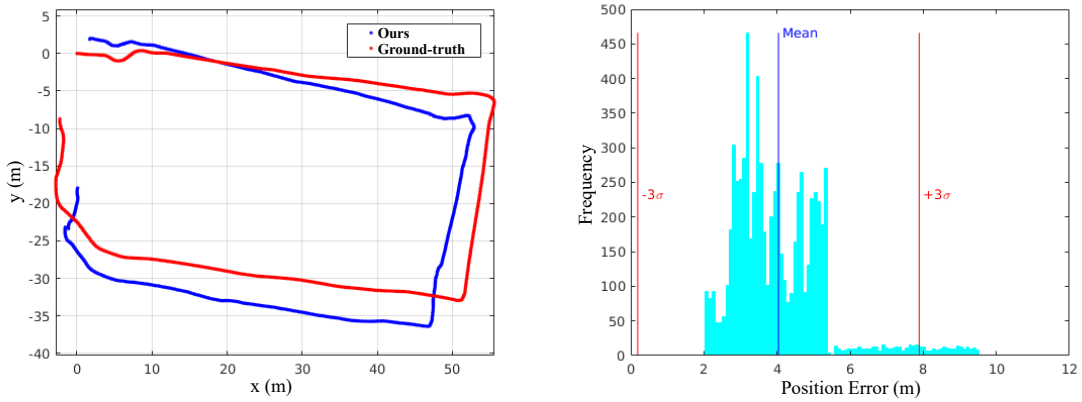


(a) *(b)*

FIGURE *4.8: Estimated trajectory (a) along with the corresponding ATE occurrences (b) of the original method.*

We can see that the trajectory estimated using the proposed features initialization strategy is more consistent than the one using a uniform initialization strategy. The estimation drift is reduced when considering the proposed method. This is supported by the graphics of the ATE occurrences, which show a mean error decrease of more than 30%. The initialization of reliable features has, in fact, an important impact on the estimation, as features with longer lifetime and higher matching percentage result in more precise estimates. The matching percentages of both strategies is shown in figure 4.10.

The proposed strategy permits to increase the features matching percentage all along the navigation, inducing an increase in the features lifetime and a decrease of



(a) *(b)*

FIGURE *4.9: Estimated trajectory (a) along with the corresponding ATE occurrences (b) of the proposed method.*

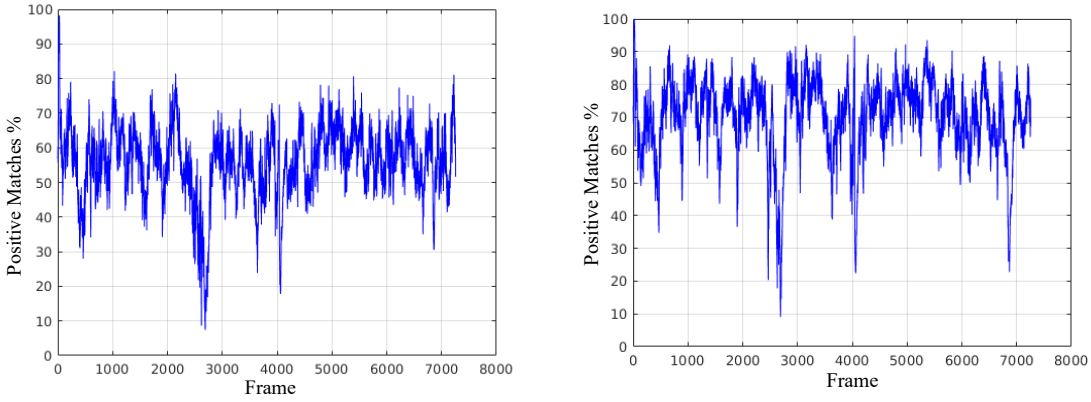
TABLE *4.1: Statistics on the mapped features.*

	Original	Ours
Average features needing to be initialized per frame	10 features	**4** features
Average Initialized Features per Frame	7 features	**4** features
Average Matching Percentage	56.2 %	**68.3** %
Mean Features Lifetime	35 frames	**45** frames

the number of features needed to be initialized. This is shown in Table 4.1, which presents the statistics related to features initialization, matching, and lifetime. Such a behavior permits a more precise estimation of the features' locations and by this mean a better estimation of the camera pose.

4.3.2 RIEKF-VIO

In this part, the proposed initialization strategy is integrated to a RIEKF as proposed by (Wu et al., 2017) and compared to a uniform features initialization strategy on the EuroC dataset (Burri et al., 2016) in the same settings. This dataset was acquired using a Micro Aerial Vehicle (MAV) and consists in pairs of stereo images in addition to synchronized IMU measurements, from which only monocular images and IMU measurements were used for the estimation. Example images of the EuroC dataset are shown in figure 4.11. Furthermore, the corner point detector used in these tests is the minimum eigen value feature detector (Jianbo Shi and Tomasi, 1994). The computed RMSE (Root Mean Squared Error) on the *V1-01-Easy*, *V1-02-Medium*, *V2-01-Easy*,



(a) *(b)*

FIGURE *4.10: Matching percentages of the original strategy (a) and the proposed one (b).*

and *V2-02-Medium* sequences over different executions is shown in figure 4.12.

We can see that the proposed method permits to increase the estimation precision on position in all sequences. Note that the difference between the two methods is more important in the *medium* sequences, which were recorded during fast MAV displacements. This permits to say that detecting features in highly distinguishable image regions plays an important role in the quality of the estimation like was the case for the purely monocular case.

The proposed initialization strategy integrated to the RIEKF is compared with other state-of-the-art algorithms in this part. We accordingly considered the systems studied in (Brossard et al., 2018) :

- The right and left UKF-LG (Unscented Kalman Filter on Lie Groups) proposed by (Brossard et al., 2018) denoted L-UKF-LG and R-UKF-LG respectively.
- The $SE(3)$-based UKF introduced by (Loianno et al., 2016) denoted SE(3)-UKF.
- A UKF that represents the attitude in $SO(3)$ denoted UKF.

The position RMSE of these methods is shown in figure 4.13, where we can see that the mean position RMSE is smaller for the proposed algorithm. The integration of the proposed initialization strategy permitted to increase the estimation precision in the various tests that were conducted, considering different systems and sensor configurations.

FIGURE *4.11: Example images of the EuroC dataset.*

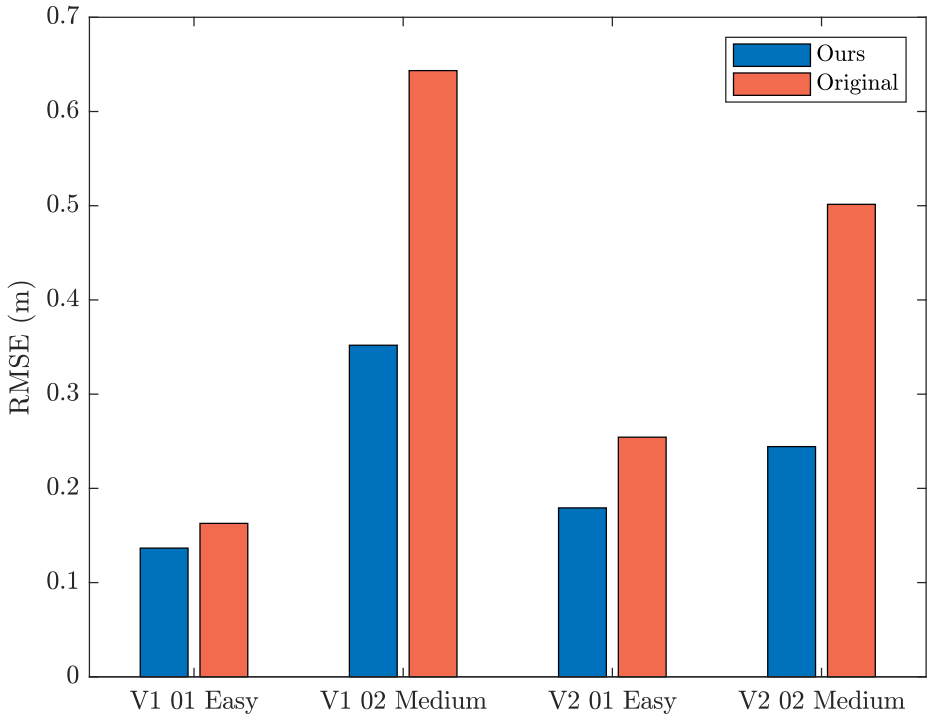


FIGURE *4.12: Root Mean Squared Error on position w.r.t. ground-truth of the proposed method and the reference one.*

V1-03-Difficult

This sequence is challenging because the camera exhibits fast displacements in addition to important variations in the scene illumination. We therefore consider for both methods the SURF descriptor (Bay et al., 2006), known to be robust to these variations in illumination, and show the results of the estimation in figure 4.14.

As can be seen in figure 4.14, the consideration of SURF features permitted all the algorithms to estimate the trajectory of the moving camera. However, the consideration of the proposed features initialization strategy integrated to the RIEKF provides the best estimation precision, whether it be in comparison with the original RIEKF or the other considered algorithms.

4.4 Conclusion

In this chapter, we described and validated a features initialization strategy based on the image saliency information. The proposed approach permits to locate most of the features on the most distinguishable image regions while keeping a whole image

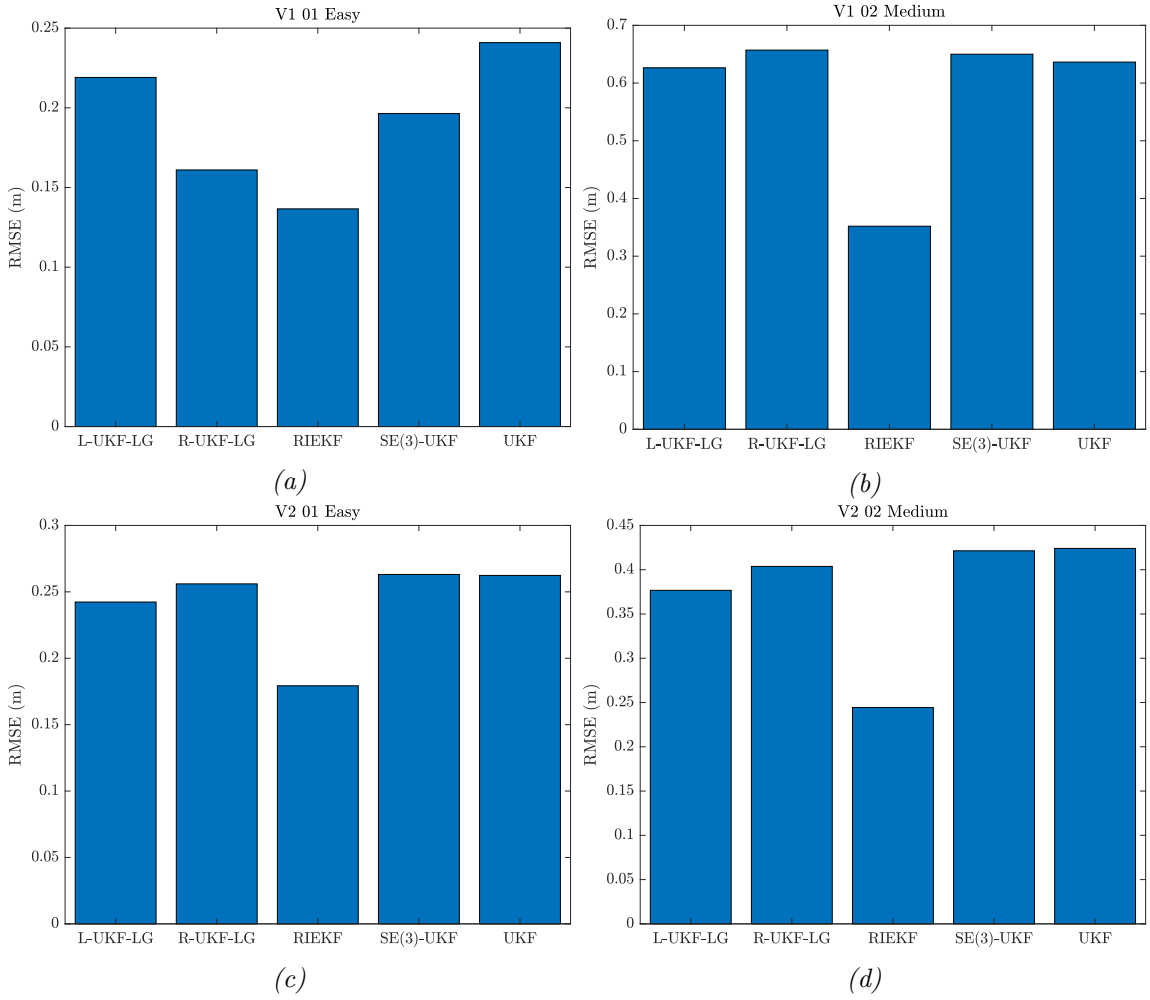


FIGURE *4.13: Root Mean Squared Error on position w.r.t. ground-truth for the four different sequences. RIEKF is considered here with the proposed initialization strategy.*

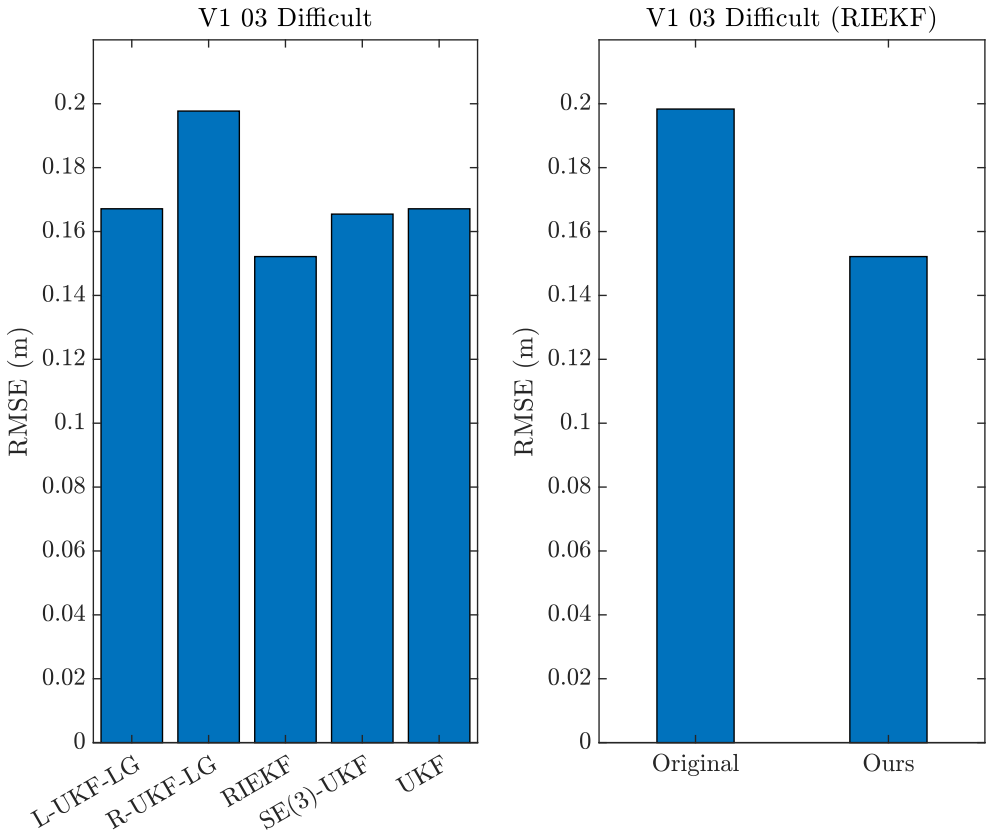


FIGURE *4.14: Root Mean Squared Error on position w.r.t. ground-truth for the V1-03-Difficult. Left : The proposed method is compared to other state-of-art algorithms. Right : The proposed method is compared to the original feature initialization approach.*

coverage, through the exploitation of the image saliency information. In fact, the conducted experiments showed the interest of such a strategy to increase of the estimation precision. Furthermore, other information sources can be considered for the enhancement of VO and V-SLAM systems as the camera motion information that can be obtained from an IMU sensor, which can be the object of future researches.

Chapitre 5

Scale-Space Image Alignment

This chapter focuses on providing a way of using the scale-space representation of images in the context of **direct** image alignment, which can be considered as the problem of maximizing the similarity between two images according to some geometric model. Noting that "direct" in this context means that the algorithm uses directly the pixel brightness of images. The scale-space representation permits to control the image smoothness (blurring) during the optimization process ; permitting to eliminate local minima, which may cause direct alignment algorithms to fail.

A work that is related to the approach presented in this chapter is (Lee et al., 2013), which can jointly estimate geometric (translation and rotation), and scale parameters between images. However, their approach differs from ours in the sense that it uses the scale-space representation to handle the difference in resolution between images in the context of super-resolution image alignment. Our method on the other hand considers the scale-space representation to control the amount of image details used during the optimization to enlarge the basin of convergence of the algorithm.

In this chapter, the used methodology to integrate the scale-space representation of images to the direct optimization scheme is first shown in section 5.1 ; focusing on the following geometric models : translations (usually used for optical flow estimation) and homographies (considered for plane tracking and augmented reality applications). The results of the various experiments that were conducted to evaluate the proposed methods is then presented and discussed in section 5.2. Finally, a conclusion about the work presented in this chapter is proposed in section 5.3.

5.1 Method Description

The goal is to align image pairs $\{\boldsymbol{I}_1, \boldsymbol{I}_0\}$ obtained from a moving camera in a static scene (or at least mostly static) according to some predefined geometric model. It can therefore be posed as the problem of finding the parameter vector $\boldsymbol{\xi}$ of a warping function $\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi})$ that maps between the two image coordinates, where $\mathbf{x} = (u, v)^T$ represents pixel coordinates. By making the photo-consistency assumption this is equivalent to the following optimization problem (Lucas and Kanade, 1981) :

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}} \left[\boldsymbol{I}_1(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi})) - \boldsymbol{I}_0(\mathbf{x}) \right]^2 . \tag{5.1}$$

We propose to use the scale-space representation (Lindeberg, 2014) $G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}); \lambda)$ and $G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref})$ of $\boldsymbol{I}_1(\mathbf{x})$ and $\boldsymbol{I}_0(\mathbf{x})$, respectively, according to the scale parameters λ and λ_{ref} :

$$G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}); \lambda) = \boldsymbol{I}_1(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi})) * \mathbf{g}(\mathbf{x}; \lambda), \tag{5.2}$$

and

$$G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref}) = \boldsymbol{I}_0(\mathbf{x}) * \mathbf{g}(\mathbf{x}; \lambda_{ref}), \tag{5.3}$$

where $*$ is the convolution operator and $\mathbf{g}(\mathbf{x}; \lambda)$ is an isotropic Gaussian kernel around mean $\boldsymbol{\mu}$:

$$\mathbf{g}(\mathbf{x}; \lambda) = \frac{1}{2\pi\lambda^2} \exp(-\frac{(\mathbf{x} - \boldsymbol{\mu})^2}{2\lambda^2}). \tag{5.4}$$

An illustration of the scale-space representation of an example image is shown in Figure 5.1. This results in the following reformulation of equation (5.1) :

$$\tilde{\boldsymbol{\xi}}^* = \underset{\tilde{\boldsymbol{\xi}}}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathbf{x}} \left[G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}); \lambda) - G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref}) \right]^2 , \tag{5.5}$$

where $\tilde{\boldsymbol{\xi}} = (\boldsymbol{\xi}, \lambda)^T$ is the augmented vector containing the geometric and scale parameters. This parameter vector can be iteratively optimized by adopting a *forwards additive* approach, as shown in the expression below :

$$\tilde{\boldsymbol{\xi}} \leftarrow \tilde{\boldsymbol{\xi}} + \alpha \Delta \tilde{\boldsymbol{\xi}}, \tag{5.6}$$

here α is a damping parameter. Equation (5.5) is a nonlinear least squares problem that can be solved by considering a Gauss-Newton scheme, similarly to the approach of (Baker and Matthews, 2004). The expression of the increment is then :

$$\Delta \tilde{\boldsymbol{\xi}} = \mathbf{H}^{-1} \sum_{\mathbf{x}} \left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial \tilde{\boldsymbol{\xi}}} \right]^T \left[\mathbf{G}_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref}) - G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}); \lambda) \right] , \tag{5.7}$$

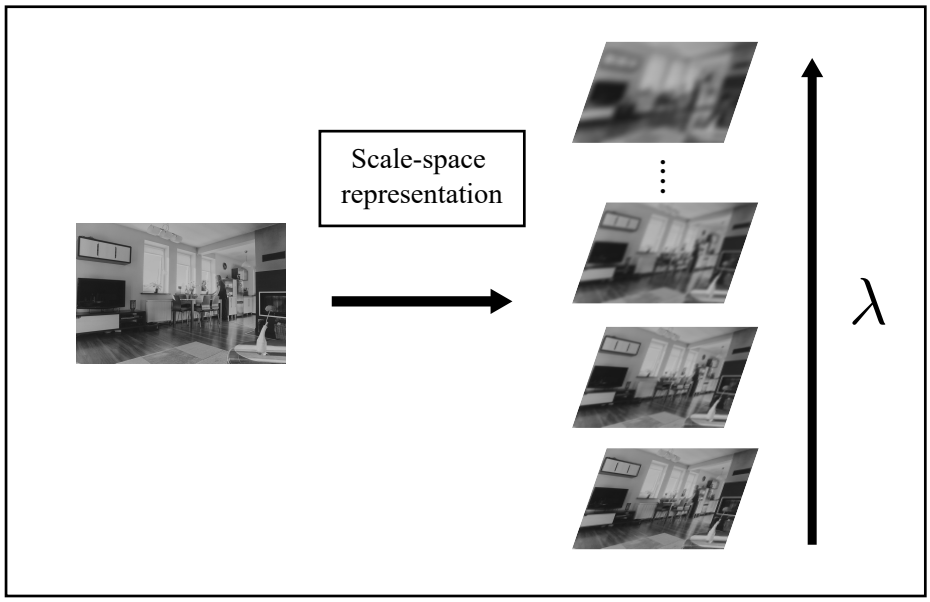


FIGURE *5.1: The scale-space representation of an example image. The value of the scale parameter controls how much the image is blurred.*

H represents the (Gauss-Newton approximation to the) *Hessian* matrix :

$$\mathbf{H} = \sum_{\mathbf{x}} \left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial \tilde{\boldsymbol{\xi}}} \right]^T \left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial \tilde{\boldsymbol{\xi}}} \right]. \tag{5.8}$$

The parameters $\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})$ and λ of $G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi});\lambda)$ have been omitted for compactness in the writing of the *Jacobian* line $\frac{\partial G_{\boldsymbol{I}_1}}{\partial \tilde{\boldsymbol{\xi}}} = \left[\nabla_{u,v} G_{\boldsymbol{I}_1} \frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}}, \nabla_\lambda G_{\boldsymbol{I}_1} \right]$ of a pixel $\mathbf{x}$. Gradients of $G_{\boldsymbol{I}_1}$ in space and scale are evaluated at $\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})$ using finite differences :

$$\nabla_u G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left. \frac{G_{\boldsymbol{I}_1}(u+l_u,v;\lambda) - G_{\boldsymbol{I}_1}(u-l_u,v;\lambda)}{2l_u} \right|_{\mathbf{x}=\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})} \tag{5.9}$$

$$\nabla_v G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left. \frac{G_{\boldsymbol{I}_1}(u,v+l_v;\lambda) - G_{\boldsymbol{I}_1}(u,v-l_v;\lambda)}{2l_v} \right|_{\mathbf{x}=\boldsymbol{\mathcal{W}}(\mathbf{x};\boldsymbol{\xi})} \tag{5.10}$$

$$\nabla_\lambda G_{\boldsymbol{I}_1}(u,v;\lambda) \approx \left. \frac{\boldsymbol{I}_1 * (\mathbf{g}(\mathbf{x};\lambda+l_\lambda) - \mathbf{g}(\mathbf{x};\lambda-l_\lambda))}{2l_\lambda} \right|_{\mathbf{x}=\mathcal{W}(\mathbf{x};\boldsymbol{\xi})}, \tag{5.11}$$

where l_u, l_v and l_λ are the finite differences step sizes. The *Jacobian* needed to compute the increments depends on the gradients of $G_{\boldsymbol{I}_1}$ and on the derivative of the warping function w.r.t. the parameters $\boldsymbol{\xi}$, which are presented in the next part for the following transformations : translations and homographies. It is worth noting that the intensity values of the images are normalized between 0 and 1 to prevent the optimization

from being unstable, due to important increment values. Furthermore, $l_u = l_v = 1$ as usually done to compute image gradients and $l_\lambda = 0.5$, permitting to have a unitary denominator.

5.1.1 Translation

The warping function modeling translations uses a 2-vector $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$ to account for the displacement in each image axis direction :

$$\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}) = \begin{pmatrix} u + \xi_1 \\ v + \xi_2 \end{pmatrix}. \tag{5.12}$$

Consequently, its *Jacobian* is expressed in the following manner :

$$\frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{5.13}$$

5.1.2 Homography

The warping function modeling a homography transformation $\mathbf{M}$ uses a 8-vector $\boldsymbol{\xi} = (\xi_1, \xi_2, ..., \xi_8)^T$:

$$\mathbf{M} = \begin{pmatrix} \xi_1 & \xi_4 & \xi_7 \\ \xi_2 & \xi_5 & \xi_8 \\ \xi_3 & \xi_6 & 1 \end{pmatrix}, \tag{5.14}$$

leading to :

$$\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi}) = \begin{pmatrix} \frac{\xi_1 u + \xi_4 v + \xi_7}{\xi_3 u + \xi_6 v + 1} \\ \frac{\xi_2 u + \xi_5 v + \xi_8}{\xi_3 u + \xi_6 v + 1} \end{pmatrix}. \tag{5.15}$$

As a result, the expression of its *Jacobian* is :

$$\frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}} = \frac{1}{c} \begin{pmatrix} u & 0 & -u.a & v & 0 & -v.a & 1 & 0 \\ 0 & u & -u.b & 0 & v & -v.b & 0 & 1 \end{pmatrix}, \tag{5.16}$$

where

$$a = \frac{\xi_1 u + \xi_4 v + \xi_7}{c}$$
$$b = \frac{u\left[\xi_2 u + \xi_5 v + \xi_8\right]}{c}$$
$$c = \xi_3 u + \xi_6 v + 1.$$

5.1.3 Wrap-up

The previously obtained expressions permit to iteratively estimate the transformations parameters between image pairs. It is thenceforth possible to evaluate the proposed approach, referred as AFA-LK (*Adaptive Forwards Additive* Lucas Kanade), which is the object of the next section.

5.2 Experiments

5.2.1 Overview

The proposed method is evaluated for the two types of transformations mentioned before : pure translations and homographies. Concerning translations, the AFA-LK is compared with the original *Forwards Additive* LK (FA-LK) algorithm (Baker and Matthews, 2004) with various initialization settings. When homography transformations are considered, the comparison is done between the AFA-LK and state-of-the-art non-learning and learning-based methods (RANSAC+SIFT Homography, ESM, Supervised Descent Method, Conditional-LK) (Vedaldi and Fulkerson, 2008; Benhimane and Malis, 2004; Xiong and la Torre, 2014; Lin et al., 2016).

The considered algorithms are evaluated using the MS-COCO dataset (Lin et al., 2014) and the Yale face database (Belhumeur et al., 1997). Furthermore, the 110000 images used for the evaluation of alignment with translations and homographies are available as the AFAMIS dataset[1].

5.2.2 Translation Transformation

Both algorithms are evaluated using all the images of the validation set of the MS-COCO dataset (Lin et al., 2014), which consists of 5000 images of complex everyday scenes, by following the subsequent process for the generation of a testing bench (Figure 5.2). Every image is used to generate a triplet $\{\boldsymbol{I}_1, \boldsymbol{I}_0, \boldsymbol{\xi}_{ref}\}$ by first translating the original image according to $\boldsymbol{\xi}_{ref}$, which is the reference translation applied in the u and v axes of this original image according to a uniform distribution within the range $[-10, 10]$ pixels. $\boldsymbol{I}_0$ is set to be the square region (29x29 pixels) around a randomly picked point of interest in the translated image and $\boldsymbol{I}_1$ is the square region of the same size and at the same location in the image, as shown in Figure 5.2. This transformation range ($[-10, 10]$ pixels) is considered because optical flow estimation is done for small

[1] `http://mis.u-picardie.fr/~g-caron/pub/data/AFAMIS_dataset.zip`

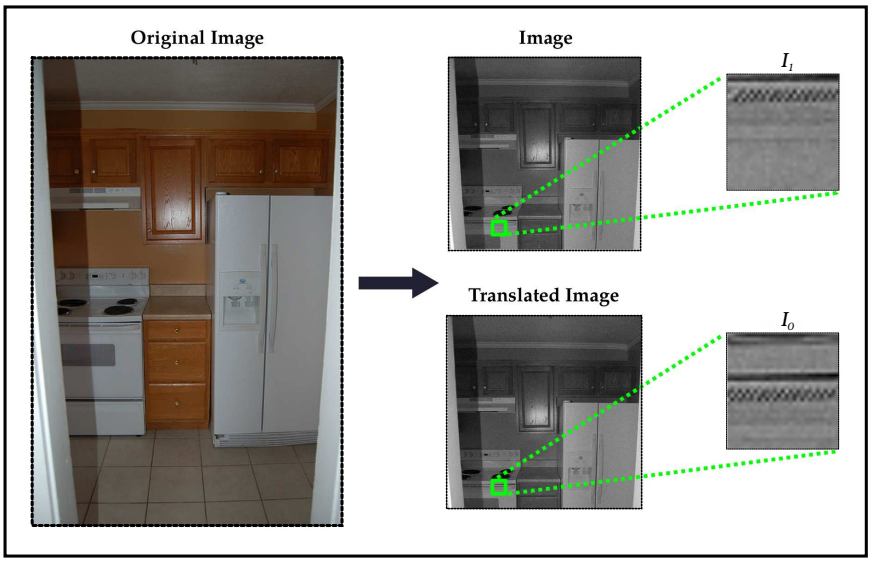


FIGURE *5.2: Generation of* $\boldsymbol{I}_0$ *and* $\boldsymbol{I}_1$*. The original image is translated according to* $\boldsymbol{\xi}_{ref}$*.* $\boldsymbol{I}_0$ *is the square region (29x29) around a randomly picked corner in the translated image and* $\boldsymbol{I}_1(\boldsymbol{x})$ *is the square region of the same size and at the same location in the image.*

displacements and consequently the used images sizes are small (29x29 pixels). Each algorithm is provided with the input and the template images $\boldsymbol{I}_1$ and $\boldsymbol{I}_0$, in addition to the initial parameter values $\boldsymbol{\xi}_{init} = [0, 0]^T$.

Cost Function : A 3D visualization of the cost function (equation 5.5), at different scales, where the proposed algorithm was able to converge towards the correct value is presented in Figure 5.3. The red marker corresponds to the location of the reference translation ($\boldsymbol{\xi}_{ref}$). It shows the effect of the scale parameter on the shape of the cost function. When its value is high, the cost function is smoothed (typically at the beginning of the optimization), which permits to avoid the local minima. Whereas when the value of the scale parameter is small (typically at the end of the optimization) the cost function is sharp, which allows reaching a high estimation precision. This behavior permits to implement a coarse-to-fine approach, where the scale parameter is automatically tuned according to the needs of the optimization process and not empirically as done in (Sharmin and Brad, 2012).

Experimental Study of Parameters Impact : The results of the evaluations on the validation set of the MS-COCO dataset are presented in Figures 5.4 and 5.5, which represent the Error Cumulative Distribution (ECD) of the AFA-LK and FA-LK algorithms for different values of α and λ_{init} (the initial value of λ for $\boldsymbol{I}_1$). The

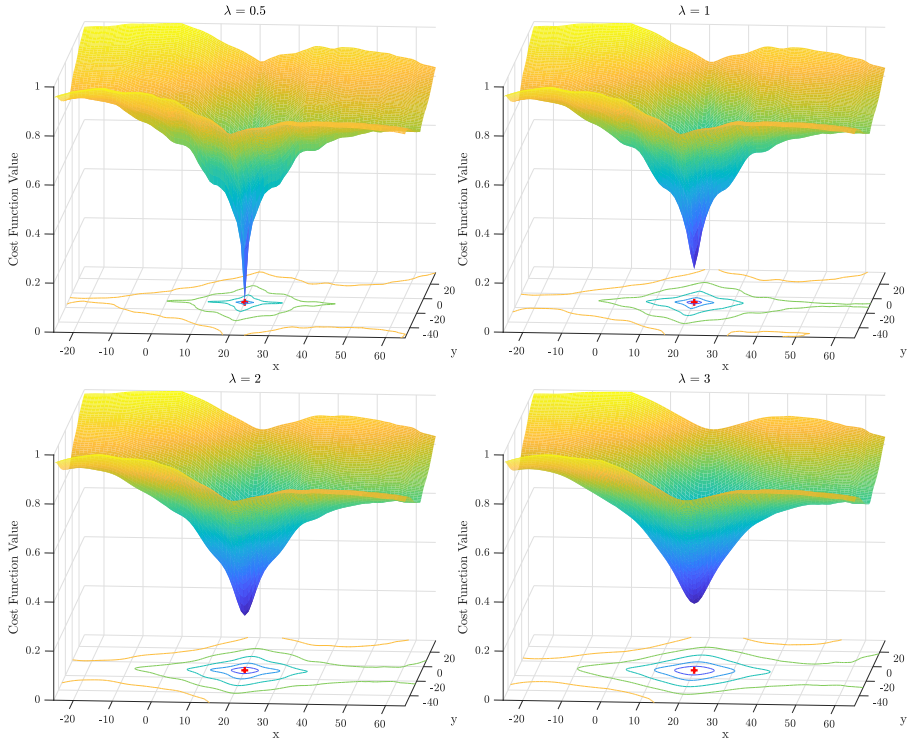


FIGURE *5.3: Cost functions for different values of the scale parameter λ, the greater is λ the smoother is the cost function.*

---

metric error used for computing the ECD is :

$$e_c = ||\boldsymbol{\xi} - \boldsymbol{\xi}_{ref}||_2 \tag{5.17}$$

where $||.||_2$ represents the L^2 norm of the error vector $(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref})$. This metric permits to evaluate the ability of each algorithm to converge under important displacements, which gives us insights about the basin of convergence of the AFA-LK and the FA-LK, the effect of the values of α and λ_{init}, and the advantage of considering the optimization of the parameters in the scale-space domain. Each algorithm is considered to have converged when the metric error is less than 1.0 pixel (10^0 in the figures) (Baker and Matthews, 2004). The maximum number of iterations is fixed to 30 iterations to allow the algorithms converging. $\lambda_{ref} = 0.5$ to obtain a scale-space representation close to $\boldsymbol{I}_0$, permitting to increase the estimation precision. $\lambda_{init} = 4$ for the evaluation on the value of α (Figure 5.4), and $\alpha = 0.3$ for the evaluation on the value of λ_{init} (Figure 5.5).

The proposed algorithm permits to increase the convergence domain in comparison to the original Lucas-Kanade algorithm without loss in estimation precision.

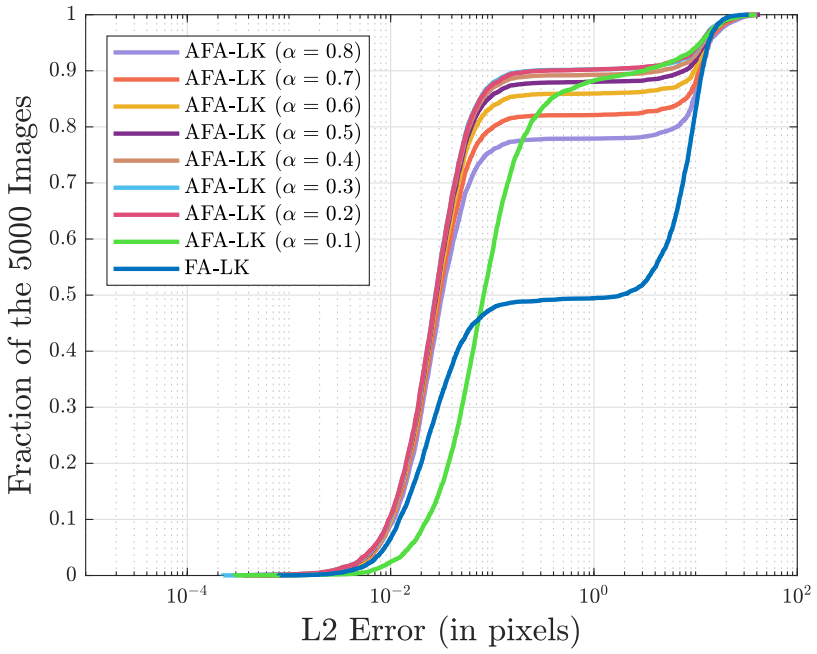


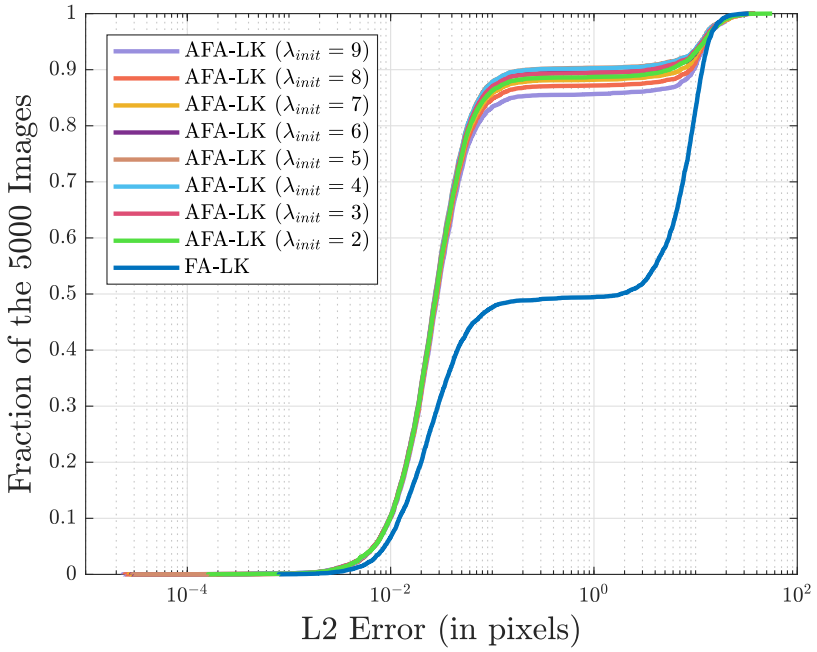
FIGURE *5.4: ECD of the AFA-LK for different values of α compared to the FA-LK.*

First, concerning the effect of the value of α presented in Figure 5.4, we can see that values inferior or equal to 0.6 allows to have a convergence rate superior to 85%, and the values of α equal to 0.2 and 0.3 provide the best results. This parameter, indeed, controls the step size in each iteration. When its value is too large, the algorithm tends to be unstable and diverge. Conversely, when the value of α is too small the speed of convergence becomes too slow. The value $\alpha = 0.3$ will be used in the following validations.

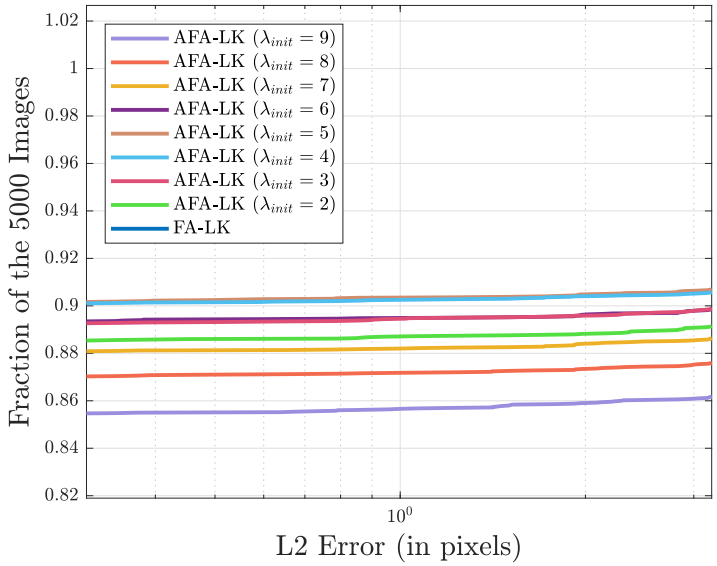
Second, regarding λ_{init}, Figure 5.5 shows that the AFA-LK performs similarly for the values of $\lambda_{init} \in [3; 6]$. In fact, on the one hand, when the value of the scale parameter is too large, the algorithm needs a greater number of iterations to converge. On the other hand, when its value is too small, there is not enough smoothing and the algorithm is unable to deal with challenging cases (large motions, local minima).

5.2.3 Homography Transformation

In this part, the scale-space alignment is evaluated with respect to state-of-the-art methods for the task of 2D homography estimation. First, the comparison is done between the AFA-LK and non-learning-based approaches (similarly to (Chang et al., 2017)) : the variants of the LK algorithm (*forwards additive*, *inverse compositional*, *forwards compositional*) (Baker and Matthews, 2004) and the ESM (Benhimane



(a)



(b)

FIGURE *5.5: (a) ECD of the FA-LK and AFA-LK for different values of* λ_{init}*. (b) is a zoom of (a) around the error of 1 pixel.*

and Malis, 2004), as dense direct methods ; along with SIFT+RANSAC Homography (Hartley and Zisserman, 2004) (we used the implementation given in the vlfeat[1] library), as a feature-based method. We used the validation set of the MS-COCO dataset (Lin et al., 2014) for this comparison.
Second, the AFA-LK is compared to learning-based methods : the Conditional-LK (Lin et al., 2016) and the SDM (Supervised Descent Method) (Xiong and la Torre, 2014) in a set of images from the Yale face database (Belhumeur et al., 1997) (sample images are shown in Figure 5.10).

Comparison With Geometric Methods

We considered a procedure similar to the one of (Chang et al., 2017) for the validation of the algorithms. Every image of the validation set of the MS-COCO dataset was used to generate a template and an input image. The image was first down-sampled so that the shorter side was equal to 240 pixels. Then, a square was randomly cropped in the resized image and set to be the input image $\boldsymbol{I}_1$ (192x192 pixels). Next, in order to generate the template $\boldsymbol{I}_0$, we selected a 128x128 square region centered in $\boldsymbol{I}_1$ and perturbed the four corners of the square using a uniform distribution within the range $[-42, +42]$ pixels (represented by the red quadrilateral in Figure 5.6). After that, the warp $\mathcal{W}(\mathbf{x}; \boldsymbol{\xi}_{ref})$ was defined by the homography that maps the template corners to the perturbed ones, and $\boldsymbol{I}_0$ was generated by applying the reference warp to the input image $\boldsymbol{I}_1(\mathcal{W}(\mathbf{x}; \boldsymbol{\xi}_{ref}))$. The initial warp $\mathcal{W}(\mathbf{x}; \boldsymbol{\xi}_{init})$ is the translation that maps the template domain to the image domain.

In order to avoid unrealistic shape distortions resulting from the homography transformation, every angle of the quadrilateral was restricted to be less than $\frac{3}{4}\pi$. Figure 5.7 shows examples of the pairs $\{\boldsymbol{I}_1, \boldsymbol{I}_0\}$. A Gaussian noise of standard deviation 0.02 [2] was added to both the intensity values of $\{\boldsymbol{I}_1$ and $\boldsymbol{I}_0\}$ to make the dataset more challenging. Based on (Chang et al., 2017), we used the *corner error* as the metric for the tests, which is defined as follows :

$$e_c(\boldsymbol{\xi}, \boldsymbol{\xi}_{ref}) = \frac{1}{4} \sum_{j=1}^{4} ||\mathcal{W}(\mathbf{e}_j; \boldsymbol{\xi}) - \mathcal{W}(\mathbf{e}_j; \boldsymbol{\xi}_{ref})||_2 \tag{5.18}$$

where $\mathbf{e}_j$ represents the jth corner coordinates. Figure 5.8 shows the *corner error* cumulative distribution of the following algorithms :

[1] http ://www.vlfeat.org/

[2] corresponding to 2% of pixel intensities since they belong to $[0, 1]$ (see section 5.2.1)

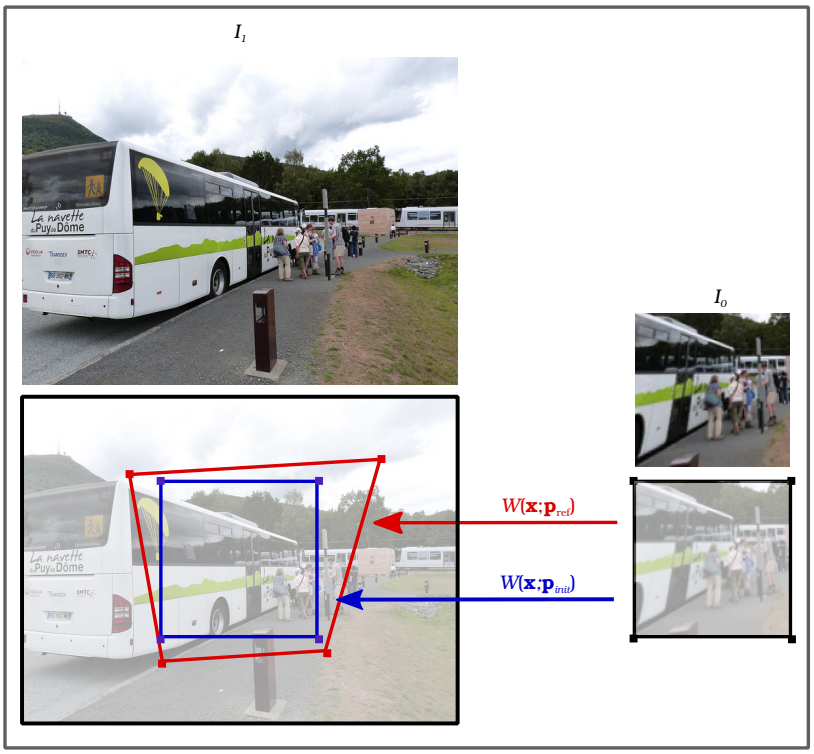


FIGURE *5.6: Generation of the template image* $\boldsymbol{I}_0$ *from the input image* $\boldsymbol{I}_1$. $\mathcal{W}(\boldsymbol{x};\boldsymbol{\xi}_{init})$ *represents the translation that maps the template domain (the dark square) to the image domain (blue square),* $\mathcal{W}(\boldsymbol{x};\boldsymbol{\xi}_{ref})$ *represents the homography that maps the template corners (in black) to the perturbed corners (in red).*

---

1. **AFA-LK :** Parameters are set to the following values $\alpha = 0.3$, $\lambda_{ref} = 0.5$, and $\lambda_{init} = 12$ because we noticed in experiments that a large value of the initial scale parameter is needed when the considered images are large (128x128 pixels) in comparison with the ones of the validations presented in section 5.2.2 (29x29 pixels). The number of iterations is equal to 30.

2. **LK variants :** We considered the *forwards additive*, *inverse compositional*, and *forwards compositional* variants of the LK algorithms and used the code provided by (Baker and Matthews, 2004). The integration of a pyramid approach to the *forwards additive* LK (3 levels) is also presented. The number of iterations is fixed to 30.

3. **ESM :** We used the code provided by (Mei et al., 2008) and set the number of iterations to 30.

4. **SIFT + RANSAC :** A feature-based homography estimation algorithm implemented in the VLFeat library (Vedaldi and Fulkerson, 2008).

FIGURE *5.7: Samples of the input images from the MS-COCO dataset and the corresponding templates used for testing the algorithms.*

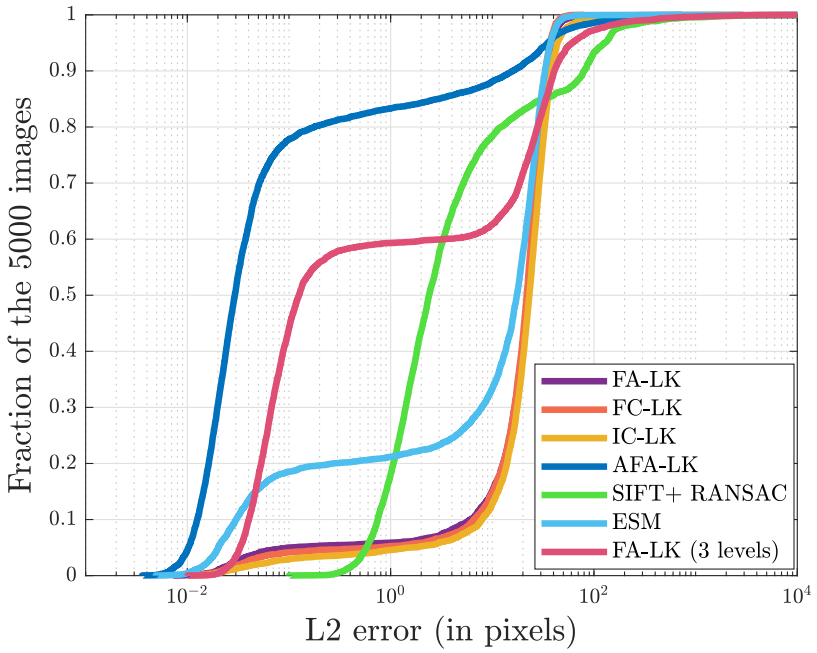
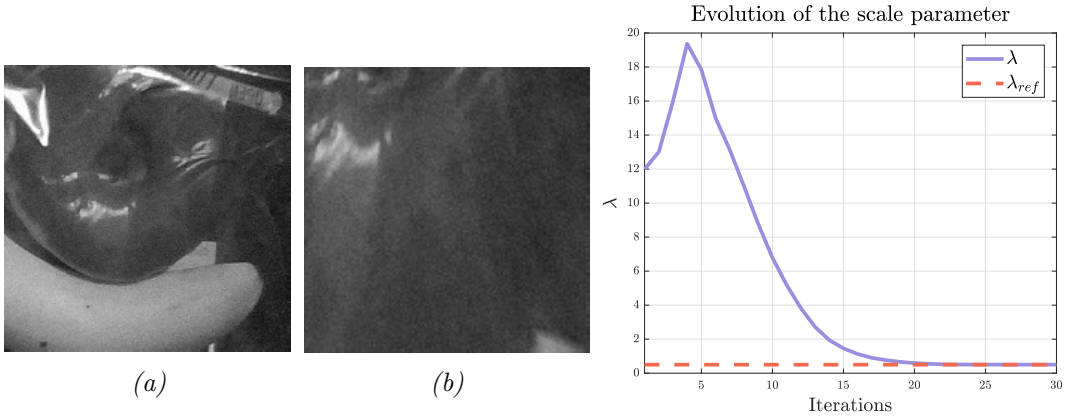


FIGURE *5.8: Corner error cumulative distribution for the different algorithms.*



(a) *(b)* *(c)*

FIGURE *5.9: (a, b) Example where the SIFT+RANSAC algorithm was unable to converge ((a) represents $\boldsymbol{I}_1$ and (b) represents $\boldsymbol{I}_0$). (c) Example where the AFA-LK was the only algorithm to converge. The value of λ increases, permitting to suppress local minima then converges to λ_{ref}.*

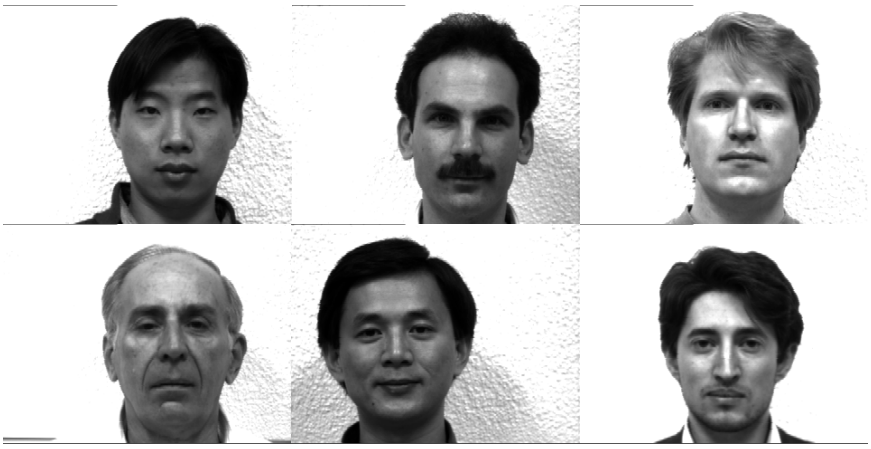


FIGURE *5.10: Samples of the subjects of the Yale face database.*

Figure 5.8 shows that among the various algorithms, the variants of the LK (FA, IC, FC) exhibit the lowest convergence rate because they are unable to cope with huge displacements (up to 42 pixels, as a remembering). The ESM, for its part, provides better results and we can notice its good accuracy when converging. The SIFT + RANSAC algorithm exhibits a comparable convergence domain but is less precise than the pixel-based methods. Because it is a feature-based algorithm, the SIFT + RANSAC fails to deal with low textured images (Figure 5.9). Our method outperforms largely the other algorithms in terms of precision and convergence domain, even when comparing it to a pyramidal approach (3 levels pyramid), which appears unable to handle cases where the motion is too large (Figure 5.9 shows an example where the algorithm increases the value of λ to remove local minima then converges). We can clearly see that the AFA-LK is able to deal with important displacements thanks to its ability to continuously adopt the scale, even when it must increase to get in the convergence domain (Figure 5.9(c)). Furthermore, because it is a pixel-based method, it is able to provide accurate results even with low textured images.

Comparison With Learning Methods

In this evaluation, we followed the methodology of (Lin et al., 2016) in order to compare our algorithm to their method. Every input image $\boldsymbol{I}_1$ is used to generate a template $\boldsymbol{I}_0$ in the following manner. A manually selected square region is first defined in the image. The four square corners are then individually perturbed using independent and identically distributed Gaussian noise of standard deviation σ in addition to a single translational noise of the same distribution applied to every corner. Finally, template $\boldsymbol{I}_0$ is generated from the perturbed corners similarly to

section 5.2.3. We followed the procedure described in (Lin et al., 2016) to train the different regressors of the Conditional-LK and the SDM.

As mentioned before, convergence is achieved when the point RMS error defined in equation (5.19) is less than 1.0 pixel. The convergence rates of the considered algorithms can hence be computed for different values of σ (the convergence rates of each subject are computed from a total of 1000 trials for each value of σ), as shown in Figure 5.11. It is worth noting that a subset of the Yale face database (Figure 5.10) is used for this validation. Moreover, the learning methods were trained using a value of $\sigma = 1.2$ pixel (represented by a vertical dashed line in the figure) as done in (Lin et al., 2016).

$$P_{RMSE} = \sqrt{\frac{1}{4}\sum_{j=1}^{4}\Big(\mathcal{W}(e_j;\boldsymbol{\xi}) - \mathcal{W}(e_j;\boldsymbol{\xi}_{ref})\Big)^2} \tag{5.19}$$

We can see in Fig 5.11 that the AFA-LK shows superior convergence properties in comparison to both the Conditional-LK and the SDM. In addition to that, the proposed method has the advantage of requiring no training and consequently permits a considerable save of time.

The previous results showed the interest of considering the scale-space image representation for leveraging the robustness against important transformations when considering translation and homography transformations.

5.3 Conclusion

This chapter presented an image alignment algorithm, based on the *forwards additive* variant of the Lucas-Kanade algorithm, that considered a scale-space image representation. The integration of the scale parameter in the optimization process permits to control the degree of smoothness of the cost function in an automatic manner. The proposed algorithm is therefore able to tune the scale parameter according to its needs. The value of the latter, being high in the beginning of the optimization, permits to enlarge the basin of convergence. Towards the end of the optimization, when the algorithm tends to converge, the value of the scale parameter is decreased, resulting in a fine final estimation. However, some limitations still persist. Notably, concerning the initialization of the scale parameters, which is still not done automatically.

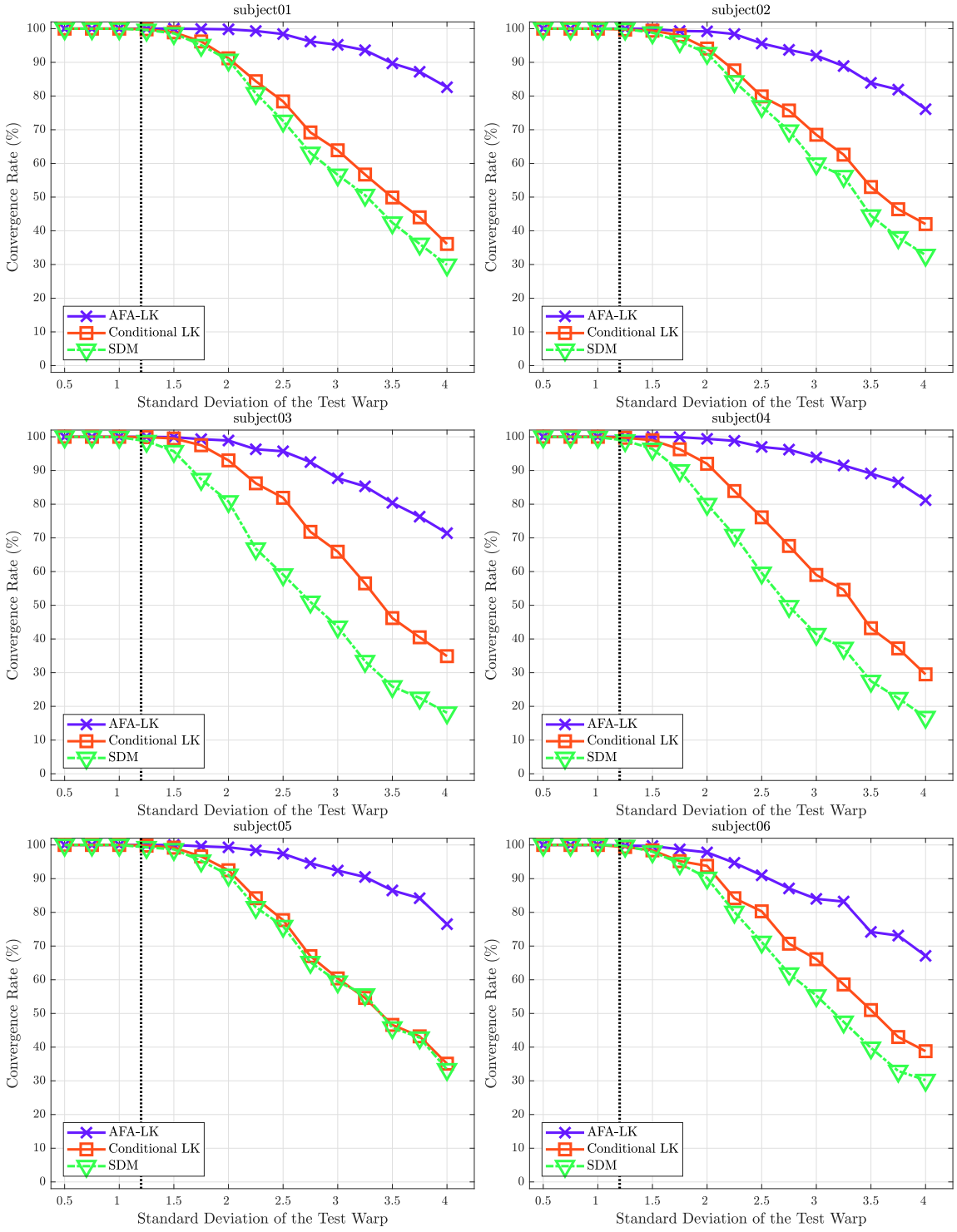


FIGURE *5.11: Convergence rates for different subjects of the Yale database, computed over a total of 15000 image pairs for each subject.*

Chapitre 6

Combining Vision and Depth Information for VO and SLAM

Following the way paved in the previous chapter, we consider the pixel brightness of images in addition to the depth information provided by a depth sensor to estimate the displacement of a moving agent in the 3D space. The resulting algorithm represents the elementary building block applied in the SLAM system presented in this chapter.

A work related to the proposed image alignment algorithm is the CVO (Continuous direct sparse Visual Odometry) of (Jadidi et al., 2019). This approach considered a continuous formulation for RGB-D image alignment, which permitted to increase the estimation precision in comparison to existing direct approaches. Such a method can be seen complementary to ours that aims at increasing the robustness with respect to large camera displacements.

The rest of this chapter is organized in the following manner. First, the proposed image alignment algorithm that takes advantage of the scale-space theory to increase the robustness towards large displacements is described in section 6.1. Second, section 6.2 shows how this image alignment algorithm is integrated to a SLAM system that permits the fusion of the information provided by vision and depth sensors. Third, the results of the conducted experiments are shown in section 6.3, exhibiting the performances of the considered algorithms. Finally, conclusions and perspectives about the proposed algorithm are provided in section 6.4.

6.1 RGB-D Image Alignment

This section presents two RGB-D image alignment algorithms that use, at each time t, the last consecutive undistorted images ($\boldsymbol{I}_t$, $\boldsymbol{I}_{t-1}$) and the corresponding depth images ($\boldsymbol{D}_t$, $\boldsymbol{D}_{t-1}$) to estimate the relative 3D motion (rigid body transformation) of the camera. The proposed approach uses the scale-space representation of images in order to increase its robustness towards large camera displacements. This is done through the continuous and automatic adaptation of the images scale within each level of a multi-resolution image pyramid. The use of image pyramids and the consideration of 3D camera motions are, in fact, the main differences with the AFA-LK, presented in Chapter 5. The other approach presented here is the algorithm proposed by (Steinbruecker et al., 2011), which also uses multi-resolution image pyramids. The main difference between the two approaches described in this section being that the proposed one automatically adapts the scale of the images within each pyramid level during the optimization and is consequently referred to as OP-B (Optimized Pyramid-Based), while the approach of (Steinbruecker et al., 2011) does not, hence called PP-B (Photometric Pyramid-Based). PP-B uses a constant scale linked to the resolution in the multi-resolution image pyramid. An example showing the benefit of the proposed OP-B method is presented in Figure 6.1.

6.1.1 PP-B Image Alignment

In order to align the image pair $\{\boldsymbol{I}_0, \boldsymbol{I}_1\}$ using the corresponding image depth pair $\{\boldsymbol{D}_0, \boldsymbol{D}_1\}$, the PP-B algorithm considers the photo-consistency assumption mentioned in section 5.1. This assumption permits to express the RGB-D image alignment problem through the optimization of a cost function similar to the one of equation (5.1). This cost is described in a linearized form (first order approximation) below :

$$\boldsymbol{\xi}^* = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_i \left[\mathbf{r}_i(0) + \mathbf{J}_i \, \Delta\boldsymbol{\xi}\right]^2, \tag{6.1}$$

where $\mathbf{r}_i(0) = \boldsymbol{I}_1(\boldsymbol{\mathcal{W}}(\mathbf{x}_i, 0)) - \boldsymbol{I}_0(\mathbf{x}_i)$ is the residual and $\mathbf{J}_i = \frac{\partial \mathbf{r}_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ the ith row of the *Jacobian* matrix corresponding to the ith pixel. The specific aspect of this alignment problem consists, however, in the parameter vector $\boldsymbol{\xi}$ used to describe the rigid body transformation and the corresponding warping function $\boldsymbol{\mathcal{W}}(\mathbf{x}; \boldsymbol{\xi})$. The former is represented using twist coordinates :

$$\boldsymbol{\xi} = \begin{pmatrix} \mathbf{v}^T & \mathbf{w}^T \end{pmatrix}^T = \begin{pmatrix} v_1 & v_2 & v_3 & w_1 & w_2 & w_3 \end{pmatrix}^T, \tag{6.2}$$

FIGURE *6.1: The considered algorithms aim at aligning the image pair* $\{\boldsymbol{I}_0, \boldsymbol{I}_1\}$*, where the images resulting from the alignment are* $G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}, \lambda)$ *and* $\boldsymbol{I}_1(\boldsymbol{\mathcal{W}})$*. Here the fixed scale pyramid-based method (PP-B) was unable to align the image pair because it fell in a local minimum, while the proposed optimized scale pyramid-based method (OP-B) succeeded because it is able to automatically adapt the image smoothing to suppress local minima.*

where $\mathbf{v}$ and $\mathbf{w}$ are the 3-vectors of linear and angular velocities respectively. Despite the fact that this representation is minimal in the sense that a 6-vector is used to describe a transformation with 6 degrees of freedom, it is not adapted for warping pixels from one image to another. In such a case, this representation is mapped to SE(3) through the exponential map described in section 3.4 :

$$\mathbf{T} = \Gamma(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}}), \tag{6.3}$$

where $\hat{\boldsymbol{\xi}}$ is the skew symmetric matrix of $\boldsymbol{\xi}$. The warping function can then be expressed using the following equation :

$$\mathbf{x}' = \boldsymbol{\mathcal{W}}(\mathbf{x}, \boldsymbol{\xi}) = \boldsymbol{\Pi}\left(\Gamma(\boldsymbol{\xi}) \,.\, \boldsymbol{\Pi}^{-1}(\mathbf{x}, \boldsymbol{D}_1(\mathbf{x}))\right), \tag{6.4}$$

where $\mathbf{x}'$ is the pixel coordinates in $\boldsymbol{I}_0$ corresponding to $\mathbf{x}$ (the pixel coordinates in $\boldsymbol{I}_1$) and $\boldsymbol{\Pi}(.)$ is the projection function described in equation (4.8). The inverse projection function $\boldsymbol{\Pi}^{-1}(.)$ permits to reconstruct a 3D point $\mathbf{p}$ from the pixel coordinates $\mathbf{x}$ and the corresponding depth $z = \boldsymbol{D}_1(\mathbf{x})$ as follows :

$$\mathbf{p} = \boldsymbol{\Pi}^{-1}(\mathbf{x}, z) = \begin{pmatrix} z\frac{u-c_u}{f_u} \\ z\frac{v-c_v}{f_v} \\ z \\ 1 \end{pmatrix}, \tag{6.5}$$

The update equation permitting to iteratively optimize the transformation matrix $\mathbf{T}$ according to a Gauss-Newton scheme can then be expressed in the following manner :

$$\mathbf{T} \leftarrow \Gamma(\alpha\, \boldsymbol{\xi}) \,.\, \mathbf{T}, \tag{6.6}$$

where α is damping parameter, the increment $\Delta\boldsymbol{\xi}$ being computed according to the following equation :

$$\Delta\boldsymbol{\xi} = -\mathbf{H}^{-1} \sum_i \mathbf{J}_i^T \mathbf{r}_i. \tag{6.7}$$

Noting that $\mathbf{H} = \sum_i \mathbf{J}_i^T \mathbf{J}_i$ is the (Gauss-Newton approximation to the) *Hessian* matrix. Furthermore, the expression of the ith *Jacobian* row is :

$$\mathbf{J}_i = \frac{\partial \boldsymbol{I}_1}{\partial \boldsymbol{\xi}} = \left[\nabla_{u,v} \boldsymbol{I}_1 \, \frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}}\right]. \tag{6.8}$$

Here $\nabla_{u,v}\boldsymbol{I}_1$ represents the gradient of $\boldsymbol{I}_1$ according to u and v (equations (5.9) and (5.10)). The *Jacobian* $\frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}}$, for its part, is expressed as follows (Kerl, 2012) :

$$\frac{\partial \boldsymbol{\mathcal{W}}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} f_x \frac{1}{z'} & 0 & -f_x \frac{x'}{z'^2} & -f_x \frac{x'y'}{z'^2} & f_x(1+\frac{x'^2}{z'^2}) & -f_x \frac{y'}{z'} \\ 0 & f_y \frac{1}{z'} & -f_y \frac{y'}{z'^2} & -f_y(1+\frac{y'^2}{z'^2}) & f_y \frac{x'y'}{z'^2} & f_y \frac{x'}{z'} \end{pmatrix}. \tag{6.9}$$

The image alignment consists then in iteratively computing the increment $\Delta\boldsymbol{\xi}$ (equation (6.7)) and updating the parameter vector $\boldsymbol{\xi}$ (equation (6.6)) until convergence. Each time the algorithm converges, the resulting estimate is propagated to the next level of the multi-resolution image pyramid from the coarsest to the finest level.

6.1.2 OP-B image alignment

This proposed approach represents the images $\{ \boldsymbol{I}_0, \boldsymbol{I}_1 \}$ in the scale-space domain, as described in equations (5.2) and (5.3). The resulting residual can then be expressed as :

$$G_{\mathbf{r}_i}(\boldsymbol{\xi}) = G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}, \boldsymbol{\xi}); \lambda) - G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref}). \tag{6.10}$$

After that, by stacking the twist vector and the scale parameter $\tilde{\boldsymbol{\xi}} = (\boldsymbol{\xi}^T, \lambda)^T$, and after the linearization of equation (6.10), we obtain the following linearized cost function :

$$\tilde{\boldsymbol{\xi}}^* = \underset{\tilde{\boldsymbol{\xi}}}{\operatorname{argmin}} \sum_i \left[G_{\mathbf{r}_i}(0) + \tilde{\mathbf{J}}_i \, \Delta\tilde{\boldsymbol{\xi}} \right]^2, \tag{6.11}$$

where $G_{\mathbf{r}_i}(0) = G_{\boldsymbol{I}_1}(\boldsymbol{\mathcal{W}}(\mathbf{x}, 0); \lambda) - G_{\boldsymbol{I}_0}(\mathbf{x}; \lambda_{ref})$ and the augmented *Jacobian* row corresponding to the ith pixel is computed as follows :

$$\tilde{\mathbf{J}}_i = \left[\frac{\partial G_{\boldsymbol{I}_1}}{\partial \boldsymbol{\xi}}, \frac{\partial G_{\boldsymbol{I}_1}}{\partial \lambda} \right]. \tag{6.12}$$

where $\frac{\partial G_{\boldsymbol{I}_1}}{\partial \lambda}$ is computed using finite differences, as in equation (5.11). This results in the following increment expression :

$$\Delta\tilde{\boldsymbol{\xi}} = \begin{pmatrix} \Delta\boldsymbol{\xi} \\ \Delta\lambda \end{pmatrix} = -\tilde{\mathbf{H}}^{-1} \sum_i \tilde{\mathbf{J}}_i^T G_{\mathbf{r}_i}, \tag{6.13}$$

where $\tilde{\mathbf{H}} = \sum_i \tilde{\mathbf{J}}_i^T \tilde{\mathbf{J}}_i$. This results in the following update rule for the scale parameter :

$$\lambda \leftarrow \lambda + \Delta\lambda. \tag{6.14}$$

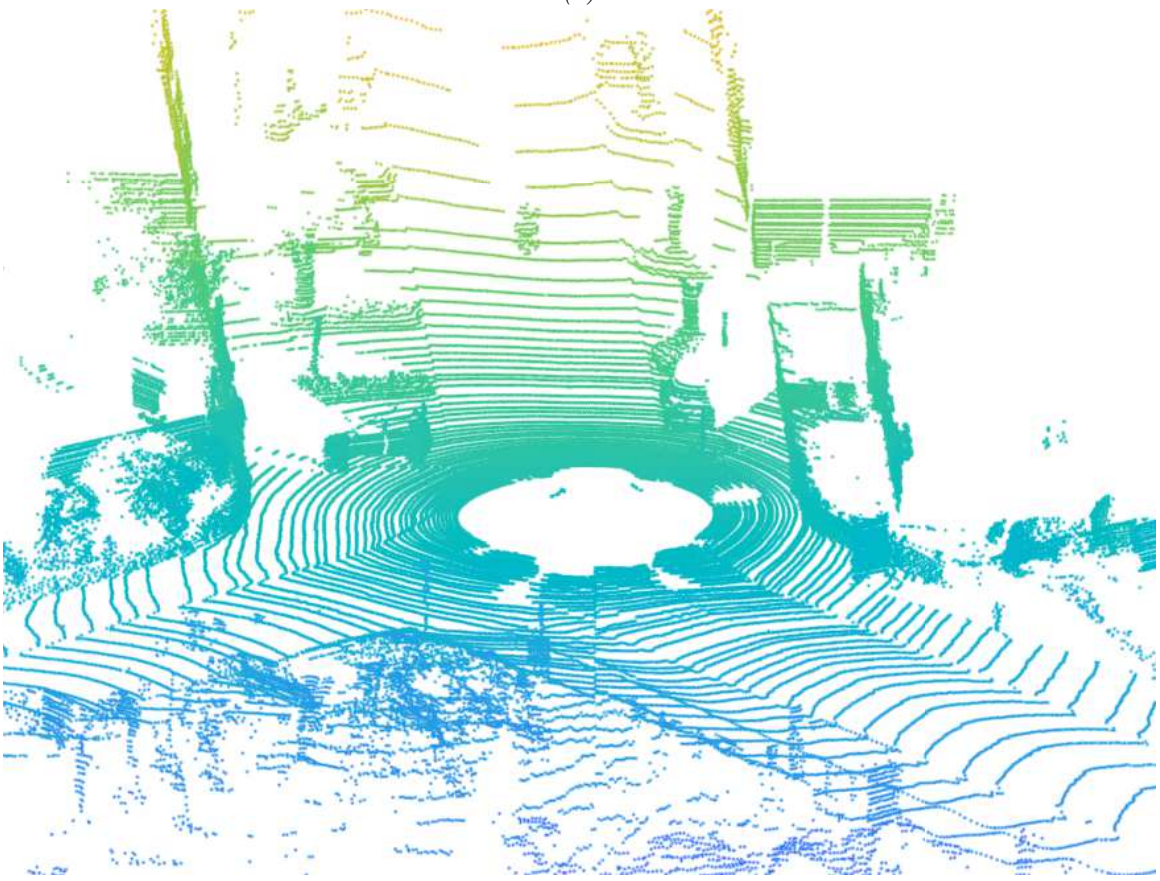
The transformation matrix is updated according to equation (6.6). This iterative computation of the increment (equation (6.13)) followed by the parameters update (equations (6.6) and (6.14)) is repeated within each level of the image pyramid until the convergence of the algorithm. The resulting estimate is then propagated to the next finer level from the coarsest to the finest pyramid level.

6.2 SLAM Based on Vision and Depth Information

This section describes how the proposed OP-B image alignment algorithm is integrated into a SLAM system, which permits the fusion of the information provided by a Laser Range Finder (LRF) and a camera. Figure 6.2 shows example data acquired from the LRF and the camera. This SLAM algorithm consists of the front-end part, which permits to estimate the relative camera motion from $t-1$ to t each time a new frame is available. The second part, the back-end, ensures the global consistency of the estimate by building and optimizing a pose graph of key-frames. The key-frames mentioned here are the set of frames that begins with the first acquired frame and exhibit a displacement greater than a predefined threshold between each other. A frame is consequently considered as key-frame if its displacement is greater than the threshold. The nodes of the pose graph are therefore the set of key-frames absolute poses, while its edges are constraints between the nodes represented in the form of relative transformations. Furthermore, additional constraints can be added to the pose graph when the agent goes through previously visited places (loop closure detection), which permits to reduce the drift of the estimate and to ensure its global consistency (Grisetti et al., 2010).

A typical algorithm workflow can be summarized in the following manner. The pose graph is first initialized using the first frame, which is considered as the world frame $\{\mathcal{W}\}$ origin. After that, as the agent acquires images and corresponding LRF scans, the algorithm estimates the relative motion between pairs of consecutive frames through the front-end tracker ; the resulting estimate permits to check whether the last frame is a new key-frame. At this point, when a new key-frame is detected, the algorithm integrates it as an additional node into the pose graph. The relative transformation between the last and the new key-frame is, for its part, added as a constraint. Furthermore, the added key-frame is used for loop closure detection. This workflow and the main elements involved in the proposed SLAM algorithm are shown in figure 6.3.

(a)



(b)

FIGURE *6.2: Example of the data used by the SLAM algorithm for the estimation process. (a) is a gray-scale image acquired from the camera and (b) is a 3D scan acquired from the LRF.*

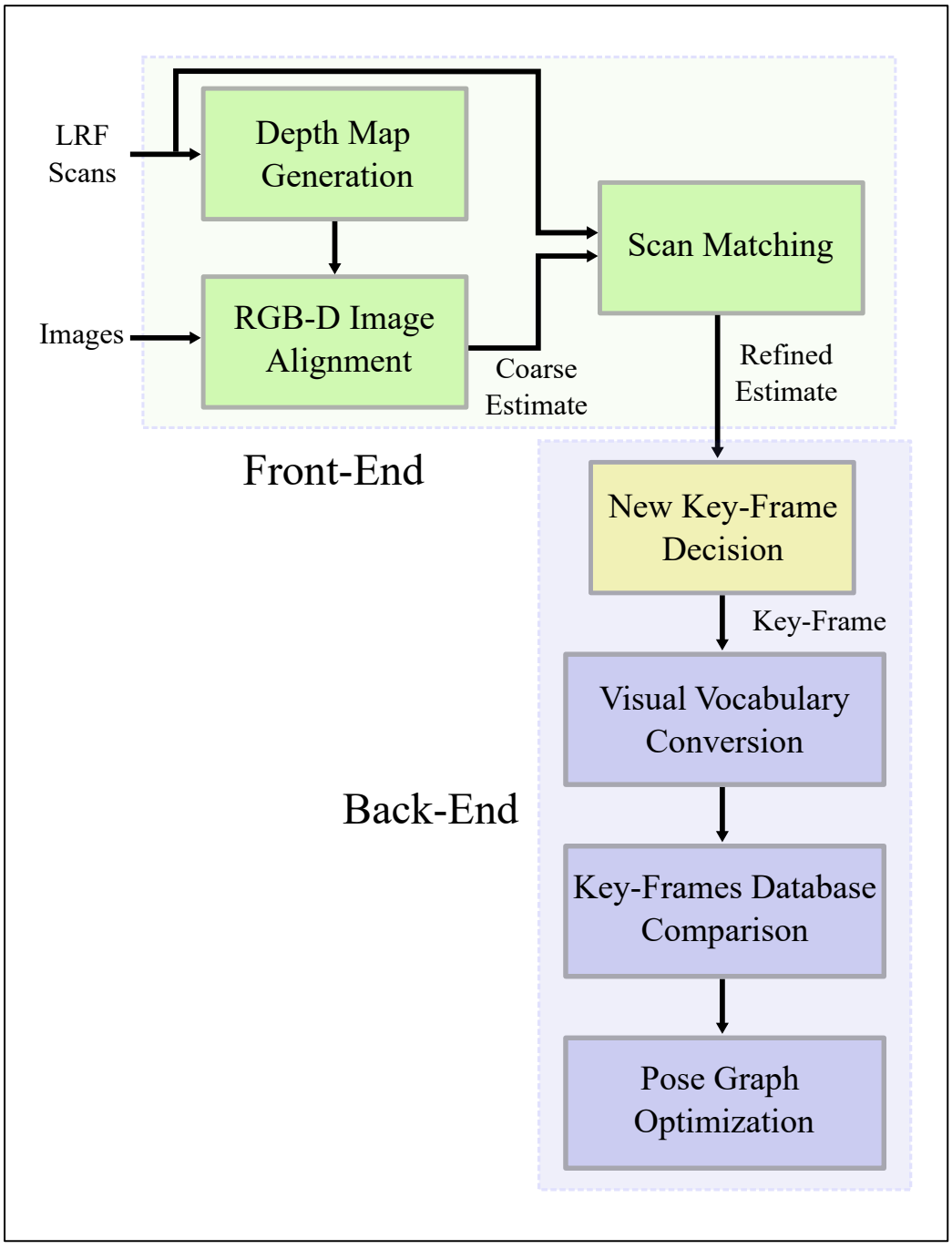


FIGURE *6.3: Overview of the proposed SLAM system, showing all the steps involved in the estimation process.*

6.2.1 Depth Map from Laser Scans

The first module to be described consists in a depth map generator based on the point cloud obtained from the LRF. The generation of the depth map is done through the transformation of the point cloud from the LRF reference frame $\{\mathcal{L}\}$ to the camera reference frame $\{\mathcal{C}\}$, followed by its projection into the image plane, as in equation (6.15).

$$\mathbf{x} = \mathbf{\Pi}(\mathbf{T}^{\mathcal{CL}}.\,\mathbf{p}), \tag{6.15}$$

where $\mathbf{x}$ represents the coordinates of the 3D point $\mathbf{p}$ in the image plane of the depth map. $\mathbf{T}^{\mathcal{CL}}$ is the transformation matrix that transforms the 3D point from $\{\mathcal{L}\}$ to $\{\mathcal{C}\}$. After that, the depth map is computed by first rounding the pixel coordinates of the projected points, and defining the depth of these pixels as the z component of the points coordinates expressed in $\{\mathcal{C}\}$. The depth of the pixels for which no point was projected is set to undefined.

6.2.2 Tracking

This module estimates the relative transformation between two consecutive frames through two stages. First, an RBG-D image alignment algorithm permits to compute a coarse estimate of the relative transformation using the image pair and the computed depth map. This coarse estimate is then refined using the Iteratively Re-weighted Least Squares Iterative Closest Point (IRLS-ICP) algorithm (Bergström and Edlund, 2014) in the second stage of this module. This approach differs from other SLAM algorithms, as (Zhang and Singh, 2015), since it considers a direct approach to estimate the relative camera motion.

RGB-D Image Alignment

The RGB-D image alignment algorithm used to compute the first estimate of the relative motion between the last two consecutive frames is the OP-B algorithm – described in section 6.1.2. This algorithm uses the triplet $\{\boldsymbol{I}_{t-1}, \boldsymbol{I}_t, \boldsymbol{D}_t\}$ for the estimation process. Besides, the estimate is computed using only the two coarsest pyramid levels ; i.e., the considered image resolution ranges from 76×22 to 152×44. Considering only the two coarsest pyramid levels permits indeed to save time for the second stage of the tracker.

Scan Matching

The computed coarse estimate (on the two coarsest pyramid levels) is refined in this stage using the Iteratively Re-weighted Least Squares Iterative Closest Point (IRLS-ICP) algorithm (Bergström and Edlund, 2014) – summarized in algorithm (6.1). This algorithm based on the Huber estimator uses the last two LRF scans to minimize the following cost function :

$$\underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} E(\mathbf{R},\mathbf{t}) = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} w_i ||\mathbf{r}_i||_2^2, \tag{6.16}$$

where $\mathbf{r}_i = (\mathbf{R}\,\mathbf{s}_t^i + \mathbf{t}) - \mathbf{s}_{t-1}^i$ and N is the number of considered 3D points. $\mathbf{s}_{t-1}^i$ and $\mathbf{s}_t^i$ are respectively the ith points of the previous and current LRF scans. Furthermore, the Huber weight w_i is computed as follows :

$$w_i = \begin{cases} 1 & if|\mathbf{r}_i| < \kappa_H \\ \frac{\kappa_H}{|\mathbf{r}_i|} & if|\mathbf{r}_i| < \kappa_H \end{cases} \tag{6.17}$$

where κ_H is the Huber design parameter. Noting that the ground points in the LRF scans are fitted using the algorithm of (Torr and Zisserman, 2000) to avoid considering them for the optimization. This permits to increase the estimation precision as the displacement of camera is in the 2D space.

Algorithm 6.1: The IRLS-ICP algorithm.

1: **for** $i = 0$ to maxIterations **do**
2: **for** all $\mathbf{s}_t^i \in \mathbf{S}_t$ **do**
3: Find the closest point to $\mathbf{s}_t^i$ in the point cloud $\mathbf{S}_{t-1}$.
4: **end for**
5: Compute $\mathbf{R}_i^*$ and $\mathbf{t}_i^*$ that minimize equation (6.16) using the found closest point.
6: Apply $\mathbf{R}_i^*$ and $\mathbf{t}_i^*$ to $\mathbf{S}_t$.
7: Compute $||E_{i-1}(\mathbf{R}_i^*, \mathbf{t}_i^*) - E_i(\mathbf{R}_i^*, \mathbf{t}_i^*)||$. If this difference is smaller than a threshold ε, break.
8: **end for**

6.2.3 Loop Closure Detection

Every time a new key-frame is detected, it is used to check for a loop closure. This is done by first converting the new key-frame into a set of words according to a pre-computed bag of visual words based on SURF features (Bay et al., 2006). The

resulting visual word set is then compared to the image database of previous key-frames (Philbin et al., 2007). When correspondences are found, the loop detection module selects the valid frame from the three best candidates, and estimates the relative transformation between the corresponding frames.

First, an estimate of the relative transformation between the current and each candidate frame is computed using the feature-based approach of (Hartley and Zisserman, 2004). This step permits to select a valid frame – the one that has the most inliers above a minimal number of 20 features. The estimate computed in the first step is then refined using the OP-B followed by the IRLS-ICP as done in tracker. Besides, the new key-frame is added to the image database for future loop detection.

6.2.4 Pose Graph Optimization

The pose graph used in this SLAM system is a graph whose nodes are the absolute poses of the set of key-frames, while its edges are constraints represented in the form of relative transformations between pairs of nodes (Grisetti et al., 2010). These constraints, in addition, are modeled as Gaussian distributions computed from both the estimates of the font-end tracker and the loop closure detection module. Considering $\mathbf{z}_{ij}$ and $\mathbf{\Omega}_{ij}$, that represent respectively the mean and information matrix (inverse of the covariance matrix) of the constraint between nodes $(\mathbf{x}_i, \mathbf{x}_j)$, permits to define the following error function :

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \bar{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j), \tag{6.18}$$

where $\bar{\mathbf{z}}_{ij}$ is the prediction of the relative pose between the nodes $(\mathbf{x}_i, \mathbf{x}_j)$ – computed from the absolute pose values. It is worth noting that the constraint is considered as a virtual measurement of the relative pose between the nodes $(\mathbf{x}_i, \mathbf{x}_j)$.

Let ζ be the set of pairs of node indices for which a constraint exists. The objective is then to find the nodes configuration that best describes the virtual measurements in the least squares sense. This can be done through the minimization of the following cost function :

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{<i,j>\in\zeta} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}. \tag{6.19}$$

Such an optimization problem can be solved using the Gauss-Newton or Levenberg–Marquardt algorithms described in section 3.3. More details concerning the

implementation of the pose graph optimization can be found in (Grisetti et al., 2010) and (Kümmerle et al., 2011).

6.3 Experiments

This section shows the results of the validation of the different algorithms presented in this chapter. Two datasets are therefore considered for this purpose. The TUM dataset (Sturm et al., 2012) for validating the RGB-D image alignment approaches, and the Kitti dataset (Geiger et al., 2012) for validating the proposed SLAM algorithm.

6.3.1 RGB-D image Alignment

We first present qualitative results about the proposed method. We show and discuss the effect of the scale parameter on the shape of the cost function, then we present the evolution of λ during the optimization process. After that, we present quantitative results about the comparison between the proposed OP-B and the PP-B methods through the experiments that were conducted. The maximum number of iterations considered for both the PP-B and OP-B methods is set to 40 iterations. Such a high value was chosen to avoid PP-B misses reaching its minimum. Furthermore, as our method aims at increasing the robustness towards large camera motion, we considered an additional pyramid level for the PP-B (five levels) compared to the OP-B (four levels) for fairer comparison.

Cost Function Qualitative Analysis

Figure 6.4 presents a 3D representation of the cost function for each geometric Degree Of Freedom (DoF) combined with the scale-space parameter for the coarsest level of the multi-resolution pyramid. The values of the scale parameter range from 0.1 to 10, and the values of each geometric parameter range from -1.0 to 1.0, while the others are fixed to their reference value. λ_{ref} is set to 1. Figures 6.5 and 6.6 show a section view for two values of the scale parameter $\lambda = \{0.1, 5\}$. Figure 6.5 is equivalent to the usual fixed scale within a pyramid level.

Figures 6.4, 6.5, and 6.6 show that the scale parameter has a smoothing effect on the cost function, i.e. for large values of this parameter the local minima of the cost function are suppressed, inducing an enlargement of the basin of convergence around the global minimum (figure 6.6). On the other hand, when the value of the scale parameter is small, which is equivalent to the purely photometric case, the global

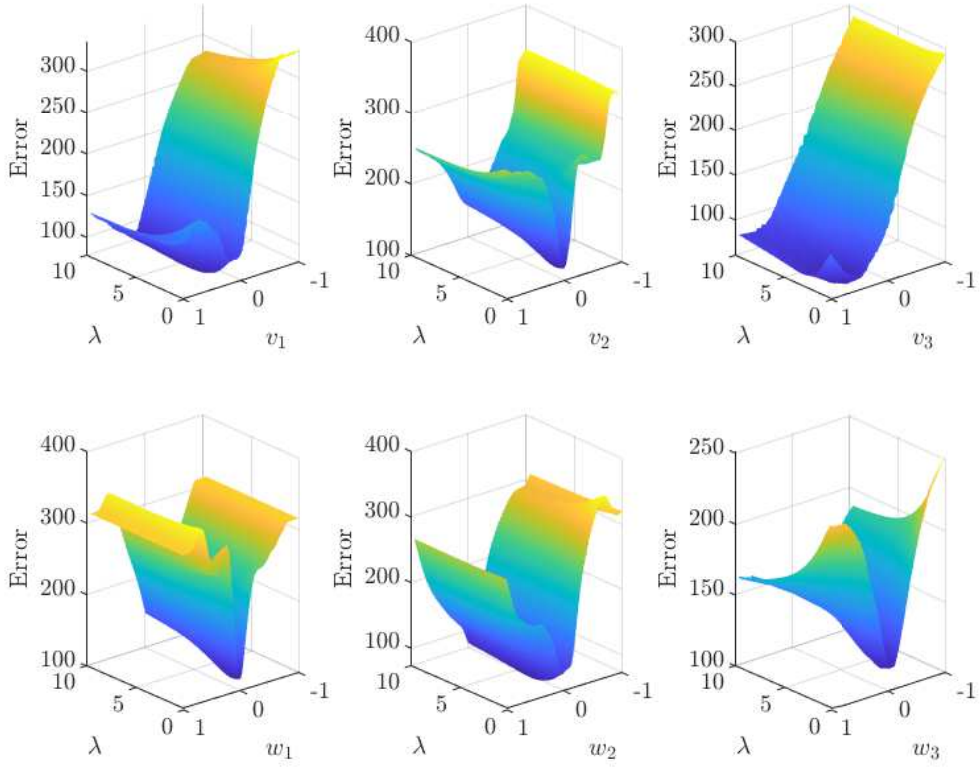


FIGURE *6.4: Visualisation of the cost function for the geometric DoF combined with the scale-space DoF.*

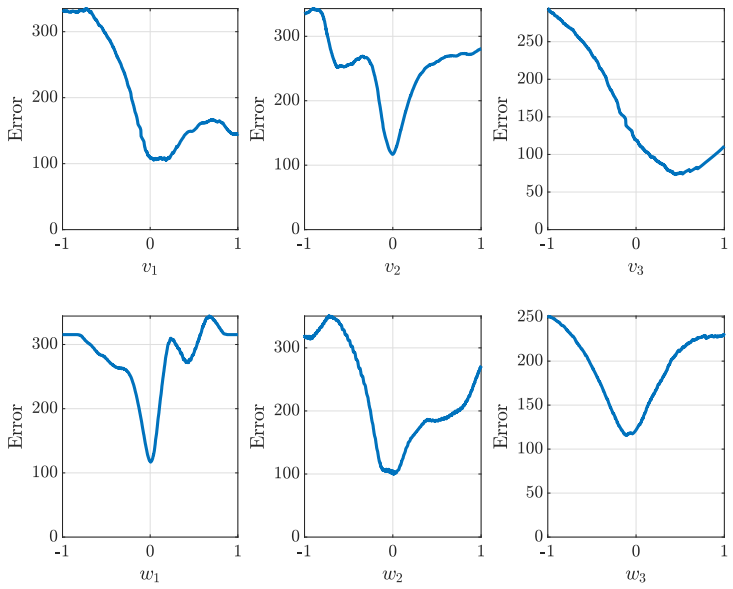


FIGURE *6.5: Section view of the 2D cost function per geometric degree of freedom for* $\lambda = 0.1$ *and* $\lambda_{ref} = 1$*, similar to the photometric case.*

---

minimum is sharp but the local minima are not suppressed (figure 6.5). It is worth noting that the shape of the cost function in the case of figure 6.5 corresponds to the coarsest level of the PP-B that is not enough to suppress local minima.

During optimization, figure 6.7 shows that the scale parameter (λ) first increases in the pyramid level 3. This shows that the proposed method can increase the value of the scale parameter if its initial value is too low. After that, λ continuously tends towards the reference (λ_{ref}) in each pyramid level resulting in a coarse to fine approach. An interesting aspect of the proposed method is that λ_{ref} acts as an implicit objective value for λ that permits the automatic adaptation of the scale within each pyramid level, being accurate at convergence.

Quantitative Results

We present here a quantitative comparison between the OP-B and PP-B methods. We consider for the evaluation the fr1/room, fr1/desk, fr2/desk, and fr1/floor sequences (Sturm et al., 2012). The fr1/room, fr1/desk, and fr2/desk were recorded in an office environment and contain translational and rotational motion with varying speeds. The fr1/floor is a sequence that containing images of a sweep over a wooden floor.

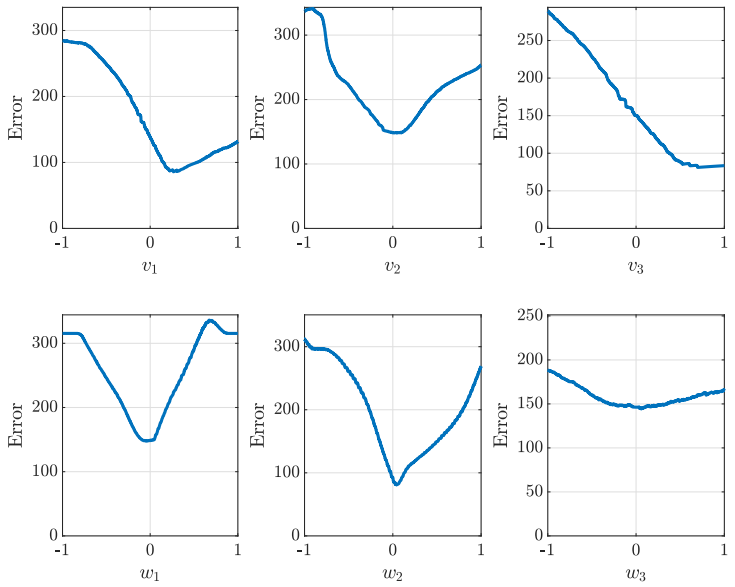


FIGURE *6.6: Section view of the 2D cost function per geometric degree of freedom for* $\lambda = 5$ *and* $\lambda_{ref} = 1$*, the local minima are suppressed and the function is smoothed.*

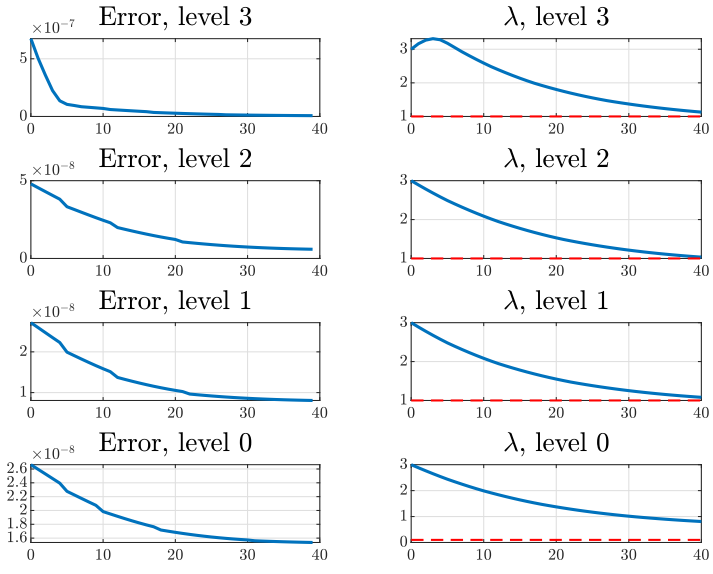


FIGURE *6.7: Evolution of the normalized photometric error during the optimization (for each pyramid level) and the corresponding scale parameter. The blue line represents the value of* λ *at each iteration (x-axis) and the red dashed line represents* λ_{ref}*.*

TABLE *6.1: Statistics of the RPE of the proposed OP-B and PP-B methods on the fr1/room sequence.*

	rms (m/s)	mean (m/s)	median (m/s)
OP-B	**0.4348**	**0.3767**	0.3502
PP-B	0.4351	0.3768	**0.3495**

TABLE *6.2: Statistics of the RPE of the proposed OP-B and PP-B methods on the fr1/desk sequence.*

	rms (m/s)	mean (m/s)	median (m/s)
OP-B	**0.553**	**0.4987**	0.5139
PP-B	0.555	0.4991	**0.5136**

It is worth noting that the mean execution time of the proposed method is 1.2 times the mean execution time of the PP-B method.

Validation in Office Environment

First, the RPE (relative pose error) per second was computed using the the ground truth data and the validation tools proposed by Sturm et al. (2012) to evaluate the drift of the algorithms, as in Kerl et al. (2013). The statistics of the RPE per second for the fr1/room, fr1/desk, and the fr2/desk sequences are presented in Tables 6.1, 6.2, and 6.3 respectively.

We can see that both methods have similar error statistics in terms of rms, mean, and median for the three sequences. This is consistent, since the proposed approach provides enlarged basin of convergence but does not aim at increasing the estimation precision in the nominal case of the PP-B.

TABLE *6.3: Statistics of the RPE of the proposed OP-B and PP-B methods on the fr2/desk sequence.*

	rms (m/s)	mean (m/s)	median (m/s)
OP-B	**0.270**	**0.233**	**0.217**
PP-B	0.272	0.234	0.218

In order to simulate larger displacements and study the behavior of each algorithm in such conditions, we followed the methodology used in Steinbruecker et al. (2011). Hence, we estimate the motion between pairs of images $\boldsymbol{I}(n)$ and $\boldsymbol{I}(n+k)$ for $k = 1, 2, 3, 4$ (image steps). Example images are shown in figure 6.8. The evolution of the mean and the rms errors of the RPE w.r.t. image steps for the fr2/desk, fr1/desk, and fr1/room sequences is shown in tables 6.4, 6.5, and 6.6 respectively.

TABLE *6.4: The rms and mean values of the RPE (m/s) for the proposed OP-B and the PP-B methods in the fr2/desk sequence.*

		Every image	1 out of 2	1 out of 3	1 out of 4
rms	OP-B	**0.270**	**0.274**	**0.276**	**0.270**
	PP-B	0.272	0.277	0.279	0.273
mean	OP-B	**0.233**	**0.235**	**0.237**	**0.230**
	PP-B	0.234	0.237	0.238	0.232

TABLE *6.5: The rms and mean values of the RPE (m/s) for the proposed OP-B and the PP-B methods in the fr1/desk sequence.*

		Every image	1 out of 2	1 out of 3	1 out of 4
rms	OP-B	**0.553**	**0.554**	**0.555**	**0.544**
	PP-B	0.555	0.557	0.576	0.608
mean	OP-B	**0.4987**	**0.500**	**0.501**	**0.489**
	PP-B	0.4991	0.502	0.515	0.537

TABLE *6.6: The rms and mean values of the RPE (m/s) for the proposed OP-B and the PP-B methods in the fr1/room sequence.*

		Every image	1 out of 2	1 out of 3	1 out of 4
rms	OP-B	**0.4348**	**0.4342**	**0.429**	**0.483**
	PP-B	0.4351	0.4344	0.443	0.523
mean	OP-B	**0.3767**	**0.37627**	**0.372**	**0.394**
	PP-B	0.3768	0.37629	0.385	0.423

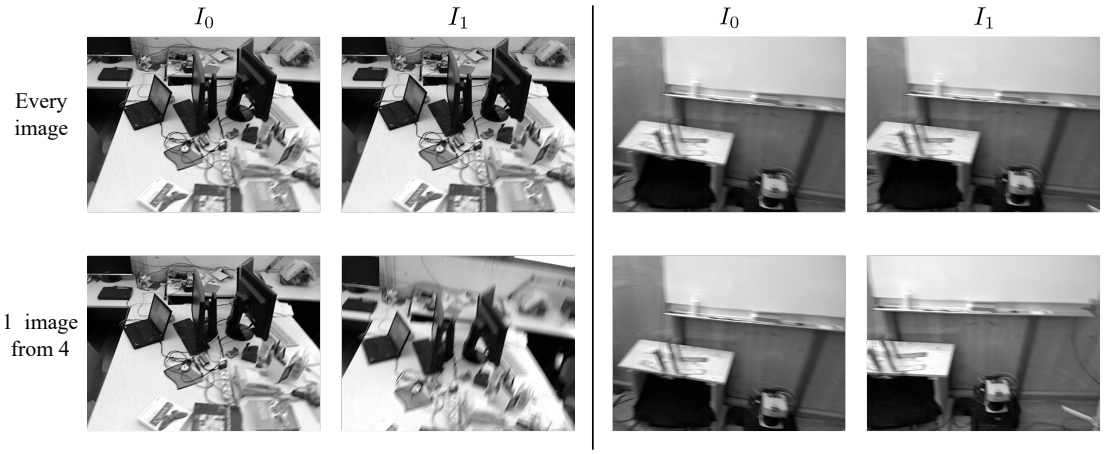


FIGURE *6.8: Example image pairs for different images steps from the fr1/desk and fr1/room sequences.*

TABLE *6.7: The rms values of the ATE (cm) for the proposed OP-B and PP-B methods in the fr1/room sequence.*

	Every image	1 out of 2	1 out of 3	1 out of 4
OP-B	36.5	37.3	**37.8**	**66.1**
PP-B	**36.1**	**36.9**	54.9	90.4

As can be seen in table 6.4 the difference between the methods is not significant for the fr2/desk. This is due to the fact that the camera movement in this sequence is slow with a mean translational velocity of 0.2 m/s, especially in comparison with the fr1/room and fr1/desk sequences with a mean translational velocity of 0.4 m/s and 0.3 m/s respectively. On both latter sequences, methods show similar results in terms of accuracy when the image steps are small ($k \leq 2$). However, when the displacement is more important ($k > 2$), our method have less drift than the PP-B (tables 6.5 and 6.6). Indeed, the rms of the RPE for the proposed method is inferior by around 4 cm and 6 cm for the fr1/room and fr1/desk respectively.

In order to evaluate the consistency of the estimated trajectories w.r.t. the ground-truth data, we computed the Absolute Trajectory Error (ATE) shown in tables 6.7, 6.8, and 6.9 in addition to figures 6.12, 6.13, and 6.14. Furthermore, the estimated trajectories along with the ground-truth trajectory are presented in figures 6.12, 6.13, and 6.14.

We can see from these results that the proposed method is able to maintain a better level of consistency (ATE) compared to the photometric-based method for

TABLE *6.8: The rms values of the ATE (cm) for the proposed OP-B and PP-B methods in the fr1/desk sequence.*

	Every image	1 out of 2	1 out of 3	1 out of 4
OP-B	10.4	10.9	**11.1**	**13.0**
PP-B	**10.0**	**10.1**	35.8	56.3

TABLE *6.9: The rms values of the ATE (cm) for the proposed OP-B and PP-B methods in the fr2/desk sequence.*

	Every image	1 out of 2	1 out of 3	1 out of 4
OP-B	59.1	58.6	58.3	58.8
PP-B	**58.6**	**58.0**	**57.6**	**57.7**

image steps of 4, which are equivalent to a mean displacement of 4.0 cm and 4.9 cm for the fr/room and fr1/desk sequences respectively. When the image step is small ($k \leq 2$), the difference in the ATE obtained from the estimates of the PP-B and the OP-B is not as significant as for $k > 2$. Similar results can be seen in Table 6.9, as the slow camera motion in the fr2/desk sequence induces similar results even when considering 4 image steps. This slight difference in precision results from the fact that the scale parameter of the input images does not always attain its reference, notably due to the motion blur present in the acquired images. However, when the image step is important ($k > 2$) the error increases more for the PP-B than the OP-B proposed method. The latter shows, thus, more consistency of the estimates w.r.t. the ground-truth trajectory.

FIGURE *6.9: Example images from the sequence fr1/floor.*

TABLE *6.10: The rms and mean values of the RPE (m/s) for the proposed OP-B and PP-B methods in the fr1/floor sequence.*

		Every image	1 out of 2
rms	OP-B	**0.421**	**0.417**
	PP-B	0.436	0.435
mean	OP-B	**0.356**	**0.356**
	PP-B	0.369	0.368

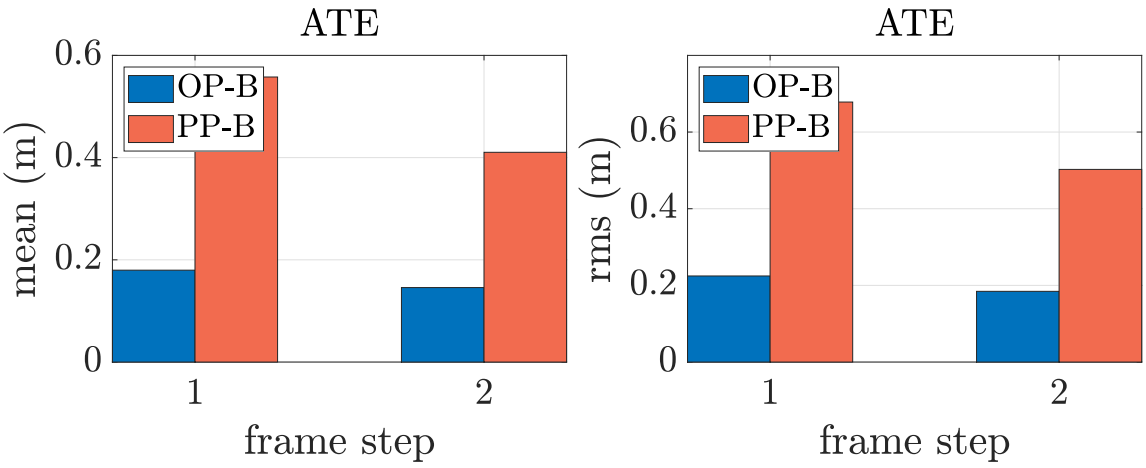


FIGURE *6.10: Evolution of the mean and the rms of the ATE in meters w.r.t. image steps for the fr1/floor sequence.*

Fr1/Floor Sequence

We now consider the fr1/floor sequence, which is a challenging sequence (with rapid camera movement and a lot of reflections on the wooden floor). Example images of this sequence are shown in figure 6.9. Figure 6.10 presents the evolution of the mean and rms values of the ATE w.r.t. image steps. The rms and mean values of the RPE are shown in Table 6.10. Furthermore, the estimated trajectories and their corresponding errors are shown in figure 6.15.

As can be seen in Table 6.10 the proposed method shows more accurate results than the PP-B. Furthermore, The estimated trajectory using the proposed OP-B is more consistent with the ground-truth. This permits to say that the proposed method is more accurate and consistent than the PP-B at frame rate and when considering 1 out of 2 images. When larger image steps are considered, both methods failed because of the large camera displacement in addition to the important illumination changes. Figure 6.11 shows an example, where both algorithms were unable to successfully

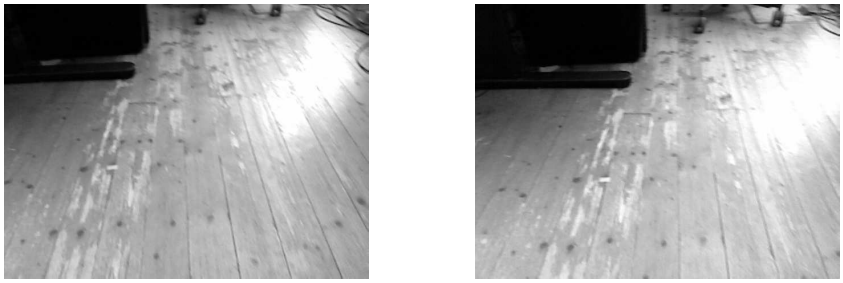


FIGURE *6.11: Example image pair where both algorithms were unable to successfully estimate the camera motion.*

estimate the camera motion.

6.3.2 Integration of the OP-B into a SLAM system

In this part, the integration of the proposed OP-B algorithm into a SLAM system is validated using the Kitti dataset (Geiger et al., 2012), which consists of a set of image pairs acquired from a stereo camera mounted on a car in addition to scans acquired using a 360° laser scanner. Each sequence of this dataset was acquired outdoor through the navigation in a city-like environment. The following parts show the results of the study and experiments on the proposed SLAM system.

Tracking Module

As mentioned, the tracking module considered in the proposed SLAM system follows a two steps scheme : RGB-D image alignment followed by scan matching. An alternative and more compact approach would be however to only use the RGB-D image alignment algorithm (OP-B) and compute the estimate on the full levels of the multi-resolution image pyramid. We therefore consider in this part a comparison between these two approaches using the sequence 07 of the Kitti dataset. Table 6.11 shows the median RPE values for the OP-B (using all pyramid levels) along with the tracker described in section 6.2.2.

TABLE *6.11: The median values of the RPE (m/s) for the two tracking approaches.*

	RGB-D Image Alignment	RGB-D Image Alignment + IRLS-ICP
RPE	0.973	**0.247**

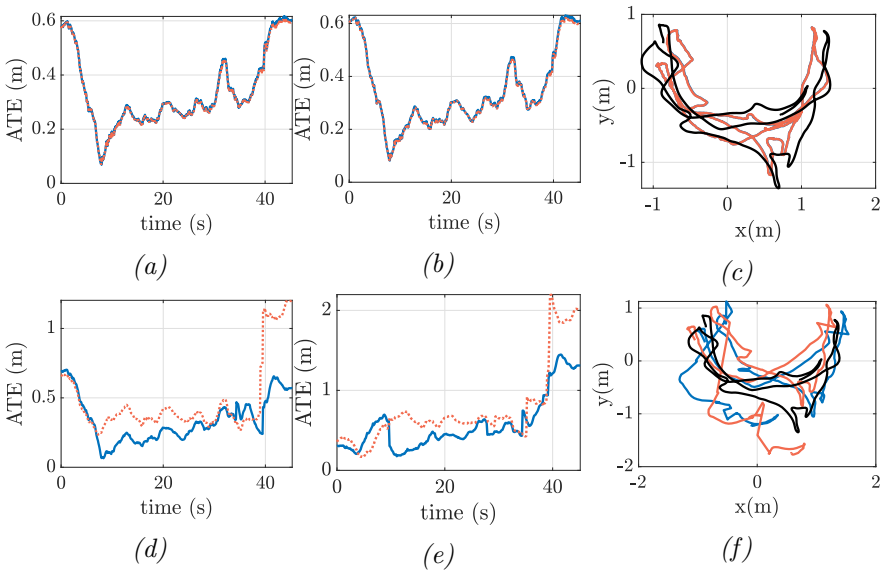


FIGURE *6.12: Visualization of the absolute errors of each trajectory, for the fr1/room sequence, considering the different image steps : 1 (a), 2 (b), 3 (d), and 4 image steps (e). The estimated trajectories along with ground-truth (on the xy plane) for 1 and 4 image steps are shown in (c) and (f) respectively. The PP-B and the new OP-B methods are, respectively, in red and blue. The ground-truth trajectory is in black.*

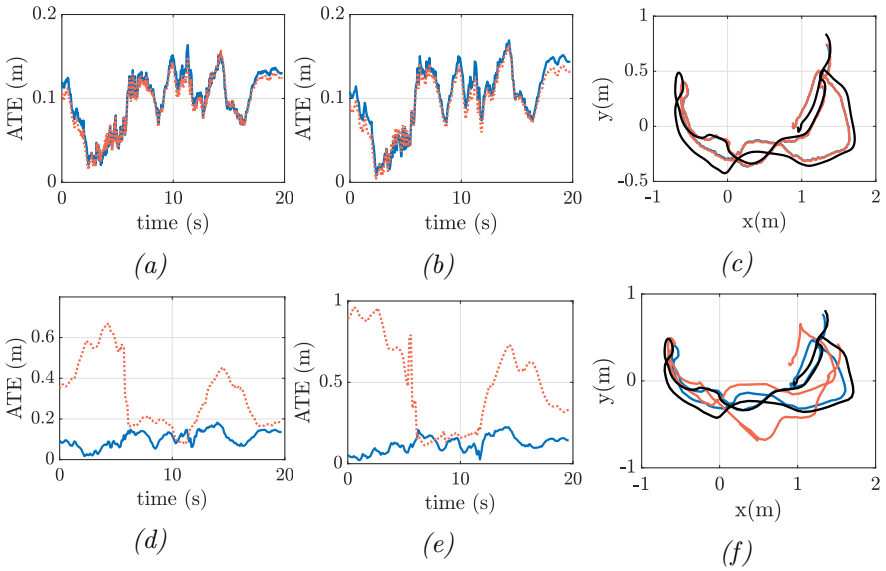


FIGURE *6.13: Visualization of the absolute errors of each trajectory, for the fr1/desk sequence, considering the different image steps : 1 (a), 2 (b), 3 (d), and 4 image steps (e). The estimated trajectories along with ground-truth (on the xy plane) for 1 and 4 image steps are shown in (c) and (f) respectively. The PP-B and the new OP-B methods are, respectively, in red and blue. The ground-truth trajectory is in black.*

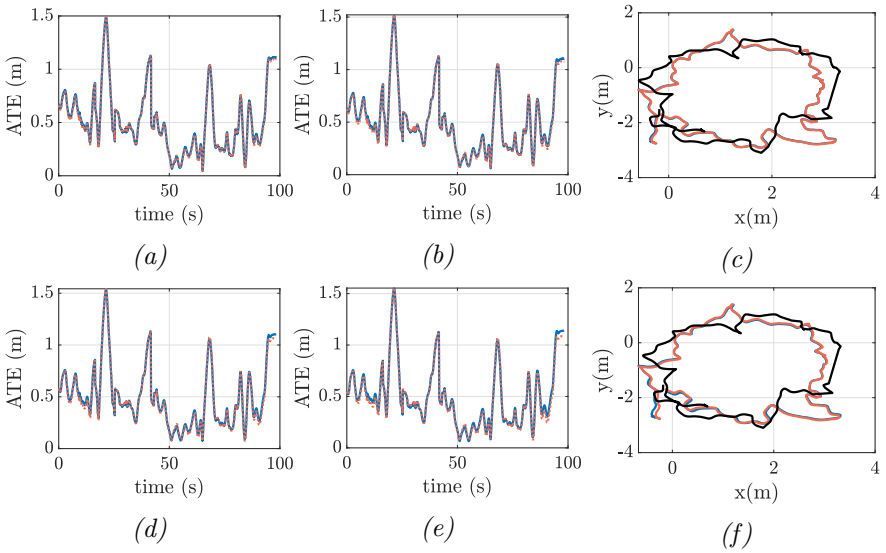


FIGURE *6.14: Visualization of the absolute errors of each trajectory, for the fr2/desk sequence, considering the different image steps : 1 (a), 2 (b), 3 (d), and 4 image steps (e). The estimated trajectories along with ground-truth (on the xy plane) for 1 and 4 image steps are shown in (c) and (f) respectively. The PP-B and the new OP-B methods are, respectively, in red and blue. The ground-truth trajectory is in black.*

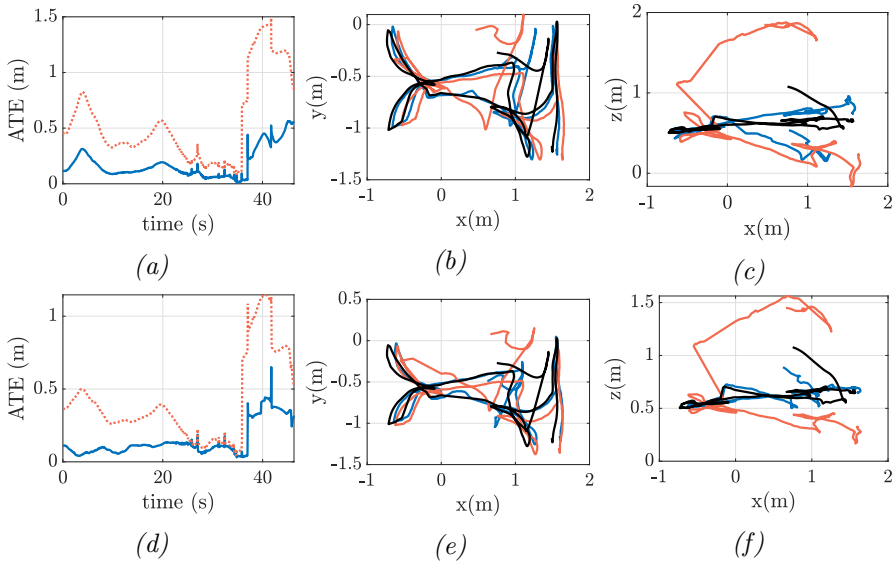


FIGURE *6.15: Visualization of the absolute errors of each trajectory for the fr1/floor sequence considering every image (a) and 1 out 2 images (d). The estimated trajectories along with ground-truth (on the xy and xz planes) for 1 and 2 image steps are shown in (b), (c), (e), and (f) respectively. The PP-B and the new OP-B methods are, respectively, in red and blue. The ground-truth trajectory is in black.*

We can see that the median value of the RPE is smaller for the two steps scheme. The difference between the two approaches is due to the fact that the depth map used by the OP-B is approximative, as the depth assigned to the pixels is based on the rounded values of the projected points coordinates. The error introduced by the depth map computation results therefore in estimation errors and increased RPE. The considered tracker conversely uses the OP-B to reach estimates near the correct values and refines them using the IRLS-ICP, which justifies its consideration for the proposed SLAM system.

Loop Closure Module

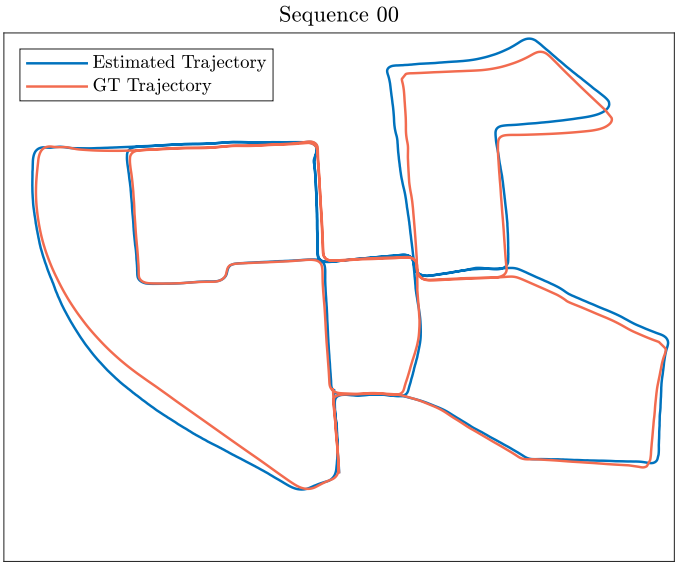
The considered loop closure module uses a feature-based approach to compute a first estimate of the relative pose between two corresponding frames. This first estimate is then refined using the OP-B algorithm followed by the IRLS-ICP. In order to evaluate the benefit of this approach, the trajectory of the platform on the sequence 00 is estimated using the loop closure module with refinement and without it. Figure 6.16 shows the estimated trajectories along with the ground truth trajectory for the two approaches.

As shown in figure 6.16 the trajectory estimated using the loop closure module with refinement provides a more consistent estimate than without it. This justifies our choice of using this approach in the proposed SLAM system.

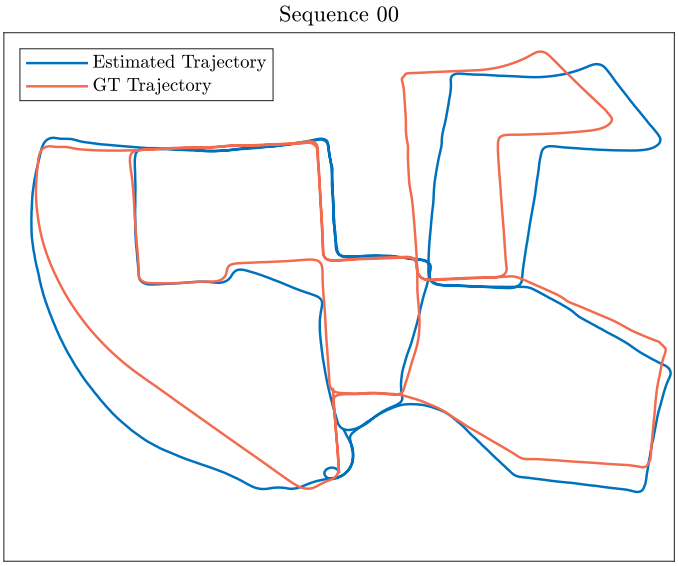
Full SLAM System

Figure 6.17 shows the estimated trajectories and maps using the proposed SLAM system for the sequences 00, 03, 05, 06, and 07.

As can be seen figure 6.17 shows that the proposed SLAM system permits to consistently estimate the trajectory of the camera on all the considered sequences. This system provides dense point clouds (map) describing the explored environment as shown in figure 6.18. These maps permit to distinguish elementary elements of the environment such as trees, roads, and buildings. The quantitative results presented in Table 6.12 (computed using the evo package (Grupp, 2017)) show that the proposed SLAM method permits to reach a small pose error except for sequences 01 and 02. The high error value of the former is related to the fact that the sequence does not contain loop closures, which prevents the algorithm from correcting its estimation drift. Concerning the error related to sequence 02 it is due to the fact that the algorithm was not able to detect loop closures, thus preventing the drift correction as



(a)



(b)

FIGURE *6.16: Estimated and ground-truth trajectories for the sequence 00 of the Kitti dataset. (a) The loop closure module uses the refinement step, while (b) does not.*

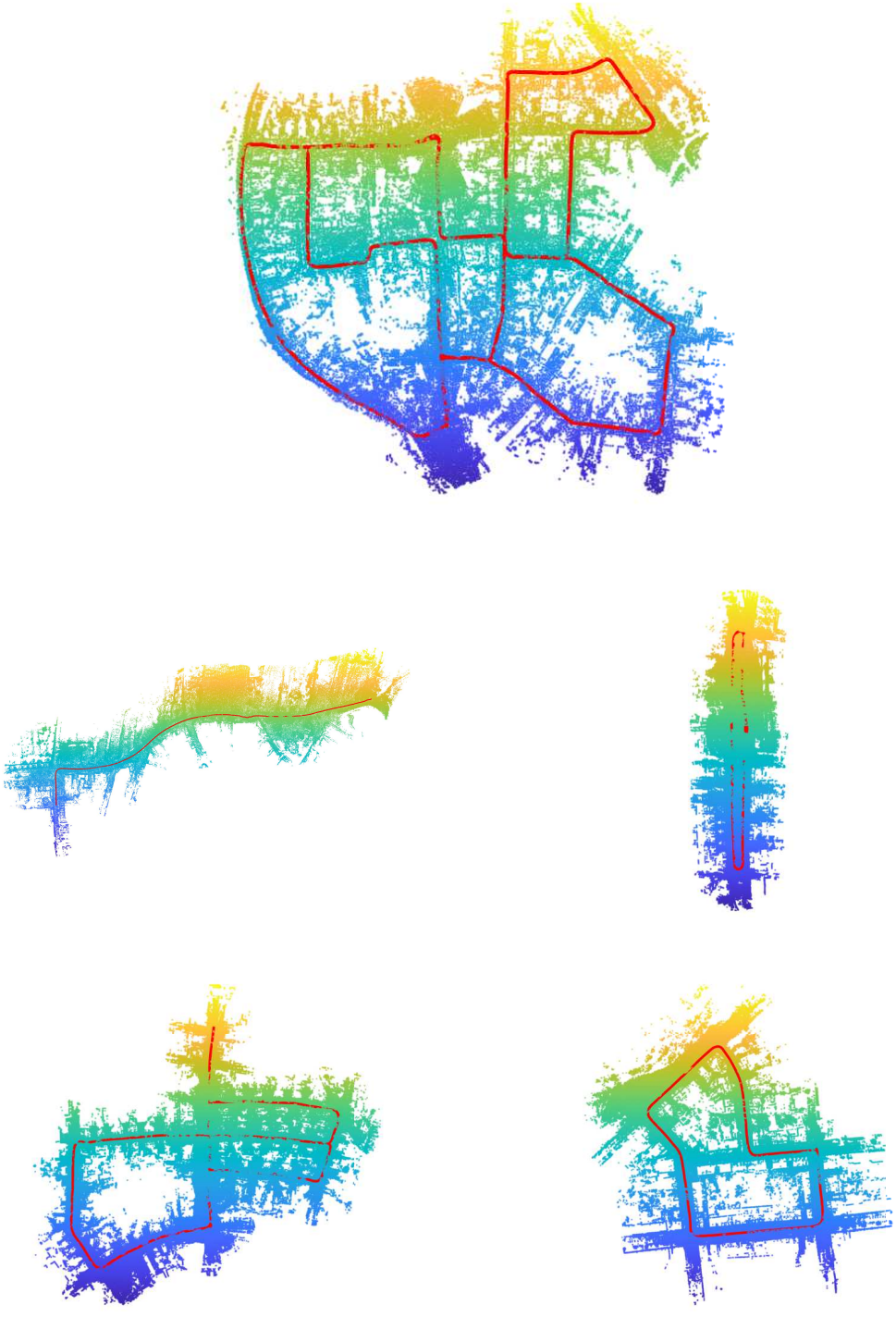


FIGURE *6.17: Estimated trajectory along with the map for the sequences 00, 03, 05, 06, 07 of the Kitti dataset.*

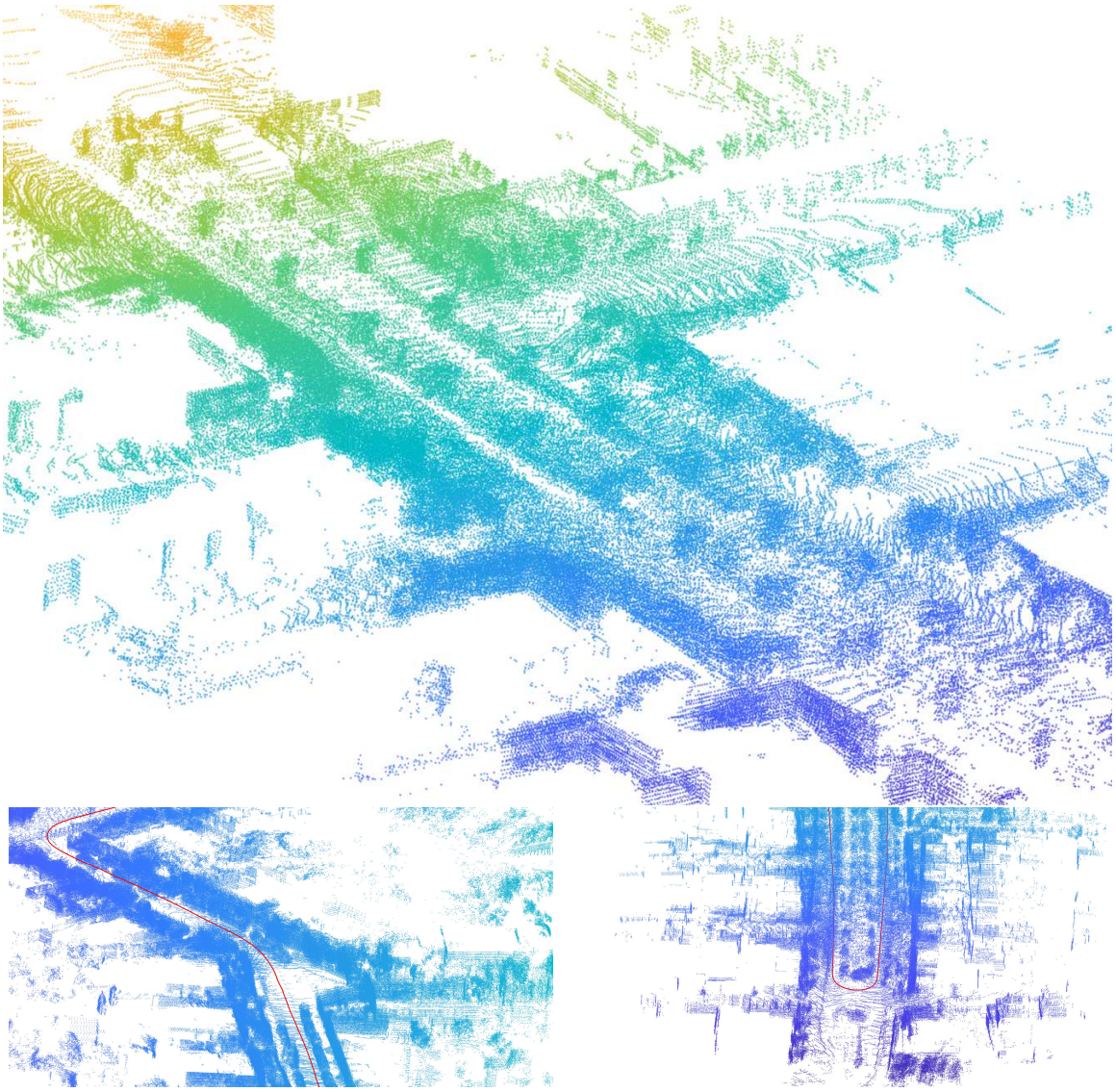


FIGURE *6.18: Examples of the estimated maps using the proposed SLAM algorithm.*

TABLE *6.12: The median value of the absolute pose errors in meters of the proposed SLAM algorithm.*

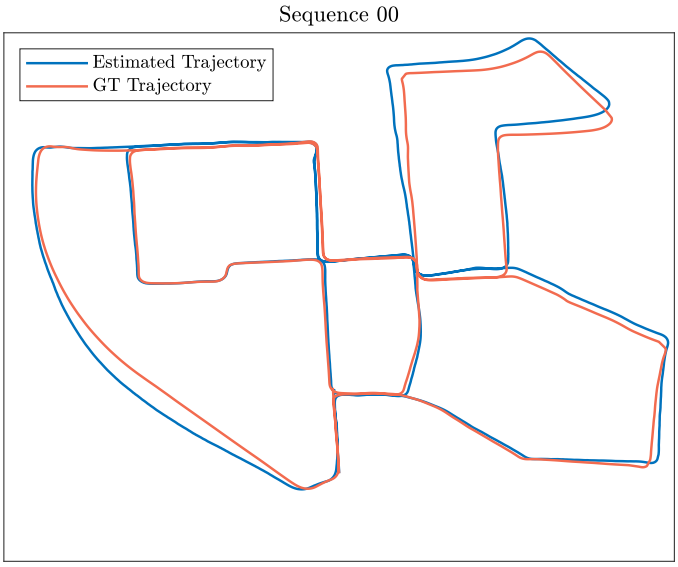
Sequence	Median
00	6.064
01	103.325
02	77.322
03	2.309
04	5.280
05	5.154
06	2.008
07	6.942
08	3.446

shown in figure 6.19. This is notably due to the presence of dense trees in the images which makes it hard to find correct frame correspondences.

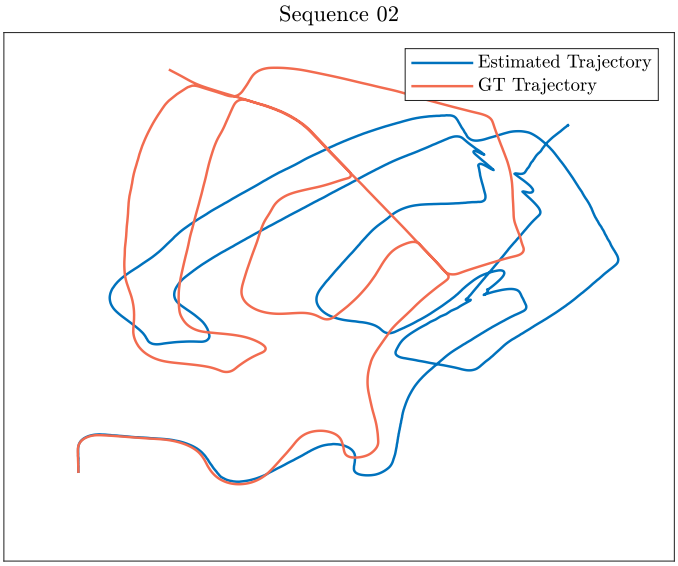
6.4 Conclusion

This chapter presented an image alignment algorithm that permits to control the continuous evolution of the image scale within each level of multi-resolution pyramid. This proposed approach is built upon the scale-space representation of the images that is automatically adapted according to the needs of the algorithm, permitting to handle large camera displacements. Furthermore, the integration of this image alignment algorithm into a SLAM system permitted the successful estimation of the camera motion in addition to a dense map of the environment as was shown in the validation.

One of the perspectives of this work is to consider the combination of the OP-B, which permits to increase the robustness towards large camera displacements, with the CVO of (Jadidi et al., 2019), which permits to increase the estimation precision. The resulting algorithm would take the best of both worlds, being robust to large displacements and more precise at convergence than the PP-B. Furthermore, as was shown in the results the loop closure module of the proposed SLAM system is still not able to be proficient in all the sequences of the Kitti dataset. One possible solution to increase its efficiency would be the consideration of the combination of the images and scans information, permitting the use of additional clues for the detection of loop closures.



(a)



(b)

FIGURE *6.19: Estimated and ground-truth trajectories for the sequences 00 (a) and 02 (b) of the Kitti dataset. The estimation drift for the sequence 00 was corrected because the loop closure was able to correctly detect revisited places, while it failed on doing so in the sequence 02.*

Chapitre 7

Conclusions and Perspectives

The research work presented in this thesis revolved around two kinds of approaches related to SLAM and visual odometry : feature-based and direct approaches. The former were considered in Chapter 4, showing how features initialization can have substantial effects on the estimation precision of VO algorithms. As the more a feature is "reliable", in the sense that it is easily re-detected, the better the estimation is. The proposition of this chapter was consequently to build an algorithm for the initialization of "reliable" features. This algorithm uses Gaussian Mixture Models to model the image saliency information and distribute the initialized features accordingly ; permitting to initialize more features in the most salient image regions and less in the least salient ones. As was shown in the conducted experiments, such an approach results in features that have a better matching percentage and a longer lifetime, since they are principally located in easily discriminated image regions. Furthermore, the experiments permitted to validate the proposed features initialization strategy, showing its effectiveness in increasing the estimation precision and consistency in the context of visual odometry.

Direct approaches for their part were considered in Chapters 5 and 6. In Chapter 5, the proposed algorithm was able to estimate the parameters of a geometric transformation (translation or homography) between a pair of images using their scale-space representation. Such a method permits to increase the robustness towards large transformations, as direct approaches suffer from limited basin of convergence. This increase in the robustness towards large transformations is possible because of the automatic adaptation of the input image scale through the joint optimization of its scale parameter along with the geometric parameters in the same framework. As was shown in the various experiments that were conducted, this proposed image align-

ment algorithm has a larger basin of convergence, when compared to state-of-the-art learning and non-learning approaches.

In chapter 6, the images scale-space representation were considered to build an algorithm that continuously optimizes the scale parameter of the input image within each level of a multi-resolution image pyramid. This algorithm was used to estimate the 3D camera displacement between a pair of RGB-D images, permitting to notably increase the convergence domain of the algorithm. Furthermore, the proposed algorithm was integrated to a SLAM system that permits the fusion of the 3D scans of a laser range finder with the images of a camera. This enlargement of the basin of convergence was shown throughout the conducted experiments, as the proposed algorithm was more robust to large camera displacements in comparison to the standard photometric-based RGB-D image alignment algorithm.

One of the perspectives related to the proposed feature initialization strategy concerns its integration to other state-of-the-art algorithms as the ORB-SLAM (Mur-Artal et al., 2015) or the SVO (Forster et al., 2014). The consideration of additional information sources such as IMUs or LIDARs represent also an interesting research track. These sensors can hence be used to define the most suitable image regions to initialize features according to the camera movement. Image regions that are likely to be unseen in next frames would thus be discarded, while the ones that would exhibit substantial parallax would be favored. The consideration of learning approaches either to compute saliency maps or to discard irrelevant image regions (moving objects) may also be a promising research track. As various methods were proposed to compute saliency maps using Convolutional Neural Networks (CNNs) (Simonyan et al., 2014; Monroy et al., 2018) or to detect moving objects Zhu et al. (2020).

In Chapter 5, the use of images scale-space representation showed its interest for increasing the algorithm basin of convergence through the optimization of the scale parameter. One of the limitations, however, of the proposed approach is related to this scale parameter. Since its initial value is not computed automatically. This is still an open problem as no method to our knowledge permits its automatic computation. One possibility would be to compute a measure related to the image structure and uniformity (lack of contrast), since the smoothing effect of the scale parameter highly depends on the image contrast. Such a measure would hence permit to set the scale parameter's initial value. It would also be possible to integrate an illumination model into the proposed approach in order to account for illumination variations of the scene. In such a case the use of learning approaches (Riegler and Koltun, 2021) may

be of good help. Furthermore, optimizing the scale of the whole cost function instead of only the one of the input image may permit to have more direct control of the cost function level of smoothness. Such a change in the approach would however result in the introduction of additional constraints to unsure the adapted evolution of the scale parameter for the image alignment purpose.

The RGB-D image alignment algorithm proposed in Chapter 6 permitted an increase of the robustness towards large camera displacements in the 3D space, and was shown to be effectively integrated into the proposed SLAM algorithm. Despite these encouraging results, some improvements can be considered regarding this algorithm, such as its combination with the Continuous direct sparse Visual Odometry (CVO) algorithm of (Jadidi et al., 2019). The resulting algorithm would take the best of both approaches, being robust to large camera displacements and more precise than the standard photometric-based approaches. Furthermore, as was mentioned in Chapter 6, the proposed algorithm permits the continuous optimization of the scale parameter within each level of a multi-resolution image pyramid. An attractive expansion of the proposed method would be to use area-based resampling to build a fully continuous pyramid-based image alignment instead of common discrete pyramids considered in Chapter 6. Concerning the proposed SLAM algorithm, the loop closure module can still be improved. Notably by considering the additional information provided by the laser range finder, which may result in increased performances of this module. It is also possible to consider the integration of the proposed feature initialization strategy into the loop closure module, in order to increase its efficiency.

An additional perspective to this research work would be to develop a hybrid approach between feature-based and direct approaches. One possibility would hence be to directly estimate geometric transformations using features descriptors. Such an approach would be computationally more expensive while also more robust to illumination variations.

Bibliographie

Akaike, H., Petrov, B. N., and Csaki, F. (1973). Second international symposium on information theory. 53

Alismail, H., Browning, B., and Lucey, S. (2016). Robust tracking in low light and sudden illumination changes. In *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*. 24

Angeli, A., Doncieux, S., Meyer, J., and Filliat, D. (2009). Visual topological slam and global localization. In *2009 IEEE International Conference on Robotics and Automation*, pages 4300–4305. 17

Antonakos, E., Alabort-i Medina, J., Tzimiropoulos, G., and Zafeiriou, S. P. (2015). Feature-Based Lucas–Kanade and Active Appearance Models. *IEEE Transactions on Image Processing*, 24(9) :2617–2632. 24

Bailey, T., Nieto, J., and Nebot, E. (2006). Consistency of the fastslam algorithm. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 424–429. 14

Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on : A unifying framework. *International Journal of Computer Vision*, 56(3) :221–255. 7, 24, 65, 68, 70, 71, 74

Barrau, A. and Bonnabel, S. (2015). Invariant filtering for pose ekf-slam aided by an imu. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2133–2138. 23

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf : Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg. Springer Berlin Heidelberg. 60, 89

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces : recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7) :711–720. 68, 73

Benhimane, S. and Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 1, pages 943–948. IEEE. 24, 68, 71

Bergström, P. and Edlund, O. (2014). Robust registration of point sets using iteratively reweighted least squares. *Computational Optimization and Applications*, 58 :543–561. 88, 89

Blochliger, F., Fehr, M., Dymczyk, M., Schneider, T., and Siegwart, R. (2018). Topomap : Topological mapping and navigation based on visual slam maps. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3818–3825. 17

Bouguet, J.-Y. (2000). Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*. 25

Bresson, G., Féraud, T., Aufrère, R., Checchin, P., and Chapuis, R. (2011). A New Strategy for Feature Initialization in Visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment*, San Francisco, United States. 23, 43

Bresson, G., Féraud, T., Aufrère, R., Checchin, P., and Chapuis, R. (2015). Real-time monocular slam with low memory requirements. *IEEE Transactions on Intelligent Transportation Systems*, 16(4) :1827–1839. 23

Bristow, H. and Lucey, S. (2016). In Defense of Gradient-Based Alignment on Densely Sampled Sparse Features. In *Dense Image Correspondences for Computer Vision*. Springer, Cham. 24

Brossard, M., Barrau, A., and Bonnabel, S. (2019a). Exploiting symmetries to design ekfs with consistency properties for navigation and slam. *IEEE Sensors Journal*, 19(4) :1572–1579. 17

Brossard, M., Barrau, A., and Bonnabel, S. (2019b). Exploiting symmetries to design ekfs with consistency properties for navigation and slam. *IEEE Sensors Journal*, 19(4) :1572–1579. 23

Brossard, M., Bonnabel, S., and Barrau, A. (2018). Invariant kalman filtering for visual inertial slam. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2021–2028. 16, 48, 58

Brown, D. C. (1976). The bundle adjustment - progress and prospects. *Intemational Archives of. Photogrammetry*, 21(3) :3–03(33). 14

Brown, M. and Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. In *International Journal of Computer Vision*. 24

Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck : Combining local and global optic flow methods. *International Journal of Computer Vision*. 23

Bryner, S., Gallego, G., Rebecq, H., and Scaramuzza, D. (2019). Event-based, direct camera tracking from a photometric 3d map using nonlinear optimization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 325–331. 16

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*. 54, 57

Capel, D. (2004). Image Mosaicing. In *Image Mosaicing and Super-resolution*. Springer, London. 23

Carlone, L., Du, J., Kaouk Ng, M., Bona, B., and Indri, M. (2014). Active SLAM and Exploration with Particle Filters Using Kullback-Leibler Divergence. *Journal of Intelligent & Robotic Systems*, 75(2) :291–311. 17

Chang, C., Chou, C., and Chang, E. Y. (2017). Clkn : Cascaded lucas-kanade networks for image alignment. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3777–3785. 24, 71, 73

Civera, J., Davison, A. J., and Montiel, J. M. M. (2008). Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5) :932–945. v, 45, 46

Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J. M. M. (2009). 1-point ransac for ekf-based structure from motion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3498–3504. 3, 44, 50, 54, 56

Comport, A., Malis, E., and Rives, P. (2010). Real-time quadrifocal visual odometry. *The International Journal of Robotics Research*, 29(2-3) :245–266. 24

Crivellaro, A. and Lepetit, V. (2014). Robust 3D Tracking with Descriptor Fields. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3414–3421. IEEE. 24

Csorba, M. (1997). *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, Oxford. 14

Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE. 23, 24

Davison, A. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, Nice, France. IEEE. 14, 17

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Self-improving visual odometry. *ArXiv*, abs/1812.03245. 17

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3) :229–241. 14

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping : part I. *IEEE Robotics & Automation Magazine*, 13(2) :99–110. 14

Eade, E. and Drummond, T. (2007). Monocular SLAM as a Graph of Coalesced Observations. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil. IEEE. 15, 23

Engel, J., Koltun, V., and Cremers, D. (2018). Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3) :611–625. 15, 17, 24

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM : Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, volume 8690, pages 834–849. Springer International Publishing, Cham. Series Title : Lecture Notes in Computer Science. 15, 17, 23

Eyice, K. and Çulha, O. (2018). Lrf assisted slam for airborne platforms. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1131–1134. 23

Fontana, G., Matteucci, M., and Sorrenti, D. (2014). Rawseeds : Building a benchmarking toolkit for autonomous robotics. *SpringerBriefs in Applied Sciences and Technology*, 7 :55–68. 54

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO : Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, Hong Kong, China. IEEE. 15, 17, 23, 110

Frintrop, S. and Jensfelt, P. (2008). Attentional landmarks and active gaze control for visual slam. *IEEE Transactions on Robotics*, 24(5) :1054–1065. 4, 23, 51

Gallego, G., Lund, J. E. A., Mueggler, E., Rebecq, H., Delbruck, T., and Scaramuzza, D. (2018). Event-based, 6-dof camera tracking from photometric depth maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10) :2402–2412. 16

Gallier, J. (2011). *Geometric Methods and Applications*, volume 38 of *Texts in Applied Mathematics*. Springer New York, New York, NY. 37

Gao, X. and Zhang, T. (2017). Unsupervised learning to detect loops using deep neural networks for visual SLAM system. *Autonomous Robots*, 41(1) :1–18. 16

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving ? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 91, 100

Geneva, P., Eckenhoff, K., and Huang, G. (2019a). A linear-complexity ekf for visual-inertial navigation with loop closures. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3535–3541. 23

Geneva, P., Maley, J., and Huang, G. (2019b). An efficient schmidt-ekf for 3d visual-inertial slam. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12097–12107. 23

Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2 :31–43. 11, 85, 90, 91

Grupp, M. (2017). evo : Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`. 103

Guivant, J. E. and Nebot, E. M. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3) :242–257. 14

Hahnel, D., Burgard, W., Fox, D., and Thrun, S. (2003). An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 206–211, Las Vegas, Nevada, USA. IEEE. 14

Hall, B. C. (2015). *Lie Groups, Lie Algebras, and Representations*, volume 222 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham. 37

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, second ed. edition. 36, 73, 90

Herrera, D. C., Kim, K., Kannala, J., Pulli, K., and Heikkilä, J. (2014). Dt-slam : Deferred triangulation for robust slam. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 609–616. 15

Huang, S. and Dissanayake, G. (2016). A critique of current developments in simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 13(5) :172988141666948. 19

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11) :1254–1259. 23

Jadidi, M. G., Clark, W., Bloch, A., Eustice, R., and Grizzle, J. W. (2019). Continuous direct sparse visual odometry from rgb-d images. In *Proceedings of Robotics : Science and Systems*. 80, 107, 111

Jianbo Shi and Tomasi (1994). Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. 57

Johannsson, H., Kaess, M., Fallon, M., and Leonard, J. J. (2013). Temporally scalable visual slam using a reduced pose graph. In *2013 IEEE International Conference on Robotics and Automation*, pages 54–61. 21

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). iSAM2 : Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2) :216–235. 15, 17, 21

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam : Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6) :1365–1378. 36

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45. 29

Kerl, C. (2012). Odometry from rgb-d cameras for autonomous quadrocopters. Master's thesis, Technical University Munich, Germany. 84

Kerl, C., Sturm, J., and Cremers, D. (2013). Dense visual SLAM for RGB-D cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106, Tokyo. IEEE. 15, 24

Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754. 16, 95

Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Nara, Japan. IEEE. 15, 24

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G2o : A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. 91

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). G^2o : A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, Shanghai, China. IEEE. 15

Lee, J., Gutierrez-Osuna, R., and Young, S. S. (2013). Silk : Scale-space integrated lucas-kanade image registration for super-resolution from video. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2282–2286. 64

Leonard, J. J. and Feder, H. J. S. (2000). A Computationally Efficient Method for Large-Scale Concurrent Mapping and Localization. In Hollerbach, J. M. and Koditschek, D. E., editors, *Robotics Research*, pages 169–176. Springer London, London. 14

Lin, C.-H., Zhu, R., and Lucey, S. (2016). The conditional lucas & kanade algorithm. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 793–808. Springer International Publishing, Cham. 25, 68, 73, 77, 78

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO : Common Objects in Context. In *Computer Vision – ECCV*, pages 740–755. Springer International Publishing. 68, 73

Lindeberg, T. (2014). Scale selection. In Ikeuchi, K., editor, *Computer Vision : A Reference Guide*, pages 701–713. Springer US, Boston, MA. 6, 25, 65

Loianno, G., Watterson, M., and Kumar, V. (2016). Visual inertial odometry for quadrotors on se(3). In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1544–1551. 58

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2) :91–110. 23, 24

Lu, F. and Milios, E. (1997). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4) :333–349. 20

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference*

on Artificial Intelligence - Volume 2, IJCAI'81, pages 674–679, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 6, 24, 50, 65

Mallios, A., Ridao, P., Ribas, D., Maurelli, F., and Petillot, Y. (2010). Ekf-slam for auv navigation under probabilistic sonar scan-matching. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4404–4411. 23

Matthies, L. and Shafer, S. (1987). Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3) :239–248. 13

Mei, C., Benhimane, S., Malis, E., and Rives, P. (2008). Efficient homography-based tracking and 3-d reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics*, 24(6) :1352–1364. 74

Mobahi, H., Zitnick, C. L., and Yi Ma, Y. (2012). Seeing through the blur. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1736–1743. IEEE. 25

Monroy, R., Lutz, S., Chalasani, T., and Smolic, A. (2018). Salnet360 : Saliency maps for omni-directional images with cnn. *Signal Processing : Image Communication*, 69 :26–34. 110

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). Fastslam : A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, page 593–598, USA. American Association for Artificial Intelligence. 14, 19

Moravec, H. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, Stanford. 13

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real Time Localization and 3D Reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 363–370, New York, NY, USA. IEEE. 15

Mu, B., Giamou, M., Paull, L., Agha-mohammadi, A., Leonard, J., and How, J. (2016). Information-based active slam via topological feature graphs. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5583–5590. 17

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM : A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5) :1147–1163. 15, 16, 24, 110

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM : Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, Barcelona, Spain. IEEE. 15

Newman, P. and Kin Ho (2005). SLAM-Loop Closing with Visually Salient Features. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 635–642, Barcelona, Spain. IEEE. 15

Nister, D., Naroditsky, O., and Bergen, J. (2004). 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. 14

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, second edition. 33

Oron, S., Bar-Hille, A., and Avidan, S. (2014). Extended lucas-kanade tracking. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 142–156. Springer International Publishing, Cham. 24

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 90

Pirchheim, C. and Reitmayr, G. (2011). Homography-based planar mapping and tracking for mobile phones. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 27–36. 15

Pire, T., Fischer, T., Castro, G., Cristóforis, P. D., Civera, J., and Berlles, J. J. (2017). S-ptam : Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93 :27 – 42. 17

Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F. (2017). Pl-slam : Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508. 3, 23, 43

Pupilli, M. and Calway, A. (2005). Real-Time Camera Tracking Using a Particle Filter. In *BMVC*. 14

Riegler, G. and Koltun, V. (2021). Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12216–12225. 110

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511. 56

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision (to appear)*. 56

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4) :80–92. 2, 16

Schöps, T., Sattler, T., and Pollefeys, M. (2019). Bad slam : Bundle adjusted direct rgb-d slam. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144. 16, 17

Schwiegerling, J. (2004). *Field Guide to Visual and Ophthalmic Optics (SPIE Vol. FG04)*. SPIE Publications. 4, 51

Sharmin, N. and Brad, R. (2012). Optimal Filter Estimation for Lucas-Kanade Optical Flow. *Sensors*, 12(9) :12694–12709. 25, 69

Shin, Y.-S., Park, Y. S., and Kim, A. (2020). DVL-SLAM : sparse depth enhanced direct visual-LiDAR SLAM. *Autonomous Robots*, 44(2) :115–130. 17

Siciliano, B. and Khatib, O., editors (2016). *Springer Handbook of Robotics*. Springer International Publishing, Cham. 21

Silveira, G., Malis, E., and Rives, P. (2008). An Efficient Direct Approach to Visual SLAM. *IEEE Transactions on Robotics*, 24(5) :969–979. 16

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks : Visualising image classification models and saliency maps. 110

Smith, R., Self, M., and Cheeseman, P. (1988). Estimating Uncertain Spatial Relationships in Robotics. In LEMMER, J. F. and KANAL, L. N., editors, *Machine Intelligence and Pattern Recognition*, volume 5, pages 435–461. North-Holland. 13, 19

Stachniss, C., Hahnel, D., and Burgard, W. (2004). Exploration with active loop-closing for FastSLAM. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1505–1510, Sendai, Japan. IEEE. 15

Steinbruecker, F., Sturm, J., and Cremers, D. (2011). Real-time visual odometry from dense rgb-d images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*. 9, 81, 95

Strasdat, H. (2012). *Local accuracy and global consistency for efficient SLAM*. PhD thesis, Imperial College London, UK. v, 28, 34

Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Real-time monocular slam : Why filter ? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664. 15

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 91, 93, 95

Sturm, P. and Triggs, B. (1996). A factorization based algorithm for multi-image projective structure and motion. In Goos, G., Hartmanis, J., van Leeuwen, J., Buxton, B., and Cipolla, R., editors, *Computer Vision — ECCV '96*, volume 1065, pages 709–720. Springer Berlin Heidelberg. 14

Sumikura, S., Shibuya, M., and Sakurada, K. (2019). OpenVSLAM : A Versatile Visual SLAM Framework. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2292–2295, Nice France. ACM. 16

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT Press, Cambridge, Mass. 29, 30, 32

Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics Research*, 23(7-8) :693–716. 14

Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical report, International Journal of Computer Vision. 50

Torr, P. and Zisserman, A. (2000). Mlesac : A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1) :138–156. 89

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle Adjustment — A Modern Synthesis. In Goos, G., Hartmanis, J., van Leeuwen, J., Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms : Theory and Practice*, volume 1883, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg. 14

Varadarajan, V. S. (1984). *Lie Groups, Lie Algebras, and Their Representations*, volume 102 of *Graduate Texts in Mathematics*. Springer New York, New York, NY. 37

Vedaldi, A. and Fulkerson, B. (2008). VLFeat : An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`. 68, 74

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017). Deepvo : Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. 17

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. *Inside Data Association*, 57(12) :1188–1197. 15

Williams, S. B., Newman, P., Dissanayake, G., and Durrant-Whyte, H. (2000). Autonomous underwater simultaneous localisation and map building. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 1793–1798, San Francisco, CA, USA. IEEE. 14

Wu, K., Zhang, T., Su, D., Huang, S., and Dissanayake, G. (2017). An invariant-ekf vins algorithm for improving consistency. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1578–1585. 44, 54, 57

Xiong, X. and la Torre, F. D. (2014). Supervised descent method for solving nonlinear least squares problems in computer vision. *CoRR*, abs/1405.0601. 68, 73

Yu, K.-T., Yu, J.-Y., Liu, F.-C., Tseng, S.-H., Fu, L., and Hsieh, H.-W. (2011). An integrated feature selection strategy for monocular slam. 23

Zhang, J. and Singh, S. (2015). Visual-lidar odometry and mapping : low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. 88

Zhang, J., Tai, L., Boedecker, J., Burgard, W., and Liu, M. (2017). Neural SLAM : Learning to Explore with External Memory. *arXiv :1706.09520 [cs]*. arXiv : 1706.09520. 16

Zhou, H., Fan, H., Peng, K., Fan, W., Zhou, D., and Liu, Y. (2019). Monocular visual odometry initialization with points and line segments. *IEEE Access*, 7 :73120–73130. 3, 23, 43

Zhu, H., Yan, X., Tang, H., Chang, Y., Li, E., and Yuan, F. (2020). Moving object detection with deep cnns. *IEEE Access*, PP :1–1. 110

Zhu, M., Ramalingam, S., Taguchi, Y., and Garaas, T. (2012). Monocular Visual Odometry and Dense 3D Reconstruction for On-Road Vehicles. In Fusiello, A., Murino, V., and Cucchiara, R., editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, pages 596–606, Berlin, Heidelberg. Springer Berlin Heidelberg. 16

Résumé

Pour une plateforme mobile qui explore un environnement inconnu, la capacité de se localiser est primordiale pour mener à bien sa mission. Or, quand les systèmes de localisation globale ne sont pas accessibles, ce processus de localisation nécessite une carte de l'environnement qui lui-même est inconnu. Afin de résoudre ce problème, l'approche communément adoptée consiste en l'estimation concurrente de la pose de la plateforme ainsi que la carte de l'environnement. Approche connue dans la communauté robotique sous l'acronyme SLAM (Simultaneous Localization And Mapping) – signifiant localisation et cartographie simultanées. C'est dans ce contexte que s'inscrit cette thèse, qui a pour objectif de proposer des solutions aux problématiques d'estimation du déplacement de caméras embarquées qui sont aussi liées à l'odométrie visuelle. Les contributions de la thèse peuvent ainsi se résumer dans les points suivants :

- **Proposition d'une stratégie d'initialisation de points d'intérêts basée sur l'information de saillance des images pour l'estimation de la trajectoire d'une caméra en mouvement.**
- **Utilisation de la représentation continue des images dans l'espace d'échelles afin d'augmenter le domaine de convergence des algorithmes d'alignement d'images directes pour les transformations projectives et 3D.**
- **Intégration de l'algorithme d'alignement d'images proposée à un système SLAM pour la fusion des données 3D d'un télémètre laser et des images acquises par une caméra.**

L'ensemble des expérimentations réalisées a ainsi permis de valider les approches proposées, montrant leur avantage en terme de précision d'estimation et de robustesse aux transformations importantes.

Mots clés : SLAM, odométrie visuelle, vision, télémètre laser, filtre de Kalman, Gauss-Newton, alignement d'images direct.

Abstract

During the exploration of an unknown environment, a mobile platform needs to know its location to accomplish its mission. However, in GPS-denied situations, this localization requires the use of a map of the environment, which is unknown. This problem is typically solved through the joint estimation of the platform pose and the environment map and is known in the robotics community under the acronym SLAM (Simultaneous Localization And Mapping). It is in this context that this research work is conducted, aiming at providing solutions to the problematics related to embedded camera motion estimation, also related to visual odometry. The main contributions of this thesis can hence be summarized in the following points:

- **Proposition of a feature initialization strategy based on image saliency information for the estimation of a moving camera trajectory.**
- **Use of the continuous images scale-space representation to build algorithms with increased basin of convergence for the estimation of projective and 3D transformations.**
- **Integration of the proposed image alignment algorithm into a SLAM system for the fusion of laser range finder 3D scans and images acquired by a camera.**

The conducted experiments showed the benefits of the proposed approaches concerning the estimation precision and the robustness towards large transformations.

Keywords: SLAM, visual odometry, laser range finder, Kalman filter, Gauss-Newton, direct image-alignment.