Universitat Autònoma de Barcelona

Alignement de données 2D, 3D et applications en réalité augmentée

Youssef El Rhabi

Université de Caen Normandie

École doctorale SIMEM

Thèse de doctorat

présentée et soutenue le :

par

Youssef El Rhabi

pour obtenir le

Doctorat de l'Université de Caen Normandie
Spécialité : Informatique

Alignement de données 2D, 3D et applications en réalité augmentée

Directeur de Thèse : Luc Brun
Co-directeur de Thèse : Josep Llados

Jury

Rapporteurs

Table des matières

Résumé

Cette thèse s'inscrit dans le contexte de la réalité augmentée (RA). La problématique majeure consiste à calculer la pose d'une caméra en temps réel. Ce calcul doit être effectué en respectant trois critères principaux : précision, robustesse et rapidité.

Dans le cadre de cette thèse, nous introduisons certaines méthodes permettant d'exploiter au mieux les primitives des images. Dans notre cas, les primitives sont des points que nous allons détecter puis décrire dans une image. Pour ce faire, nous nous basons sur la texture de cette image. Nous avons dans un premier temps mis en place une architecture favorisant le calcul rapide de la pose, sans perdre en précision ni en robustesse. Nous avons pour cela exploité une phase hors ligne, où nous reconstruisons la scène en 3D. Nous exploitons les informations que nous obtenons lors de cette phase hors ligne afin de construire un arbre de voisinage. Cet arbre lie les images de la base de données entre elles. Disposer de cet arbre nous permet de calculer la pose de la caméra plus efficacement en choisissant les images de la base de données jugées les plus pertinentes. Nous rendant compte que la phase de description et de comparaison des primitives n'est pas suffisamment rapide, nous en avons optimisé les calculs. Cela nous a mené jusqu'à proposer notre propre descripteur. Pour cela, nous avons dressé un schéma générique basé sur la théorie de l'information qui englobe une bonne part des descripteurs binaires, y compris un descripteur récent nommé BOLD [BTM15]. Notre objectif a été, comme pour BOLD, d'augmenter la stabilité aux changements d'orientation du descripteur produit. Afin de réaliser cela, nous avons construit un nouveau schéma de sélection hors ligne plus adapté à la procédure de mise en correspondance en ligne. Cela permet d'intégrer ces améliorations dans le descripteur que nous construisons. Procéder ainsi permet d'améliorer les performances du descripteur notamment en terme de rapidité en comparaison avec les descripteurs de l'état de l'art.

Nous détaillons dans cette thèse les différentes méthodes que nous avons mises en place afin d'optimiser l'estimation de la pose d'une caméra. Nos travaux ont fait l'objet de 2 publications (1 nationale et 1 internationale) et d'un dépôt de brevet.

Réalité augmentée : SFM,SLAM, estimation de pose temps réel, description, Apprentissage,recalage 2D/3D

Title : 2D, 3D data alignment and application in augmented reality

Abstract

This thesis belongs within the context of augmented reality. The main issue resides in estimating a camera pose in real-time. This estimation should be done following three main criteria: precision, robustness and computation efficiency.

In the frame of this thesis we established methods enabling better use of image primitives. As far as we are concerned, we limit ourselves to keypoint primitives. We first set an architecture enabling faster pose estimation without loss of precision or robustness. This architecture is based on using data collected during an offline phase. This offline phase is used to construct a 3D point cloud of the scene. We use those data in order to build a neighbourhood graph within the images in the database. This neighbourhood graph enables us to select the most relevant images in order to compute the camera pose more efficiently. Since the description and matching processes are not fast enough with SIFT descriptor, we decided to optimise the bottleneck parts of the whole pipeline. It led us to propose our own descriptor. Towards this aim, we built a framework encompassing most recent binary descriptors including a recent state-of-the-art one named BOLD. We pursue a similar goal to BOLD, namely to increase the stability of the produced descriptors with respect to rotations. To achieve this goal, we have designed a novel offline selection criterion which is better adapted to the online matching procedure introduced in BOLD.

In this thesis we introduce several methods used to estimate camera poses more efficiently. Our work has been distinguished by two publications (a national and an international one) as well as with a patent application.

Augmented Reality : SFM,SLAM, real time pose computation, keypoint description, Machine learning, 2D/3D registration

Títol : enregistrament 2D/3D i aplicació en realitat augmentada

Abstract

Aquesta tesi s'emmarca en el context de la realitat augmentada. La problemàtica més gran consisteix en l'estimació de la posició de la càmera en temps real. Aquesta estimació s'ha de fer seguint tres criteris principals: precisió, robustesa i eficiència computacional.

En el marc d'aquesta tesi, establim alguns mètodes que permetin un millor ús de les primitives de les imatges. En el nostre cas, les primitives de les imatges són els punts característics. Per aconseguir aquest objectiu, ens basem en la textura d'aquesta imatge. Primerament, establim una arquitectura que faciliti una estimació de la posició més ràpida, sense pèrdua de precisió o robustesa. Aquesta arquitectura es basa en la utilització de la informació recollida durant una fase offline, en la qual reconstruïm l'escena en 3D. Utilitzem tota aquesta informació per a construir un graf de veïnatge dins de les imatges de la base de dades. Aquest graf de veïnatge ens permet seleccionar les imatges més rellevants per tal de calcular la posició de la càmera de manera més eficaç. En tant que els processos de descripció i matching no són prou ràpids, s'han optimitzat els càlculs, la qual cosa ens ha portat a proposar el nostre propi descriptor. Amb aquesta finalitat, hem construït un esquema genèric basat en la teoria de la informació la qual engloba una bona part dels descriptors binaris, inclòs el recent descriptor anomenat BOLD. El nostre objectiu ha estat, com per a BOLD, incrementar l'estabilitat dels descriptors produïts en els canvis d'orientació. Per aconseguir-ho, hem dissenyat un nou esquema de selecció offline que s'adapta millor al procés de matching online, que ens permet integrar les millores al descriptor que hem construït. Tot això ens permet millorar les actuacions del descriptor especialment en termes de rapidesa en comparació amb els descriptors de l'estat de l'art.

En aquesta tesi descrivim diversos mètodes utilitzats per a estimar la posició de la càmera més eficientment. Dels resultats del treball n'han sorgit dues publicacions (una nacional i una altra internacional) així com una sol · licitud de patent.

Realitat augmentada : SFM,SLAM, estimació de la posició a temps real, descriptors basats en punts característics, aprenentatge, enregistrament 2D/3D

Discipline : Informatique et applications.

Université de Caen Basse-Normandie, ENSICAEN, CNRS
GREYC - Équipe image
6 Boulevard Maréchal Juin, 14050 Caen cedex, France

44screens
7 rue Bondor, 50100,Cherboug, France

Computer Vision Center
Edificio O
Campus UAB
08193 Bellaterra (Cerdanyola)
Barcelona, Spain

Extended abstract

This thesis is focused on the resolution of several bottlenecks within the augmented reality framework. The main issue resides in estimating a camera pose in real-time. This estimation should be done following three main criteria : precision, robustness and computation efficiency. Within this thesis we established methods enabling a better use of image primitives. As far as we are concerned, we limit ourselves to key-point primitives. Within this framework, one first bottleneck concerns the huge amount of keypoint comparisons performed by usual augmented reality pipelines. We first set an architecture enabling faster pose estimation without loss of precision or robustness. This architecture is based on using data collected during an offline phase. This offline phase is used to construct a 3D point cloud of the scene. We use those data in order to build a neighbourhood graph within the images in the database. This neighbourhood graph enables us to select the most relevant images in order to compute the camera pose more efficiently. The second bottleneck concerns the important execution time required to compare keypoint's descriptors such as SIFT. It led us to propose our own descriptor. Towards this aim, we built a framework encompassing most recent binary descriptors including a recent state-of-the-art one named BOLD. We pursue a similar goal to BOLD, namely to increase the stability of the produced descriptors with respect to rotations. To achieve this goal, we have designed a novel offline selection criterion which is more suited to the online matching procedure introduced in BOLD. In this thesis we introduce several methods used to estimate camera poses more efficiently. Our work has been distinguished by two publications (one national and one international one) as well as with a patent application.

Chapter 1 - Introduction

Augmented reality is a field that attracts more and more. We can highlight it with the appearance of this technology in movies like terminator (1984)

in which the robot's view is enhanced by additional information. Augmented reality requires three things :

- A camera in order to acquire the scene
- A processor enabling the camera's pose computation in real time
- A screen in order to print the augmented scene

This Phd was made in partnership with GREYC laboratory, CVC laboratory, and 44screens a digital company. In its activities the 44screens company often creates augmented reality applications in outdoors contexts. In many of there applications they work on the outside views of buildings. Such buildings are not easy to scan and it's not so easy to control the environment surrounding them. Our aim is to cope with issues raised by augmented reality in not supervised outdoor context. The problematic are the following ones :

- 3D model acquisition can be complex,
- Strong risk of object occlusions,
- Acquisition conditions can be highly varying,
- Since applications are based on mobiles there is few available computational power

The computation time problem on mobile device is the one we address the most since it's the most challenging one.

We want to have a system which will be precise, fast, robust and flexible. To reach that we decided to set a framework based on texture analysis in order to compute device poses in a scene. By doing this we were able to cope occlusions issues.

Chapter 2 - State of the art

In chapter 2 we focus on the state of the art. We first introduce tools and notions required to compute the position of a device in a scene. We also provide the state of the art of device localisation in a scene. So in this chapter we describe how an image capture device works. The principle behind it is the protective geometry. Once we have a correct representation of capture devices we can set the theory of 3D scene reconstruction and locate ourselves properly in the scene. Then we describe how all those steps can be combined to create a structure

from motion experience so that a 3D scene can be reconstructed. Finally, we introduce some recent methods of device location.

We first introduce mathematical tools required to get an augmented reality experience. The book [HZ03] details precisely the tools to achieve those steps. We start by describing the chosen camera model, the pinhole camera model. The pinhole camera model consists in considering that the camera can be represented by a focal point (the optical center of the camera) and a focal plane on which 3D points will be projected. The pinhole camera model is represented in figure 1. We see that a projection camera can be represented as a point in space combined with a focal plane. 3D points from the real world (X in figure 1) are projected onto the focal plane by intersecting the line joining them and the optical center (line d_1 in figure 1) and the focal plane.

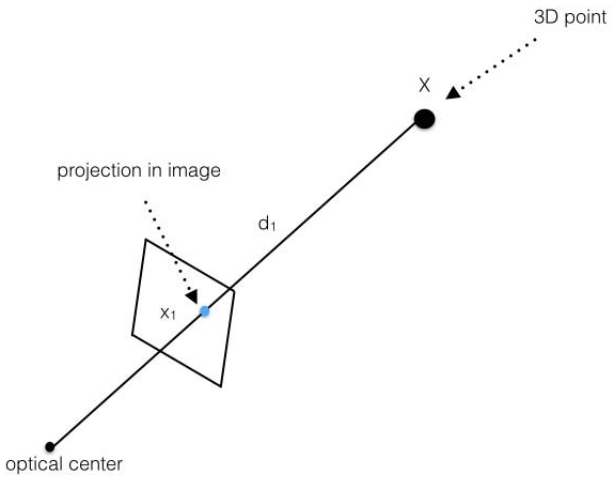


FIGURE 1 – Pinhole camera model

We then describe how to retrieve the relative positions of two cameras having a set of correspondences as an input. To achieve this goal we rely on epipolar geometry. Epipolar geometry can be summed up as the geometry of different views of the same scene. In practice we don't have any 2D correspondences between images. Hence we need detectors and descriptors. Among detectors two classes can be pointed out : the first class is based on scale spaces and differences of gaussians, the other one is based on accelerated segments tests (AST) and detects corners in an image. In terms of descriptors we can distinguish two different categories : the full spectrum descriptors, and the binary ones. Concerning the first family of descriptors, the full spectrum ones, we focus on histogram of oriented gradients (HOG) methods. Very often this kind of descriptors are combined with scale spaces. The binary descriptors on the other hand are usually combined with AST methods (for

their computational efficiency). We also give details about learning methods such as convolutional neural networks (CNN) which have proven to be very effective in the last few years. In the general scenario descriptor based on CNN also learn how to detect points in an image.

Having those detectors and descriptors we are able to link points between different images. Still a strategy in order to achieve this task optimally is required. According to the type of descriptors different approaches are possible. For example when using binary descriptors local sensitive hashing (LSH) is a good option since it exploits hamming distance. When using HOG methods kd-tree implementation is preferred. Even if pairs of points are linked carefully some of them can be linked by mistake. In order to cope with this fact some correspondence filtering is required. For that matter approaches such as RANSAC like methods [FB81, RDGM10, MMM12a] are very efficient. Once we dispose of those filtered matches classical optimisation methods such the Levenberg-Marquardt [Lev44, Mar63] one can be used to solve the pose computation problem. Then all those tools can be used to reconstruct a 3D scene namely in the frame of the SFM approach. Chapter 3 relates how those tools can further be used in order to create an augmented reality experience.

We also present some recent augmented reality solutions. Augmented reality consists in inserting virtual objects into the real world. We want this insertion to be as natural as possible. For that aim computing accurately the camera pose is crucial. In terms of augmented reality first instances were based on markers like with [Vuf13]. But more and more solutions tend to be based on the scene itself like with Metaio, Diotasoft,Wikitude. To do so methods either used the silhouette of an object in order to compute camera pose, or otherwise use texture of elements in the scene. We put a particular focus on markerless approaches which can be divided between :

- silhouette based approaches
- reprojection error minimization based methods

In conclusion we can say that we detailed how to build a 3D scene and how to locate a device in it. We namely focused on the real time issue. Our approach can be seen as a constrained SLAM method in which the reconstruction phase and the localization phases are decoupled. Hence our approach is divided in two different steps :

- A pre-processing step in which images of the scene are acquired to reconstruct the scene
- Another step in camera pose is computed in real time

In our researches we focused on methods that improve computation robustness while ensuring real time pose computation. For that aim in chapter 3 a framework based on SIFT descriptor is proposed. We propose a structure based on graphs allowing to reduce the set of images to focus on in order to compute the image pose. We also propose a filter on SIFT points in order to select the more robust ones. Then still in the idea of getting results even faster we proposed our own binary descriptor in chapter 4.

Chapter 3 - presentation of our methodology

In augmented reality our purpose is to insert virtual objects in a given spot of a real scene. Augmented reality can be summed as the problem of projecting in a realistic manner an object into the scene. For that aim knowing the position of the device in the scene is necessary. Since the pose computation problem boils down to computing 2D/3D correspondences and that to a given 2D point there is an infinity of possible 3D points it is an ill-posed problem. In order to simplify this problem we build during a pre-processing step a list of 2D/3D correspondences thanks to the structure from motion (SFM) method by using the bundle adjustment method. Then in order to locate a new image in the scene we compute 2D/2D correspondences between the new image and the images acquired during the pre-processing. Then 2D/3D correspondences can be retrieved transitively. In figure 2 the whole process we set to solve the pose computation problem is described. First step consist in creating links between 2D points (green lines) in image acquired from the scene. Once those steps are done the 3D reconstruction part can start. Here the purpose is to construct a point cloud of the scene having the knowledge of 2D correspondences between different pairs of images. In this points cloud each 3D point is linked to a set of 2D points in the preprocessed images (the 2D points that helped to construct it). In order to have a finer approximation of the relative positions of cameras and the point cloud a bundle adjustment [LA09] is operated. Once all those operations are made it is possible to compute the pose of a new image i_1. The main idea is to link 2D points and retrieve 2D/3D links by transitivity. When those 2D/3D links between the new image and the point cloud are available it is possible to compute the pose of the camera that took image i_1. With the knowledge of the pose we can then project objects in the scene.

Noticing that all those steps were too slow we focused on how to make them faster, with no considerable loss of robustness. To cope with this problematic we deported the task with the highest complexity (the 2D/3D correspondences) in a pre-processing step. Furthermore we propose some structures enabling reduction of the pose computation of an image while using SIFT descriptor.

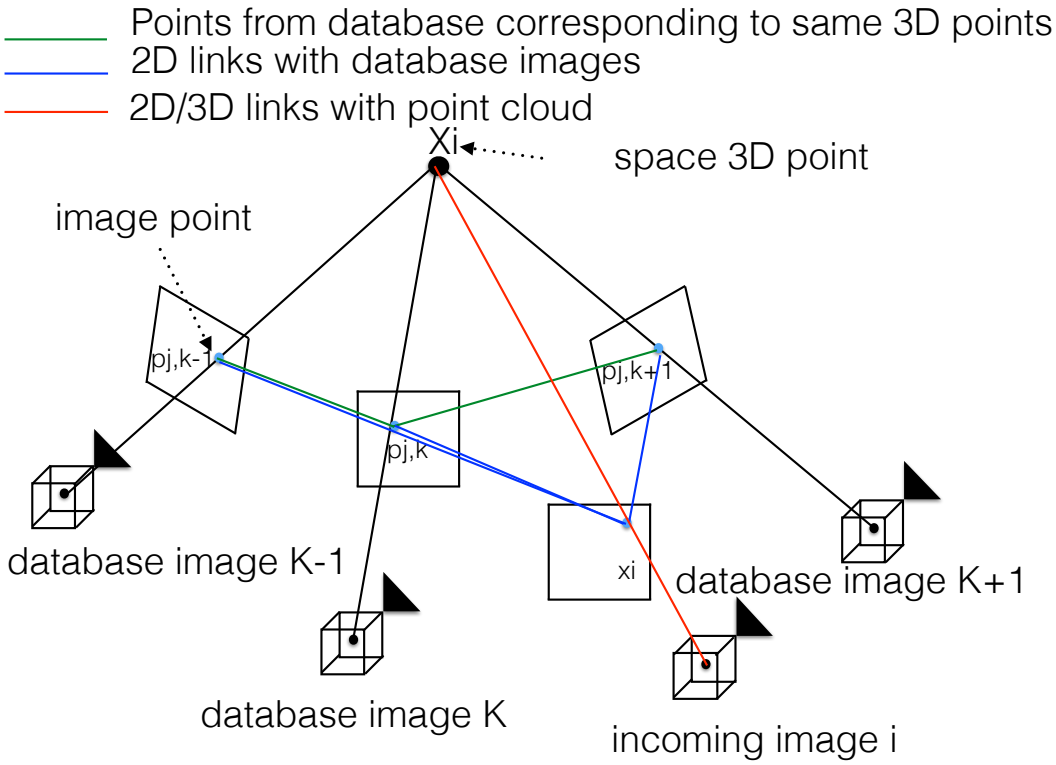


FIGURE 2 – 3D reconstruction process and camera pose estimation

The first proposal is to filter the SIFT points we want to match to one another. In order to do so we keep a given ratio of SIFT points that have the highest scale. Furthermore we propose a graph structure enabling to create a neighbourhood evaluation. With this neighbourhood evaluation we are able to select database images with which the incoming frame will be compared. Thanks to those improvements computation time decreased from 9.77s to 0.12s for pose computation. The loss of precision is about 3 pixels for images of resolution 5312x2988. Detailed results can observed in table 1.

Chapter 4 - Contribution to binary descriptors

Since the advent of SIFT [Low99], extracting local descriptors has become a common practice in order to assess the similarity of image regions. Applications of local descriptors have been considerable, such as image stitching to build panoramas [BL07], context-based image retrieval, visual odometry or multi-

SIFT points ratio		100%	80%	40%	20%	10%
Convergence time	B_d complete	9.77 s	7.29 s	3.81 s	1.93 s	1.01 s
	relevant images	1.69 s	1.36 s	0.78 s	0.46 s	0.31 s
Precision	B_d complete	0 px	2.5 px	3.6 px	4.1 px	3.9 px
	relevant images	2.65 px	3.2 px	3.8 px	3.0 px	4.9 px

TABLE 1 – table highlighting performances observed after our optimisations

view 3D reconstruction [MMM13a]. As a result of its success, this line of research has greatly impacted our everyday behaviour, be it by our use of efficient exemplar based image search engine, or the pervasive introduction of computer vision in mobile devices. Due to this important economical and societal repercussions, the design of ever improving descriptors has drawn a strong interest [BTVG06, MS05]. One of the main enhancements relates to data-driven construction schemes, where a typical database of image correspondences is leveraged to learn an efficient descriptor [HBW07, TCFL13]. In particular, recent approaches based on deep learning techniques [ZK15] have shown a strong improvement on the state of the art.

However, some kind of "no free lunch" principle applies in that quest. Depending on the targeted application, the desired properties of the descriptor may differ significantly, leading to several trade-offs and design principles. Among others, the following questions are recurrent. Is the computational complexity of paramount importance ? Does accuracy matter more than completeness ? What class of invariance is required ? For instance, in context-based image retrieval, a query image is proposed to a system that should propose several images similar to the query. But often semantic similarity is more crucial than purely visual resemblance. On a different note, perspective or affine invariance is a desirable asset in a multi-view reconstruction system but not in a tracking scenario. These central questions become further more complicated in practice, since descriptors are no more than a brick in a complex pipeline. Therefore, some properties of the descriptors can be destroyed or corrected by other parts of the systems. For instance, invariance can be embedded in the design of a descriptor or provided by detecting an orientation and scale before computing a non invariant descriptor. A more sophisticated case is exposed in [FKS+16], where the authors acknowledge

the benefit of binary descriptors for real-time applications but claim that in a 3D reconstruction system, typical descriptors like SIFT provide a better compromise between accuracy and run time, in particular when matching is accelerated thanks to adapted data structures [ML12].

We here intend to improve the state-of-the-art of descriptors with real-time applications in mind (e.g. SLAM). We therefore focus on low-complexity binary descriptors based on image intensity comparisons [CLSF10, AOV12]. This active line of research lies at the crossroad of several intertwined areas such as feature selection [PLD05, SNB14] and hashing [IM98]. Our contributions include a clear exposition of a generic framework encompassing the typical state-of-the-art descriptor pipelines, as well as the design of an elegant information theoretic criterion used in the feature selection process. It yields a consensus between the discriminative power of the descriptor and its resilience to rotations. This contribution is evaluated on classical benchmarks and decreases the time and space complexity by a factor 2 compared to a recent state-of-the-art technique [BTM15]. We used this descriptor in the context of augmented reality reaching better results than in chapter 3. Indeed while keeping decent approximation errors as compared to image sizes (3.97 pixel errors) we reached a computation time as low as 0.09 seconds.

In conclusion we have developed a novel binary image descriptor that finds its roots in the context of real time applications. To construct this descriptor, we have laid a common foundation based on feature selection. This framework covers most of the recent data-driven binary descriptors including a recent one called BOLD. We have also complemented the online selection mechanism proposed in BOLD by an adapted offline criterion applied beforehand. This new mechanism presents an elegant information theoretic interpretation and above all a perceptible practical influence. The immediate comparison to BOLD conveys that in most cases, our descriptor carries as much useful information while being twice more compact. Such an asset is an important benefit in the considered applications. Comparisons to a few other classical descriptor show that our approach obtains favorable results under mild geometric transforms. This situation arises easily in applications on mobile devices where guessing a rough estimate is often possible thanks to additional sensors.

chapter 5 - Conclusion

We can here conclude that in non-controlled environment there are few systems that are good enough. In particular in the frame of exterior environment and without any 3D model available. The best possible solution in such a case is

to work on objects textures. In our approach we detect and describe points of interest. While working on mobile devices two main issues are to be addressed :

- First description and linking of 2D points is a task that can be greedy in terms of computational time
- The second issue concerns the design of a descriptor robust enough to several transformations while being fast enough.

In this thesis we had as a goal to improve some methods of the state of the art in augmented reality. We focused on two topics : setting an augmented reality framework enabling computation time reduction and setting a global framework encompassing most recent binary descriptor. We even extended this frame by proposing some improvements on those descriptors.

In the future we aim to work on :

- **Managing objects occlusions.** Having new material we aim to be able to detect when parts of some objects are to be hidden due to occlusions that occur in the scene.
- **Using convolutional neural network to learn detector and descriptor.** We aim to go deeper in the learning aspects in order to be able to learn features from an image database.
- **Estimating scene illumination.** Being able to estimate scene illumination will enable us to apply it to the object so that the rendering might seem even more real.

Chapitre 1

Introduction

Nous présentons ici une thèse CIFRE financée par 44screens. La thèse s'est déroulée dans le cadre d'une cotutelle entre le laboratoire GREYC (Groupe de Recherche en Informatique, Image, Automatique et Instrumentation) à Caen et le CVC (Computer Vision Center) de l'université autonome de Barcelone à Cerdanyola.

1.1 Introduction de la réalité augmentée

La réalité augmentée (RA) est un domaine qui suscite de l'intérêt. Depuis des années la RA attise l'imagination de l'Homme.La réalité de cet attrait est attestée dans des films de science-fiction tels que Terminator. Dans ce film sorti en 1984, le robot Terminator observe le monde réel avec une série d'informations qui défilent en superposition (voir figure 1.1a). La RA est un problème qui requiert trois éléments :

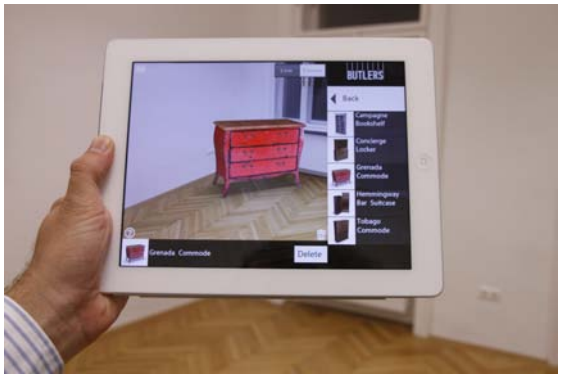
- une caméra pour filmer la scène
- un processeur permettant le calcul temps réel de la pose du dispositif
- un écran pour afficher la scène augmentée en temps réel

Ce sont des caractéristiques que l'on retrouve de plus en plus communément sur les appareils mobiles (smartphones, tablettes, smartglasses...). Le caractère transportable des appareils mobiles en fait un outil de choix pour de telles applications.

La force de la RA réside dans ses utilisations hétéroclites. Elle se sert de l'environnement utilisateur afin de faire remonter des informations à un endroit

(a) Terminator et réalité augmentée

(b) RA dans le contexte du pilotage



(c) RA dans le contexte de la vente

FIGURE 1.1 – Réalité augmentée dans plusieurs contextes

choisi. Cela permet d'imaginer des interactions bien plus diversifiées que ce qui se faisait jusque là, et plus adaptées au cas d'usage également. Cette souplesse lui permet d'être appliquée à des domaines aussi vastes que la maintenance, la médecine, le patrimoine, ou encore l'aide à la vente.

Dans le cas de la maintenance, ou encore de la médecine en chirurgie, la RA peut être mise à profit pour faciliter certaines tâches. L'idée est ici de localiser des éléments de la scène avec lesquels l'utilisateur doit interagir, et lui indiquer quelles actions effectuer. La figure 1.1b illustre comment la RA peut servir au pilotage d'un avion.

Dans des contextes comme la vente ou le tourisme la RA peut être vue comme un moyen de transformer un élément de la scène afin de l'observer de la manière désirée. En ce qui concerne le tourisme cela peut se concrétiser par la visualisation d'un château à sa plus belle époque ou encore à plusieurs époques différentes. C'est, entre autres, ce que propose 44screens (figure 1.3b). Pour la vente la RA peut être exploitée afin de de visualiser et personnaliser un objet ou un lieu (personnalisation du placement d'un élément dans une scène, de sa couleur, de ses motifs, design...) comme on peut le voir dans la

FIGURE 1.2 – Système de RA haptique

figure 1.1c. Nous pouvons constater dans cette figure que l'objet peut être placé dans son contexte d'utilisation. Cela facilite ainsi la visualisation par le client d'un produit personnalisé répondant entièrement à ses envies. Ces mêmes arguments peuvent s'appliquer aux domaines de l'architecture et du design. La RA peut apporter aux architectes, ou au designers, la représentation visuelle de leur projet en cours de construction, les aidant ainsi à faire les choix qui leur semblent être les plus judicieux.

L'image n'est pas la seule option qui ait été considérée afin de localiser une caméra dans une scène, et ainsi être en mesure de mettre en place des expériences de RA. Les premières instances de la RA impliquaient l'utilisation de dispositifs mécaniques comme dans [Sut68]. De prime abord, cette approche semble parmi les plus simples. Généralement il s'agissait d'exploiter la configuration d'éléments mécaniques connectés avec des pièces électromécaniques. De cette configuration est déduite la pose du dispositif. Dans les systèmes haptiques (voir figure 1.2) ce genre de processus est encore utilisé. L'inconvénient majeur de cette approche reste que les déplacements et mouvements sont limités. De plus, ces dispositifs sont souvent encombrants.

Une autre approche possible consiste à s'appuyer sur des données inertielles, composées par un gyroscope et un accéléromètre. Ces éléments sont de plus en plus compacts et précis, ce qui en fait une alternative intéressante. Cependant, pour localiser un appareil avec suffisamment de précision, le biais et la dérive induit par ce type de capteurs sont trop importants. Ces capteurs peuvent toutefois être utilisés en complément d'autres méthodes [YNA99, LKPB02].

Des capteurs acoustiques basés sur la différence de phase entre un signal émis et le même signal reçu par le dispositifs ont aussi été introduits par Sutherland. Dans ce genre de méthodes la différence de phase est utilisée afin de calculer la distance de l'objet par rapport aux émetteurs, sa position peut donc être calculée par triangulation. Cette méthode présente deux inconvénients conséquents :

- Certains objets reflètent les ondes sonores provoquant des phénomènes de retour. Ces ondes réfléchies seront à nouveau reçues par le dispositif (avec des phases différentes encore et cela provoquera une perturbation de l'estimation).
- Malgré les améliorations apportées à cette méthode, elle reste très sensible aux autres sources de sons (au bruit).

Afin de calculer l'orientation d'un appareil l'exploitation de capteurs magnétiques peut aussi être envisagée [HCG+96, L+97]. Ces capteurs sont usuellement calibrés sur l'orientation terrestre, celle-ci étant relativement stable en tout point de la Terre. Cependant certains objets, ferromagnétiques ou encore conducteurs, altèrent les champs magnétiques. Cela influence l'estimation de l'orientation. Cette approche reste intéressante car elle peut permettre de suivre des objets de petites tailles (voire même de suivre plusieurs objets), et n'est pas gênée par les occlusions car les champs passent à travers la plupart des objets.

Dans cette constellation de capteurs, les capteurs optiques tirent leur épingle du jeu. Dans les milieux non contrôlés nous pouvons trouver tous type de perturbations : des sons, des objets, des champs magnétiques et autres. Ces perturbations ne permettent pas aux autres capteurs de fournir des estimations suffisamment précises de la pose d'un objet. Les méthodes basées sur la vision offrent plus de possibilités. Les évolutions en vision par ordinateur permettent des traitements de l'image efficaces et robustes aux bruits. La diversité des approches possibles en fait un outil puissant. Les méthodes de vision peuvent en effet se baser sur l'anticipation des déplacements en se basant sur des filtres bayesiens [ZF12, Dav03, ED06], sur la forme d'un objet à suivre ou encore sur la texture des objets dans une scène. Cela donne une gamme d'outils flexibles et robustes permettant de faire face aux défis soulevés par la RA. De plus le fait d'avoir à portée de mains des dispositifs mobiles de plus en plus puissants (combinant la présence d'une caméra ainsi que d'un terminal de calcul et d'affichage) rendent les méthodes de vision par ordinateur d'autant plus pertinentes. Le fait que ces dispositifs possèdent le plus souvent des gyroscopes, d'un accéléromètre et d'autres capteurs permettent de combiner la vision avec ces approches et d'améliorer ainsi les estimations de pose.

1.2 Contexte de la thèse

Cette thèse a été réalisée au sein de 44screens qui est une société spécialisée dans l'innovation numérique interactive. L'activité de cette entreprise se concentre tout particulièrement sur la création d'applications mobiles exploitant la 3D. 44screens fait entre autre de :

- La modélisation, de l'animation 3D, et de la réalité virtuelle,
- De la réalité augmentée,
- Du scan 3D, du prototypage 3D

La société s'engage sur le développement d'applications dans les deux contextes suivants :

- Appliquer son savoir-faire afin de valoriser le patrimoine dans des contextes comme le tourisme. Cela est fait dans un programme qu'ils appellent Visites Augmentées.
- Apporter au monde de la maintenance son expertise afin d'optimiser l'exécution de certaines tâches.

À l'heure actuelle, elle a déjà fourni des applications de qualité dans le patrimoine. L'application d'Arromanches en Normandie (illustrée figure 1.3a) en est un parfait exemple. Cette application consiste à nous montrer, sur les plages d'Arromanches, les ponts suspendus qui ont été érigés pendant la seconde guerre mondiale. Elle a, depuis, participé à de nombreux projets comme :

- la Hunaudaye (voir figure 1.3b) : application permettant de visualiser le château de la Hunaudaye à plusieurs époque différentes.
- Mortain (voir figure 1.3c) : application ayant vocation à entrainer les utilisateurs dans les légendes de Mortain.
- Manzanares el real (voir figure 1.3d) : application permettant de visiter le château de Manzanares el real en Espagne comme il se présentait au Moyen Âge.
- Beaugency (voir figure 1.3e) : Application relatant, de façon à la fois ludique et immersive, l'histoire de Jean de Dunois compagnon d'armes de Jeanne d'arc.

(a) Arromanches

(b) Hunaudaye

(c) Mortain

(d) Manzanares el real

(e) Beaugency

FIGURE 1.3 – Quelques applications produites par 44screens

Comme nous pouvons le constater, dans leurs projets, 44screens produisent le plus souvent des expériences de RA sur des bâtiments et en extérieur. Cela rend les conditions de calcul de pose contraignantes dans la mesure où :

- L'acquisition de modèles 3D dans ce contexte est difficile
- Il y a de forts risques d'occlusions, dû au passage de personnes etc..

- Les conditions d'acquisitions risquent d'être fortement changeantes
- Les applications étant basées sur mobile il y a peu de puissance de calcul à dispositon

De là est né le besoin chez 44screens d'entamer un programme de recherche et développement permettant de faire face à ces contraintes. C'est dans ce contexte que le sujet de cette thèse a été proposé.

1.3 Problématique et objet de la thèse

Dans le cadre de cette thèse nous visons à offrir des solutions permettant de faire face aux problèmes liés à la RA en extérieur dans le cadre décrit précédemment. Nous avons décidé d'estimer la pose de la caméra dans des environnements peu contrôlés sans avoir de contraintes telles que fournir un modèle 3D de l'objet. Pour cela nous avons pris le parti de nous appuyer sur la texture des éléments de la scène pour localiser la caméra. Cela permet en plus d'être robuste à certaines occlusions. En effet ainsi même si certaines parties de la scène ne sont pas visibles, d'autres éléments de textures le seront.

Nous l'avons compris, le but de la RA est de localiser un dispositif dans une scène afin d'augmenter la vue en y insérant des objets virtuels. Nous avons argumenté sur le fait qu'utiliser les outils de vision par ordinateur, notamment sur dispositifs mobiles, est une approche attrayante. Nous avons aussi vu que, comme nous sommes confrontés à diverses occultations et à l'absence de modèle 3D, l'utilisation de primitives dans ces images est l'approche la plus pertinente. Nous désirons en effet un système à la fois :

- Précis : la localisation du dispositif doit permettre de recaler les objets virtuels dans le monde réel sans déviation et sans vibration (effet de jittering)
- Rapide : l'objet doit être localisé à la volée, lorsque les images sont acquises. Tous les calculs doivent être effectués en temps réel
- Robuste : les altérations de la scène ne doivent pas influencer outre mesure la précision de la localisation. Les altérations peuvent être : les changements d'illumination, de points de vue etc.
- Souple : différents types d'objets dont la forme peut être très changeante doivent pouvoir être suivis dans des environnements peu contrôlés. Il ne faut pas oublier que des occultations peuvent aussi se produire à tout moment.

Pour faire face à l'ensemble de ces contraintes nous proposons certaines solutions que nous présentons dans cette thèse. Dans un premier temps dans le chapitre 2, nous décrivons les outils que nous jugeons nécessaires à la compréhension de cette thèse.

Ensuite dans le chapitre 3 nous nous appuyons sur les principes liés à l'utilisation de primitives pour le calcul de pose dans des environnements peu contrôlés. Nous abordons également certaines méthodes qui ont permis d'en augmenter la rapidité sans perdre en précision. Cela a donné une contribution dans [ERSB15]. Dans [ERSB15] nous avons aussi utilisé une méthode permettant d'utiliser les points SIFT [Low99] à moindre coût.

Dans un deuxième temps dans le chapitre 4 nous analysons plus en profondeur les contraintes liées au temps réel. L'idée étant d'allier de nouveau rapidité et robustesse. Pour cela nous avons créé un descripteur binaire, qui tout en étant rapide à calculer reste robuste à certaines déformations de l'image. Ce travail a été valorisé par une publication internationale dans [ERSB+16] et à un dépôt de brevet.

Nous finissons par un bilan de ce qui a été fait dans le chapitre 5 ainsi que sur les perspectives à envisager.

Chapitre 2

État de l'art

Le domaine abordé dans cette thèse est la réalité augmentée. Nous introduisons ici des notions et outils permettant le calcul de position d'un dispositif dans une scène. Nous dressons aussi l'état de l'art des méthodes de localisation dans une scène. Pour cela nous devrons commencer par décrire la façon dont un dispositif de capture fonctionne. Le principe utilisé est la géométrie projective vue dans la section 2.1. Une fois que nous disposons d'une représentation correcte de nos dispositifs de capture, nous pouvons mettre en place la théorie permettant de reconstruire une scène en 3D et de se localiser correctement dans cette scène. Cela est abordé dans les parties 2.1.5 à 2.1.12. Dans la partie 2.1.13 nous voyons comment toutes ces étapes peuvent être exploitées dans le cadre de la "structure from motion" afin de construire une scène en 3D. Nous voyons ensuite quelques méthodes récentes de localisation dans une scène, cela est fait dans la section 2.2.

2.1 Reconstruction 3D et localisation d'un dispositif dans une scène

.

Comme nous l'avons décrit plus haut, cette partie est dédiée à introduire les outils mathématiques nécessaires à la réalité augmentée. Ce sont ici des rappels, plus de détails peuvent être trouvés dans [HZ03]. Dans les parties 2.1.2 à 2.1.4 nous décrivons le modèle de caméra choisi, nous définissons au préalable dans la partie 2.1.1 certaines notations ainsi qu'une formalisation de la notion de coordonnées. Dans la partie 2.1.5 nous décrivons comment, à partir de correspondances entre des points $2D$ de deux images d'une même scène, retrouver leur position relative. En pratique nous ne disposons à priori d'aucune

correspondance $2D$ entre les images. Nous avons donc besoin de détecteurs et descripteurs. Dans les sections 2.1.6 à 2.1.8 nous décrivons les principes liés à la détection et la description. Cela constitue une des étapes clés pour le bon déroulement de tout le processus. Une fois ces étapes réalisées, nous voyons dans 2.1.9, il s'agit ensuite d'apparier les points détectés dans les images afin de calculer la position relative de celles-ci. Par contre les correspondances fournies par les descripteurs ont un fort taux de faux positifs, aussi appelés aberrations. Pour faire face à cela des outils de filtrage, notamment basés sur des contraintes liées à la géométrie épipolaire, ont été mis en place, comme cela est détaillé dans la section 2.1.10. Dans la partie 2.1.11 nous décrivons quelques méthodes de résolutions de problèmes d'optimisation fréquemment utilisés dans le domaine du traitement de l'image. Enfin lorsque nous disposons de tout cela nous pouvons effectivement reconstruire la scène, comme vu dans les sections 2.1.12, où les outils sont décrits, et 2.1.13, où la notion de structure from motion est décrite. Pour la réalité augmentée ce procédé peut être exploité de plusieurs façons différentes, nous en donnerons un exemple dans le chapitre 3.

2.1.1 Coordonnées homogènes

Posons $\mathbb{P}^n$ l'espace projectif de dimension n comme étant l'ensemble des classes d'équivalence de $\mathbb{R}^{n+1}$ formé par la relation qui suit :

$$\forall u, v \in \mathbb{R}^{n+1*}, u \sim v \iff \exists \lambda \in \mathbb{R}^* \text{ tel que } u = \lambda v \tag{2.1}$$

Ce qui nous intéresse ici est de définir la notion de points homogènes, un vecteur de $\mathbb{P}^n$ pourra ainsi être noté :

$$w = [w_1 \dots w_{n+1}]^T \tag{2.2}$$

Avec au moins un des w_i non nul. Si w_{n+1}, le dernier paramètre, est non nul alors le vecteur $\bar{w} = [w_1/w_{n+1} \dots w_{n+1}/w_{n+1}]^T$ est le représentant de la classe de w. On nomme $\bar{w}$ vecteur homogène. Si w_{n+1} est nul, alors w représente un point à l'infini. Nous pourrons noter Ψ la fonction faisant passer de coordonnées homogène aux coordonnées euclidiennes. Nous pouvons la noter ainsi :

$$\begin{array}{c} \Psi = \\ \end{array} \begin{array}{c} \mathbb{P}^n \longrightarrow \mathbb{R}^n \\ [w_1 \dots w_{n+1}]^T \longmapsto [w_1/w_{n+1} \dots w_n/w_{n+1}]^T \end{array} \tag{2.3}$$

2.1.2 Modélisation d'une caméra

Nous allons ici présenter le modèle de caméra sténopé. Nous avons choisi ce modèle car les smartphones du marché configurés de façon usuelle le respectent suffisamment. D'autres modélisations ou traitements mathématiques sont envisageables, elles sont présentées plus en détail dans [HZ03]. Nous avons pris le parti ici de décrire seulement ceux que nous avons utilisés, et qui permettent de comprendre notre approche.

Puisque le processus de prise de photo peut être mathématiquement assimilé à une projection d'une scène 3D dans un plan 2D, on peut le modéliser ainsi (figure 2.1) : la caméra est représentée par un point. La projection de la scène 3D sur l'image se fera dans un plan placé dans le plan focal de l'appareil (plan situé à la distance focale du dispositif). Pour déterminer la position d'un point de l'espace dans l'image on tracera la droite de l'espace passant par le centre du dispositif et par le point 3D en question. L'intersection de cette droite avec le plan focal nous donnera la position 2D du point 3D dans l'image. Ce dispositif est nommé caméra sténopée (en anglais pinhole camera model).

La matrice de projection d'une telle caméra peut être définie ainsi :

$$P = K[R|t] \tag{2.4}$$

où K est la matrice de calibration, R la matrice de rotation, et t le vecteur de translation.

2.1.3 Matrice de calibration d'une caméra

La matrice de calibration est celle qui renseigne sur les paramètres intrinsèques de la caméra. Ces paramètres intrinsèques correspondent aux réglages internes de la caméra, ils sont aussi appelés paramètres internes. La matrice de calibration K se présente sous la forme suivante :

$$K = \begin{pmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Nous avons ici α_u et α_v qui correspondent aux facteurs d'échelle sur chacun des axes principaux de la caméra. Ils sont liés à la longueur focale de la caméra par les formules $\alpha_u = k_u f$ et $\alpha_v = k_v f$, où k_u et k_v représentent le nombre de pixels par unité correspondant à chaque axe principal.

Le vecteur $c = [u_0, v_0]^T$ représente les coordonnées image de l'intersection de l'axe optique avec le plan focal de l'appareil, ce point est appelé point principal.

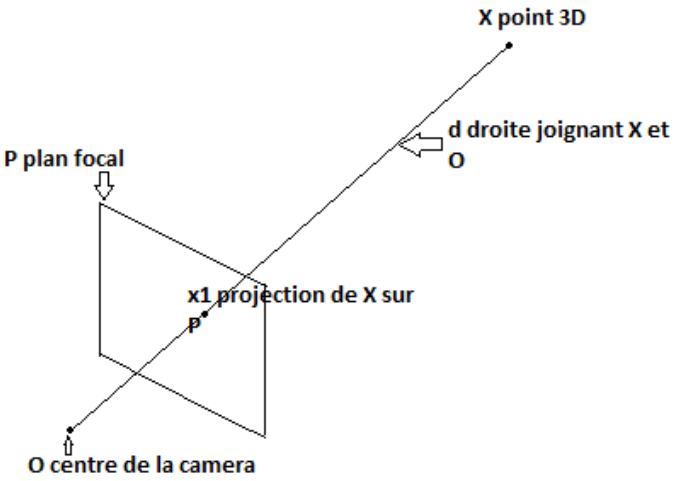


FIGURE 2.1 – modèle caméra sténopé : X point de l'espace observé O centre optique de l'appareil x projection du point 3D

s est un facteur d'obliquité. Dans la vaste majorité des cas il est nul ce qui signifie que les axes principaux sont perpendiculaires.

Dans la plupart des cas le point principal c sera suffisamment proche du centre, et les pixels seront carrés donc α_u et α_v sont égaux. Il existe plusieurs algorithmes permettant de calibrer une caméra comme l'algorithme proposé dans [JY08].

2.1.4 Caméra normalisée

Pour faciliter certains calculs liés à la géométrie épipolaire, nous avons besoin de nous placer dans le contexte où la calibration de la caméra est connue. Cela permet de simplifier certaines représentations et surtout de plus facilement déterminer les positions relatives de deux caméras. Pour cela nous introduisons la normalisation des caméras et des coordonnées.

En partant du principe que la calibration de l'appareil utilisé K est connue, nous pouvons poser :

$$\hat{x} = K^{-1}x \tag{2.5}$$

x sont les coordonnées d'un point dans l'image. $\hat{x}$ est nommé coordonnées normalisées de x. Cela permet ensuite d'exprimer $\hat{x}$ comme suit :

$$\hat{x} = [R|t]X \tag{2.6}$$

où R la matrice de rotation, et t le vecteur de translation, X est un point de l'espace correspondant à x dans l'image.

De la même manière nous nommons caméra normalisée la matrice qui suit :

$$\hat{P} = K^{-1}P = [R|t] \tag{2.7}$$

Nous avons maintenant introduit la plupart des outils dont nous avons besoin pour la représentation d'une caméra. Nous pouvons donc aborder des notions de géométries plus particulières, permettant par la suite le calcul des poses de caméra.

2.1.5 Géométrie épipolaire

La géométrie épipolaire est la géométrie de différentes représentations d'un même objet. Étant donné que le processus de prise de photo peut être assimilé à une projection d'une scène 3D dans un plan 2D, on peut le modéliser, comme vu dans la partie 2.1.2, ainsi :

- la caméra est considérée comme un point
- la projection de la scène 3D sur l'image se fera dans un plan placé dans le plan focal de l'appareil (plan situé à la distance focale du dispositif).
- Pour déterminer la position d'un point 2D sur l'image on tracera la droite de l'espace passant par le centre du dispositif et par le point 3D. L'intersection de cette droite avec le plan focal nous donnera la position du point 2D sur l'image.

Ce dispositif est nommé modèle sténopé (figure 2.1).

Matrice Fondamentale

L'avantage de la géométrie épipolaire est qu'elle nous permet d'avoir des informations sur deux images d'un même objet sans avoir besoin de connaitre la pose des caméras a priori, cas dans lequel nous nous trouvons. La géométrie épipolaire à deux vues se caractérise par l'existence de la matrice fondamentale F, qui a plusieurs propriétés intéressantes. La matrice F permet de modéliser une relation entre deux points x_1 d'une image I_1 et x_2 d'une image I_2, qui sont en correspondance (c'est à dire qu'ils représentent le même point X de l'espace). À noter qu'ici les images I_1 et I_2 sont deux représentations d'un même objet sous deux angles de vues différents. En voici la formule :

$$x_1^T F x_2 = 0 \tag{2.8}$$

où F une matrice de taille 3x3.

Cette expression est vraie pour tous points en correspondance. Considérer cette formule lorsque seul x_1 est connu revient à dire que x_2 peut être trouvé dans l'image I_2 sur la droite d'équation $x_1^T F$. Cette droite est nommée droite épipolaire du point x_1 dans l'image I_2.

La matrice F a donc un intérêt non négligeable : elle nous permet de vérifier par la suite si des points que nous avons mis en correspondance sont cohérents. Elle nous aide ainsi à filtrer les points mis en correspondance à tort.

F étant une matrice 3x3, il faut théoriquement 9 correspondances pour la déterminer. Cependant F est définie à un facteur multiplicatif près. Il est donc possible de l'obtenir à partir de 8 correspondances. De plus F est de rang 2. On peut donc la paramétrer avec 7 paramètres indépendants. L'algorithme des 7 points consiste à trouver le noyau du système linéaire constitué par 7 correspondances 2D. Ces correspondances fournissent un système d'équation de rang 2. Ce noyau est de dimension 2, la matrice F s'exprime sous la forme :

$$F = F1 + \alpha F2 \tag{2.9}$$

Où $F1$ et $F2$ sont deux matrices 3×3 indépendantes solutions du système linéaire posé précedemment. Il suffit alors de trouver α pour que $\det(F)=0$. Il s'agit d'une équation polynomiale de degré 3.

Matrice Essentielle

La matrice fondamentale est centrale dans la mesure où elle nous aide à calculer les matrices de projection des dispositifs ayant permis l'acquisition des deux images. Cependant elle ne nous procure pas suffisamment d'informations.

En effet avec F, il est possible de déterminer les matrices de projection de chacune des deux vues, dans un référentiel que nous fixons, seulement à une transformation projective près. Dans la pratique, pour éliminer cette ambiguïté projective, on a recours à la matrice essentielle E. Pour disposer de la matrice essentielle il est nécessaire de connaître la matrice de calibration du dispositif de capture. Sachant qu'une matrice de projection peut être définie comme dans 2.1.2

À partir des correspondances normalisées (vue dans 2.1.4) nous calculons la matrice essentielle E comme suit :

$$\hat{x}^T E \hat{x}' = 0 \tag{2.10}$$

Pour tout couple de coordonnées normalisées $\hat{x}$ et $\hat{x}'$ en correspondance.

La matrice essentielle peut être calculée grâce à l'algorithme 5 points. Cela est possible car E peut être exprimée comme suit :

$$E \propto [t]_\times R \tag{2.11}$$

où R est la rotation relative des deux caméras et t leur translation relative. Nous avons donc :

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad [t]_\times = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.12}$$

De ce fait, E peut être représentée par 6 paramètres et peut donc être déterminée à un facteur multiplicatif près grâce à 5 correspondances. Comme pour l'algorithme de 7 points, il s'agit de trouver les 4 matrices E_1,..,E_4 du noyau du système linéaire induit par les 5 correspondances. La matrice essentielle peut alors être cherchée sous la forme

$$E = E_1 + aE_2 + bE_3 + cE_4 \tag{2.13}$$

de façon à vérifier les contraintes internes dues au fait que la matrice E est de rang 2 et que ces valeurs propres non nulles sont égales. C'est ce que fait l'algorithme des 5 points décrit dans [Nis04]. On peut reformuler ces contraintes (dites contraintes internes) sous la forme suivante :

$$\begin{aligned} \det(E) &= 0 \\ 2EE^TE - 1/2\,\mathrm{Tr}(EE^T)E &= 0 \end{aligned} \tag{2.14}$$

Nous pouvons résumer les choses ainsi, la matrice essentielle est un cas particulier de matrice fondamentale pour lequel les images sont normalisées.

Dans ce contexte les matrices de projection sont calculées à : un facteur d'échelle près et une ambiguïté de quatre dispositions près comme nous pouvons le voir figure 2.2.

Calcul de matrice de projection à partir de la matrice essentielle

Nous avons vu précédemment comment calculer la matrice essentielle entre deux images. Ici nous abordons la façon dont nous pouvons estimer les matrices de projection relatives de ces deux images en se servant de la matrice essentielle E. L'avantage d'utiliser la matrice essentielle, plutôt que la matrice fondamentale, est d'obtenir les matrices de projections à ambiguïté d'échelle et à quatre dispositions possibles près. La matrice fondamentale ne nous donne les positions des caméras qu'à une ambiguïté projective près. Cela limite grandement le champ des poses possibles. D'autant plus que parmi les quatre dispositions possibles une seule est retenue, celle pour laquelle les deux caméras font face à tous les points 3D générés.

Pour simplifier l'écriture nous supposons qu'une des caméras, notons la c_1, est à l'origine du référentiel $\mathbb{F}_c$ dans lequel nous nous plaçons. Toutes les preuves sont détaillées dans le livre [HZ03]. Il y est démontré que E est une matrice essentielle si et seulement si ses deux premières valeurs singulières sont égales et la troisième est nulle. Ils introduisent dans [HZ03], les matrices Z, W et $\text{diag}(1,1,0)$ définies comme suit :

$$Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{diag}(1,1,0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.15}$$

En se basant sur le fait que l'on ait l'équation 2.11, E peut donc être exprimée ainsi :

$$E \propto [t]_\times R = SR \tag{2.16}$$

Où S est une matrice antisymétrique qui peut être exprimée ainsi

$$S = kUZU^T \text{ où } U \text{ est une matrice orthogonale} \tag{2.17}$$

En considérant que la SVD de E est $U\text{diag}(1,1,0)V^T$, cela débouche sur le fait que $E = SR$ peut avoir deux factorisations possibles :

$$\begin{aligned} &S \propto UZU^T \\ &R = UWV^T \text{ ou } R = UW^TV^T \end{aligned} \tag{2.18}$$

Cette factorisation se déduit de l'unicité de la décomposition en SVD et du fait que R peut s'écrire sous la forme $R = UXV^T$ où X est une matrice de rotation. On obtient alors l'égalité :

$$U\text{diag}(1,1,0)V^T = E \propto SR = (UZU^T)(UXV^T) = U(ZX)V^T \tag{2.19}$$

On en déduit que $X = W$ où $X = W^T$, et donc deux possibilités pour R. De plus comme le signe de E n'est pas connu, sachant que $St = 0$ et donc $t = U(0,0,1)$, alors le signe de t n'est pas connu non plus. D'où les quatre configurations de caméras possibles, en considérant la caméra de c_1 à l'origine et donc $P_1 = [I|0]$:

$$P_2 = [UWV^T| + u_3] \text{ ou } [UWV^T| - u_3] \text{ ou } [UW^TV^T| + u_3] \text{ ou } [UW^TV^T| - u_3] \tag{2.20}$$

Pour retrouver la bonne combinaison parmi les quatre nous cherchons celle pour laquelle les points 3D font face aux deux caméras, c'est le cas en rouge dans la figure 2.2. Utiliser la matrice essentielle plutôt que la matrice fondamentale lève une grand partie de ambiguïté. L'ambiguïté sur l'échelle de E, ne peut quant à elle être levée, et résulte en une ambiguïté sur la translation relative des caméras et sur l'échelle globale de la reconstruction. Il s'agit donc d'une simple convention des unités métriques pour la reconstruction.

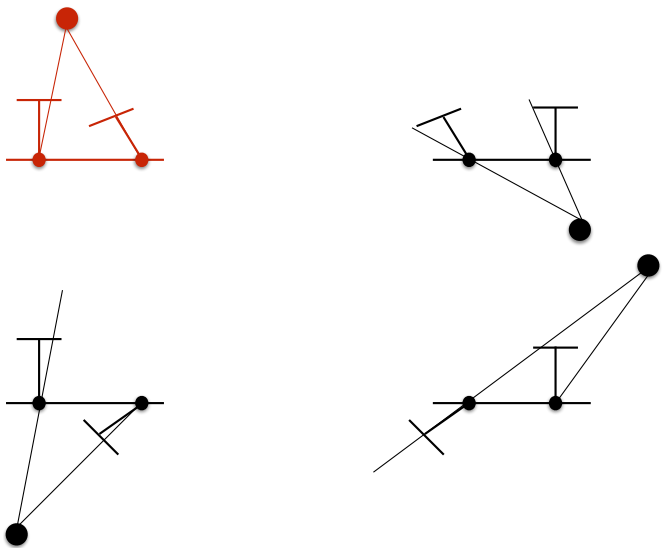


FIGURE 2.2 – quatre dispositions différentes possibles pour la caméra

Nous avons, dans cette section, établi comment se servir de la géométrie épipolaire afin d'extraire des informations de paires d'images. Toutes ces théories supposent cependant qu'en amont nous ayons déjà connaissance de points correctement en correspondances, mais dans les faits ce n'est pas le cas. Pour cela nous devons passer par la mise en place et l'exploitation de points d'intérêts comme nous le verrons dans les parties 2.1.6, 2.1.7, 2.1.8.

2.1.6 Points d'intérêts

Il convient, avant de définir comment détecter et décrire des points d'intérêts dans des images, de savoir comment ces points sont caractérisés. Savoir comment trouver ces points est une étape préliminaire à de nombreuses applications. Nous pouvons entre autre nommer la reconstruction 3D, le suivi d'objets, la détection de personnes voire aussi la reconnaissance de gestes. Ces points sont souvent choisis comme étant des discontinuités des niveaux de gris comme montré dans la figure 2.3.

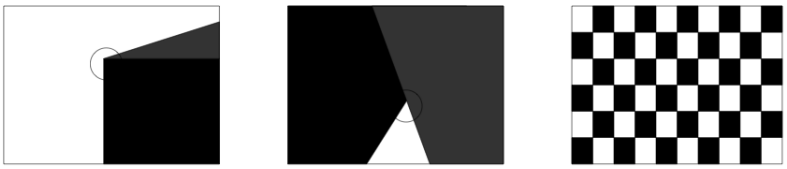


FIGURE 2.3 – Discontinuités possibles : coin, jonction en T, damier ou forte variation de texture

Il est courant que ce type de discontinuités apparaissent lors de modification de texture, de structure ou encore de géométrie dans l'image. Étant donné que notre approche est basée sur la géométrie épipolaire, qui s'appuie sur la mise en correspondance de points, nous avons d'emblée éliminé les méthodes basées sur les contours comme vue dans [AB86], repris par [DG93] plus tard.

Parmi les détecteurs deux grandes classes peuvent être extraites : une première classe de méthodes est basée sur les pyramides d'échelles et de différences de gaussiennes pour la détection de points, une autre est basée sur les tests de segments accélérés (AST) afin de détecter des coins. Nous verrons cela dans la section 2.1.7.

Concernant les descripteurs nous pouvons distinguer deux grandes catégories, les descripteurs à virgules flottantes et les descripteurs binaires. Pour les descripteurs à virgules flottantes nous nous intéresserons en particulier aux cas des HOG, qui se présentent sous formes d'histogrammes d'orientations de gradients. Ces mêmes HOG existent en version circulaire ou carrée. Nous

présenterons aussi les descripteurs basés sur des techniques d'apprentissage qui ont suscité dans les dernières années beaucoup d'intérêt. Cela sera l'objet de la section 2.1.8.

2.1.7 Détection

La détection consiste à chercher les points d'une image dont la répétabilité est forte. La répétabilité peut être vue comme la propriété caractérisant le fait que, pour deux images différentes représentant une même scène, des points représentant les mêmes zones d'intérêts soient détectés. C'est une propriété d'importance notamment afin de simplifier par la suite la mise en correspondance des points détectés.

Les premières instances en termes de détecteurs se sont concentrées sur la détection de régions ou de points distribués de façon éparse dans l'image. Cela génère en moyenne quelques milliers de points dans chaque image. Usuellement les points choisis sont les coins : le détecteur de coins de Harris [HS88] en est un bel exemple. Du fait qu'ils présentent de multiples discontinuités dans la fonction d'intensité de l'image, les coins sont faciles à détecter. Une alternative aux détecteurs de coins de type Harris est l'utilisation d'extrema locaux de certains filtres. C'est ce que nous aborderons dans la partie qui suit.

Méthodes basées sur différence de gaussiennes

SIFT [Low99] détecte les points d'intérêt dans des espaces d'échelles. L'espace d'échelle est défini ainsi :

$$L(x;\sigma) = g_\sigma * I(x) \tag{2.21}$$

où g_σ est un flou gaussien et $I(x)$ la valeur de la fonction intensité en x. Les images ainsi générées seront groupées par octaves (nous passons d'une octave à une autre en doublant σ). Suite à cela des différences de gaussienne sont opérées pour la détection multi-échelle. La différence de gaussienne (DOG) s'exprime ainsi :

$$D_k(x;\sigma) = L(x;k\sigma) - L(x;\sigma) \tag{2.22}$$

k représente la finesse de l'espace d'échelle désirée (souvent égal à $\sqrt{2}$). Les espaces d'échelle ainsi que la formation des DOG sont représentées figure 2.4. Les DOG sont effectuées entre deux images consécutives d'une octave. Une fois les DOG obtenues on y détecte les extrema locaux, à plusieurs échelles. Afin de choisir si un point est un extremum local, il est comparé à ces 8

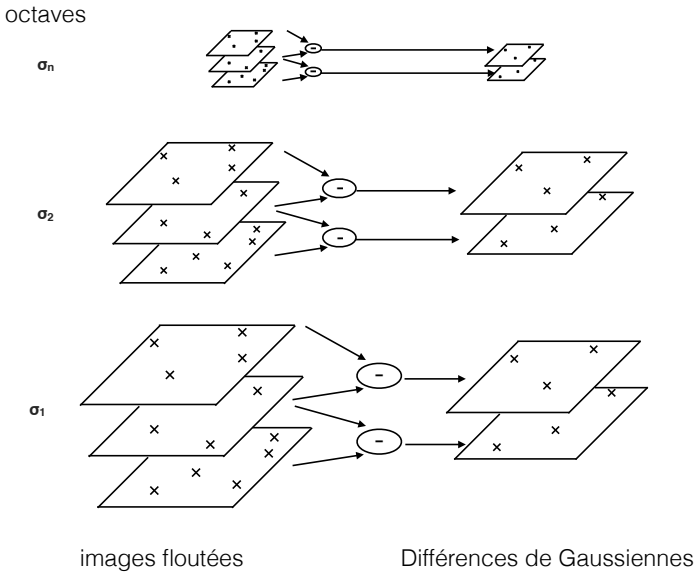


FIGURE 2.4 – représentation des espaces d'échelle et formation des différences de gaussiennes

voisins dans la même échelle et à 9 pixels correspondant dans d'autres espaces d'échelles. Le point sera considéré comme point d'intérêt si c'est un extremum dans toutes les configurations énoncées précédemment (inspiré de la détection de blob de Lindeberg [Lin93]). Ce procédé fournissant une quantité de points trop importantes, Lowe [Low99] propose certains filtres. Il utilise par exemple l'expansion de Taylor

$$D_k(x) \approx D_k(x_0) + \frac{\partial D_k^T(x_0)}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D_k^T(x_0)}{\partial x^2} x \tag{2.23}$$

afin de déterminer si un point candidat est vraiment un extremum, et interpoler sa position. Ainsi les points de faibles contrastes, détectés en vérifiant si l'élément de second ordre de l'expansion de Taylor est inférieur à 0.03, ne sont pas retenus. Enfin les DOG ont tendance à détecter beaucoup de points sur les contours. Les points sur les arêtes doivent de préférence être éliminés car ils ne possèdent pas d'échelle caractéristique. Pour cela il utilise la matrice Hessienne en s'appuyant sur le fait que pour un coin les variations sont fortes sur les deux axes.

Il existe d'autres approches, nous pouvons mentionner celle de SURF [BTVG06]. En s'inspirant de SIFT [Low99], SURF approxime les filtres

gaussiens par des filtres carrés de taille 9×9 et avec un écart type σ=1.2. Pour rendre le processus plus rapide encore des images intégrales sont utilisées :

$$S(x, y) = \Sigma_{i \leq x} \Sigma_{j \leq y} I(i, j) \tag{2.24}$$

Afin de détecter les points d'intérêts SURF utilise la matrice Hessienne. Ils exploitent son déterminant afin d'avoir une mesure des changements locaux. La hessienne est aussi utilisée pour obtenir l'échelle, comme chez Lindeberg [Lin98]. En général l'échelle est obtenue en créant une pyramide d'image. Chez SURF les auteurs modifient la taille du filtre carré (approximant la gaussienne). L'échelle est ainsi obtenue par :

$$\sigma_{approx} = \text{échelle du filtre de base} \times \frac{\text{taille du filtre utilisé}}{\text{taille du filtre de base}} \tag{2.25}$$

Les méthodes décrites dans [MTS+05, MCUP04] sont elles aussi basées sur des propriétés de covariance. Cela permet de détecter des régions similaires, même lorsque les images subissent des transformations géométriques. Bien que très efficaces ces méthodes ne permettent pas d'être suffisamment proche du temps réel. Pour cela nous parlons dans le chapitre qui suit de méthodes adaptées aux détecteurs binaires. Bien que moins performantes elles ont l'avantage de fournir des points plus rapidement.

Détecteurs pour descripteurs binaires

Les détecteurs de ce type consistent en des détecteurs de coins. Nous détaillerons dans cette partie quelques unes des méthodes les plus connues de l'état de l'art.

CENSURE [AKB08] est construit en trois étapes. En premier lieu un laplacien de gaussienne bi-niveau est appliqué. Les réponses faibles sont filtrées, les éléments restant sont les contours détectés. Dans un deuxième temps, parmi ces contours, les extrema locaux sont détectés. Enfin, en utilisant la mesure de Harris [HS88] sur ces extrema locaux, ceux avec une forte réponse sont gardés. Ce sont les plus à même d'être des coins.

Dans [RD06] les auteurs introduisent la notion de critère de détection FAST. Ils y utilisent la méthode des tests de segment accéléré (AST) afin de détecter des points. Le voisinage d'un pixel p correspond à un ensemble de 16 pixels autour de celui ci, ils sont repartis comme vu sur la figure 2.5. Les pixels voisins sont classifiés comme étant 'clairs' si leur intensité est plus grande que celle de p par au moins un seuil t fixé. Ils sont classifiés comme 'sombres' si ils sont

plus sombres d'au moins t. Cela se représente ainsi, soit p_1 un pixel voisin de p :

$$\text{si} \begin{cases} p_1 > p + t \text{ alors } p_1 \text{ sera considéré comme 'clair'} \\ p - t \leq p_1 \leq p + t \text{ alors } p_1 \text{ sera considéré comme 'semblable'} \\ p_1 < p - t \text{ alors } p_1 \text{ sera considéré comme 'sombre'} \end{cases} \tag{2.26}$$

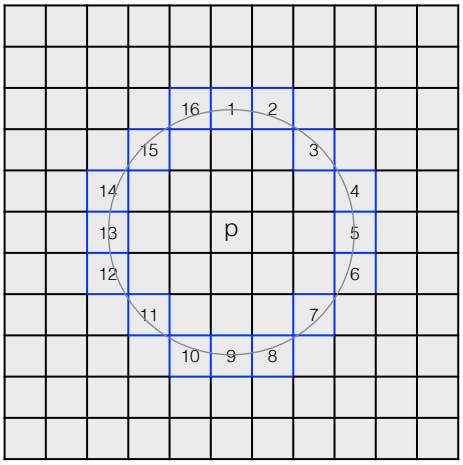


FIGURE 2.5 – représentation du voisinage d'un pixel p chez FAST

Le pixel candidat sera considéré comme point d'intérêt s'il est déclaré comme étant un coin. C'est pour vérifier cela que le test des segments intervient. Si un ensemble de n voisins consécutifs sont considérés comme 'clairs' ou 'sombres' , alors on dit que p est un coin. Ils ajoutent à cela une couche d'apprentissage basé sur l'algorithme ID3 [Qui86] afin de choisir, parmi les pixels voisins, ceux qui apportent le plus d'informations. Cela permet de contrôler l'ordre des comparaisons effectuées et ainsi d'en réduire le nombre nécessaire en moyenne pour classifier un pixel comme un coin. Ceci permet d'améliorer encore le temps de calcul du détecteur. Ils ont aussi rajouté des mesures pour supprimer les non maximum (NMS). C'est un problème rencontré lorsque dans un voisinage donné des points multiples sont détectés. Pour régler cela ils comparent un point détecté à deux points adjacents et ne gardent que celui qui a la plus forte probabilité d'être un coin. Afin de rendre l'approche encore plus efficace la NMS est appliquée seulement sur une partie des points qui ont passé positivement le test des segments.

Dans ORB [RRKB11], le détecteur FAST est utilisé afin de trouver des points d'intérêts. Ensuite la mesure de coins de Harris [HS88] est utilisée sur ces

points afin de sélectionner les N meilleurs candidats. Ils complètent également leur détecteur en y ajoutant une pyramide d'échelle, cela donne la possibilité de détecter des points sur plusieurs échelles. Pour pallier au fait que FAST ne donne pas d'orientation, ils utilisent des barycentres pondérés par l'intensité des pixels [Ros99]. Ces barycentres sont centrés sur le coin détecté. Le postulat est que l'intensité d'un coin est décalée de son centre, l'orientation du point est donnée par le vecteur représentant le décalage (voir figure 2.6). Pour renforcer l'invariance aux échelles, les moments, qui permettent le calcul du barycentre, sont calculés de sorte à ce qu'ils soient dans le rayon associé à la taille du patch.

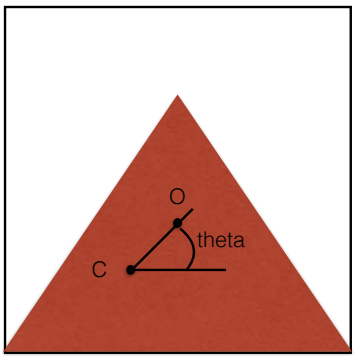


FIGURE 2.6 – Schéma représentant le décalage entre le centre du coin O et le moment de la centroide C.

Dans le cas de BRISK [LCS11], leur descripteur est lui inspiré d'AGAST [MHB+10]. AGAST est lui même une amélioration de FAST [RD06], dans laquelle les performances sont augmentées en construisant les arbres de décision de façon plus optimale. Il effectue également une pyramide d'échelle. La suppression des non-maximum et l'interpolation des points est faite à travers toutes les échelles.

2.1.8 Calcul des descripteurs

Une fois les points détectés, il est primordial de pouvoir les caractériser. Pour un point de l'espace projeté sur l'image il est important que la représentation soit :

- suffisamment discriminante pour permettre de distinguer des points ne représentant pas le même point de l'espace.
- suffisamment générique afin de pouvoir reconnaitre des points représentant un même point de l'espace malgré certaines altérations. Ces

altérations peuvent être des variations de la scène telles que l'illumination, les distorsions liées au changement de point de vue, aux rotations etc.

Il s'agit dans les faits de trouver un équilibre entre ces deux points.

Il existe plusieurs familles de descripteurs. Nous nous concentrerons dans notre cas sur les descripteurs locaux. En effet ce sont ceux qui caractérisent des points de l'image. Parmi les descripteurs locaux nous aborderons ceux basés sur les histogrammes (ceux-ci ont eu un franc succès sur la dernière décennie), ceux qui sont appris notamment grâce aux réseaux neuronaux convolutifs (CNN en anglais) puis les descripteurs binaires (qui ne sont pas aussi robustes que ceux basés sur histogramme mais présentent l'avantage de pouvoir être calculés rapidement).

Descripteurs basés sur histogrammes

Les descripteurs basés sur histogrammes restent les plus couramment utilisés. L'idée sous-jacente consiste à dresser l'histogramme d'une des caractéristiques du voisinage d'un point, le plus souvent les intensités de pixels ou les orientations de gradient sont choisis. Dans [SB91] nous retrouvons les premières instances de ce type d'idées. Nous pouvons extraire deux familles, les histogrammes de pixels [SB91], puis ceux basés sur histogrammes de gradient [Low99, BTVG06, MS05]. La seconde famille est la plus répandue, car elle fournit en effet de meilleures performances.

Histogrammes d'intensité de pixel

Ceux-ci permettent d'avoir une idée de la distribution des valeurs de pixel dans le voisinage d'un point. Ce processus permet de donner un résumé de l'information contenue dans l'image. Dans ce contexte l'histogramme n'est pas appliqué à l'image entière, mais dans un voisinage du point détecté. L'histogramme se dresse ainsi :

$$h(i) = \frac{1}{card(\Omega)} \Sigma \mathbb{1}(I(x) = i) \text{ avec } i \in \text{Q=[0 ;255]} \tag{2.27}$$

Cela revient au final à compter le nombre de pixels ayant la même valeur.

Pour comprendre les forces et les faiblesses de ce type de descripteur, il faut s'intéresser à ces invariances. Un histogramme local est clairement invariant à des transformations géométriques telles que des rotations. Hélas l'orientation étant déjà estimée lors de la détection, cette invariance n'est que de peu d'utilité. De plus un histogramme, est aussi invariant a des transformations

plus agressives telles que des permutations aléatoires des pixels. Cela est clairement contre-productif pour caractériser la structure d'un voisinage. Enfin, un histogramme de niveau de gris, n'est pas invariant aux changements de contraste, ce qui est encore un autre défaut.

Histogrammes de gradient

Les histogrammes de gradients proposent une analyse du voisinage d'un point basée sur la structure qui l'entoure. Ils sont basés sur le calcul de variations d'une fonction dans un voisinage donné. Souvent, pour une image, la fonction sur laquelle les variations sont opérées est la fonction d'intensité des pixels comme illustré dans la figure 2.7.

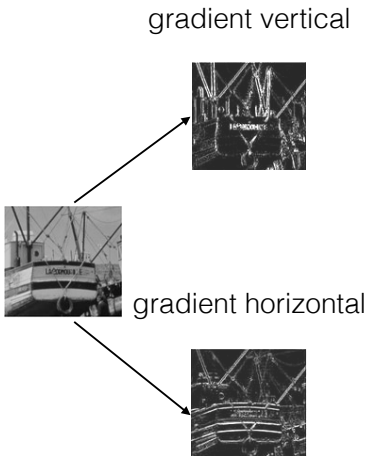


FIGURE 2.7 – représentation des gradients horizontaux et verticaux

Les histogrammes peuvent ainsi permettre d'extraire des informations ciblées dans un voisinage donné. Plus précisément ces derniers permettent d'extraire des informations sur la silhouette (ou la forme) d'un objet en question.

SIFT

En plus d'un détecteur, Lowe propose dans son article [Low99] un descripteur basé sur un histogramme de gradients. Ainsi, une fois que la position et l'échelle d'un point sont détectées, le processus de description débute. Pour cela la méthode est basée sur trois étapes. Dans un premier temps les orientations et amplitudes de gradients des points, dans un voisinage du point détecté, sont calculées. Toutes ces orientations de gradients sont stockées dans un histogramme de 36 intervalles, qui sera pondéré par l'amplitude du

gradient. De cet histogramme sont extraites la ou les orientations principales. Quand cela est obtenu, la phase de description à proprement parler est entamée. Un histogramme de gradients, de 8 intervalles, basé sur un voisinage de 4×4 zones de 4×4 pixels chacune est calculé. Ce voisinage est placé selon l'orientation du point détecté. Une fois traitée numériquement la concaténation des histogrammes de chaque zone forme un descripteur. On peut remarquer deux contributions majeures dans cette procédure qui répondent aux insuffisances d'un histogramme de niveau de gris. Premièrement, le découpage en sous-régions, permet de contrecarrer les invariances aux permutations de pixels. D'autre part, le calcul de statistiques sur les gradients permet d'obtenir une meilleure invariance aux changements de contraste.

Le premier histogramme d'orientations des gradients est dressé sur un voisinage du point détecté. Il y a en tout un intervalle tous les 10° soit 36 intervalles. Voici la formule des orientations locales en chaque point (x, y) :

$$\Theta(x, y) = \arctan\left(\frac{L(x, y+1, \sigma_0) - L(x, y-1, \sigma_0)}{L(x+1, y, \sigma_0) - L(x-1, y, \sigma_0)}\right) \tag{2.28}$$

L étant défini comme dans 2.21, (x, y) représentant les coordonnées du point détecté, et σ_0 l'échelle correspondante. Afin de prendre en compte l'échelle du point (x, y) détecté l'histogramme est pondéré par un filtre gaussien qui est une fois et demi plus grand que l'échelle de (x, y) d'une part. D'autre part l'histogramme est aussi pondéré par l'amplitude m calculé ainsi :

$$m(x, y) = \sqrt{(L(x, y+1, \sigma_0) - L(x, y-1, \sigma_0))^2 + (L(x+1, y, \sigma_0) - L(x-1, y, \sigma_0))^2} \tag{2.29}$$

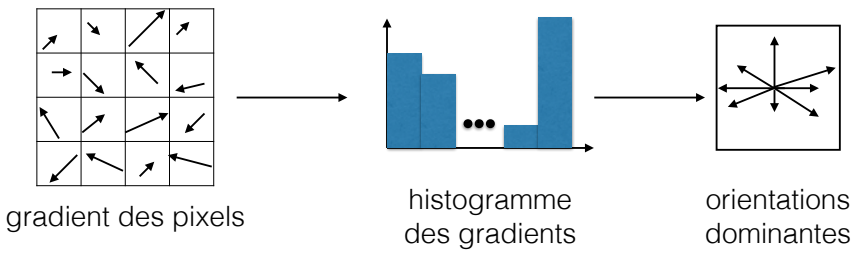


FIGURE 2.8 – représentation de l'histogramme des orientations

Dans la figure 2.8 sont résumées les étapes permettant l'obtention des orientations principales d'un point détecté. Pour chaque point du voisinage choisi une orientation Θ et une amplitude est calculée, comme vu précédemment. Ensuite avec ces orientations un histogramme sur 36 intervalles est dressé, c'est

ce que nous voyons sur la deuxième étape dans la figure. Cet histogramme est pondéré à la fois par l'amplitude du gradient en question ainsi que par une convolution gaussienne, cela représente l'étape finale de la figure. À noter que l'on voit sur la figure seulement 8 intervalles par soucis de clarté, dans les faits il y en a 36.

Dans l'histogramme final, celui à 8 intervalles, toutes les orientations ayant une amplitude supérieure à 80% de la valeur maximale sont retenues pour former un nouveau point d'intérêt. Chacun de ces points ayant la même échelle, et la même localisation (x, y) mais avec une orientation θ différente.

Pour avoir le meilleur rendement entre performances et temps de calcul Lowe conseille d'utiliser un voisinage de 16×16 pixels autour d'un point, regroupés en 4×4 zones de 4×4 pixels chacune. Pour assurer l'invariance aux rotations, la grille placée sur le voisinage du point (x, y) est tournée d'un angle égal à l'orientation θ du point détecté. Dans chacune des 16 zones, un histogramme basé sur 8 intervalles (espacés de $\pi/4$ chacun) est dressé. Les descripteurs sont la concaténation de ces histogrammes. Ils sont ensuite seuillés (les valeurs inférieures à 0.2 sont ramenées à 0) pour enfin être normalisés. Lowe dans son article démontre que son approche obtient des performances remarquables en montrant que même pour des changements de point de vue supérieurs à 50°, 50% des correspondances prédites sont correctes.

SIFT reste une méthode de référence et plusieurs méthodes essayent d'obtenir des performances similaires en réduisant le temps de calculs. Pour cela ils approximent l'approche introduite par SIFT.

SURF

Influencé par SIFT, Bay propose dans [BTVG06] une méthode de description locale nommée SURF (speeded up robust features). Dans leur approche les auteurs combinent le recalage de la grille et la mise en place d'un histogramme de gradient orienté. Pour cela, dans un premier temps, ils cherchent à détecter rapidement l'orientation de la grille. À cette fin ils appliquent des ondelettes de Haar, en passant par des images intégrales. Ces ondelettes servent à estimer des gradients verticaux et horizontaux.

Afin de trouver l'angle de recalage les auteurs cherchent dans un arc d'angle $\pi/3$, la répartition majoritaire des réponses des ondelettes. Le principe est illustré dans la figure 2.9, dans un premier temps une zone circulaire de taille $6s$ (où s est l'échelle du point détecté) est choisie autour du point, ensuite les ondelettes de Haar y sont appliquées pour estimer l'angle de recalage.

Dans le graphique, les ondelettes de Haar sont constituées d'une partie noire qui vaut -1 et une blanche qui vaut +1. Les ondelettes sont de taille $4s$.

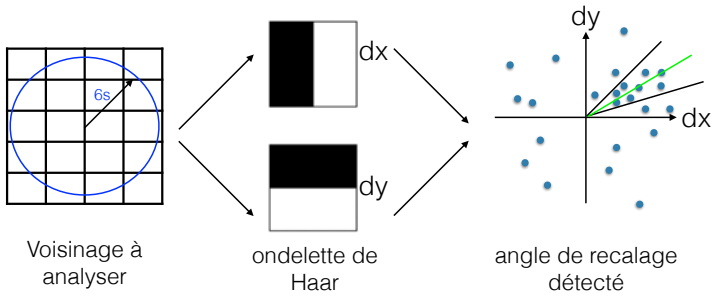


FIGURE 2.9 – Représentation de l'histogramme des orientations pour surf

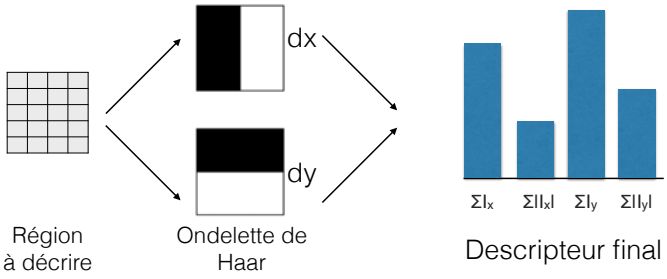


FIGURE 2.10 – Illustration du procédé visant à extraire les différentes composantes du descripteur SURF

Pour décrire le voisinage du point en question, les ondelettes horizontales et verticales sont appliquées au sein des pixels d'une région. Pour chacune des composantes, chacune des composantes sont sommées ainsi que leurs normes. Le processus de description est illustré dans la figure 2.10. Il y a chez SURF 4×4 régions elles même divisées en 5×5 sous régions. Une analyse en ondelette sur chaque région est effectuée. Le descripteur est la concaténation de ces éléments.

GLOH

En 2005 les auteurs de [MS05] ont proposé une méthode permettant d'améliorer SIFT. L'idée est de considérer un plan circulaire constitué de 17 zones, afin de calculer un gradient orienté dans chacune d'entre elles. La figure 2.11 illustre chacune des zones. Ces zones sont caractérisées par 3 paramètres radiaux (en noir sur la figure) et 8 paramètres angulaires (en vert sur la figure).

Ici 16 classes sont possibles pour les orientations (intervalles de $\pi/8$), l'histogramme de gradients orientés est effectué dans chacune des 17 zones. L'histogramme final est constitué de 272 données seuillées et normalisées. Les expériences montrent que la démarche améliore SIFT en terme de robustesse et de nombre de matches corrects trouvés sur plusieurs types de scènes différentes.

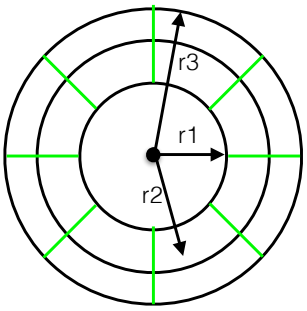


FIGURE 2.11 – Représentation du masque utilisé par GLOH

Descripteurs basés sur réseaux neuronaux convolutifs

De nombreuses applications des réseaux neuronaux convolutifs se sont montrées très compétitives lors des dernières années, dépassant avec aisance l'état de l'art pour les tâches considérées. Les domaines d'application s'étendent de la reconnaissance d'objets en passant par l'identification d'images et la prédiction de cartes de profondeur [SRASC14, SZS+13, TYRW14, EPF14]. Mais de plus en plus, les méthodes de ce type s'élargissent en s'appliquant à l'apprentissage de descripteurs et à l'estimation des fonctions de similarité permettant de les comparer.

Il convient dans un premier temps de décrire les réseaux de neurones convolutifs. Nous allons dans la suite décrire quelques unes des contributions ayant abouti à la construction de descripteurs basés sur les réseaux neuronaux convolutifs.

Introduction aux réseaux de neurones convolutifs

Cette partie est une introduction aux réseaux de neurones convolutifs (aussi appelés CNN) et n'a pas vocation à détailler tous les aspects de ce sujet. Le livre de Goodfellow [GBC16] en fait une présentation détaillée. L'idée derrière cette méthode est fortement inspirée de la biologie et de la structure du cerveau. Nous retrouvons ainsi une architecture de traitement multicouche pour les CNN.

Les réseaux de neurones ont été introduits pour la première fois dans [MP43] en 1943. D'un point de vue informatique un neurone effectue une somme pondérée de ses entrées et y applique une fonction d'activation. Le poids associé aux entrées peut être appelé poids synaptique, on retrouve le lien avec la biologie. La fonction d'activation est définie ainsi :

$$\begin{aligned} \mathbb{R} &\longrightarrow [0,1] \\ x &\longmapsto f(x) \end{aligned} \tag{2.30}$$

L'idée derrière les réseaux de neurones est de placer les neurones en parallèle, on appelle cela des perceptrons. Des détails sur cette méthode et ses limitations sont données dans [MS69]. Très vite ces perceptrons ont été placés sur différentes couches [Bry75, RMG+88]. Chaque neurone possédant ces paramètres (les poids synaptiques), le principe est de trouver la combinaison de paramètres idéale afin de réaliser une tâche.

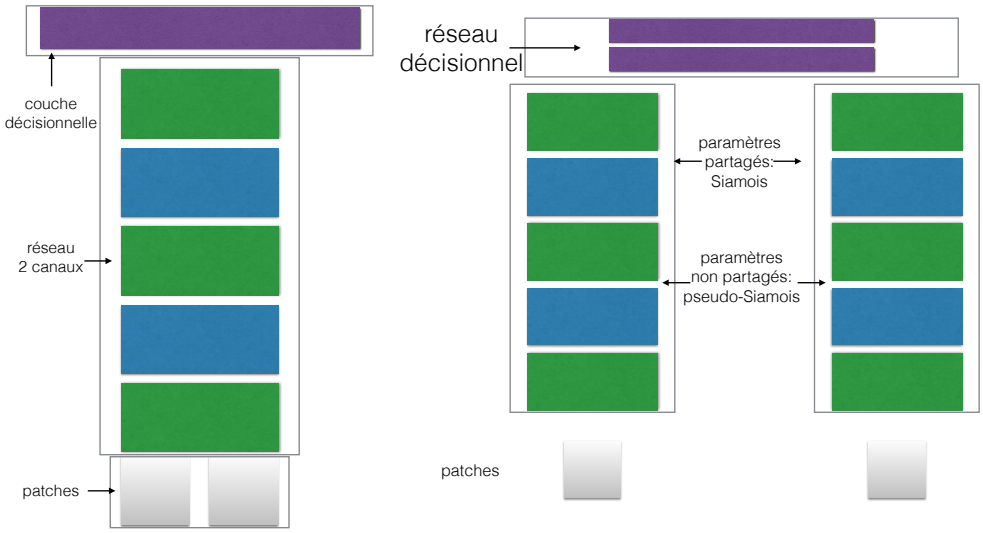
La méthodologie la plus répandue est celle de la propagation et retropropagation de gradients. Dans le cas de réseaux non profonds l'apprentissage des paramètres est effectué de façon supervisée. Ainsi dans un premier temps les entrées sont placées et diffusées à travers les différentes couches du réseau, cette étape correspond à la propagation. À la fin de la propagation nous pouvons disposer de l'erreur entre la sortie obtenue et la sortie souhaitée, et en dresser la dérivée. La dérivée de l'erreur pourra simplement être exprimée en fonction des paramètres du réseau, notamment quand la fonction de réseau est correctement choisie. L'étape de retropropagation de gradients consistera à optimiser les paramètres en remontant de la sortie vers l'entrée afin de réduire l'erreur.

Dans les dernières années les efforts se sont concentrés sur la mise en place de réseaux profonds. Ils présentent l'intérêt d'offrir une meilleure représentation de fonctions de probabilité complexes. Ils ont néanmoins l'inconvénient d'être complexes à optimiser et le problème ainsi posé est non convexe.

Les réseaux de neurones convolutifs rajoutent à cela le fait d'utiliser un poids unique pour tous les signaux entrant dans des neurones faisant partie du même noyau de convolution. La convolution vient du fait que les neurones ont les même paramètres, mais appliqués à des pixels décalés. Chaque noyau de convolution analyse une caractéristique particulière de l'image d'entrée. Cela rend le réseau moins couteux en mémoire, plus performant et robuste aux translations grâce à la convolution.

Deepcompare

Paru en 2015 dans [ZK15], les auteurs y présentent comment utiliser plusieurs architectures connues de CNN afin de construire une fonction de similarité entre patchs. Leur but est d'automatiquement générer une fonction. Cela induit qu'il n'y a aucun schéma manuel de construction d'une telle fonction. Pour ce faire ils se sont posés la question de l'architecture la plus adaptée à cette tâche.



(a) réseau 2-canaux (b) réseau siamois et pseudo-siamois

FIGURE 2.12 – représentation d'architectures connues dans le cadre des réseaux de neurones

Ils introduisent quelques architectures comme les réseaux de neurones siamois, semi-siamois, 2 canaux vus entre autre dans [KSH12, LCTHS88], ils sont représentés dans la figure 2.12. Le point commun de tous ces réseaux est qu'ils mettent en entrée une paire de patchs en correspondance ou en non correspondance. La base d'apprentissage est donc composée de paires avec les informations de correspondance.

L'apprentissage est effectué ainsi :

- L'apprentissage est supervisé. La fonction objectif suivante est établie afin d'optimiser les paramètres :

$$\arg\min_{w} \frac{\lambda}{2}||w||_2 + \Sigma_1^n \max(0, 1 - y_i o_i^{net}) \tag{2.31}$$

où w représente le poids du réseau de neurones et λ est un scalaire permettant de pondérer l'importance de $||w||_2$ dans la procédure d'optimisation. Enfin o_i^{net} est la sortie du réseau et y_i le label correspondant au fait que la paire soit une correspondance ou non. La partie $\Sigma_1^n \max(0, 1 - y_i o_i^{net})$ représente l'attache au donnée. Elle peut être interprétée comme étant une fonction de type hinge-loss grandement exploitée dans le SVM (support machine vector). La partie $\frac{\lambda}{2}||w||_2$ quant à elle est une régularisation.

- La minimisation de l'équation 2.31 est réalisée par l'utilisation de l'algorithme de descente de gradient stochastique moyennée (ASGD) présenté dans [Xu11]. Les paramètres sont initialisés aléatoirement et les modèles sont appris ex nihilo.

Bien que toutes ces approches apportent une amélioration significative, l'approche 2 canaux se démarque particulièrement. Elle se compose assez simplement : une paire de patchs entrant est considérée comme une image a deux canaux. Les couches basses sont une série de convolutions, avec une fonction d'activation ReLU (Rectfied Linear Units, cela apporte de la non-linéarité à la convolution) et de couche max pooling (cela consiste à réduire la dimension d'une image en ne prenant que le maximum des pixels dans une zone, cela à pour effet de faire face au problème d'overfitting). Ce réseau, contrairement au réseau siamois, n'est pas spécifiquement adapté aux images mais est très flexible. Il est rapide à apprendre, mais plus long à exécuter.

Un inconvénient majeur de ces méthodes, d'après les auteurs, reste que l'implémentation GPU (sur une GPU Titan dans Torch [CKF11]) de ces approches est deux fois plus lente que l'implémentation CPU de SIFT [Low99].

MatchNet

Les auteurs de MatchNet dans [HLJ+15] proposent une architecture et une organisation différente pour comparer les patchs. Ils proposent une approche unifiée qui utilise les CNN pour donner une représentation de la paire de patchs permettant de les comparer de façon robuste. Pour cela MatchNet met en place deux réseaux de neurones différents, un pour la description des patchs et un pour leur comparaison.

Le réseau nécessaire à la description est fortement inspiré de AlexNet [KSH12]. Cependant l'architecture qu'ils ont utilisés requiert moins de paramètres. Ils n'utilisent pas de normalisation de la réponse locale (local response normalisation), ni de dropout. Ils ont opté pour la ReLU comme fonction d'activation.

Le réseau utilisé pour la métrique est basé sur 3 couches avec une fonction d'activation ReLu. Sur la troisième couche la fonction d'activation Sofmax est utilisée. En entrée il accueille la concaténation des descripteurs de la paire de patchs que nous voulons comparer. La sortie est composée de deux valeurs dans [0,1], ces valeurs peuvent être interprétées comme étant une condition de correspondance ou non-correspondance.

Pour combiner ces deux réseaux lors de l'apprentissage la structure à deux-tours avec paramètres liés est utilisée. Elle est également mise en œuvre dans

[ZL15]. Afin de pouvoir comparer la description des patchs, il faut que les deux patchs soient passés par le même réseau, ainsi lors de l'apprentissage, la partie description est composée de deux réseaux avec des paramètres similaires tous deux connectés au réseau de comparaison des patchs. Cette structure peut être comparée aux réseaux siamois [BBB+93, CHL05].

Dans MatchNet le maxpooling est aussi utilisé. Via le réseau de neurones convolutif construit le descripteur et la mesure de similarité sont apprises ensemble. L'apprentissage se fait avec une descente de gradient stochastique. La fonction objectif est la suivante :

$$E = -\frac{1}{n}\Sigma_1^n[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{2.32}$$

Ici y_i représente le label indiquant si une paire de patchs est en correspondance (label 1) ou non (label 0). $\hat{y}_i$ est la réponse estimée par le réseau à l'issue de l'activation Softmax. E peut est interprétée comme une entropie croisée, elle sera basse lorsque $\hat{y}_i$ approxime y_i correctement.

Ces méthodes bien qu'offrant des performances, en terme de robustesse, restent assez lentes. Elles sont moins rapides encore que des méthodes de type SIFT. Leur exploitation sur mobile n'est donc pour le moment toujours pas envisageable.

Descripteurs binaires

Les descripteurs binaires ont été considérablement mis en avant durant la dernière décennie. Ces descripteurs ont en effet les deux propriétés suivantes : ils requièrent peu de mémoire et peuvent être calculés efficacement. Dans le cadre d'applications temps réels embarquées sur mobile, ce sont des propriétés de choix. De façon empirique le constat est le suivant, un descripteur binaire usuel nécessite le stockage de 512 bit. Les descripteurs basés sur histogrammes de leur côté nécessitent 512 valeurs de type flottante (et requièrent donc 32 fois plus de mémoire). Afin de réduire la taille en mémoire de ces descripteurs il est possible d'utiliser des méthodes de réduction de dimension ou des méthodes de hashing sur les descripteurs basés sur histogrammes [TCFL13, KS04]. De leur côté, les descripteurs binaires n'ont pas besoin de cette étape puisqu'ils génèrent directement un nombre restreint d'éléments binaires simples (appelés tests). Il en résulte qu'ils sont non seulement économes en mémoire, mais aussi plus rapides à calculer.

Toutefois, les points forts des descripteurs binaires ont un prix. La simplicité des tests ne permet généralement pas de distinguer les points clés aussi efficacement que les descripteurs basés sur des attributs complexes (tels que les

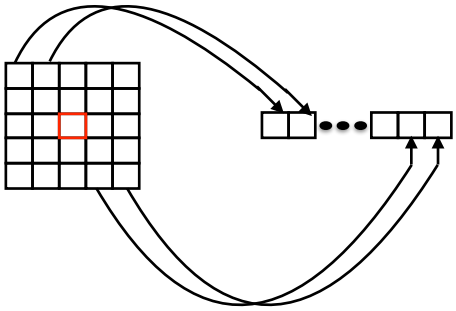


FIGURE 2.13 – Mode de fonctionnement de CENSUS

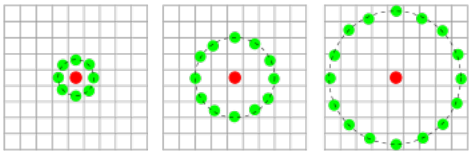


FIGURE 2.14 – Mode de fonctionnement de LBP

histogrammes). Le problème n'est donc que partiellement résolu. Ainsi l'axe de recherche principal dans ce domaine est d'augmenter le potentiel de distinction des descripteurs binaires tout en gardant un bon compromis entre mémoire et temps de calcul. Nous allons ici décrire quelques travaux représentatifs.

Parmi les premières instances concernant les descripteurs binaires figurent CENSUS et LBP [ZW94, OPH96]. Ils sont basés sur la comparaison systématique des pixels dans un voisinage du pixel central. Les deux méthodes diffèrent dans la forme du voisinage : un carré plein pour CENSUS et un anneau pour LBP. De telles procédures génèrent des codes binaires dont la longueur dépend directement de la taille du voisinage. Ainsi afin qu'il soit court et rapide, le descripteur doit être calculé sur une petite région. Cela a pour effet de restreindre leur pouvoir de distinction.

BRIEF [CLSF10] est une approche plus récente qui propose un compromis entre localité et efficacité. Il est construit sur l'idée de locality sensitive hashing (LSH) [IM98] dont l'application principale est la recherche de plus proches voisins. Cette méthode est basée sur la pré-sélection aléatoire de paires de pixels dans un large voisinage. Après cela, le descripteur d'un patch donné est calculé en fusionnant les résultats des comparaisons binaires opérées entre chacune des paires. De cette manière, la taille du voisinage et la longueur du descripteur peuvent être choisies indépendamment (par exemple un descripteur de taille 512 pour un patch de 32). Les auteurs d'ORB [RRKB11] avancent que le mécanisme de sélection doit prendre en compte la distribution des données. Ils proposent un schéma permettant de sélectionner les tests les plus pertinents. Leur approche est de sélectionner des tests décorrélés présentant une variance maximale en se basant sur un algorithme glouton. Dans la section 4.1 nous donnerons une interprétation de cette procédure comme étant une maximisation de la quantité d'information. Ce qui importe est que la variance et la corrélation soient estimées sur des bases de données représentatives. Cette

solution permet d'atteindre un équilibre entre la complexité du descripteur et son pouvoir de distinction en se référant à la distribution des données.

Certains auteurs ont guidé la sélection en s'appuyant sur d'autres principes. Par exemple, dans BRISK [LCS11], un échantillonnage concentrique est réalisé dans un patch de l'image, et des paires sont sélectionnées à partir de chaque paire de pixels qui sont suffisamment proches. Dans la même lignée FREAK [AOV12] construit un échantillonnage concentrique qui mime l'architecture rétinale en densifiant l'échantillonnage au niveau du centre. Ensuite l'algorithme glouton initié par ORB est utilisé pour sélectionner les bonnes paires.

Rendre les descripteurs binaires robustes aux transformations naturelles est également une tâche importante. Par construction les descripteurs basés sur des différences locales sont invariants aux changements de contrastes. D'un autre côté, le bruit est par défaut mal géré. Cela peut être compensé en lissant le patch choisi avant de le décrire. Des tests binaires plus sophistiquées ont aussi été créés tels que LDB [YC12] où les données concernant les pixels sont agrégées dans des grilles. En ce qui concerne les déformations géométriques, leur impact peut être considérablement réduit si l'orientation principale (ainsi que l'échelle) du point détecté est correctement estimée. C'est le cas d'ORB et AKAZE [ANB13]. Plus récemment les auteurs de BOLD [BTM15] ont proposé une alternative où la robustesse est introduite par une sélection en ligne des tests à sélectionner.

2.1.9 Appariement des points 2D des images

Comme le processus de prise de photo peut être assimilé à une projection d'une scène 3D dans un plan 2D, on peut le modéliser selon le modèle sténopé (figure 2.1 page 26). Les smartphones du marché configurés de façon usuelle respectent suffisamment ce modèle.

Plusieurs stratégies peuvent être utilisées pour apparier efficacement des points 2D dans plusieurs images. Dans [LLF05] l'approche est d'utiliser des méthodes d'apprentissages. Les auteurs effectuent une sélection d'un certain nombre de points d'intérêts qui sont souvent vus dans des images prises au préalable. Sur cette base d'images les points qui sont en correspondance sont supposés connus et il est possible d'entrainer un classifieur.

Ce dernier est basé sur un arbre binaire encodant la distribution a posteriori. Chaque classe représente une famille de points 2D représentant le même point de l'espace. Ensuite pour la phase de mise en correspondance l'idée est de se servir du classifieur afin de savoir si le point testé fait partie d'une classe, et si oui laquelle. Dans [OCLF10] la même approche est abordée mais cette fois-ci en

s'affranchissant de la structure d'arbre. L'idée est de trouver des sous-groupes de classes indépendantes au sens des probabilités. Ils appellent ces sous-groupes FERN. Cette approche permet de rendre l'algorithme plus rapide.

Dans d'autres approches il est nécessaire d'extraire des points d'intérêts dans les images, puis d'apparier directement les points les uns aux autres. Pour cela, il est essentiel de disposer d'un descripteur efficace. Les points SIFT sont connus pour être les plus robustes [MS05]. Certaines méthodes basées sur des kd-trees du type de FLANN [ML14, ML09, ML12], sont utilisées afin d'estimer le plus proche voisin d'un descripteur.

2.1.10 Filtrage épipolaire

Une fois que nous disposons de correspondances entre points 2D de deux images, notre désir est d'estimer la matrice essentielle E. La matrice nous permettra ensuite de disposer d'une première estimation de position entre deux caméras comme vu dans 2.1.5. Afin d'estimer E nous pourrions passer par un calcul aux moindres carré comme vu dans la partie 2.1.11, mais ceux-ci ne sont pas robustes aux aberrations. Il existe pour faire à cela des méthodes comme les M-estimateurs, mais nous avons ici trop d'aberrations pour qu'ils soient utilisés. Nous abordons dans cette partie plusieurs alternatives afin de pallier aux aberrations, dont LMDes [LR87], RANSAC [FB81] et certaines extensions de RANSAC par les méthodes a contrario [RDGM10, MMM12a]. Ces méthodes permettent à la fois de filtrer les correspondances aberrantes tout en estimant la matrice E. Les méthodes a contrario offrent une alternative intéressante que nous avons décidé d'exploiter dans le cadre de nos travaux. Le socle commun de toutes ces méthodes étant qu'elles cherchent un ensemble d'éléments correctement liés (les inliers) afin de générer un modèle. Une fois cela effectué, il est possible d'estimer les poses de caméras en se servant des correspondances jugées correctes.

LMedS

Cette méthode consiste à estimer l'erreur médiane des résidus. Cela revient à réduire l'erreur médiane des résidus définie dans l'équation 2.34 page 53. Elle permet d'avoir une estimation directe du modèle, spécialement lorsque le taux de points non-aberrants est supérieur à 50%. Le critère d'optimalité est fonction de la médiane de $||f(\hat{x}_i) - y_i||^2$. Le meilleur tirage sera celui qui donnera l'erreur la plus faible en se basant sur la médiane.

RANSAC

RANSAC est un algorithme permettant de retrouver un modèle, ici E vérifiant l'équation 2.8, en y incluant un nombre fixé de données en entrée. Pour ce faire à partir d'un sous-groupe minimal de données tirées aléatoirement il calcule E. Ici 5 correspondances suffisent pour déterminer E. Il vérifie au fur et à mesure si d'autres correspondances vérifient ce modèle à une déviation près (le seuil choisi étant un paramètre de l'algorithme).

Si tel est le cas il les rajoute à l'ensemble des données cohérentes ou inliers. Sinon elles sont exclues et considérées comme outliers. Si à la fin de ce processus il y a plus de N correspondances, N valeur fixée au préalable, alors le modèle est accepté. Sinon le processus est réitéré en tirant un autre sous-groupe aléatoirement.

Si plusieurs modèles sont fournis alors RANSAC choisit celui qui a inclus le plus de données (ici de correspondances). En sortie de RANSAC il fournit donc la matrice E et des correspondances entre points 2D des images respectant la géométrie épipolaire à un seuil donné. Cela rajoute un niveau de confiance aux correspondances dont nous disposons déjà.

RANSAC avec une approche a contrario

Les méthodes a contrario [RDGM10, MMM12b] visent a automatiser le choix du seuil de RANSAC. Elles se basent sur le principe de Helmholtz, selon lequel aucune structure n'apparait dans le bruit. Par exemple, si l'on tire un ensemble de points indépendamment et de façon uniforme sur un carré, la probabilité qu'un nombre k d'entre eux soit bien alignés est extrêmement faible dès lors que k est grand. Ce principe intuitif a été formalisé par [DMM07] dans le cadre des tests d'hypothèse. En résumé, le cadre qu'ils proposent, requiert une hypothèse nulle (encore dénommée hypothèse de fond), ainsi qu'une statistique $t(x, M)$ révélatrice de l'adéquation entre le point x et le modèle M. Dans l'exemple précédent (recherche d'alignement) la statistique peut-être la distance entre le point et le modèle correspondant à la droite envisagée.

Comme dans RANSAC, on pourrait se contenter d'appliquer un seuil fixe à la statistique $t(x_k, M)$ pour déterminer la liste des points x_k en accord avec le modèle M. L'apport de la méthode à contrario consiste à tester toutes les valeurs de seuils possibles et à calculer pour chacun d'entre eux une mesure appelée nombre de fausses alarmes (NFA) caractérisant la plausibilité de la configuration des éléments vérifiant le modèle (inliers) sous l'hypothèse nulle. Moins cette configuration d'inlier est plausible sous l'hypothèse nulle, plus elle est caractéristique de la présence d'une structure non accidentelle. Evidemment, l'hypothèse nulle est souvent choisie en tenant compte de la

Algorithme 1 : Recherche de correspondances correctes entre paires d'images, et calcul de la matrice essentielle

Entrées : 2 images,matrice intrinsèque K
Sorties : matrice essentielle, correspondances entre images

```
1 correspCalc()          // Calcul des correspondances basé SIFT
2 ACRANSAC()     // Filtrage géometrique des correspondances
```

1 Procédure *correspCalc()*
- **Entrées** : images
- **Sorties** : ensemble de correspondances éstimées par un descripteur
- **2** Détection de points
- **3** Description de chacun de ces points
- **4** Mise en correspondance des descripteurs

1 Procédure *ACRANSAC()*
- **Entrées** : ensemble de correspondances éstimées par un descripteur,nombre maximal de tirage,K
- **Sorties** : matrice essentielle, correspondances réelles entre images
- **2** Échantillonage aléatoire de points Sélection de correspondances correctes minimisant le nombre de fausses alarmes
- **3** Validation du modèle
- **4** Optimisation du modèle

complexité des calculs de NFA, ce qui peut en partie la rendre artificielle. Par exemple le calcul du NFA pour un problème de correspondance 2D-2D est donné dans [MMM12b] sous une hypothèse nulle de tirage aléatoire uniforme dans les deux images.

En pratique, l'algorithme AC-RANSAC est proche de RANSAC. On commence par choisir un sous-groupe minimal pour lequel on détermine le modèle putatif correspondant. Pour ce modèle M, on classe les données par statistique croissante $t(x_1, M) \leq \cdot \leq t(x_n, M)$. On ne considère pour seuils τ que les valeurs distincte des $t(x_k, M)$, et pour chacun d'entre eux on calcul le NFA. On associe au modèle putatif considéré, le NFA minimal trouvé et le seuil correspondant, noté $\tau(M)$. Enfin on choisit parmi tous les modèles, celui ayant le NFA minimal, et on ne l'accepte que si ce NFA est plus petit qu'une valeur préétablie ayant typiquement peu d'influence sur les performances. Les inliers sont alors déterminés grâce au seuil $\tau(M)$ associé au modèle sélectionné. L'approche associée est résumée dans l'algorithme 1.

2.1.11 Méthodes d'estimation connues

En vision par ordinateur, un des problèmes à résoudre peut être vu comme la recherche d'un ensemble de valeurs d'entrées $x = x_1, \ldots, x_n$ tel que :

$$f(x) = y \tag{2.33}$$

où f est une fonction qui modélise le problème qui est étudié et $y = y_1, \ldots, y_p$ est un ensemble de mesures observées. Dans les faits, dus aux erreurs de mesure ou encore aux approximations faites lors des calculs, l'égalité 2.33 ne peut pas être obtenue. C'est pour cela que l'erreur résiduelle est prise en compte, elle consiste à faire la différence entre les mesures et le modèle appliqué aux paramètres estimés on la définit ainsi :

$$r(x) = f(x) - y \tag{2.34}$$

La résolution du problème revient ainsi à minimiser l'erreur résiduelle.

Nous sommes ici dans un cas où nous disposons de correspondances entre images, elles ont été fixées suite à l'étape de détection et description vu dans les sections 2.1.7 et 2.1.8. Dans la section 2.1.10 nous avons vu comment filtrer les correspondances et enlever les aberrations. Il nous reste maintenant à estimer, grâce aux correspondances filtrées, la position du dispositif de capture d'image.

Plusieurs méthodes existent afin de résoudre ce genre de systèmes. Ces méthodes peuvent aller des moindres carrés, rapides mais peu robustes au bruit (le bruit ici étant les correspondances effectuées à tort), à des méthodes non linéaires. Celles-ci peuvent être des méthodes comme la descente de gradient ou encore l'algorithme de Levenberg-Marquardt. L'algorithme de Levenberg-Marquardt combine les avantages de la descente de gradient et de l'algorithme de Gauss-Newton. Les parties qui suivent détaillent ces approches, notre modèle étant non-linéaire la deuxième approche nous intéresse le plus. Mais par souci de clarté nous commençons par expliquer le cas où la fonction considérée est linéaire.

Moindres carrés linéaires

Dans le cas où f est une fonction linéaire, il existe une matrice A pour laquelle pour tout x : $f(x) = Ax$. Deux cas peuvent être considérés, les cas homogènes et non homogènes.

Le cas homogène correspond au cas où y est nul. Résoudre un tel système, au sens des moindres carrés, revient à minimiser le critère :

$$\varepsilon(x) = ||Ax||^2 \tag{2.35}$$

En prenant pour $||x|| = 1$, la solution du système ainsi posée est donnée par le vecteur singulier associé à la plus petite valeur singulière. Usuellement ce vecteur est obtenu en effectuant une décomposition en valeurs singulières (SVD) :

$$A = U\Sigma V^T \tag{2.36}$$

La solution est ici donnée par la colonne de V associée à la plus petite valeur singulière.

L'autre cas, lorsque $y \neq 0$, correspond au cas où le système est dit non homogène. Dans ce cas, la formulation au sens des moindres carrés de ce problème reviendra à minimiser la fonction suivante :

$$\varepsilon(x) = ||Ax - y||^2 \tag{2.37}$$

La solution sera donnée par les équations normales induites par :

$$\begin{aligned} Ax &= y \\ A^T Ax &= A^T y \\ x &= (A^T A)^{-1} A^T y \end{aligned} \tag{2.38}$$

L'opérateur $(A^T A)^{-1} A^T = A^+$ est appelé pseudo-inverse de A.

Méthode de résolution non linéaires

Nous nous plaçons ici dans le cas où la fonction f est non linéaire. Dans la plupart des cas, un tel problème est résolu itérativement. Le principe sous-jacent pour ce type de méthodes est de considérer que, localement, la fonction à optimiser est linéaire. L'idée est d'itérativement réduire l'erreur résiduelle. Pour cela il sera nécessaire de trouver le pas et la direction adéquate.

Nous cherchons ainsi à estimer le paramètre $\hat{x}$ minimisant $||r|| = ||f(\hat{x}) - y||$. Pour ce faire, une estimation initiale de la solution x_0 sera posée. Cette estimation est itérativement modifiée pour converger vers une solution du problème. Pour cela à chaque étape k, le but sera de calculer un pas δ_k vérifiant les conditions suivantes :

$$\begin{cases} x_{k+1} = x_k + \delta_k \\ ||f(x_{k+1}) - y|| \leq ||f(x_k) - y|| \end{cases} \tag{2.39}$$

En procédant ainsi nous nous assurons de la décroissance de l'erreur. Nous avons donc la convergence de la suite $(x_k)_{k\in(N)}$ vers un minimum local.

Le pas δ_k peut être interprété comme suivant une direction d_k avec une amplitude α_k. Les méthodes de résolution non linéaires se différencient sur la stratégie choisie afin de déterminer la direction et l'amplitude du pas. Nous détaillons dans ce qui suit certaines méthodes usuellement utilisées en vision par ordinateur. Ces méthodes sont :

- La descente de gradient
- La méthode de Gauss-Newton
- Levenberg-Marquardt qui combine les deux précédentes

La description détaillée de ces méthodes d'optimisation peut être trouvée dans [HZ03].

2.1.12 Reconstruction 3D

Dans ce qui a précédé nous avons vu comment extraire les données pertinentes d'images pour la 3D. Nous avons vu dans la section 2.1.2 comment une caméra est mathématiquement représentée. Dans la section 2.1.5 nous avons abordé la géométrie épipolaire, qui donne des informations sur les contraintes géométriques liées à la représentation d'un point de l'espace sous différentes vues. Sachant que dans le cadre de la géométrie épipolaire (mais aussi pour le calcul de la pose de la caméra que nous verrons dans cette section) des correspondances sont nécessaires, nous avons abordé dans les sections 2.1.7 et 2.1.8 les notions de détection et description de points dans une image. Les manières de filtrer les correspondances ainsi obtenues ont été vues dans la section 2.1.10. Enfin nous avons aussi introduit certaines des méthodes d'estimation les plus connues dans la section 2.1.11.

Nous pouvons maintenant adresser la question suivante : Comment, à partir des informations 2D que nous possédons, retrouver les positions relatives des points 3D d'une scène ainsi que de leur caméra ?

Pour y répondre nous abordons, dans un premier temps, les méthodes connues pour construire un nuage de points en 3D. Ensuite nous verrons comment calculer la position d'une caméra en ayant connaissance de liens entre la 2D et la 3D. Nous concluons sur la méthode des ajustements de faisceaux, permettant de raffiner les positions relatives des points et des caméras.

Reconstruction d'un nuage de points 3D

Elle est aussi communément appelée triangulation. Le principe de la reconstruction d'un nuage de point 3D est de retrouver la position 3D, d'un point de l'espace, à partir de ces observations 2D sur le plan focal d'au moins deux caméras dont la pose est connue. À ce stade, supposer la pose d'une caméra connue n'est pas invraisemblable. En effet sachant que des correspondances 2D entre images sont connues, il est possible d'estimer la matrice essentielle. Nous avons vu dans la partie 2.1.5 que le calcul de la pose d'une paire de caméras est possible à partir de la matrice essentielle.

Dans le cas idéal où il n'y a aucun bruit, résoudre cela est simple. Il faut avant tout disposer du plan focal de deux caméras c_1 et c_2 et de leur centre optique o_1 et o_2. Ensuite en ayant la connaissance des projections x_1 et x_2 (respectivement sur c_1 et c_2) d'un point de l'espace X sur chacun des plans focal, l'idée sera de tracer la droite d_1 liant o_1 et x_1 ainsi que d_2 liant o_2 et x_2. L'intersection de d_1 et d_2 sera le point de l'espace X en question. La figure 2.15 illustre cela. Dans les faits il y a du bruit dans les données, et les droites ne s'intersectent pas. Plusieurs méthodes ont été mises en place afin de faire face à cela.

Méthode du point milieu

Une première approche consiste à prendre l'intersection entre le plan médian aux deux droites et la droite reliant la projection de d_1 sur d_2 à la projection de d_2 sur d_1. Cette approche se nomme la méthode du point milieu, elle est l'une des plus rapides. Lorsqu'il y a plus de 2 caméras, on applique cette approche pour chaque paire de caméras. Le barycentre des points ainsi calculés sera l'estimation du point de l'espace X.

Transformation linéaire directe

Une deuxième approche est la méthode appelée transformation linéaire directe (DLT). Son introduction est attribuée à [Sut74], qui est du domaine de la photogrammétrie. Une implémentation pour le calcul de point 3D est donnée dans [HZ03]. Appliquée à notre problème la DLT peut s'exprimer de façon naturelle. La DLT consiste à résoudre des équations du type :

$$x_k \propto Ay_k \text{ où } k \in [1, N],\ N \in \mathbb{N} \tag{2.40}$$

x_k,y_k sont des vecteurs connus. A est une matrice où se trouvent les inconnues. Dans cette équation les éléments de droite et de gauche sont proportionnels. Une façon de les résoudre est de passer par leurs coordonnées

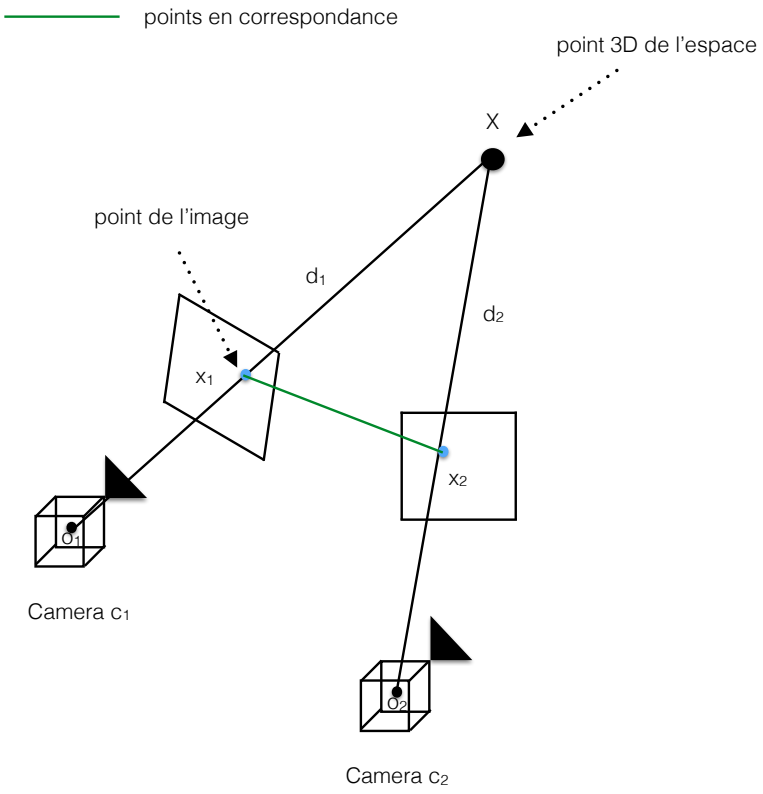


FIGURE 2.15 – Triangulation d'un point de l'espace

homogènes, puis d'utiliser des techniques standard de résolutions comme la SVD.

En disposant des deux projections x_1 et x_2 du point X sur les caméras c_1 et c_2 nous avons :

$$\begin{aligned} x_1 &\propto P_1 X \\ x_2 &\propto P_2 X \end{aligned} \tag{2.41}$$

où P_1 et P_2 sont les matrices de projection de c_1 et c_2. Les deux équations figurant dans 2.41 peuvent être combinées en une équation du type $AX = 0$, qui est linéaire en A. Dans [HZ03] ils proposent d'éliminer le facteur d'échelle homogène en effectuant un produit vectoriel cela donne 3 équations formant un système de rang 2. Voici leur expression pour la caméra c_1 :

$$\begin{aligned} x_1(1)(P_{1,3}^T X) - (P_{1,1}^T X) &= 0 \\ x_1(2)(P_{1,3}^T X) - (P_{1,2}^T X) &= 0 \\ x_1(1)(P_{1,2}^T X) - x_1(2)(P_{1,1}^T X) &= 0 \end{aligned} \tag{2.42}$$

ici $P_{i,j}$ représente la ligne j de la caméra i. Dans le contexte que nous décrivons ici, avec deux caméras et un point de l'espace, la matrice A peut s'exprimer comme suit :

$$A = \begin{bmatrix} x_1(1)P_{1,3}^T - P_{1,1}^T \\ x_1(2)P_{1,3}^T - P_{1,2}^T \\ x_2(1)P_{2,3}^T - P_{2,1}^T \\ x_2(2)P_{2,3}^T - P_{2,2}^T \end{bmatrix} \tag{2.43}$$

Nous pouvons aisément imaginer que si nous avons d'autres projections x_i sur des caméras c_i, la méthode se généralise en concaténant ces valeurs dans A. Ce système peut être résolu en réalisant une SVD sur la matrice A. La solution sera le vecteur associé à la plus petite des valeurs singulières de A.

Méthode optimale

Dans cette méthode le but est de trouver un couple de points $(\hat{x}_1, \hat{x}_2)$, dans les caméras c_1,c_2, qui soit le minimum global de la fonction de coût qui suit :

$$C(x_1, x_2) = d(x_1, \hat{x}_1)^2 + d(x_2, \hat{x}_2)^2 \tag{2.44}$$

Nous ajoutons à la fonction de coût 2.44 la contrainte épipolaire suivante : $\hat{x}_1^T F \hat{x}_2 = 0$. Résoudre $\arg\min_{\hat{x}_1, \hat{x}_2} C(x_1, x_2)$ revient à chercher la paire de points de c_1 et c_2 :

- Les plus proches de x_1 et x_2.
- Vérifiant la contrainte épipolaire.

L'avantage d'une telle approche est ainsi de raffiner encore la précision des points mis en correspondance afin de les faire correspondre à la géométrie épipolaire. Ainsi les droites d_1 et d_2 entreront en intersection.

L'algorithme de Levenberg-Marquardt vu dans la section 2.1.11 peut être utilisé pour minimiser de telles fonctions de coûts.

Une autre approche consiste à tracer des lignes épipolaires sur les deux plans focaux correspondant aux caméras. Dans cet algorithme une ligne épipolaire l_i est choisie dans l'image c_i avec un paramètre t, la ligne épipolaire correspondante dans c_2 est calculée. Suite à cela, la distance entre x_i et la ligne épipolaire l_i dans chacune des deux images est calculée. L'équation 2.44 devient ainsi :

$$\arg\min_t C(x_1, x_2) = d(x_1, l_1(t))^2 + d(x_2, l_2(t))^2 \tag{2.45}$$

Minimiser ce problème revient à résoudre un polynôme de degré 6.

Estimation de la pose d'une caméra

Nous nous plaçons dans cette section dans le cas où les correspondances entre des points 3D de l'espace $\{X_j\}_{j\in[1,N]}$ et des points 2D $\{x_j\}_{j\in[1,N]}$ $(N \in (N))$ du plan focal image d'une caméra c_i sont connues. La connaissance de tels liens nous permet de calculer la pose de la caméra, dans le repère fixé par les points 3D pris en compte. Dans le cas où il n'y a pas de bruit dans les données, et où la matrice de calibration n'est pas connue, six correspondances entre points 3D et 2D sont nécessaires. Si la calibration est connue, seulement trois correspondances suffisent pour calculer la pose d'une caméra. Le calcul de pose dans un tel contexte est rapide et ne pose pas d'ambiguïté sur l'échelle, dans la mesure où les points 3D nous donnent toutes les informations nécessaires. Nous abordons dans cette partie les deux méthodes les plus utilisées dans l'état de l'art à savoir : le calcul par transformation linéaire directe, et par formulation PNP (perspective-N-point) du problème.

calcul de la pose par transformation linéaire directe

De la même manière que nous l'avons vu dans la section précédente, la DLT peut être exploitée afin de résoudre le problème du calcul de la pose [Fau93, HZ03]. Dans ce cas chaque correspondance 3D-2D donne deux équations linéairement indépendantes comme suit, pour un point x_i et un point de l'espace X_i :

$$\begin{aligned} x_i(1) &= \frac{p_{11}X_i(1) + p_{12}X_i(2) + p_{13}X_i(3) + p_{14}}{p_{31}X_i(1) + p_{32}X_i(2) + p_{33}X_i(3) + p_{34}} \\ x_i(2) &= \frac{p_{21}X_i(1) + p_{22}X_i(2) + p_{23}X_i(3) + p_{24}}{p_{31}X_i(1) + p_{32}X_i(2) + p_{33}X_i(3) + p_{34}} \end{aligned} \tag{2.46}$$

où les p_{ij} sont les coefficients de la matrice de projection P définie dans l'équation 2.11. Elle s'écrit ainsi :

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \tag{2.47}$$

En posant p le vecteur défini par p=$[p_{11}\ p_{12}\ \ldots\ p_{34}]$ et A la matrice ainsi construite :

$$A = \begin{pmatrix} X_i(1) & X_i(2) & X_i(3) & 1 & 0 & 0 & 0 & 0 & -X_i(1)x_i(1) & -X_i(2)x_i(1) & -X_i(3)x_i(1) & -x_i(1) \\ 0 & 0 & 0 & 0 & X_i(1) & X_i(2) & X_i(3) & 1 & -X_i(1)x_i(2) & -X_i(2)x_i(2) & -X_i(3)x_i(2) & -x_i(2) \end{pmatrix} \tag{2.48}$$

Nous pouvons exprimer l'équation 2.46 par une équation de la forme $Ap = 0$, que la DLT permet de résoudre. La solution sera, comme vu précédemment, le vecteur propre correspondant à la plus petite valeur propre de la décomposition en SVD de A.

Dans les faits la question de connaître, ou non, la calibration peut avoir un impact considérable sur l'estimation de la position d'une caméra dans une scène. Dans certaines configurations les erreurs d'estimations peuvent être dûes au bruit ou à la géométrie de la scène.

Savoir extraire la matrice de projection normalisée $\hat{P}$, vue dans l'équation 2.11, peut s'avérer nécessaire dans certaines applications. Un exemple classique est la localisation 3D d'une caméra. Dans ce contexte il est nécessaire de savoir séparer les paramètres intrinsèques des paramètres extrinsèques. Dans le cas où la matrice de calibration K est connue cela est simple, il suffit de faire l'opération : $\hat{P} = K^{-1}P$. Afin de s'assurer que la rotation R (trois premières colonnes de P) est bien une rotation des corrections peuvent être effectuées comme vue dans [Zha00].

Néanmoins, avoir cette connaissance de la calibration n'est pas toujours possible. Dans le cas non calibré, il existe des méthodes telles que la décomposition RQ détaillée dans [HJ12]. Elle consiste à décomposer une matrice M en deux matrices Q, qui est orthogonale, et R, qui est triangulaire supérieure. Cette décomposition est vérifiée pour tout M inversible, sinon Q sera une matrice unitaire. Ainsi nous aurons avec M inversible :

$$M = QR \tag{2.49}$$

Nous cherchons plutôt à obtenir $P_{3\times3} = KR$, où $P_{3\times3}$ correspond à la matrice 3×3 que nous pouvons extraire des trois premières colonnes de P. Dans cette relation K est triangulaire supérieure et R est une matrice de rotation. Ce que nous voulons faire est donc d'obtenir une décomposition RQ de $P(1:3, 1:3)$. Cela est fait à partir de la décomposition QR en opérant des transformations sur la matrice $P(1:3, 1:3)$ afin de retrouver la matrice de calibration K et de rotation R. La position t peut en être directement retrouvée.

La méthode Perspective-n-Point (PNP)

Lorsque nous disposons d'un ensemble de points 3D dans l'espace et de leur projection dans l'image, le problème visant à trouver la position d'où la caméra voit la scène se nomme Perspective-n-Point (PNP). Elle consiste à trouver la pose permettant de réduire l'erreur de reprojection entre le point de l'espace et son correspondant 2D dans l'image. Ce problème revient à minimiser la fonction de coût suivante :

$$\arg\min_{P} \Sigma_{j=0}^{N} ||x_j - PX_j|| \tag{2.50}$$

Très souvent dans ce genre de méthodes il est supposé que la caméra est déjà calibrée. Le problème quand $N = 3$ est nommé P3P, c'est celui le plus couramment résolu. Plusieurs méthodes de résolution de ce problème ont été proposées notamment dans [HLON94, QL99].

La méthode suivie dans notre approche est la suivante. La relation liant un point X_i et un point x_i de la frame est pour tout i :

$$P_{frame} X_i \propto x_i \tag{2.51}$$

x_i étant en coordonnées homogènes.

La matrice P_{frame} est de taille 3×4, elle possède donc 12 éléments. Nous considérons que notre caméra n'est pas calibrée, et recherchons les paramètres intrinsèques en opérant une décomposition RQ. Or nous avons ici un système qui possède deux équations par correspondances SIFT calculées (sachant l'équation 2.51). Notre problème est donc sur-contraint. Nous avons donc recours à une méthode de moindres carrés permettant de trouver la matrice P_{frame} répondant au mieux à l'ensemble des contraintes. Nous chercherons donc à minimiser la fonction de coût :

$$costFun(P_{frame}) = \sum_i \|\hat{x}_i - x_i\|^2 \tag{2.52}$$

avec $\hat{x}_i = \frac{P_{12} X_i}{P_3 X_i}$

Où P_{12} est la matrice 2x3 composée des deux premières lignes de P. P_3 est la troisième ligne de P.

Le problème à résoudre étant de type moindre carré "non-linéaire" et non convexe, nous le résolvons par une méthode de type Levenberg-Marquardt [Mor78] décrite dans 2.1.11.

Une approche plus récente a été présentée dans [LMNF09] nommée EPNP. Ils supposent ici la caméra calibrée. Ils ont pris le parti de s'affranchir des problèmes de calibration en normalisant les points 2D, il s'agit de les multiplier

par l'inverse de la matrice intrinsèque comme vu dans 2.1.4. Ils rajoutent à cela le fait de paramétrer la pose de la caméra en passant par 4 points de contrôles, assurant que la transformation estimée soit rigide. Le fait de procéder ainsi permet de rendre les temps de calcul moins longs.

Des méthodes telles qu'UPNP [PSACMN13] proposent des alternatives à ce problème permettant de travailler sur des caméras non calibrées.

Ajustement de faisceaux

À ce stade nous disposons d'une première estimation de la position des points 3D X_i et de la position des caméras c_j. À partir de cette estimation initiale il est possible de raffiner les positions par le biais d'une étape d'optimisation. Une solution possible pour cela est d'utiliser la méthode d'ajustement de faisceaux. Cette méthode a été grandement utilisée dans le cadre de la photogrammétrie. Elle s'est néanmoins fortement répandue en vision par ordinateur notamment dans des domaines comme la structure from motion (SFM).

En partant des estimations de $\hat{P}_i$ des poses des caméras c_i et $\hat{X}_j$ des points 3D dont nous disposons déjà, l'ajustement de faisceaux permet d'affiner leur estimation en réduisant l'erreur de reprojection. L'idée derrière le principe de minimiser les erreurs de reprojection est la suivante : disposant des points $\hat{X}_j$ et des matrices de projections approximées $\hat{P}_i$ nous obtenons la reprojection approximée $\hat{x}^i_j$, elle s'exprime comme dans l'équation 2.53. Tous les points 3D et 2D sont représentés en coordonnées homogènes.

$$\hat{P}_i \hat{X}_j \propto \hat{x}^i_j \tag{2.53}$$

Une fois que nous avons obtenu cette reprojection approximée nous pouvons la comparer avec la position projetée x^i_j du point 3D X_j dans la caméra c_i. Cela nous donnera une idée de l'erreur de reprojection commise. L'idée afin de se rapprocher de la disposition réelle des caméras et des points sera de minimiser l'erreur de reprojection comme suit :

$$\min_{\hat{P}_i, \hat{X}_j} \sum_{ij} d(\hat{P}_i \hat{X}_j, x^i_j)^2 = \min_{\hat{P}_i, \hat{X}_j} \sum_{ij} d(\hat{x}^i_j, x^i_j)^2 \tag{2.54}$$

d est une distance euclidienne entre deux points 2D d'une image. Résoudre cela permet d'obtenir une estimation précise de la position du nuage de points et des caméras. Par conséquent nous avons la position des caméras de la base et du nuage de points 3D (figure 2.15), ainsi que la calibration à priori de la caméra ayant capturé la scène. Nous nommons *liste de correspondances* $2D$-$3D$

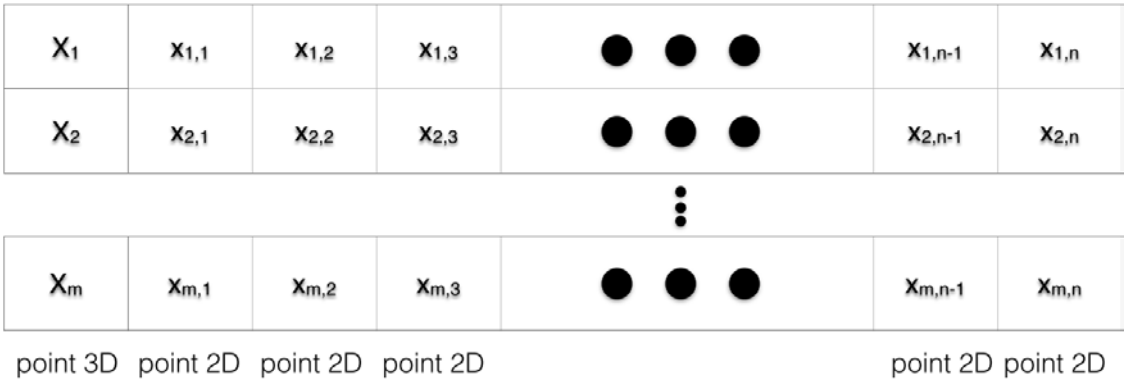


FIGURE 2.16 – liste de m correspondances 2D-3D. A noter que toutes les listes n'ont pas forcément le même nombre de points 2D.

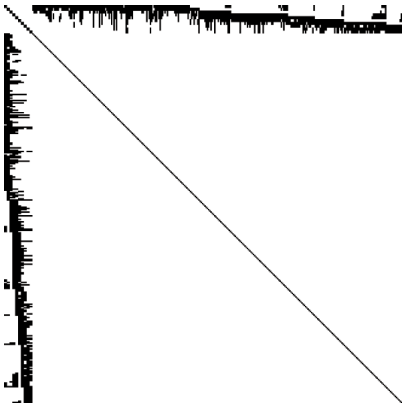


FIGURE 2.17 – représentation de l'approximation de la matrice Hessienne. Les zones blanches correspondent aux zéros, la nature éparse de la matrice est mise en avant.[1]

l'ensemble constitué par un point 3D et les points 2D des images de la base qui ont servi à le calculer comme représenté dans la figure 2.16.

Dans les faits, minimiser l'équation 2.54 est un problème de minimisation non linéaire au sens des moindres carrés. Nous sommes donc dans le cadre introduit dans la partie 2.1.11. C'est une étape à la fois cruciale et délicate. Elle est cruciale car elle permet d'avoir une représentation fidèle d'une scène réelle. Elle est délicate car le problème est non-convexe et de grande dimension. Pour faire face aux problèmes de dimension la nature creuse de J est exploitée. Comme expliqué dans [LA09], l'ajustement de faisceaux rentre parfaitement dans le cadre de l'algorithme de Levenberg-Matquardt vu dans la partie 2.1.11.

Dû à la nature du problème, certaines optimisations sont possibles. En effet tous les points ne sont pas visibles dans toutes les images et donc la matrice jacobienne J possède beaucoup de zéros. Cela lui confère une structure creuse permettant de simplifier certains calculs. Calculer J est relativement simple, par contre le calcul de l'inverse de la matrice H peut s'avérer plus complexe. Pour cela des méthodes comme le complément de Schur permettent de grandement simplifier les calculs. Cette méthode se sert de la structure éparse de J (figure 2.17). Une méthode optimisée afin de calculer l'inverse de H en utilisant ces principes est évoquée dans [LA09].

Concrètement une première paire d'image est choisie, elle doit fournir le plus d'information possible et facilite la reconstruction initiale. Puis au fur et à mesure d'autres paires d'images sont ajoutées en sélectionnant la paire d'images qui a le plus de points en commun avec les points 3D déjà reconstruits. Et à chaque itération un ajustement de faisceaux est effectué. L'algorithme s'arrête lorsque plus aucune paire d'image ne peut être ajoutée.

2.1.13 Méthodes "Structure From Motion" (SFM) : mise en application des méthodes de reconstruction

Cette section peut être vue comme un récapitulatif des outils vus précédemment. Elle présente la façon dont ces outils peuvent être utilisés conjointement afin de parvenir à la reconstitution 3D d'une scène. En effet le principe des méthodes de type SFM se base sur l'enchainement des méthodes présentées jusqu'à présent.

Dans les faits nous ne connaissons a priori ni la configuration du nuage 3D, ni la localisation des caméras. Pour cela les méthodes de type SFM sont souvent utilisées afin de déterminer la pose des caméras dans une scène, relativement à celle du nuage de point 3D qui lui est associé. Les méthodes de ce type ont été introduites dès les années 1980 et ont connu un essor considérable depuis. Leur principe est de se servir de plusieurs images d'une même scène afin de retrouver sa structure 3D ainsi que la position d'où les images ont été prises. Des méthodes comme celle de Snavely [SSS06, SSS08] utilisées dans PhotoTourism attestent de la popularité et de l'efficacité de ce type de méthodes. L'approche globale de cette méthode est donnée dans l'algorithme 2.

Dans ce processus trois étapes clés se distinguent :

[1]L'image est libre et vient de : https://upload.wikimedia.org/wikipedia/en/e/ea/Bundle_adjustment_sparse_matrix.png

Algorithme 2 : SfM incrémentale

Entrées : K, correspondances filtrées
Sorties : nuage de point $3D$, pose de caméras

1 calculer liste de correspondances $2D - 3D$ t `// à ce stade la position 3D des points n'est pas encore connue nous avons seulement l'ensemble des points 2D qui correspondent au même point de l'espace`
2 calcul d'un graphe de connectivité G `// un nœuds par vue, une arête lorsqu'il y a suffisamment de correspondances entre vues`
3 choisir une arête e permettant une bonne initialisation `// une arête pour laquelle il y ait un nombre conséquent de correspondances`
4 estimation par ACRANSAC de la matrice essentielle entre images
5 triangulation des éléments de $t \cap e$ `// donne une reconstruction initiale, donne une première estimation des positions 3D des éléments de` $t \cap e$
6 retirer e de G
7 **while** $G \neq \emptyset$ **do**
8 choisir e$\in$ G qui maximise liste(e)$\cap points3D$
9 estimation de la pose de la vue par ACRANSAC
10 triangulation des éléments de $t \cap e$
11 retirer e de G
12 faire un ajustement de faisceaux

- la détection et la description de primitives vues dans les sections 2.1.7, 2.1.8
- le filtrage des données aberrantes vu dans la section 2.1.10
- le respect des contraintes géométriques dû à l'observation des primitives sous plusieurs vues vu dans la section 2.1.12

De nombreux axes de recherches ont visé à améliorer chacun de ces trois axes. Le but final étant de toujours raffiner la reconstruction tout en étant de plus en plus compétitif en terme de temps de calculs.

Concernant l'axe sur les primitives, lors des dernières décennies, nombres de méthodes de détection et de description de point d'intérêts ont émergés. Leur but étant de décrire un point de l'image de façon suffisamment générique. Il s'agit d'être en mesure de reconnaitre, de plus en plus rapidement et avec

robustesse, un même point de l'espace projeté sur des images différentes. Cela même lorsque les prises de vue des images apportent de grosses distorsions dans le voisinage des points projetés. L'état de l'art en a été dressé dans les parties 2.1.7, 2.1.8. Dans [Wu13], l'auteur donne des indices sur comment choisir les images à comparer. Il se base sur l'idée qu'il suffit de comparer les points d'intérêt de grandes échelles afin de savoir si deux images se chevauchent ou non. Cela permet de réduire le nombre de comparaisons à effectuer.

En ce qui concerne l'axe sur le filtrage de données aberrantes des méthodes comme openMVG [MMM13b], ont travaillé sur l'élaboration de méthodes de filtrage RANSAC optimisées et adaptées au problème. Ils préconisent par exemple d'utiliser le filtrage ACRANSAC, notamment dans [MMM12b], afin d'obtenir des résultats encore plus pertinents. Cela est décrit dans 2.1.10.

Le respect des contraintes géométriques est effectué en deux parties. Une première où un système non linéaire au sens des moindres carrés est résolu. Pour cela des méthodes de type Levenberg-Marquardt sont utilisées. Ce même algorithme est utilisé dans la phase d'ajustement de faisceaux, afin de minimiser les erreurs de reprojections. Des méthodes comme celles introduites dans [RCC15] proposent une bibliothèque nommée LMA qui est un méta-programme. Ce méta-programme génère un code permettant de résoudre des problèmes d'optimisation tout en s'adaptant automatiquement aux paramètres du problème. D'autres approches comme celles vues dans [Wu13] modifient la façon dont l'ajustement de faisceau est effectué en opérant un ajustement complet ou partiel selon la nécessité. Il a implémenté sa méthode sous forme de logiciel dans VisualSfm [W+11].

De récentes approches tendent à porter la SFM sur appareils mobile. Dans [KTSP14], un nuage de point basé sur l'intégration d'hypothèses de modèle est généré. Ces modèles sont regroupés en un modèle 3D unique et consistant. MobileFusion [OKI15], va plus loin en fournissant directement un modèle 3D volumétrique (pas en nuage de point). De telles approches rendent possible l'utilisation des technologies de scan 3D directement sur nos appareils portables en des temps très compétitifs.

2.2 Problématique du temps réel

Dans la section précédente nous avons introduit certains des outils nécessaires à la reconstruction d'une scène 3D, et à la localisation d'un dispositif dans une telle scène. Nous verrons ici certaines des solutions récentes proposées en réalité augmentée. Comme nous l'avons déjà détaillé, la réalité augmentée (RA) consiste à insérer des objets virtuels dans une scène réelle en temps réel. Le but

étant que cette insertion paraisse naturelle, et que l'utilisateur ait l'impression que l'objet inséré fait parti de la scène.

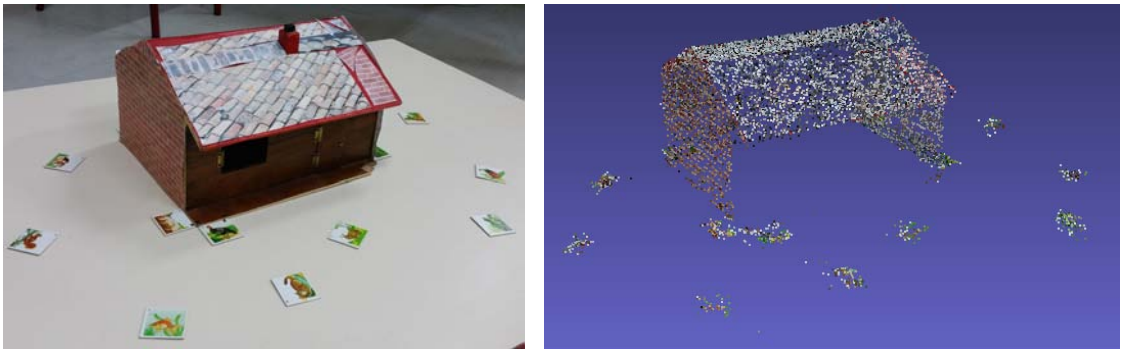
Pour atteindre un tel résultat, le calcul de la pose de l'appareil est central dans le contexte de la réalité augmentée. L'idée est, pour chaque image d'une séquence vidéo, de calculer la pose de façon robuste et rapide. Si cela est réalisé, alors des objets virtuels pourront être placés dans une scène de manière naturelle. Cela fait du calcul de la pose un problème classique et très étudié de la réalité augmentée. Dans la littérature, plusieurs capteurs ont été pensés pour cela. Une analyse en a été faite dans [WF02]. On peut citer, parmi d'autres, les capteurs mécaniques, ultrasoniques tous présentés dans [Sut68], magnétiques introduits dans [RBSJ79], inertiels [Wel95], et en particulier les capteurs optiques. Ce sont ces derniers qui vont particulièrement nous intéresser. En effet, toujours d'après [WF02], les méthodes basées sur l'image sont les plus compétitives.

Les premières instances de la réalité augmentée étaient basées sur l'utilisation de marqueurs. Vuforia [Vuf13], une librairie open source, est une des solutions les plus connues dans ce domaine. Dans les dernières années de nouvelles méthodes, basées sur l'analyse de la scène, ont émergé. Dans cette mouvance, nous pouvons citer les SDK à vocation commerciale : Metaio, récemment acheté par Apple, Wikitude [Per08], AugmentedPro ou encore Diotasoft. Vuforia [Vuf13] a aussi fait quelques tentatives dans l'analyse de scène, bien qu'à l'heure actuelle leur approche reste adaptée seulement à des objets de tailles moyennes. Les librairies libres openCV, ViSP et d'autres proposent des solutions intéressantes elles aussi.

Dans cette section nous allons décrire le fonctionnement des approches en réalité augmentée basées sur l'image. Pour cela nous allons commencer par positionner le problème dans son contexte dans 2.2.1. Ensuite, dans la section 2.2.2, nous décrivons les approches de réalité augmentée qui s'appuient sur des marqueurs pour situer un appareil dans une scène. Nous abordons dans 2.2.3, les approches de réalité augmentée sans marqueurs. Parmi les approches de RA sans marqueurs, deux se distinguent : la RA basée sur la silhouette d'objet, et la RA basée sur la minimisation d'erreur de reprojection. C'est dans ce dernier contexte que la thèse se positionne. Nous concluons cette section sur un bilan des méthodes de l'état de l'art et sur la description de nos contributions.

2.2.1 Positionnement du problème de réalité augmentée

Dans le cadre de la réalité augmentée, nous disposons de deux référentiels distincts. Le premier référentiel est celui du monde réel nous le nommons $\mathbb{F}_{\mathbb{R}}$, c'est celui attaché à la scène où l'utilisateur se trouve physiquement. C'est dans



(a) scène du monde réel : référentiel $\mathbb{F}_{\mathbb{R}}$ (b) scène du monde virtuel : référentiel $\mathbb{F}_{\mathbb{V}}$

FIGURE 2.18 – représentation des deux référentiel : le réferentiel $\mathbb{F}_{\mathbb{R}}$ et $\mathbb{F}_{\mathbb{V}}$

$\mathbb{F}_{\mathbb{R}}$ que l'utilisateur doit être localisé. Le second référentiel est celui du monde virtuel $\mathbb{F}_{\mathbb{V}}$, c'est celui où les éléments 3D sont placés. Dans les faits, les deux référentiels $\mathbb{F}_{\mathbb{R}}$ et $\mathbb{F}_{\mathbb{V}}$ sont alignés. Cela permet ainsi, une fois que nous sommes localisés dans une scène, de pouvoir placer directement les éléments virtuels dans le monde réel. Nous en avons une illustration dans la figure 2.18.

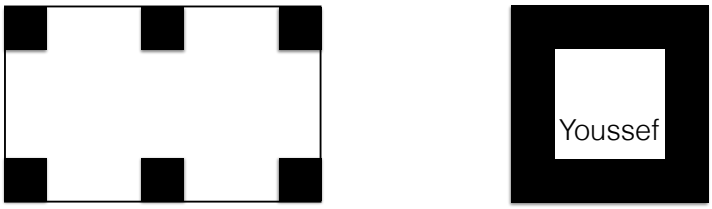
Dans un système de ce type, la seule inconnue est la pose de la caméra. Ainsi en utilisant les approches de réalité augmentée basées sur le traitement de l'image, le problème peut être résumé à un problème d'estimation de la pose d'une caméra dans une scène. C'est un problème dont nous avons présenté la résolution dans la section 2.1. Nous détaillons dans les sections qui suivent les différentes stratégies mises en place pour calculer efficacement la position d'une caméra dans une scène, l'enjeu ici étant le temps réel.

2.2.2 Réalité augmentée basée sur marqueurs

C'est l'une des premières méthodes mise en place afin de faire face aux problèmes soulevés par la réalité augmentée. Ici, plutôt que de se placer dans une scène entière, le problème est simplifié par un repère physique artificiellement généré. Dans les premières instances de cette méthode le repère était un marqueur noir et blanc ressemblant à un code. Par exemple dans ARToolkit [KB99], leur marqueur est un tableau blanc où sont placés 6 carrés noirs de tailles connues. Ils se servent de la géométrie de ce tableau pour déterminer la position de la caméra vis à vis de celui-ci.

Avec le temps, notamment avec des solutions comme Vuforia [Vuf13], ces marqueurs noir et blanc ont été remplacés par des images. Pour cela Vuforia utilise la méthode FAST [RD06] (vu dans la section 2.1.7 page 33) afin de

détecter des points dans l'image dont ils se serviront comme marqueurs. Il faut cependant veiller à ce que les contrastes de couleurs de ces images soient suffisamment forts. Il est également important que l'image ne comporte pas de symétries. Ces symétries induiraient une ambiguïté sur le calcul de la pose. Des exemples de marqueurs sont présentés dans la figure 2.19. D'autres types de marqueurs peuvent être envisagés, par exemple dans [Dor99] des boules non coplanaires sont posées dans la scène. L'algorithme POSIT [DD92] est ensuite utilisé pour calculer la pose. Une extension de POSIT pour des points coplanaires a été proposée dans [ODD96].



Marqueurs ARToolkit

Marqueurs Vuforia

FIGURE 2.19 – Exemples de marqueurs utilisés

2.2.3 Méthodes de réalité augmentée sans marqueurs

Disposer de marqueurs dans une scène peut s'avérer gênant. D'une part il faut générer des marqueurs suffisamment robustes, ce qui n'est pas forcément aisé pour un utilisateur. D'autre part les marqueurs ont tendance à dénaturer la scène où se déroule l'expérience. Pour ces raisons, les méthodes plus récentes visent à se servir de la scène elle même afin de s'y localiser.

Afin d'y parvenir deux méthodes se démarquent. Une première dans laquelle la silhouette d'un objet de la scène est utilisée afin de calculer la pose de la caméra. Ces méthodes se basent sur un modèle 3D fourni par l'utilisateur.

La deuxième approche s'appuie sur les méthodes dites SLAM, à l'origine initiées dans le monde de la robotique. Ces méthodes s'appuient, le plus souvent, sur l'extraction de points dans une scène et la minimisation d'erreur de reprojection. Dans les deux sections qui suivent nous détaillerons ces méthodes.

Méthodes basées sur modèles 3D

Cette méthode est différente au sens où elle s'appuie sur les contours d'un objet afin de déterminer la position d'une caméra. L'idée de base est de remplacer la méthode basée sur des correspondances de points vue dans la section 2.1.6. Cette approche est remplacée par la distance entre un contour dans l'image, et la projection de ce même contour issu d'un modèle 3D dans celle-ci. C'est la trame globale de ce type d'approche que l'on retrouve dans [CC12, CMPC06, L+91, PMK12, DC02]. Pour cela dans ce type d'approche il est nécessaire de disposer d'une estimation de la pose initiale $\tilde{P}$, et d'un modèle 3D d'un objet de la scène.

Lorsque nous disposons de ces éléments, le modèle 3D est projeté sur l'image en se servant de $\tilde{P}$. Usuellement le contour du modèle projeté est échantillonné en points de contrôles $C(P)$ (points en noirs sur la figure 2.20). La recherche de points voisins est effectuée le long des normales aux contours. Les points les plus proches x_i (points blancs dans la figure 2.20) sont cherchés en suivant la normale des gradients aux contours. L'idée est d'estimer la pose de la caméra qui minimise la distance entre un point du modèle projeté et les contours du même objet dans l'image.

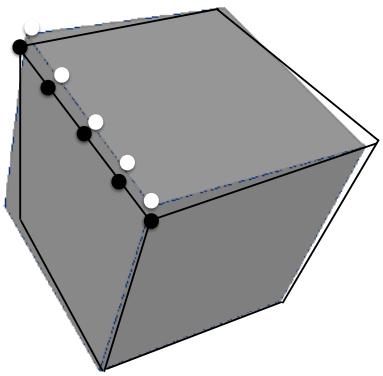


FIGURE 2.20 – modèle échantillonné à recaler

Cela résulte en la résolution d'un problème d'optimisation non-linéaire qui vise à minimiser les erreurs de reprojection. Il se définit comme suit :

$$\hat{P} = \arg\min_{P} \Sigma_i d_\perp(C(P), x_i) \tag{2.55}$$

Cette équation tend à minimiser la distance de la silhouette $C(P)$ aux points x_i, $d_\perp(C(P), x_i)$ est la distance orthogonale d'un point x_i à la projection du contour. Des méthodes telles que les algorithmes de Levenberg-Marquardt ou de Gauss-Newton vus dans la partie 2.1.11 sont usuellement utilisées. La différence majeure avec ce que nous avons vu dans le cas des problèmes PnP de la section 2.1.12 est que la distance à minimiser ici est d'un point à un contour (et non d'un point à un autre dans le cas du problème PnP).

Afin de faire face aux problèmes liés aux abberations, des méthodes comme RANSAC [FB81, BPS05] ou des M-estimateurs [CMPC06] vu dans la section 2.1.10 sont couramment exploitées. Des méthodes permettant de combiner plusieurs hypothèses ont été pensées afin de rendre l'estimation encore plus robuste [VLF04, PMK12].

Cette approche est celle utilisée dans les permières instances de Metaio, ainsi que chez ViSP [CMPC06] de l'INRIA et openTL de DLR. L'un des inconvénients majeurs de cette approche réside dans le fait qu'il faut disposer d'une estimation de la pose de la caméra, ainsi que du modèle 3D d'un objet visible dans la scène.

Méthode SLAM

Disposer de modèles 3D peut s'avérer très contraignant, surtout lorsqu'il s'agit de scènes vastes ou avec des objets imposants. Pour pallier à ces contraintes, la méthode SLAM (simultaneous localisation and mapping) s'est de plus en plus imposée ces dernières années. À la base développée dans le domaine de la robotique, cette méthode permet à la fois d'estimer la position d'une caméra dans une scène en même temps que cette scène est reconstruite. Nous en trouvons les premières instances dès la fin des années 1980 dans les articles [CL85, SSC90].

Dans les méthodes SLAM plusieurs capteurs peuvent être utilisés tels que des lasers [CMNT99], radar [DNC+01], sonar [TNNL02], images et autres. Cependant, grâce aux évolutions du traitement de l'image, dans les approches SLAM la vision a pris une place prépondérante. Cela peut s'expliquer par le fait que l'extraction et la description de points (vues dans les sections 2.1.7 et 2.1.8) dans une image est de plus en plus rapide, et que des processus comme l'ajustement de faisceaux (vu dans la section 2.1.12) ont été grandement optimisés. À tel point que dans certains systèmes la caméra est l'unique capteur, on parle de vision-based SLAM ou VSLAM [SLL05, LBJL07].

De prime abord les problèmes de type SLAM et structure from motion (SFM), vu dans la partie 2.1.13, peuvent sembler proches. Ils se distinguent cependant par la nature des données en entrée, ainsi que par la rapidité d'acquisition. Là où la SFM prend n'importe quel type d'image en entrée, les méthodes SLAM prennent des séquences d'images acquises à la volée. C'est une différence fondamentale, car elle permet à la SFM de traiter ces données hors-ligne. Les méthodes SLAM, de leur côté, permettent de calculer la pose de la caméra en ligne en temps réel. Elles s'appuient pour cela sur l'évolution dans l'optimisation des calculs. De telles qualités rendent les méthodes SLAM très intéressantes pour la réalité augmentée. En effet dans les problématiques de réalité augmentée il est primordial de pouvoir estimer la pose en temps réel. Parmi les méthodes SLAM deux courants majeurs se distinguent :

- Les approches basées sur filtres Bayesien
- Les approches basées sur minimisation d'erreurs de reprojection (basée sur l'ajustement de faisceaux)

Pour les approches basées sur filtre Bayesien, l'idée est de poser le problème comme étant la prédiction du mouvement de l'utilisateur en fonction des mouvements passés. Pour cela la mise en place d'un modèle de mouvement qui soit à la fois suffisamment complexe, pour éviter d'avoir de trop grandes incertitudes dans l'estimation, et suffisamment simple, afin de pouvoir être valide dans la majorité des cas, est nécessaire. Le modèle classiquement utilisé est celui de l'accélération constante en translation et de vitesse constante en rotation, cela est abordé dans [ZF12].

L'utilisation du filtre de Kalmann étendu a été proposé dans [Dav03]. Le filtre de Kalmann présente l'avantage d'être simple à implémenter. Son point faible reste que la preuve de convergence d'un tel filtre ne peut pas être vérifiée pour les problèmes fortement non linéaires. Il a de plus une complexité quadratique, ce qui en fait un procédé chronophage. Dans l'article présenté dans [ED06], le filtre particulaire lui est préféré. Ce filtre permet en effet d'approximer des problèmes un peu plus complexes, et est plus robuste aux non linéarités. Ces méthodes sont de nos jours de moins en moins utilisées, si ce n'est pour de la fusion de données.

Les approches basées sur la minimisation d'erreurs de reprojection leurs sont préférées, une étude dans [SMD10] indique que cette approche donne des résultats meilleurs que celles basées sur les filtres bayesiens. Les outils présentés dans 2.1.12 y sont utilisés à savoir la triangulation, le calcul de pose et l'ajustement de faisceaux. L'ajustement de faisceaux y est un principe central. Il permet de corriger les erreurs d'estimation mais est lourd en terme de calcul.

Dans [SMD10], les auteurs ont prouvé qu'une fois que la matrice Hessienne est rendue éparse pour les premières images le problème peut être simplifié en terme de calcul. La simplification est effectuée grâce au caractère séquentiel du flux vidéo. Cela rend les approches basées sur minimisation des erreurs de reprojection plus efficaces que celles basées sur filtre. Le problème peut être vu ainsi :

- Soit $[P]_t = (P_1, \ldots, P_t)$ une séquence de t positions de caméras
- Soit $[X]_N = (X_1, \ldots, X_N)$ un ensemble de N points 3D de cette scène

Le but est de minimiser l'erreur de reprojection, comme cela était le cas pour les problèmes PNP de la section 2.1.12 :

$$([\hat{P}]_t, [\hat{X}]_N) = \arg\min_{P,X} \Sigma_{i=1}^{t} \Sigma_{j=1}^{N} d(x_{ji}, P_i X_j) \tag{2.56}$$

Plus la séquence sera longue, plus la complexité du problème augmentera. À l'initialisation la géométrie épipolaire est utilisée [Nis04], et les points 3D triangulés. Ensuite plusieurs stratégies sont proposées :

- Dans [NNB04, MLD+06], l'ajustement de faisceaux est utilisé sur un voisinage de notre image. Cela a l'inconvénient de mener à des dérives au fur et à mesure du temps. Si à un moment le système dévie, il sera difficile de rattraper le coche.
- Dans des méthodes comme PTAM [KM07], un ajustement de faisceau local est effectué et en parallèle la caméra est localisée comme dans la section 2.1.12 en utilisant uniquement des points 3D déjà reconstruits.

Metaio, 13th lab (racheté par Occulus), ou encore Qualcomm (qui sont à l'origine de Vuforia) fournissent des technologies très robustes autour de cela.

Néanmoins dans de telles approches SLAM, le calcul de la position reste coûteux. Il l'est en particulier dans de vastes environnements. Le calcul est aussi peu robuste à l'ambiguïté d'échelle. Pour faire face à de telles difficultés des approches comme le SLAM contraint (CSLAM) ont été mises en place. Les méthodes CSLAM [TGBC+11, RLdTR11] ont pour vocation de donner des informations à priori comme la forme de certains objets ou des contraintes sur les types de mouvements afin d'obtenir une reconstruction de la scène et une localisation du dispositif plus précises et robustes. Plus récemment des méthodes de SLAM contraints et dynamiques ont été proposées dans [RCC15]. Les auteurs y introduisent le DCSLAM, qui permet d'utiliser un

nombre de contraintes variables. Une autre idée serait de découpler la phase de reconstruction de la scène et de calcul de la pose. Ici le principe d'ajustement de faisceaux est utilisé afin de reconstruire la scène dans une phase hors ligne, cette phase peut être assimilée à la méthode SFM abordée dans 2.1.13. Ensuite lors de la phase en ligne la pose de la caméra est trouvée en résolvant un problème PnP, voir 2.1.12. On peut voir cette approche comme du SLAM contraint où la scène est déjà connue. Cette approche a l'avantage de proposer un calcul de matrice de projection dans un référentiel fixe. C'est l'approche pour laquelle nous avons opté dans le cadre de la thèse. Elle a été suivie avec brio dans le contexte de la réalité augmentée dans ces implémentations [MMHM14, VH12, WRM+08].

DiotaSoft de leur côté ont proposé une approche intéressante mêlant les méthodes basées sur la silhouette et le SLAM, ces méthodes sont nommées méthodes hybrides. Metaio proposait elle aussi des technologies de ce type.

Conclusion

Nous avons, dans ce chapitre, présenté les outils nécessaires à la construction d'une scène en 3D et à la localisation d'un dispositif dans cette scène (section 2.1). En particulier la partie 2.1.13 décrit la méthodologie structure from motion (SFM) qui résume la façon d'utiliser les outils présentés afin de parvenir à la reconstruction d'une scène 3D, ainsi qu'à la localisation des caméras ayant permis cette reconstruction. Dans la partie 2.2, nous nous sommes concentrés sur la problématique posée par le temps réel. Nous avons présenté différentes approches pour cela : les approches basées sur marqueurs dans 2.2.2 , les méthodes s'affranchissant des marqueurs dans 2.2.3. Nous avons aussi introduit la façon dont le problème de la réalité augmentée peut être abordé dans la section 2.2.1.

Notre approche peut s'apparenter à du SLAM contraint, dans lequel les phases de reconstruction de la scène et de la localisation de la caméra ont été découplées. Notre approche est pour cela basée sur deux étapes :

- Une phase de pré-traitement où des images sont acquises puis utilisées pour reconstruire la scène.
- Une autre phase où la pose d'une caméra est calculée à la volée permettant une expérience de réalité augmentée.

Nous avons axé nos recherches sur la façon d'améliorer la robustesse des calculs tout en garantissant un temps de calcul permettant le temps réel. Pour cela nous avons proposé dans le chapitre 3 un schéma permettant de faire de la

réalité augmentée en s'appuyant sur le descripteur SIFT. Nous avons proposé une structure basée sur les graphes permettant de réduire le champ de recherche pour une image entrante. Nous avons également opéré un filtre sur les points d'intérêt afin d'utiliser seulement ceux apportant le plus d'informations utiles. Les résultats sont présentés dans le chapitre 3. Ensuite dans l'idée de fournir des résultats encore plus rapide, nous avons décidé de construire notre propre descripteur. C'est un descripteur binaire basé sur la notion de tests. Les tests qui le composent sont appris sur une base de données image jugée suffisamment diversifiée. Dans cet apprentissage nous avons pris en compte des points forts du descripteur afin de sélectionner les tests les plus pertinents. Cela fait l'objet du chapitre 4.

Chapitre 3

Présentation de notre méthodologie

Dans la RA le but est de pouvoir insérer un objet virtuel dans un emplacement donné d'une scène réelle. Nous pouvons résumer ce problème au fait de projeter de façon réaliste cet objet dans l'image. Afin de réaliser cela, la connaissance de la pose de la prise de vue par rapport à la scène est nécessaire. Dans la mesure où le calcul de la pose revient à calculer des correspondances 2D/3D et qu'un point 2D peut correspondre à une infinité de points 3D, le problème est mal posé. Afin de simplifier ce problème nous construisons en pré-traitement une liste de correspondances 2D-3D en utilisant la technique dite de structure from motion (SFM) suivi d'un ajustement de faisceaux [LA09]. Par la suite, pour situer une nouvelle image par rapport à la scène, il nous suffira de calculer des correspondances 2D-2D entre cette nouvelle image et la base de données de pré-traitement B_d. En effet, les correspondances 2D-3D s'obtiennent alors par simple transitivité. C'est à dire en chainant correspondances 2D-2D et 2D-3D. Dans la figure 3.1 tout le processus que nous avons mis en place pour résoudre la problématique de la pose d'une image entrante est décrit. La première étape consiste à établir des liens entre points 2D des images (les lignes vertes), comme vu dans 2.1.9. Il est important, afin de pouvoir correctement reconstruire la scène, d'assurer les deux propriétés suivantes :

- avoir une couverture de la scène suffisamment fournie afin de la reconstruire en entier
- avoir un recouvrement significatif entre les images capturées pour permettre l'appariement de points 2D entre celles-ci (en gardant en tête la capacité de mise en correspondance du descripteur que nous avons choisi). Les descripteurs ont été décrits dans 2.1.6.

Une fois que la prise d'image et la mise en correspondance sont effectuées, une étape de reconstruction 3D de la scène est entamée. Le but est, en ayant connaissance des correspondances entre images, de créer un nuage de points 3D. Dans ce nuage chacun des points 3D correspond à un ensemble de points 2D des images de la base B_d mis en correspondance. Afin de s'assurer des configurations relatives entre images et nuages de points une étape d'ajustement de faisceaux [LA09] est appliquée. Une fois tout cela mis en place il sera possible pour une image entrante i de créer des liaisons 2D avec les images de la base B_d (lignes bleues dans la figure 3.1). Par transitivité nous en déduisons un lien 2D-3D entre l'image entrante et la scène. En disposant de ces liens il est possible de calculer la pose du dispositif, et ainsi savoir comment il est situé dans la scène. Cela nous permettra de projeter des éléments virtuels dans notre scène réelle. Tous ces outils ont été présentés dans la section 2.1

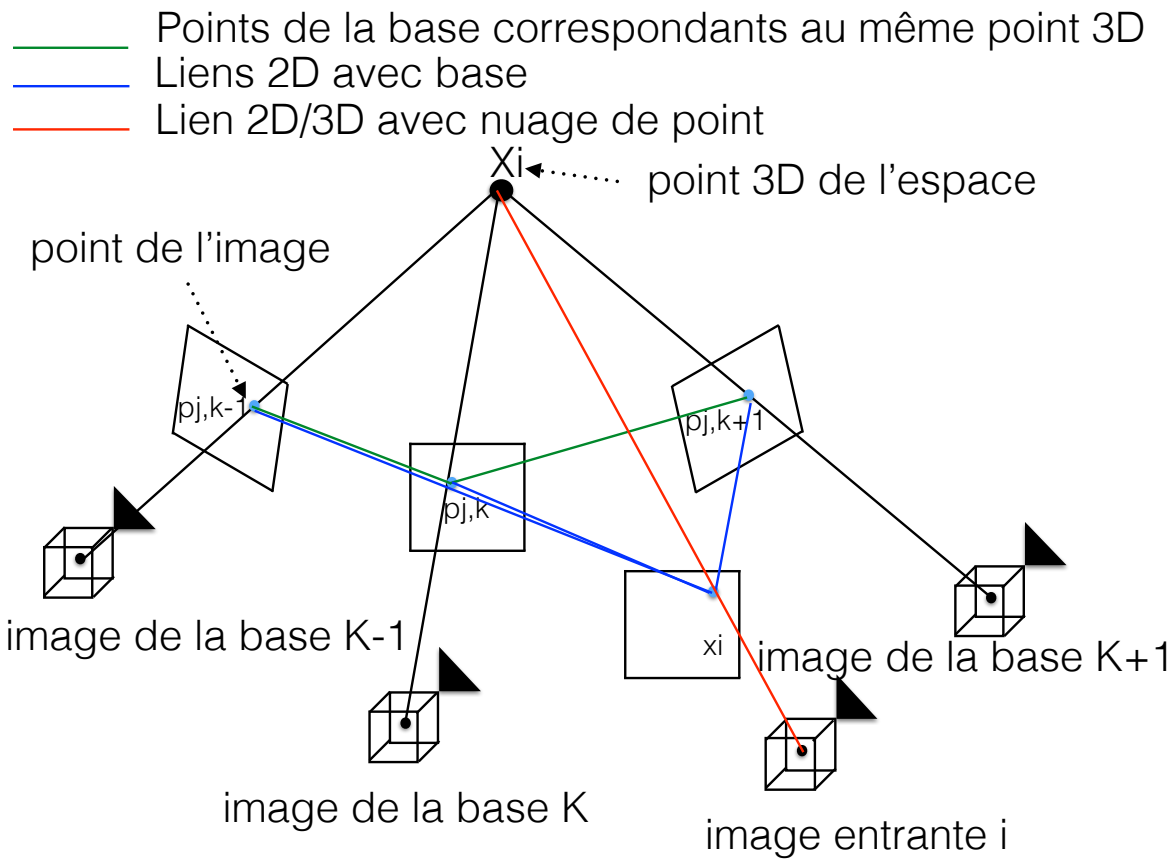


FIGURE 3.1 – Schéma expliquant la reconstruction 3D et le calcul de pose

3.1 Méthode hors ligne/en ligne avec SIFT

3.1.1 Pré-traitement : phase hors ligne

Le problème étant mal posé nous avons besoin d'une étape de pré-traitement afin d'avoir des informations complémentaires, et de simplifier sa résolution. Le principe de ce pré-traitement consiste à se servir d'une base d'images de la scène prises dans une étape hors ligne. Dans ces images nous extrayons des points SIFT [Low99]. Une fois ces points obtenus, il nous est possible de les mettre en correspondance comme vu dans la section 2.1.9. Donc contrairement aux méthodes de type SLAM vu dans la section 2.2.3, notre approche ne scanne pas la scène en même temps qu'elle se positionne dans celle-ci. Il y a une première étape où nous la scannons. Ensuite, disposant d'une représentation de la scène, nous calculons la position du dispositif dans celle-ci.

La stéréovision permet, par triangulation des points des images appariées, de retrouver des informations sur la configuration 3D de la scène, et de créer un nuage de point 3D. Cela est abordé dans la section 2.1.12. Les principes théoriques liés à cette approche sont expliqués en détails dans [HZ03]. Dans les faits nous nous sommes basés sur l'implémentation améliorée présentée dans [MMM12b] dont les algorithmes sont présentés dans l'algorithme 2 de la section 2.1.13 et l'algorithme 1 de la partie 2.1.10. Cette implémentation consiste à utiliser les points SIFT [Low99] et le procédé de Lowe, expliqué dans la section 2.1.9, pour établir des correspondances entre les points d'intérêt de deux images.

À un point 2D de l'image peut correspondre chacun des points de l'espaces présents sur la droite passant par ce point et le centre optique de la caméra (voir droite d figure 2.1). Pour pallier à ce problème l'idée que nous avons exploitée est de mettre en place un lien entre des points des images 2D et la 3D, c'est ce que nous avons vu en section 3.1.2.

Nous procédons comme suit (les étapes sont illustrées dans la figure 3.2) :

Nous prenons des photos de l'objet sur lequel nous voulons superposer un élément virtuel (camera image). Puis sur ces images nous extrayons des points d'intérêts stables aux changements de pose (points de l'image p_j). Nous mettons ces points d'intérêts en correspondance afin de définir des listes de points 2D dans les images correspondant à un même point 3D (séries en vert). Cette étape est cruciale dans la mesure où elle nous donne des informations sur la géométrie de l'objet. Une fois les correspondances faites, nous construisons le nuage de point 3D en cohérence avec les données acquises (point 3D de l'espace P_j).

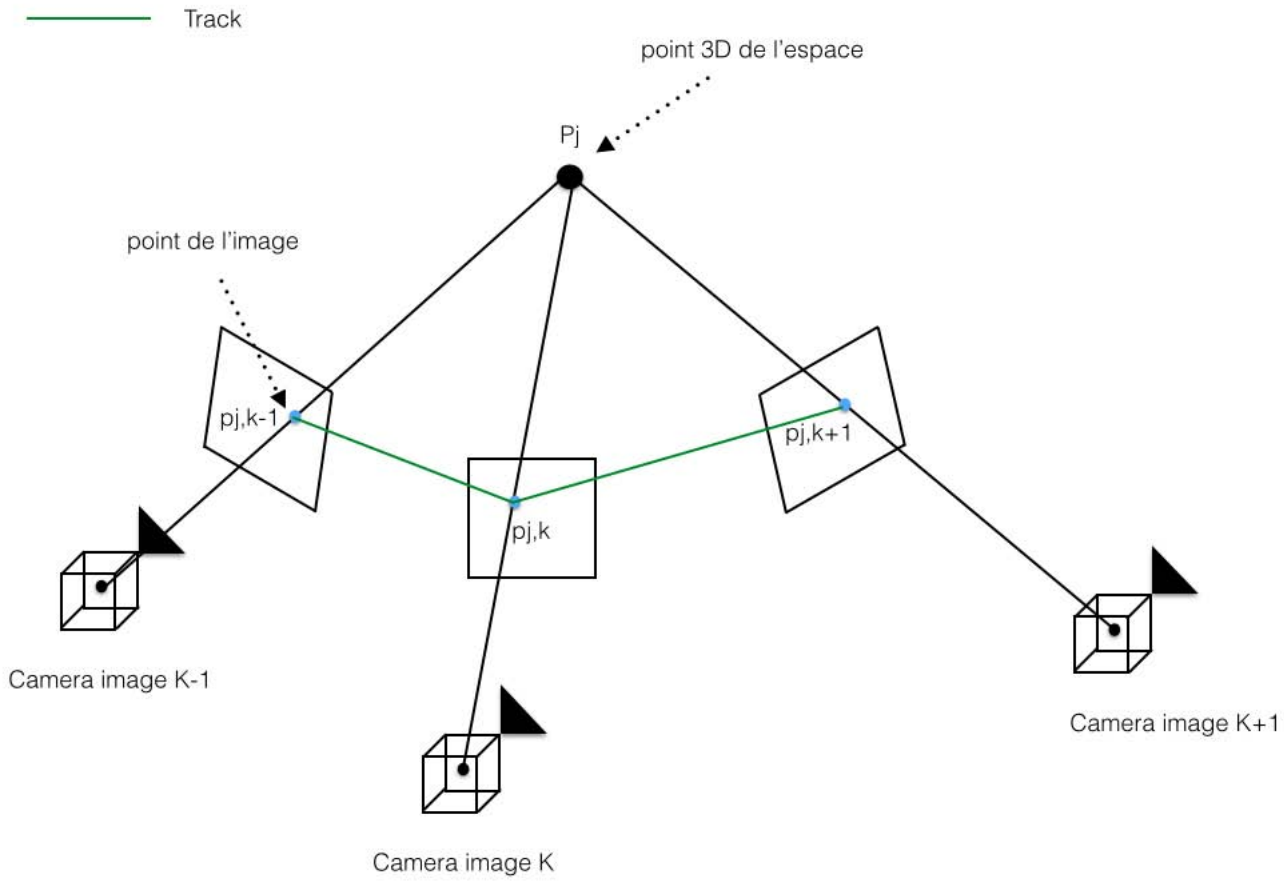


FIGURE 3.2 – Triangulation d'un point $3D$

Ceci se fait en se basant sur la géométrie épipolaire [HZ03]. Nous en donnons le détail dans ce qui suit.

À l'aide des correspondances il est théoriquement possible de déduire la matrice fondamentale E qui lie les deux images, en passant par la méthode ACRANSAC [MMM12a]. La méthode suivie pour cela est détaillée dans la section 2.1.5.

Le système à résoudre, pour calculer la matrice essentielle E, est sur-contraint. Il faut donc faire en sorte de trouver un modèle qui réponde au mieux à toutes les données. Dans les cas usuels nous utilisons une méthode de moindres carrés pour résoudre ce genre de cas. Ici nous ne savons pas si toutes les données sont cohérentes. En général il y a un fort risque que certaines

Forme de matrice de projection	expression
expression générique P	K[R|t]
expression normalisée $\hat{P}$	$K^{-1}P = [R\|t]$

TABLE 3.1 – Matrice de projection sous plusieurs formes

d'entre elles ne soient pas de réelles correspondances. Dans ce cas l'algorithme RANSAC [FB81, HZ03], ou ces dérivés a contrario tels que MAC-RANSAC [RDGM10] ou ACRANSAC [MMM12b], sont des plus adaptés. Nous les avons décrits dans la section 2.1.10.

Une fois que la matrice essentielle E est estimée, il est possible d'estimer les matrices de projection, comme vu dans 2.1.5. En disposant des matrices de projections nous pouvons, par triangulation, retrouver la position du point 3D correspondant à deux points 2D des images liées. Nous avons à ce niveau une approximation des dispositions relatives entre les points du nuage $3D$ et les caméras qui ont permis leur construction. L'ajustement de faisceaux [LA09] est ensuite utilisé afin d'améliorer les estimations de la pose des caméras et de la disposition des points. La façon dont ce procédé fonctionne est vue dans 2.1.12.

Suite à tout ce processus de reconstruction en 3D, nous disposons de listes de correspondances 2D-3D. La notion de liste de correspondances a été décrite dans la partie 2.1.12 et illustrée dans les figures 2.16 de la partie 2.1.12 et 3.1 de la partie 3.

Dans la section 2.1.6 nous avons vu qu'en utilisant directement les points SIFT le temps de description et de mise en correspondance de points clés sont lents. Afin de remédier à ce dernier point, nous avons mis en place certaines optimisations dans un schéma que nous avons créé.

3.1.2 Nuage de point 3D et liaison avec points 2D : phase en ligne

Dans la continuité de ce qui a été fait dans la section 3.1.1, l'idée est de s'appuyer sur le pré-traitement réalisé lors de la phase hors ligne. Nous nous aidons des listes de correspondances 2D-3D afin d'être en mesure d'obtenir les informations désirées sur une image acquise en ligne.

Lors de l'expérience de réalité augmentée, à chaque séquence de la vidéo, une image est acquise. Sur cette image, comme dans la phase de pré-traitement, nous allons extraire des points d'intérêts en nous servant de la méthode

développée dans SIFT. Une fois ces points d'intérêts extraits, il s'agit de mettre en correspondance les points d'intérêts de l'image extraite avec ceux qui ont été calculés sur la base de données images B_d (voir section 3.1.3). Les images de B_d étant celles qui ont permis la construction du nuage de point 3D .

Les points extraits sur les images de B_d sont eux même liés aux points 3D du nuage par les listes de correspondances 2D-3D. Ces listes ont été générées lors de la construction du nuage de point. Elles peuvent être vues comme l'ensemble constitué d'un point 3D (X_i dans la figure 3.1) et des points 2D ayant servis à le reconstruire (liés par les lignes vertes dans 3.1).

Une fois que nous disposons de ces listes de correspondances, il devient plus aisé de traiter une image entrante. En effet il s'agit de chercher des correspondances entre cette image et les images de la base (lignes bleues dans 3.1), puis par transitivité de trouver des liens 2D-3D (ligne rouge dans 3.1) afin de calculer la pose à partir de laquelle la vue a été prise. En disposant d'un nombre suffisant de liens entre la 2D et la 3D il est possible de calculer la pose de la caméra ayant assurée l'acquisition de l'image comme vu dans la section 3.1.3.

Néanmoins nous avons le même type de problématiques que dans la section 3 concernant les fausses correspondances. Afin d'y faire face nous nous appuyons à nouveau sur ACRANSAC qui estime la matrice essentielle liant notre image entrante à chacune des images de la base. Cela permet de déterminer, parmi les correspondances trouvées, lesquelles sont susceptibles d'être erronées. Une approche de ce type a été entreprise par [KMPK14] afin calculer la pose d'un dispositif.

La tâche qui consiste à lier certains points 2D de l'image entrante aux points 3D du nuage de point est centrale dans notre procédé. Il est donc essentiel qu'elle soit réalisée de manière robuste, afin d'assurer une bonne qualité des projections 3D sur l'image entrante et un suivi de qualité. Elle doit aussi être rapide pour garantir la fluidité du suivi temps réel de la scène.

3.1.3 Calcul de la pose d'une image

Notre but est de calculer la matrice de pose de la caméra normalisée $P_{frame} = [R|t]$ de la caméra au moment de la capture de l'image (où R représente la rotation de celle-ci et t sa position). On s'appuie pour cela sur le modèle sténopé (figure 2.1), et des liens 2D/3D fournis suite à la reconstruction 3D réalisée dans la section 3.1.1. L'approche est la suivante : utiliser SIFT pour détecter des points caractéristiques $\{x_1,x_2,...,x_n\}$ dans l'image exprimée en coordonnées normalisées, puis les mettre en correspondance avec l'ensemble des points SIFT des images de la base, ce processus est décrit dans la section 3.1.2. Dans notre

méthode nous avons évalué la matrice de calibration K lors du prétraitement. Nous pouvons ici nous contenter de lier les points 2D afin de calculer la pose d'une caméra entrante. Ensuite nous filtrons les fausses correspondances par le biais de ACRANSAC. Les points des images du pré-traitement étant liés aux points 3D par transitivité on obtient un lien entre un point 2D x_i de l'image entrante (en coordonnées rétinales) et un point 3D (figure 3.1).

Plus en détails, pour calculer la pose nous avons suivi une approche PnP (vue dans 2.1.11 page 53). Nous nous sommes servis des points 3D du nuage de points généré ainsi que de leur correspondants 2D dans l'image d'entrée. Pour retrouver la matrice de projection P du dispositif, nous estimons la projection $\hat{x}_i$ du point X_i lié à x_i dans l'image entrante. Nous cherchons ensuite à réduire l'erreur de reprojection entre x_i et $\hat{x}_i$ pour déduire P. Dans notre implémentation pour calculer P nous utilisons la bibliothèque ceres [AM+13].

Comme nous l'avons dit plus haut nous obtenons des liens entre points 2D et 3D par transitivité. Les liens entre points 3D et points 2D des images de B_d sont stockées dans une matrice. Ayant cette connaissance, pour une image entrante, nous cherchons ses correspondances avec les images de la base de données. Ensuite une fois ces correspondances obtenues nous cherchons parmi celles ci lesquelles nous donnent des liens avec des points 3D. Une fois que nous avons des liens entre des points $\{x_i\}_{i\in\mathbb{N}}$ de l'image entrante et $\{X_i\}_{j\in\mathbb{N}}$ de l'espace, nous cherchons à trouver P qui minimise la reprojection $\hat{x}_i$ de X_i sur l'image entrante et x_i. L'erreur peut s'exprimer comme suit $\Sigma_i||\hat{x}_i - x_i||^2$. Pour minimiser cette erreur nous utilisons la méthode d'optimisation de Levenberg-Marquardt [Lev44, Mar63] implémentée dans le solveur de ceres. Pour notre problème la convergence s'effectue en moyenne en 20ms.

Cette méthode nous permet de prédire avec robustesse l'emplacement d'une frame. Par contre elle est très lente en utilisant SIFT, à cause du temps de calcul de son descripteur. Le temps de calcul de la position d'une frame est de 8s sur un PC dernière génération. Il est donc nécessaire de trouver des moyens d'optimiser les calculs.

3.2 Accélération du suivi

3.2.1 Filtrage des points SIFT

Dans notre base de données, nous avons un nombre de descripteurs conséquent. Réduire le nombre de descripteurs à prendre en compte devient alors intéressant. Il faut tout de même choisir avec soin ceux que l'on décide de garder et ceux que l'on décide de retirer. Il est en effet souligné dans [Wu13] que

pour l'appariement de points, les points d'intérêt de plus grande échelle sont de meilleurs candidats pour les raisons suivantes :

- Les points d'intérêt de grande échelle couvrent une plus grande zone de l'image. Ainsi, même en petit nombre, ces points de grande échelle représenteront une partie conséquente de l'image.
- Dans le processus d'appariement de SIFT la structure est telle que les grandes échelles seront appariées entre elles.

Nous avons donc décidé de filtrer la base d'images prise hors ligne. Nous avons classé, pour chaque image de la base, les points d'intérêt par ordre d'échelle décroissante. Enfin nous en avons gardé un pourcentage fixé parmi les plus grandes échelles.

Partant toujours de cette idée, pour gagner du temps lors du calcul des points SIFT et descripteurs pour l'image entrante, nous décidons de ne calculer que ceux de grandes échelles en ne les cherchant que dans les octaves élevées. Dans notre cas le calcul se fait sur 3 octaves, nous ne chercherons les points d'intérêts que dans les octaves 2 et 3.

3.2.2 Structuration de la base d'image sous forme de graphe

Dans la mesure où le fait d'apparier des points à une banque d'image a un coût important. Il est de l'ordre de O(NlogN + MlogN) avec M le nombre de points dans l'image entrante, et N $\gg$ M le nombre de points dans toutes les images de la base. Il est donc essentiel de réduire le nombre de points à prendre en compte, sachant que dans les cas concrets la plupart d'entre eux ne sont pas utiles car pris d'un point de vue trop éloigné de l'image entrante. L'idée est donc de réduire le nombre d'images de la base à mettre en correspondance avec l'image entrante. Pour cela nous aimerions dresser une notion de voisinage entre images afin de restreindre la zone de recherche. La théorie des graphes fournit un cadre idéal pour cette problématique.

Considérons le graphe $G = (V, E, p)$ un graphe orienté pondéré composé d'un ensemble V=$\{i_1$, i_2 , ..., $i_n\}$ de n nœuds et E $\subset V \times V$ l'ensemble des arêtes pondérées. Une arête (i,j) $\in$ E code la similarité entre deux nœuds voisins et

est pondéré par la fonction de poids p : V × V → $\mathbb{R}^+$.

Dans notre cas, chaque nœuds représente une image et chaque arête relie deux images. Pour notre notion de poids nous avons choisi de calculer le nombre de listes de correspondances (voir section 3.1.2) que les images partagent. Dans notre représentation nous poserons $T = \{t_1,t_2,...,t_m\}$ l'ensemble des m listes de correspondances $2D$-$3D$ créées. Cela correspond aux listes de correspondances $2D$-$3D$ liant les points $3D$ aux points $2D$. Considérons f : $V \times V \times T \rightarrow \mathbb{R}^+$ telle que :

Afin d'avoir une mesure de la similarité entre deux images nous posons la fonction de poids suivante :

$$p(i_1, i_2) = \#\{t \in T | \exists x \in i_1, y \in i_2 \text{ avec } \{x,y\} \subset t\} \tag{3.1}$$

où x∈ i_1 désigne, de façon un peu abusive, le fait que le point d'intérêt x appartient à l'image i_1.

Intuitivement, $p(i_1, i_2)$ représente le nombre de points 3D visibles simultanément de i_1 et i_2. Des images correspondants à des points de vues proches auront un poids élevé. Afin d'avoir une représentation sous forme de graphe, nous avons choisi un nombre variables de voisins. Les expériences de la partie 3.3 justifient les choix que nous avons fait. Nous pouvons exprimer la façon dont les k plus proches voisins sont sélectionnés comme dans l'algorithme 3.

Algorithme 3 : Sélection des k plus proches voisins

input : image i_l dont nous cherchons les voisins, ensemble de nœudss V

output : $\mathcal{S}$ liste des k plus proches voisins

```
S = ∅
while #S < k do
    i_j = arg max_{i_q ∈ V\S} p(i_l, i_q)
    S = S ∪ i_j
```

Nous choisissons comme voisins de notre nœuds i_r les nœudss de $\mathcal{S}$, qui sont les nœudss qui ont le plus de listes de correspondances $2D$-$3D$ communes avec i_r. La notion étant fortement liée à la proximité des points de vue, si une image entrante prise en ligne est fortement liée à i_r, il y a de fortes chances qu'elle le soit aussi à ses voisins.

3.2.3 Filtrage des images de la base

Lors d'une acquisition vidéo, pour une image f_k suite au filtrage de ACRANSAC [MMM12a] seul un nombre limité d'images SI_k=$\{i_{f1},i_{f2},...i_{fs}\} \subset$ V sont utiles au calcul de sa pose. En suivant cette idée, nous avons décidé de limiter la recherche aux images qui semblent les plus pertinentes.

La mise en correspondance des points d'une image entrante f_k avec les images de la base de données hors ligne B_d est une tâche qui peut s'avérer coûteuse en terme de temps de calcul. Puisque dans la majorité des cas les images de B_d sont très éloignées de f_k, pour ces images là aucune correspondance ne sera trouvée. Ainsi pour toute image f_k notre souhait est de restreindre le champ des images à utiliser pour chercher des correspondances dans B_d. Pour cela nous exploitons la notion de voisinage définie dans la section 3.2.2. Pour l'image d'initialisation f_0 nous utilisons toutes les images de B_d, faute d'information. Suite au traitement de cette image, comme vu dans la section 3.1, nous obtiendrons en utilisant l'algorithme ACRANSAC un ensemble d'images de B_d pour lesquelles des correspondances ont été trouvées avec f_0. Nous noterons E_0 l'ensemble des images de B_d ayant des correspondances avec f_0. Ces images là sont celles qui sont les plus pertinentes à utiliser par la suite car l'image f_1, étant issue d'un flux vidéo, est proche en terme de point de vue de f_0.

Posons N(E_0) l'ensemble des voisins de chaque image présente dans E_0. Alors pour f_1, nous chercherons les correspondances des points d'intérêts avec les éléments de E_0 ainsi que $N(E_0)$. Nous noterons P_0=$E_0 \cup N(E_0)$ l'ensemble des images de B_d qui seront utilisées afin de chercher des correspondances avec f_1. La figure 3.3 détaille ces étapes.

Pour assurer le suivi sur le long terme nous cherchons des correspondances avec des images de l'ensemble P_0 qui sont construites comme dans l'algorithme 4.

Algorithme 4 : Sélection des images voisines

input : correspondances entre images
output : voisinage d'une frame entrante

$E_0 \leftarrow$ ACRANSAC(B_d)
for $k = 1..$*nombre de frames* **do**
    $P_k = E_{k-1} \cup N(E_{k-1})$
    $E_k \leftarrow$ ACRANSAC(P_k)

Nous pouvons généraliser pour une image f_k, $k \in \mathbb{N}$. Prenons E_{k-1} l'ensemble des images conservées suite au filtre ACRANSAC pour l'image

précédente f_{k-1} et de même N(E_{k-1}) l'ensemble des voisins de chaque image présente dans E_{k-1}. Alors nous chercherons des correspondances dans P_k={ $E_{k-1} \cup N(E_{k-1})$ }. Dans les cas où P_k serait l'ensemble vide, nous devrons refaire une initialisation.

Le gain de temps de cette approche est multiple. D'une part cela réduit le temps de mise en correspondance puisque nous avons éliminé les candidats qui n'offrent pas de bonnes correspondances. Nous gagnons aussi du temps au niveau du filtrage géométrique lié à ACRANSAC voir section 3.1.3. Dans les cas où l'image avec laquelle f_k est comparée n'est pas pertinente ACRANSAC ne trouve pas de solution. Alors il tire plus d'échantillons aléatoirement parmi la base. Le temps consacré à filtrer les correspondances pour une image non pertinente est donc considérablement plus rallongé.

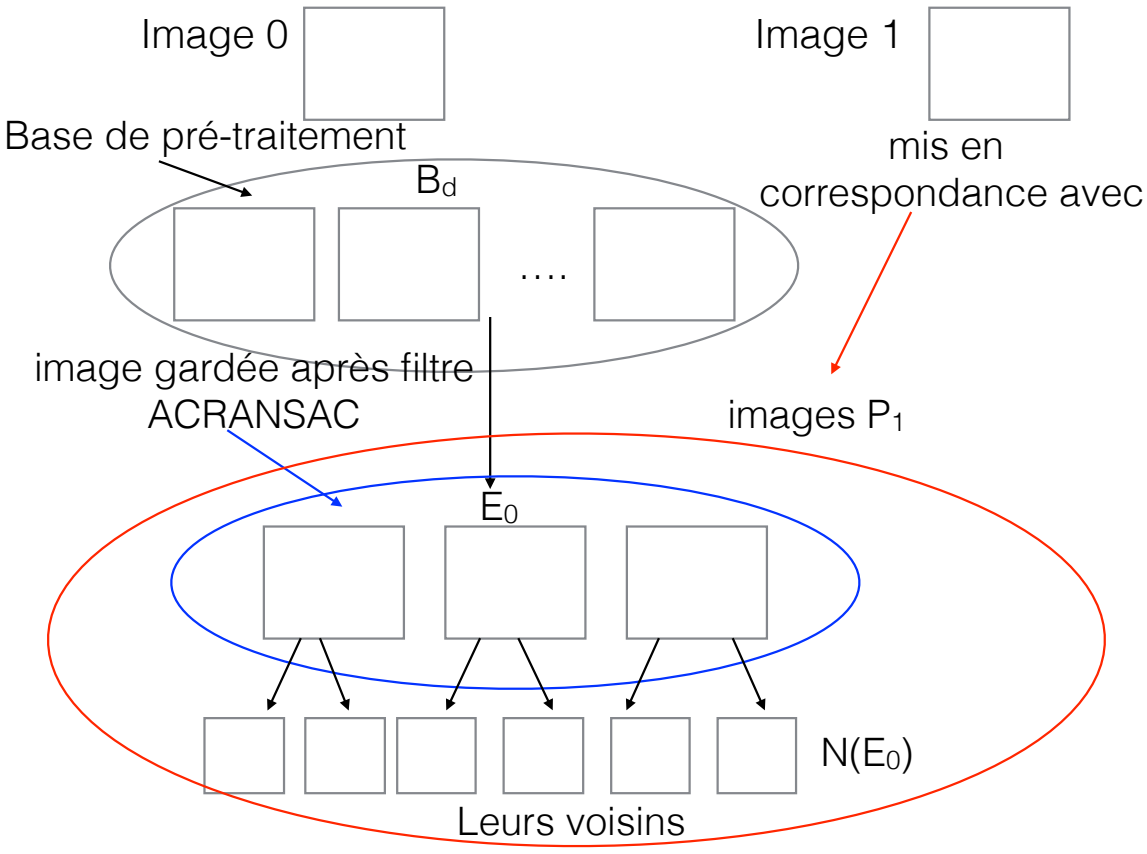


FIGURE 3.3 – schéma représentant filtrage des images de la base

3.3 Résultats

Nous avons éprouvé la robustesse de notre approximation en utilisant la base de données de Strecha [SvHVG+08]. Nous avons utilisé la fontaine, constituée de 11 images. Pour chaque image nous avons la vérité terrain, c'est à dire la matrice de projection réelle de l'appareil. Nous avons également créé notre propre jeu de donnée constitué de 41 images. Cette base de données a été créée afin que nous puissions effectuer certains de nos tests avec suffisamment de flexibilité. Il a été construit afin de rendre la construction d'un nuage de points $3D$ possible. Pour cela il est essentiel que les correspondances $2D$ fournies par SIFT soient correctes.

Nous nous sommes donc placés dans un contexte favorable à SIFT. Tout en conservant un cas d'utilisation réel car les images sont naturelles et prises avec un dispositif de capture usuel, en effectuant des captures d'images à une distance égale de notre objet, et en faisant ces captures tous les 20° en moyenne.

Concernant la scène nous l'avons aussi travaillée afin de favoriser l'utilisation des points SIFT. Nous avons ajouté de la texture sur l'objet central, et placé des imagettes texturées tout autour afin d'avoir des repères plus étalés autour de l'objet d'intérêt. L'intérêt des imagettes est triple : elles permettent d'avoir des points d'intérêt en plus dans l'image capturée, de reconnaitre plus facilement la scène, mais aussi lors du calcul de la pose de la caméra elles apportent une information géométrique capitale ; qui est de savoir si la matrice de projection calculée est cohérente sur toutes les régions de l'image. Sur celle ci nous pouvons tester nos optimisations et prendre des séquences vidéos à notre guise (à savoir en moyenne un nuage de point est composé entre 6000 et 12000 séries dans notre cas). Pour une image entrante nous avons de 1000 à 5000 points mis en correspondance. Nous avons effectué ici nos expérimentations sur un jeu de donnée différent de celui utilisé pour l'article [ERSB15], les résultats sont donc sensiblement différents.

3.3.1 Test de la robustesse de nos estimations sans optimisation

La base de données de Strecha [SvHVG+08] nous fournit une vérité terrain avec une série d'images et leurs poses. Nous nous servons de cette base pour tester la robustesse de nos estimations.

Nous avons mis une image de côté I_i, et nous avons utilisé les 10 autres comme de la base de pré-taitement. Sur cette image I_i nous nous sommes servis de notre implemtentation pour calculer P_{est}, la matrice de projection estimée. La vérité

terrain nous donne la vraie matrice de projection P_{VT}. Pour vérifier la précision de nos calculs, nous avons projeté l'ensemble des points $\{X_1,X_2,...,X_n\}$ du nuage sur le plan focal de l'image. Cela se représente comme suit :

$$\forall i \in \{1,...,n\} \begin{cases} x_{VT_i} = P_{VT} X_i \\ x_{est_i} = P_{est} X_i \end{cases} \tag{3.2}$$

Pour tout i appartenant à {1,...,n} nous avons calculé l'erreur de reprojection e_i de la façon suivante :

$$e_i = ||x_{VT_i} - x_{est_i}||^2 \tag{3.3}$$

Nous avons ainsi une mesure de la précision de nos calculs. L'erreur de reprojection moyenne est de 0.6252 en pixel. Nous avons dressé un histogramme des erreurs de reprojection (figure 3.4). Nous constatons que, pour l'ensemble des points, elle est inférieure à l'ordre du pixel.

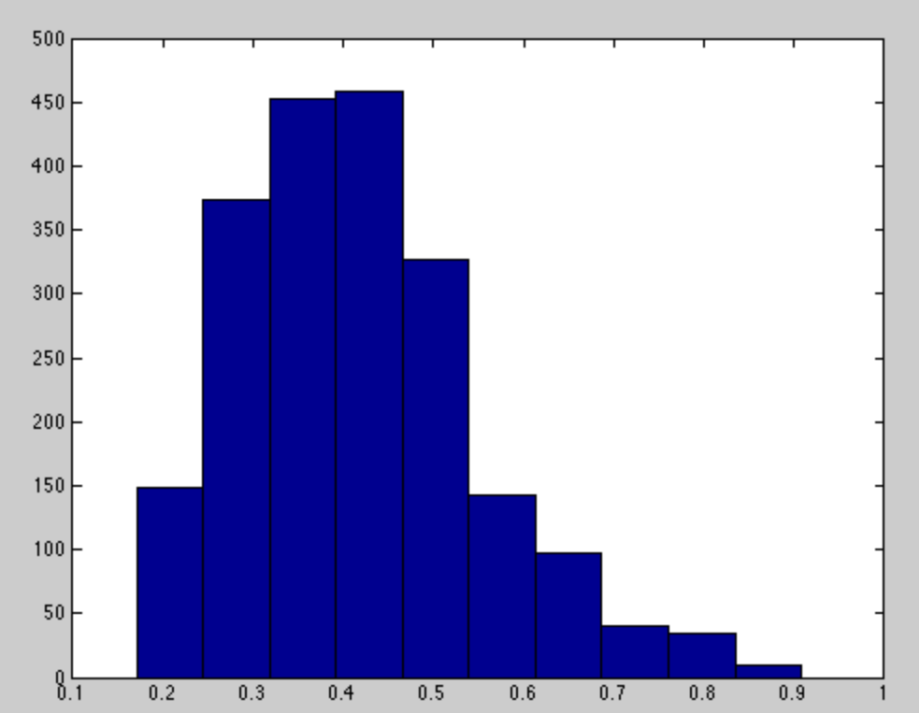


FIGURE 3.4 – histogramme de la répartition des erreurs avec 100% des points SIFT et se servant de toute la base de données. En abscisse : l'erreur de reprojection. En ordonnée : nombre d'éléments dans cette classe

Pourcentage de points SIFT		100%	80%	40%	20%	10%
Temps de convergence	B_d complète	9.77 s	7.29 s	3.81 s	1.93 s	1.01 s
	images pertinentes	1.69 s	1.36 s	0.78 s	0.46 s	0.31 s
Précision	B_d complète	0 px	2.5 px	3.6 px	4.1 px	3.9 px
	images pertinentes	2.65 px	3.2 px	3.8 px	3.0 px	4.9 px

TABLE 3.2 – table mettant en avant les performances constatées suite aux optimisations.

3.3.2 Test de la robustesse de nos estimation avec optimisations

Nous allons dans un premier temps décrire la méthodologie de test suivie. Ici nous avons pris un ensemble de 10 frames consécutives (de résolution 5312×2988) et avons calculé la pose de chaque frame sous plusieurs conditions :

- Nous avons estimé la pose de la frame f_k en conservant 100%, 80%, 40%, 20%, 10% des points SIFT comme vu dans la section 3.2.1
- Pour chacun des seuils, nous avons calculé la pose en prenant respectivement toutes les images de la base ou bien en ne prenant que les images jugées pertinentes par notre méthode comme vu dans 3.2.3

Une fois tous les calculs effectués, nous comparons les résultats trouvés dans le cas où 100% des points SIFT sont conservés et en utilisant toute la base de données B_d. C'est l'approche que nous avons validée dans la section 3.3.1 avec la base d'images de Strecha. À noter que la base B_d contient au total 41 images.

Le tableau 3.2 présente les résultats en terme de performances de notre approche. La première ligne indique le nombre de points SIFT conservés. La seconde le temps mis pour opérer le calcul de la pose, la troisième l'erreur de reprojection moyenne calculée vis à vis de la pose calculée dans le cas où 100% des points SIFT sont conservés et toutes les images de la base de données B_d sont utilisées. Le temps et l'erreur de reprojection sont considérés respectivement en utilisant toute la base ou bien simplement les images pertinentes. Le tableau 3.3 quant à lui présente certains résultats observable

SIFT conservés		mise en pair (sec)	filtre ACRANSAC (sec)	calcul de la pose (sec)	temps total (sec)	precision (pixel)
100%	B_d complète	8.435	1.422	0.027	9.88	**0**
	2 voisins	**0.502**	**0.0459**	**0.024**	**0.572**	2.02
	5 voisins	1,222	0,201	0.027	1.45	1.72
	20 voisins	4.274	0.883	0.025	5.18	1.59
80%	B_d complète	6.643	0.79	**0.0201**	7.4531	1.48
	2 voisins	**0.402**	**0.04572**	0.024	**0.472**	3.29
	5 voisins	0.967	0.181	0.026	1.174	**1.20**
	20 voisins	3.433	0.672	0.025	4.13	2.12
40%	B_d complète	3.326	0.381	**0.0192**	3.73	2.5009
	2 voisins	**0.201**	**0.0455**	0.025	**0.272**	2.0850
	5 voisins	0.488	0.177	0.026	0.691	3.3551
	20 voisins	1.71	0.438	0.0283	2.18	**1.5311**
20%	B_d complète	1.66	0.193	**0.0226**	1.87	**1.4493**
	2 voisins	**0.102**	**0.04532**	0.026	**0.173**	1.5139
	5 voisins	0.250	0.103	0.026	0.379	2.4387
	20 voisins	0.858	0.218	0.029	1.11	2.337
10%	B_d complète	0.829	0.172	**0.019**	1.02	**2.1567**
	2 voisins	**0.0513**	**0.0449**	0.0241	**0.120**	3.0411
	5 voisins	0.119	0.0870	0.0224	0.228	2.2007
	20 voisins	0.431	0.193	0.0265	0.651	2.2408

TABLE 3.3 – table mettant en avant les performances constatées suite aux optimisations.

avec la mise en place de nos optimisations. La première colonne indique le nombre de points SIFT conservés. La seconde le temps mis pour calculer les correspondances, la troisième le temps nécessaire pour opérer le filtre ACRANSAC, la quatrième celui pour calculer la pose. L'analyse du tableau 3.2 nous indique qu'utiliser 20% des points SIFT, en combinant avec un choix cohérent des images que nous sélectionnons (ici le cas où nous prenons deux voisins), nous offre un bon compromis entre précision et temps de calcul. Pour une image de taille 5312×2988 une erreur de l'ordre de 3 pixels est acceptable.

Nous atteignons un seuil lorsqu'il n'y a plus assez de points. En effet, le filtrage d'ACRANSAC trouve alors moins facilement un consensus parmi les données.

Le tableau 3.3 présente également une extension que nous avons effectuée en prenant des voisinages de tailles différentes. Nous en tirons la conclusion, comme l'on peut s'y attendre, que le voisinage a un impact considérable sur les temps de calcul. En ce qui concerne la précision de la méthode, le constat est plus mitigé. La tendance globale vise à améliorer la précision, cependant en considérant le temps mis ce gain reste marginal. En analysant ce tableau plus en détails nous constatons que la mise en paire de points est toujours l'étape qui prend le plus de temps. Nous constatons également sans surprise que les résultats ayant un voisinage de deux images sont les plus rapides, quel que soit le pourcentage de points SIFT conservés. En terme de précision les résultats sont plus subtils à analyser. Premièrement notons que nous avons utilisé la base complète en utilisant 100% des points SIFT comme référence, nous comparons toutes les autres valeurs à celle-ci. En effet dans la partie 3.3.1 nous avons validé la robustesse de celle-ci, nous nous en servons donc comme base de comparaison pour ce qui suivra. Lorsque nous conservons une grande partie des points SIFT (jusque 40%), il semble que le fait de prendre de petits voisinages (2 à 5 images voisines) aide à obtenir une meilleure précision. Cela peut s'expliquer par le fait qu'en limitant le nombre d'images à traiter nous limitons également le taux de correspondances aberrantes. Par contre dès que l'on réduit fortement le nombre de points SIFT conservés (20% et moins), avoir un voisinage constitué d'un grand nombre d'image s'avère nécessaire. En effet à ce moment il n'y a plus suffisamment de points dans la base, donc en prenant l'ensemble des images il y a plus de chances de trouver des correspondances exactes.

Dans la figure 3.5 nous pouvons observer la reprojection des points 3D sur les images et le recalage effectué. Pour chaque image, nous avons utilisé la matrice de pose estimée afin de reprojeter le nuage de points dessus.

3.3.3 Conclusion

Dans ce chapitre, nous nous sommes intéressés au calcul de la pose d'une prise de vue dans le contexte de la réalité augmentée. Dans ce cadre, la rapidité des traitements est primordiale, mais elle ne doit pas mettre en péril la robustesse des résultats obtenus. Pour répondre à cette problématique, nous avons d'abord formulé le problème en déportant la tâche de plus grande complexité, à savoir les correspondances $2D$-$3D$, dans une étape de pré-traitement. Nous avons de plus mis en place des structures permettant de réduire le calcul de la pose d'une image, tout en conservant une approche basée sur des points SIFT.

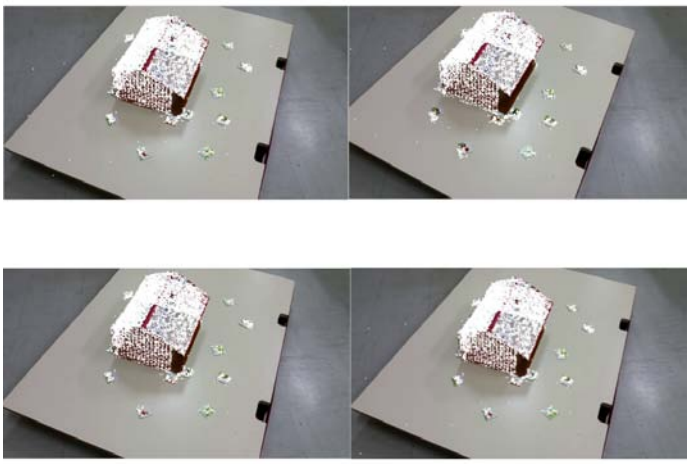


FIGURE 3.5 – en haut à gauche : reprojection en gardant 100% des points avec toute la base. En haut à droite : reprojection en gardant 10% des points en ne gardant que les images pertinentes. En bas à gauche : reprojection en gardant 20% des points avec toute la base. En bas à droite : reprojection en gardant 20% des points en ne gardant que les images pertinentes.

Nous sommes ainsi passés de 9.77 s à 0.46 s pour la totalité des étapes du calcul de la pose d'une image. La perte en précision occasionnée par cette optimisation n'est que de 3 pixels sur une image de résolution 5312x2988. Néanmoins, plusieurs approches restent à explorer pour améliorer encore les résultats. On pourra envisager entre autre le suivi via un filtrage de Kalman. Par ailleurs, la réduction du temps d'initialisation étant également essentielle, il pourrait également être envisageable d'utiliser notre structure de graphe en passant par des structures hiérarchiques [BK12]. Le résultat que nous avons obtenu, ici de passer d'un temps de calcul de 9.77 s à 0.46 s, est une bonne évolution mais cela reste loin de ce que l'on peut espérer afin d'obtenir un flux vidéo qui puisse paraître fluide à l'oeil humain. En effet, il faudrait au moins pouvoir traiter 25 images par seconde pour avoir suffisamment de fluidité, cela revient à une image toutes les 0.04s. D'autres opérations, telles que l'acquisition de l'image ou encore la détection de points d'intérêt dans cette image, sont par ailleurs nécessaires. L'idéal serait de pouvoir effectuer notre tâche dans un laps de temps de l'ordre de 0.02s à 0.03s. Ainsi après avoir réduit le temps de calcul autant que possible en nous servant d'informations dont nous disposons, nous n'atteignons pas encore des résultats suffisamment proche du temps réel. Deux alternatives sont envisageables. La première consiste à considérer les méthodes utilisant le GPU afin d'accélérer les calculs comme vu dans [Wu07], en gardant

en tête qu'à l'heure actuelle la capacité en terme de GPU des appareils mobiles reste limitée. Il existe aussi la possibilité d'utiliser des descripteurs plus rapides tels que les descripteurs binaires. Nous nous sommes naturellement tournés vers la seconde approche dans la mesure où elle nous semble plus compatible avec notre problématique. Nous présentons dans le chapitre 4 l'approche que nous avons élaborée.

Chapitre 4

Contribution aux descripteurs binaires

Ici nous décrivons un schéma générique basé sur la théorie de l'information qui englobe la plupart des descripteurs binaires. Nous étudions le calcul et l'appariement des points d'intérêt dans une perspective temps réel. Pour cela nous avons créé un descripteur binaire qui tout en étant rapide à calculer et à mettre en correspondance reste robuste à certaines déformations de l'image. Le principe sur lequel nous nous appuyons est de sélectionner les tests qui nous semblent les plus pertinents. Pour déterminer si un test est pertinent ou non, nous définissons des critères, liés à la distribution de l'information sur une base de données d'apprentissage. Nous illustrerons, en particulier, comment le descripteur BOLD [BTM15] peut être retrouvé comme un cas particulier de ce schéma. Nous en analysons aussi les limitations. En nous basant sur cette analyse, nous développons une extension possible.

Dans la section 4.1, nous établissons un cadre général comprenant notre approche que nous décrivons en détail. Ensuite nous fournissons une analyse en profondeur quant aux observations qui nous ont menées à notre conclusion. Dans la section 4.2, nous démontrons l'intérêt de nos contributions sur plusieurs benchmarks standards. Nous comparons notre descripteur avec BOLD [BTM15], notre compétiteur le plus direct. Pour résumer, notre descripteur nous permet d'obtenir des résultats similaires avec deux fois moins de tests. De plus nous fournissons une comparaison avec une large palette de descripteurs classiques tels que SIFT [Low99], SURF [BTVG06], LIOP [WFW11] and BRISK [LCS11].

4.1 Descripteur binaire robuste

Dans ce qui suit nous dénotons un patch par la lettre $w \in \mathbb{R}^{s\times s}$, s étant la taille du patch. Un test est une fonction $t : \mathbb{R}^{s\times s} \to \{0, 1\}$ qui associe un patch à une valeur binaire $t(w)$. La figure 4.1 donne une représentation de ce qu'est un test. Pour un patch w_k, on peut appeler test le résultat de la comparaison de deux pixels p_1 et p_2 du patch. Ainsi dans l'illustration la case en bleue représente $t_i(w_k)$ qui est le résultat de la comparaison des pixels p_1 et p_2 (les cases vertes et rouges), le test aura pour valeur 1 si p_1 est plus petit que p_2 et 0 sinon. Une base de données est une collection de patchs $\mathcal{D} = \{w_1, \cdots, w_d\}$ tirés d'une distribution commune (inconnue). Dans la figure 4.2 nous avons un exemple de collection de patchs. Étant donné N tests $t_1, \cdots, t_N$ nous posons $x_{k,i} = t_i(w_k)$ la collection d'échantillons binaires obtenue en appliquant chaque test à tous les patchs. Par souci de clarté, nous posons de manière générique en majuscule W la variable aléatoire suivant la même distribution que les patchs, et $X_t = t(W)$ la variable de Bernoulli induite par le test t.

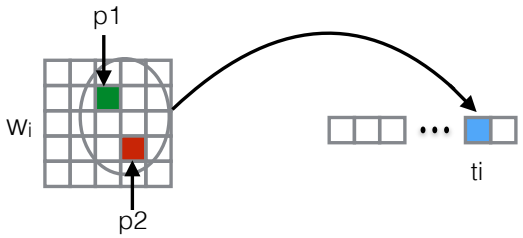


FIGURE 4.1 – Représentation d'un test

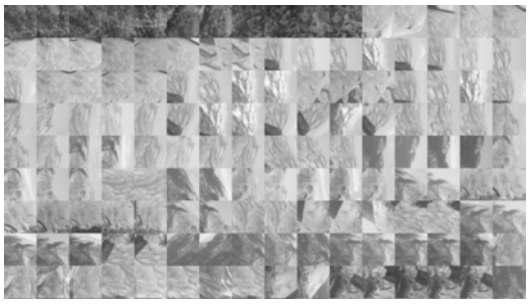


FIGURE 4.2 – Représentation d'une collection de patchs

4.1.1 Mécanisme de sélection

Notre but principal est, disposant de deux patchs w_1, w_2, de décider si ils sont similaires à certaines transformations près. Idéalement, les transformations à

prendre en compte sont les distorsions des patchs dûes aux changements de point de vue. Cela est représenté dans la figure 4.3. En effet deux patchs représentant une même zone de l'espace pourraient sembler différents à cause de ces transformations. Dans la figure 4.3 on peut en juger directement, le point de l'espace P_j est projeté sur les points $p_{j,k-1}$, $p_{j,k}$ et $p_{j,k+1}$ des plans focaux des caméras $K-1, K$ et $K+1$. Bien que représentant le même point P_j, chacun des patchs entourant les points $p_{j,k-1}$, $p_{j,k}$ et $p_{j,k+1}$ auront des distorsions les uns par rapport aux autres. Pour mesurer la similarité entre différents patchs, nous calculons une distance entre w_1 et w_2. La distance construite doit être capable de prendre en compte ce type de transformations, afin de mettre des patchs en correspondance. Nous mettons les patchs en correspondance si la distance entre ceux-ci est en dessous d'un certain seuil. Dans les faits, des descripteurs binaires sont calculés pour les deux patchs. La distance, elle, est calculée entre ces deux descripteurs. Cela revient en somme à comparer bits à bits les valeurs des descripteurs créés. La distance peut prendre plusieurs formes ; dans les équations 4.2 et 4.4 nous en avons différentes déclinaisons. Afin d'apprendre une bonne métrique il est nécessaire de choisir de bons tests, cela est typiquement fait lors d'une phase hors ligne. L'idée dans la partie hors ligne est de choisir un nombre N de tests qui apportent autant d'information que possible. Dans l'idéal, ces tests seront choisis par un apprentissage optimal sur une base de données représentative. Pour cela une approche gloutonne est intéressante, les tests sont choisis de façon itérative en maximisant une mesure liée à la quantité d'information apportée par le nouveau test relatif aux tests préalablement sélectionnés. Cette procédure est définie dans l'algorithme 5.

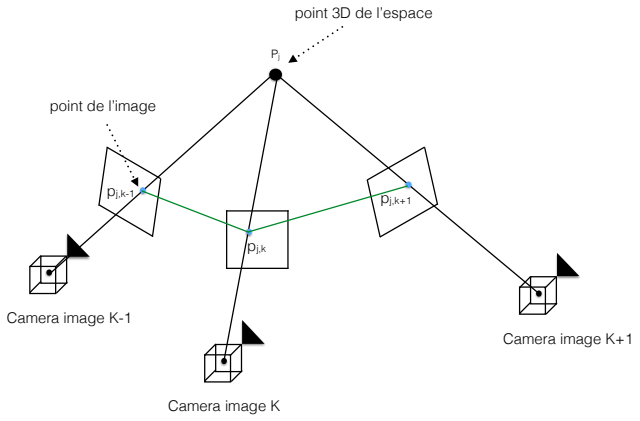


FIGURE 4.3 – patchs sous différents points de vue

Le critère de sélection des tests $J_{\mathcal{D}}(t|\mathcal{S})$ pourrait, à titre d'exemple, être une estimation de l'entropie conditionnelle $H(X_t|X_{\mathcal{S}})$. $X_{\mathcal{S}}$ serait ici la collection de variables aléatoires induites par le jeu de tests sélectionnés. De prime abord

Algorithme 5 : Algorithme hors ligne

input : base de données de patchs $\mathcal{D}$
output : $\mathcal{S}$ tests sélectionnés

1 génération d' un jeu $\mathcal{P} = \{t_1, \cdots, t_M\}$ de M tests aléatoires
  // $M \gg N$
2 $\mathcal{S} = \varnothing$
3 **for** $i = 1..N$ **do**
4 $t_i^* = \arg\max_{t \in \mathcal{P}}(J_{\mathcal{D}}(t|\mathcal{S}))$
5 $\mathcal{S} = \mathcal{S} \cup \{t_i^*\}$

une mesure de ce type semble très adaptée à notre problème. L'entropie conditionnelle, pouvant être utilisée afin de déterminer si deux variables aléatoires sont indépendantes (donc décorrélées) ou non, nous permettrait de choisir les tests apportant le plus d'information tout en étant décorrélés les uns par rapport aux autres. Néanmoins calculer l'entropie conditionnelle demande l'estimation de probabilités jointes, ce qui est coûteux. Certaines hypothèses sur l'indépendance conditionnelle (par exemple la dépendance par paire) peuvent rendre cette tâche plus simple, mais cela reste exigeant en terme de calculs. En pratique [RRKB11], il est préférable d'utiliser un critère de la forme :

$$J_{\mathcal{D}}(t|\mathcal{S}) = J_{\mathcal{D}}(t) - \infty \mathbf{1}_{|max_{s \in \mathcal{S}}(\text{corr}(X_t, X_s))| > \tau} \tag{4.1}$$

Dans cette formule $J_{\mathcal{D}}(t)$ représente la quantité d'information apportée par le test. $\mathbf{1}_{|max_{s \in \mathcal{S}}(\text{corr}(X_t, X_s))| > \tau}$ représente quant à elle la correlation du test t avec ceux de $\mathcal{S}$ déjà sélectionnés. Si t est fortement corrélé (le niveau de correlation est fixé par la constante τ) avec un seul des tests $s \in \mathcal{S}$ il sera rejeté (on lui associe $-\infty$). Un tel critère se prête à une implémentation efficace. Il pourrait être envisageable, par exemple, de maximiser la première partie $J_{\mathcal{D}}(t)$ parmi les tests qui se conforment aux seuillage sur la correlation. $J_{\mathcal{D}}(t)$ peut être une mesure de l'entropie, mais d'autres mesures existent. Par exemple, $J_{\mathcal{D}}(t) = \text{var}(X_t)$ ou $J_{\mathcal{D}}(t) = |\mathbb{E}(X_t) - 0.5|$ lui sont souvent préférés, car plus rapides à mettre en œuvre. À noter que, pour des variables de Bernoulli, ces mesures sont équivalentes.

Pour calculer la distance entre les patchs w_1 et w_2, la procédure typique de mise en correspondance consiste à calculer une distance de Hamming entre les résultats des tests $x_1, x_2 \in \{0,1\}^N$:

$$d_H(x_1, x_2) = \sum_{i=1}^{N} x_{1,i} \oplus x_{2,i} \tag{4.2}$$

Où $\oplus$ est l'opérateur *XOR* la somme peut être calculée de manière efficace grâce à la routine *popcount*.

BOLD [BTM15] améliore la robustesse aux transformations naturelles en passant par une sélection en ligne. Cela implique la mise en place et l'application d'une distance de Hamming modifiée. Cela se fait en calculant p versions transformées $w_k^1, \cdots, w_k^p$ de chaque patch w_k ($k \in \{1, 2\}$). Suite à cela, un masque $y_k \in \{0, 1\}^N$, permettant de filtrer les bits correspondants aux tests non robustes, est construit comme suit :

$$y_{k,i} = \bigoplus_{j=1}^{p} x_{k,i}^j \text{ avec } x_{k,i}^j = t_i(w_k^j) \tag{4.3}$$

Dans la formule $\bigoplus$ représente le *XOR* n-aire (vrai quand tous ces arguments sont égaux). Cette opération permet de distinguer dans le filtre les tests qui sont restés invariants aux changements induits par les versions transformées des patchs. De façon intuitive, pour un patch donné, ces tests sont les plus robustes aux transformations.

En se basant sur ce critère une distance de Hamming masquée est mise en place. Celle ci est faite pour ne prendre en compte que les tests robustes aux transformations choisies :

$$d_{masked}(x_1, x_2; y_1, y_2) = \sum_{i=1}^{N} \lambda_1 y_{1,i} \wedge (x_{1,i} \oplus x_{2,i}) + \lambda_2 y_{2,i} \wedge (x_{1,i} \oplus x_{2,i}) \tag{4.4}$$

où les poids[1] sont exprimés ainsi $\lambda_k = \frac{|y_k|}{|y_1|+|y_2|}$ avec $|y_k|$ le nombre de 1 dans y_k.

4.1.2 Analyse du mécanisme de sélection et améliorations proposées

Le fait de construire le descripteur sur une base de N tests fixés permet de faciliter la recherche de l'équilibre entre performance et temps de calcul. De plus après un nombre critique de tests, un phénomène de saturation est typiquement observé. À ce moment les performances stagnent voir régressent. Ce phénomène dépasse le cadre des descripteurs binaires et est partagé par toutes les méthodes basées sur l'analyse de données. Par exemple, dans le cas du descripteur GLOH [MS05] (basé sur la réduction de dimension), les résultats sont moins bons pour

[1] La formule correspond à l'implémentation fournie par les auteurs de BOLD et les poids y sont différents de ceux exposés dans leur article.

$\mathcal{D}'$		**estimate**	**l.b.** 95%	**u.b.** 95%
ND	a	1.083	1.078	1.088
	b	-0.043	-0.046	-0.040
Liberty	a	1.004	0.997	1.011
	b	-0.0002	-0.004	0.004

(a) $p^{\mathcal{D}}_{x_t}$ vs $p^{\mathcal{D}'}_{x_t}$

$\mathcal{D}'$		**estimate**	**l.b.** 95%	**u.b.** 95%
ND	a	1.007	1.001	1.014
	b	0.016	0.012	0.020
Liberty	a	0.938	0.930	0.946
	b	0.036	0.031	0.041

(b) $p^{\mathcal{D}}_{y_t}$ vs $p^{\mathcal{D}'}_{y_t}$

TABLE 4.1 – Regressions linéaires (avec intervalles de confiance à 95%) de la forme $p^{\mathcal{D}'} = ap^{\mathcal{D}} + b$ avec Yosemite pour $\mathcal{D}$ et deux jeux de donnés alternatifs $\mathcal{D}'$.

un descripteur de taille 272 que pour l'alternative de taille 128. Cette saturation peut être observée dans la Figure 4.5-(a) pour une sélection en ligne maximisant la variance.

Les méthodes de sélection se basant sur l'analyse de données s'appuient sur le fait que certaines estimations de distribution se généralisent bien d'une base de données à une autre. En particulier il est important que la probabilité de succès d'un test p_{x_t} soit préservée à travers les bases de données. Cela assure la préservation des tests possédant le plus d'informations pertinentes. Dans la Table 4.1-(a) nous représentons la regression linéaire entre $p^{\mathcal{D}}_{x_t}$ et $p^{\mathcal{D}'}_{x_t}$ pour plusieurs combinaisons de bases de données $(\mathcal{D}, \mathcal{D}')$ décris dans [HBW07]. Elle montre avec un taux de confiance élevé que la relation entre les probabilités est bien approchée par la fonction identité. Cela signifie qu'un même test d'une base de données à une autre aura approximativement la même probabilité de succès, et donc qu'il y a bien conservation des distributions. Nous pouvons aussi l'observer visuellement sur la figure 4.4.

À la saturation BOLD [BTM15] obtient de meilleurs résultats en ne prenant pas en compte les bits du descripteurs qui ne sont pas robustes à certaines déformations naturelles. Cette sélection de tests est faite entièrement en ligne car la robustesse d'un test dépend directement du patch choisi. Cependant on observe toujours un phénomène de saturation voir figure 4.5-(b). Dans cette figure, une fois que $N = 512$ tests sont atteints on ne remarque plus de gain en performance.

Bien que la sélection en ligne fonctionne en se basant sur un patch donné, il est intéressant de noter que certains tests sont statistiquement plus robustes aux transformations géométriques que d'autres. Ce fait devient flagrant lorsque l'on observe la généralisation des probabilités p_{y_t} qu'un test t soit gardé par

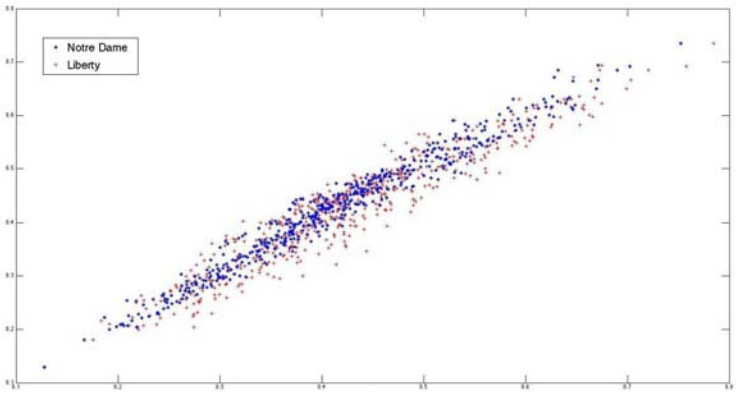
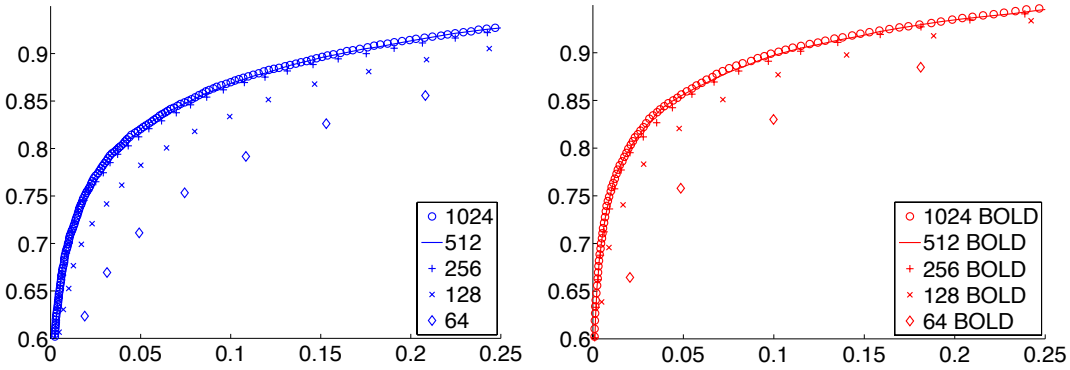


FIGURE 4.4 – Courbe représentant la fréquence que la réponse d'un test soit 1 dans les bases Notre Dame (en bleu) et Liberty (en rouge) en fonction de la fréquence que la réponse d'un test soit 1 dans Yosemite



(a) Sélection hors ligne (b) Sélection en ligne de BOLD

FIGURE 4.5 – Saturation des courbes ROC pour la sélection hors ligne et BOLD.

la partie en ligne (voir la Table 4.1-(b) concernant la regression linéaire pour plusieurs base de données différentes). Puisque p_{y_t} se généralise bien d'une base de données à une autre, les tests qui sont robustes aux transformations géométriques peuvent aussi être appris hors ligne. Le principe est donc de prendre en compte le filtrage en ligne lors de l'apprentissage hors ligne. Sans cela, la quantité d'informations estimée hors ligne ne représente pas l'information effectivement conservée en ligne. Nous proposons une version modifiée de la mesure de la quantité d'informations, adaptée à la sélection en ligne. Elle se décline comme suit :

$$H_{masked}(t) = -\left[p_{x_t}\log(p_{x_t}) + (1-p_{x_t})\log(1-p_{x_t})\right] \times p_{y_t} \tag{4.5}$$

Dans cette équation $p_{x_t} = p(X_t = 1)$ est l'estimation probabiliste que le test t soit un succès. L'interprétation de l'équation 4.5 est directe. D'un côté, $-(p_{x_t}\log(p_{x_t}) + (1-p_{x_t})\log(1-p_{x_t}))$ représente la quantité d'informations apportée par le bit t indépendemment de la phase online. D'un autre côté, cette partie est multipliée par p_{y_t} dans la mesure où l'information sera rejetée avec la probabilité $1-p_{y_t}$. Ici y_t est exprimé comme dans l'équation 4.3. Afin d'obtenir des tests qui seraient décorrélés, une définition de la quantité d'informations conditionnelles similaire à celle présentée ici est possible. Cependant, par souci d'efficacité calculatoire, nous avons décidé de nous reposer sur un schéma de décorrelation plus direct. Nous utilisons seulement la définition marginale de la décorrelation pour $J_{\mathcal{D}}(t)$, comme vue dans l'équation 4.1. Cette mesure est utilisée pour renforcer le flux global d'informations après la sélection en ligne. Cette caractéristique sera confirmée dans ce qui suit.

4.2 Résultats et apports de notre descripteur

Dans cette section, nous analysons les performances de notre descripteur sur deux bases de données standards. Pour les expériences menées, nous utilisons le filtre en ligne recommandé par BOLD [BTM15] : nous contruisons $p = 3$ rotations (de 20°) dans l'équation 4.3.

4.2.1 base de données Photo Tourism

Dans un premier temps nous présentons les résultats que nous avons obtenus sur le dataset [HBW07] avec le protocole d'évaluation proposé dans [TCFL13]. En particulier, nous comparons notre méthode avec le descripteur BOLD [BTM15] (en se basant sur son implémentation originale). Dans la figure 4.6 l'apprentissage des tests du descripteur est effectué sur le dataset Yosemite, et le descripteur a été testé sur un ensemble de 130k patchs de taille 32×32 tirés de la base de données Notre Dame. Sur cette figure on peut constater que notre descripteur avec 256 tests atteint d'aussi bons résultats que BOLD avec 512 tests. De son côté, BOLD avec 256 tests obtient de moins bons résultats. Nous pouvons de plus constater que les tests sélectionnés par notre méthode sont plus informatifs que ceux produits par BOLD. Nous remarquons en effet un écart conséquent entre notre méthode et celle de BOLD lorsque 128 tests sont concaténés. Notre méthode a une performance proche de la saturation tandis que BOLD est plus proche du régime hors ligne (ligne verte).

Nous avons testé notre approche et celle de BOLD [BTM15] sous différentes configurations (combinaisons de bases d'apprentissages/tests). Nous avons

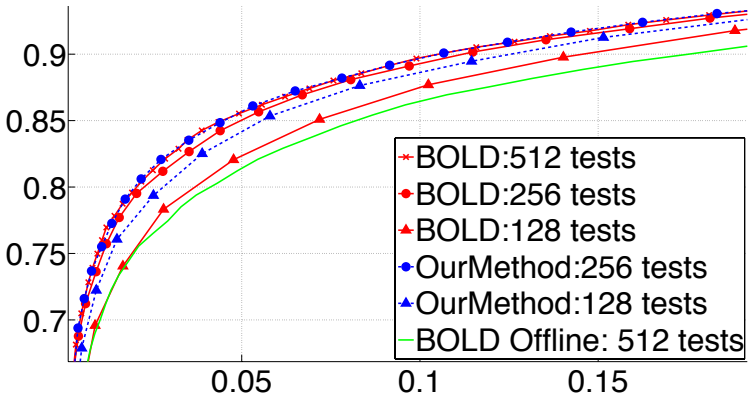


FIGURE 4.6 – Courbes ROC pour notre descripteur et BOLD sous différents régimes.

reporté les aires sous les courbes ROC dans les tableaux 4.2a, 4.2b et 4.2c. Dans toutes les configurations, nous obtenons d'aussi bons, si ce n'est de meilleurs, résultats que BOLD. Nous constatons en particulier que notre descripteur obtient de bien meilleurs résultats lorsque le nombre de tests dans le descripteur est plus petit. Aussi BOLD atteint la saturation plus lentement. À noter que nous atteignons la saturation autour de 256 tests avec notre descripteur contre 512 pour BOLD.

4.2.2 base de données Vgg

Ici nous évaluons notre descripteur sur le jeu de données proposé dans [MS05]. Ces bases de données présentent différentes conditions de tests comme des changements d'illuminations, des rotations, des zooms, du floutage et de la compression jpeg. Nous comparons également notre descripteur avec d'autres approches standards et récentes. Pour SIFT [Low99] et LIOP [WFW11], nous avons utilisé l'implémentation de vlfeat. Nous nous sommes aussi servi de l'implémentation de la "computer vision toolobox" de Matlab pour BRISK [LCS11] et pour SURF [BTVG06]. Afin que les comparaisons fassent sens, tous les descripteurs sont calculés à partir des mêmes points clés extraits par le détecteur de Harris-Laplace multi échelle [MS04]. Le tableau 4.3 montre les aires sous les courbes ROC pour différents descripteurs et différentes paires de patchs que l'on met en correspondance en utilisant la méthode du plus proche voisin. Puisque la contribution de notre méthode est liée à l'amélioration de la robustesse vis à vis des rotations, et ce dans le cadre de descripteurs binaires, nous avons organisé le tableau comme suit. Verticalement, les colonnes sont ordonnées selon le niveau de changement d'orientation entre

	Notre Dame		Liberty	
bits	**Bold**	**Nous**	**Bold**	**Nous**
1024	**0.959**	**0.959**	**0.941**	**0.941**
512	0.958	**0.959**	0.941	**0.942**
256	0.957	**0.959**	0.941	**0.942**
128	0.950	**0.954**	0.934	**0.939**
64	0.932	**0.943**	0.916	**0.929**

(a) Apprentissage sur Yosemite

Yosemite		Liberty	
Bold	**Nous**	**Bold**	**Nous**
0.951	**0.952**	0.939	**0.940**
0.952	**0.954**	0.939	**0.943**
0.951	**0.955**	0.939	**0.943**
0.946	**0.951**	0.933	**0.940**
0.930	**0.941**	0.916	**0.931**

(b) Apprentissage sur Notre Dame

Notre Dame		Yosemite	
Bold	**Nous**	**Bold**	**Nous**
0.957	**0.959**	0.952	**0.954**
0.956	**0.960**	0.953	**0.955**
0.955	**0.958**	0.952	**0.954**
0.947	**0.953**	0.944	**0.949**
0.933	**0.940**	0.931	**0.938**

(c) Apprentissage sur Liberty

TABLE 4.2 – Aires sous les courbes PR (les valeurs sont arrondies à 3 décimales).

paires d'images. Hormis pour les expériences sur la compression JPEG (les paires UBC), toutes les paires d'images correspondent à deux prises de vue indépendantes. Elles présentent donc des changements géométriques induisant des rotations d'intensité variable. Nous avons extraits 3 régimes de rotations caractéristiques. Ensuite horizontalement, les tests binaires sont séparés des descripteurs basés sur valeurs flottantes. Parmi les descripteurs binaires, nous avons calculé les résultats de BRISK [LCS11] un descripteur récent reposant sur une construction ad hoc. Nous avons également calculé plusieurs descripteurs basés sur le schéma présenté dans 4.1.1. Les dérivées du schéma 4.1.1 comprennent plusieurs mécanismes pour faire face aux rotations. Les deux premières variations du schéma correspondent à notre descripteur (avec 512 et 256 tests) et BOLD [BTM15]. Ensuite les deux autres variations ne s'intéressent qu'à la selection hors ligne. La première des deux subit une compensation de l'orientation sur le patch tandis que l'autre est appliquée directement sur le patch non transformé. Concernant les descripteurs à valeurs flottantes ils utilisent par construction des compensations explicites de la rotation. Nous

avons mis en gras le meilleur résultat parmi les descripteurs binaires d'un côté, et ceux à valeurs flottantes d'un autre côté.

Dans le premier régime (celui avec de très petits changements d'orientation), la sélection en ligne (BOLD [BTM15], Nous512, Nous256) diminue légèrement les performances par rapport à l'approche n'utilisant que la méthode hors ligne. Une compensation explicite de l'orientation exacerbe la détérioration des résultats. Cette observation renforce l'idée que forcer des invariances non nécessaires n'est souvent pas bénéfique. Le second régime (rotation d'angle moyens jusque 20°), est celui pour lequel la sélection en ligne a été pensée. Il n'est donc pas surprenant de voir que c'est dans ce régime que BOLD [BTM15] et notre approche excellent particulièrement parmi les descripteurs binaires. Ils peuvent d'ailleurs être comparés de façon favorable à certains descripteurs à valeurs flottantes. Il est à noter que la compensation de l'orientation est moins efficace que la sélection en ligne. De plus dans ce régime notre méthode réalise de meilleures performances que BOLD grâce à l'entropie modifiée proposée dans l'équation 4.5. Dans le troisième régime, la sélection en ligne ne peut pas surmonter l'intensité des rotations appliquées. À ce moment, seule la compensation de la rotation est avantageuse, avec un large avantage pour SIFT [Low99]. Pour conclure, à une exception près, la comparaison entre nous et BOLD est uniformément à notre avantage. Cela a d'ailleurs été confirmé par certaines expériences synthétiques que nous avons menées, surtout entre 15° et 30°.

4.2.3 Expériences synthétiques

Dans ce cas ci, nous travaillons toujours dans le même contexte que dans la partie 4.2.2. C'est à dire en utilisant le benchmark présenté dans [MS05]. Nous nous concentrons ici sur l'effet des rotations sur une image, afin d'en isoler l'impact. Pour ce faire, nous avons sélectionné une image de chaque base de données et lui avons appliqué une série de rotations synthétiques. Nous avons privilégié ici la comparaison entre notre descripteur et BOLD [BTM15]. Ce choix est motivé par le fait que tous deux sont censés améliorer la robustesse à ce type de rotations. Les descripteurs sont toujours calculés à partir des mêmes points extraits par le détecteur de Harris Laplace. Les tableaux 4.4a et 4.4b illustrent les aires sous les courbes ROC pour nos deux descripteurs, pour des degrés croissants de rotations. Verticalement en colonnes, est représenté l'angle de la rotation entre les deux images de la paire considérée. Pour chaque angle nous fournissons les aires sous les courbes générées par BOLD [BTM15] et notre méthode avec 512 tests. Les différentes bases de données sont présentées sur des lignes séparées. Pour chaque colonne nous considérons une image comme

pair	ubc 1 :5	bikes 1 :5	leuven 1 :5	boat 1 :5	boat 1 :2	bark 1 :2	boat 1 :4	bark 1 :4
angle	0°	6°	7°	8°	14°	31°	79°	120°
Nous512	0.880	0.828	0.820	**0.619**	**0.720**	0.179	0	0
Nous256	0.877	0.832	0.827	0.597	0.703	0.194	0	0
Bold	0.878	0.827	**0.840**	0.605	0.719	0.157	0	0
offline oriented	0.859	0.808	0.796	0.592	0.676	**0.705**	**0.592**	**0.677**
offline	**0.883**	**0.839**	0.830	0.479	0.359	0.012	0	0
BRISK	0.779	0.648	0.647	0.406	0.594	0.668	0.382	0.574
SIFT	**0.865**	**0.859**	**0.880**	**0.749**	**0.698**	0.801	**0.707**	**0.807**
SURF	0.711	0.604	0.553	0.404	0.516	0.582	0.381	0.418
LIOP	0.815	0.792	0.754	0.588	0.662	**0.804**	0.539	0.695

TABLE 4.3 – Aires sous les courbes ROC, ayant utilisé a méthode du plus proche voisin pour apparier les patchs (les valeurs sont arrondies à 3 décimales).

étant la référence. Nous lui appliquons plusieurs rotations d'angle croissant. Nous avons extrait des points sur chacune des images (celle de référence et celles modifiées par rotation), et comparons comme dans la partie 4.2.2. Cela nous a permis de confirmer l'existence de plusieurs régimes et de préciser l'apport de notre méthode. En effet pour de petites rotations (de l'ordre de 5° à 10°), aucune supériorité marquée ne peut être observée entre BOLD et notre approche. Pour des rotations de l'ordre de 15° à 25°(régime pour lequel la méthode de BOLD et la nôtre sont faites pour fonctionner), nous avons clairement l'avantage par rapport à BOLD. Cela confirme que la sélection hors ligne que nous effectuons apporte en robustesse pour des modifications de cet ordre. Au delà de 30° les résultats pour notre méthode et BOLD se dégradent trop fortement pour pouvoir envisager d'utiliser ces méthodes dans la mise en correspondance de patches.

4.2.4 Utilisation de notre descripteur dans le cadre de la réalité augmentée

Constatant les performances encourageantes de notre descripteur, nous avons voulu l'éprouver dans le contexte de la réalité augmentée. Pour cela nous avons

	5°		**10°**		**15°**		**20°**	
	Bold	Nous	Bold	Nous	Bold	Nous	Bold	Nous
trees	0.860	**0.867**	0.826	**0.832**	0.825	**0.832**	0.783	**0.792**
wall	0.871	**0.889**	0.857	**0.866**	0.826	**0.828**	0.796	**0.803**
bark	**0.924**	0.923	**0.925**	0.905	0.895	**0.905**	0.802	**0.836**
boat	0.876	**0.879**	**0.878**	0.876	0.843	**0.848**	0.800	**0.803**
leuven	**0.925**	0.923	0.934	**0.947**	**0.919**	0.913	0.836	**0.850**
ubc	**0.917**	0.909	0.909	**0.914**	0.860	**0.884**	0.810	**0.827**
bikes	0.922	**0.923**	0.916	**0.918**	0.896	**0.907**	**0.884**	0.872

(a) angles de 5 à 20

	25°		**30°**		**35°**	
	Bold	Us	Bold	Us	Bold	Us
trees	**0.577**	0.576	0.065	**0.092**	**0.004**	0.003
wall	0.608	**0.613**	0.129	**0.143**	**0.015**	0.013
bark	0.768	**0.773**	0.4125	**0.522**	0.108	**0.157**
boat	0.557	**0.581**	0.116	**0.145**	0.005	**0.010**
leuven	0.656	**0.678**	0.297	**0.320**	**0.083**	0.073
ubc	0.569	**0.572**	0.149	**0.170**	**0.016**	0.018
bikes	0.665	**0.687**	0.244	**0.274**	0.052	**0.059**

(b) angles de 25 à 35

TABLE 4.4 – Aires sous les courbes PR pour les expériences synthétiques (les valeurs sont arrondies à 3 décimales).

décidé de l'utiliser à place du descripteur SIFT [Low99] dans le mode opératoire introduit dans la partie 3.1.

Afin de nous comparer au mieux au descripteur SIFT [Low99] nous avons décidé de garder le même détecteur dans les deux expériences. Nous pourrons ainsi comparer nos résultats à ceux de la partie 3.1 en ayant comme seul élément changeant le descripteur utilisé pour apparier des points entre images. Dans nos analyses nous ne nous plongeons donc pas en détail dans les observations sur la détection, nous pouvons ceci dit noter qu'ici elle est de l'ordre de 550ms. L'étape la plus longue est la construction de l'espace d'échelle et l'extraction des patches à un flou gaussien donné. Nous pouvons ceci dit considérer des

méthodes optimisées permettant d'estimer au mieux le flou gaussien, comme les box filter [Wel86, GGBW11] ou l'approximation de Deriche [Der93].

Le tableau 4.5 dresse une analyse comparative des résultats concernant les temps d'exécution des différentes phases ainsi que de la précision des matrices de projection estimées. Globalement, nous pouvons constater que les calculs menant à l'estimation de la pose de la caméra sont plus rapides en utilisant notre descripteur. Le calcul du descripteur, en soit, est en moyenne 6 fois plus rapide en utilisant notre méthode. La mise en correspondance, quant à elle, est 2 fois plus rapide. Les autres phases sont plus ou moins comparables. Cela reste logique puisqu'elles ne font pas intervenir le descripteur mais directement les points appariés. En termes de précision nous constatons de manière générale que nos performances sont sensiblement moins bonnes que celles de SIFT [Low99]. Ceci dit en gardant en tête la résolution des images considérée (5312×2988), ces erreurs de reprojection restent acceptables. Il est à noter que pour notre descripteur plus nous filtrons les points, meilleure est la précision. Cela peut s'expliquer par le fait qu'en éliminant certains des points nous restreignons la quantité de points mis en correspondances à tort. Notre descripteur étant moins robuste que celui de SIFT [Low99], il est plus sujet aux erreurs d'approximations.

Nous pouvons donc conclure ici que notre descripteur permet d'approcher les performances atteintes par SIFT [Low99], tout en étant plus rapide.

4.3 Bilan sur les descripteurs

Ce nouveau descripteur a été construit en gardant en tête le challenge du temps réel. Pour le construire nous avons mis en place un socle commun basé sur la sélection de caractéristiques. Ce schéma englobe la plupart des descripteurs binaires construits sur la base de l'analyse de données, incluant en particulier une approche récente nommée BOLD [BTM15]. Nous avons aussi contribué à améliorer la portée du mécanisme de sélection en ligne de BOLD en rendant plus effective la sélection hors ligne appliquée en amont. Ce nouveau mécanisme présente une interprétation en terme de théorie de l'information élégante mais surtout une influence pratique perceptible. En effet, les expériences menées sur deux bases de données standards ainsi que sur une base de données synthétique que nous avons générée, confirment que la méthodologie proposée permet la sélection de meilleures caractéristiques. Elles sont, en effet, à la fois discriminantes et robustes à des transformations géométriques raisonnables. La comparaison directe à BOLD [BTM15] nous indique que, dans la plupart des cas, notre descripteur est aussi discriminant tout en étant deux fois plus compact. Un tel atout est une qualité considérable pour les applications

Points conservés		Description (sec)		mise en pair (sec)		filtre ACRANSAC (sec)		calcul de la pose (sec)		temps total (sec)		precision (pixel)	
		SIFT	Nous	SIFT	Nous	SIFT	Nous	SIFT	Nous	SIFT	Nous	SIFT	Nous
100%	B_d complète	**0.063**	$9.8e^{-3}$	8.43	3.63	1.42	1.10	0.027	0.018	9.94	4.76	**0**	**1.75**
	2 voisins	0.064	0.012	**0.50**	**0.23**	**0.046**	**0.059**	**0.024**	0.019	**0.63**	**0.32**	2.02	5.23
	5 voisins	0.068	0.010	1,22	0.51	0,20	0.22	0.027	**0.017**	1.52	0.76	1.72	5.45
	20 voisins	**0.063**	**$7.4e^{-3}$**	4.27	1.82	0.88	0.71	0.025	0.018	5.24	2.55	1.59	5.35
80%	B_d complète	0.063	$9.5e^{-3}$	6.64	2.77	0.79	0.55	**0.020**	0.017	7.51	3.34	1.48	**2.32**
	2 voisins	**0.063**	0.011	**0.40**	**0.19**	**0.046**	**0.059**	0.024	0.020	**0.53**	**0.28**	3.29	6.43
	5 voisins	0.067	0.010	0.97	0.40	0.18	0.14	0.026	0.017	1.24	0.57	**1.20**	5.09
	20 voisins	**0.063**	**$8.0e^{-3}$**	3.43	1.32	0.67	0.35	0.025	**0.015**	4.19	1.69	2.12	4.40
40%	B_d complète	0.063	$9.1e^{-3}$	3.33	1.36	0.38	0.14	**0.019**	0.020	3.79	1.53	2.50	**2.45**
	2 voisins	**0.059**	0.011	**0.20**	**0.10**	**0.045**	**0.038**	0.025	**0.019**	**0.33**	**0.17**	2.08	5.14
	5 voisins	0.067	0.011	0.49	0.21	0.18	0.078	0.026	0.020	0.76	0.32	3.36	3.69
	20 voisins	**0.059**	**$8.1e^{-3}$**	1.71	0.66	0.44	0.09	0.028	0.021	2.24	0.78	**1.53**	3.85
20%	B_d complète	0.070	$9.3e^{-3}$	1.66	0.73	0.19	0.094	**0.023**	**0.019**	1.94	0.85	**1.45**	**3.58**
	2 voisins	**0.059**	0.011	**0.10**	**0.05**	**0.045**	**0.046**	0.026	0.022	**0.23**	**0.13**	1.51	4.35
	5 voisins	0.062	0.010	0.25	0.11	0.10	0.081	0.026	**0.019**	0.44	0.31	2.44	3.62
	20 voisins	**0.059**	**$9.0e^{-3}$**	0.86	0.35	0.22	0.08	0.029	**0.019**	1.17	0.46	2.34	3.76
10%	B_d complète	0.064	$9.7e^{-3}$	0.83	0.40	0.17	0.061	**0.019**	0.019	1.02	0.49	**2.16**	**2.43**
	2 voisins	0.063	0.011	**0.051**	**0.03**	**0.045**	**0.034**	0.024	0.018	**0.12**	**0.09**	3.04	3.97
	5 voisins	0.061	0.010	0.12	0.061	0.087	0.063	0.022	**0.017**	0.23	0.15	2.20	3.95
	20 voisins	**0.059**	**$8.7e^{-3}$**	0.43	0.19	0.19	0.073	0.026	**0.017**	0.65	0.29	2.24	3.48

TABLE 4.5 – table mettant en avant les performances constatées avec notre descripteur.

considérées. La comparaison avec d'autres descripteurs standards montre que notre approche est favorable dans un certain champs de rotations. Cette situation peut être facilement atteinte pour des applications sur appareils mobiles où obtenir une estimation approximative est souvent possible grâce aux capteurs présents sur ce type de matériel. Notre descripteur est donc très adapté à des applications temps réel. Pour le futur plusieurs extensions sont envisageables. Nous sommes, par exemple, intéressés par l'impact des différentes parties du processus de mise en correspondance. Une des questions centrales est la capacité d'une paire de détecteur–descripteur à faire face au problème typique de la rapidité d'exécution tout en conservant une robustesse et une précision suffisante. Une telle étude demande plus d'attention car comparer les performances de différents détecteurs peut être délicat [RODM15].

Chapitre 5

Conclusion

5.1 Bilan des apports

À l'heure actuelle il existe peu de systèmes de réalité augmentée performants dans des contextes peu contrôlés. En particulier, dans le cas où une expérience de la réalité augmentée est désirée dans un environnement extérieur et sans avoir à fournir de modèles 3D. Dans un tel cadre, l'idéal est de se servir de la texture des objets de la scène. Pour cela l'approche, usuelle consiste à utiliser des primitives dans les images. Nous avons opté pour l'utilisation de points d'intérêt à détecter puis décrire.

Lorsque de plus le système a pour vocation d'être embarqué sur des appareils mobiles (possédant une capacité de calcul restreinte), plusieurs problèmes sont à envisager. Le premier est la gestion des multiples points à décrire et apparier dans une base d'images. En effet, cette étape peut s'avérer gourmande en temps de calcul. Le deuxième problème lui consiste à disposer d'un descripteur robuste aux transformations diverses que le voisinage d'un point peut subir, tout en restant suffisamment rapide afin de permettre le calcul en temps réel.

Dans cette thèse nous avons donc eu pour objectif d'améliorer certaines méthodes de l'état de l'art en la réalité augmentée. Nous nous sommes concentrés pour cela sur deux axes :

- Mise en place d'un schéma de réalité augmentée permettant de réduire les temps de calculs.

- Mise en place d'un cadre englobant la plupart des descripteurs binaires. Nous avons même étendu ce cadre afin de prendre en compte certaines améliorations lors du processus d'apprentissage.

Concernant le premier problème, nous avons fait face à la multiplicité des points à comparer. Pour cela nous avons établi une notion de voisinage entre images de la base ayant permis la reconstruction 3D. Nous avons aussi mis en place une notion de filtres sur les points SIFT [Low99]. Afin d'établir la notion de voisinage entre images de la base de données de pré-traitement, nous avons proposé une fonction de coût. Cette fonction de coût se base sur le nombre de correspondances entre images afin de déterminer leurs niveaux de similitudesf. Nous nous servons ensuite de cette information afin de limiter nos recherches à un nombre restreint d'images. En nous inspirant de [Wu13], nous avons décidé de filtrer le nombre de points SIFT [Low99] à utiliser, en ne gardant que ceux de plus grande échelle. Cela a abouti à une considérable réduction des temps de calcul.

En ce qui concerne le second problème que nous avons mentionné, nous avons identifié que la partie la plus chronophage est celle qui vise à décrire et mettre des points 2D en correspondance. Ainsi, dans un deuxième temps, nous avons proposé notre propre descripteur. Le descripteur que nous avons construit est binaire, et permet une mise en correspondance rapide des points d'intérêts. Pour cela, nous avons exploité l'idée de BOLD [BTM15] d'effectuer un apprentissage en ligne dans le but d'améliorer la robustesse du descripteur aux rotations. Nous l'avons étendu en rajoutant dans la phase d'apprentissage le fait de choisir les tests qui sont les plus robustes aux changements de rotations. Cela a été possible car nous avons constaté qu'il y a préservation de la robustesse d'un test aux rotations d'une base de données à une autre. En ce faisant, il nous a été possible d'obtenir un descripteur aussi robuste que BOLD tout en étant deux fois plus rapide.

5.2 Perspectives

Dans nos travaux futurs, nous adresserons certains points afin de rendre l'expérience de réalité augmentée encore plus robuste et naturelle :

- **Gestion des occlusions**. Avec l'avènement de nouveaux dispositifs, comme ceux introduits par [KVD14], de nouvelles possibilités sont envisageables. L'une des problématiques qu'il serait intéressant d'aborder est celle des occlusions vue notamment dans [LB00]. L'occlusion est le fait qu'un nouvel élément de la scène non prévu vienne se placer entre un objet et l'utilisateur. Dans certains contextes, le fait que l'objet augmenté reste entièrement visible alors qu'il est occulté peut s'avérer non naturel. Pour cela, nous pourrons exploiter la notion de voisinage afin de simplifier les calculs et déterminer quelles zones de la scène sont cachées ou non.

- **Apprentissage de détecteur et de descripteur par réseaux de neurones**. Nous désirons également approfondir ce que l'apprentissage peut apporter à la description de points. Pour cela, nous envisageons d'utiliser des réseaux de neurones que nous adapterons. Nous pourrons nous inspirer de [ZK15, HLJ+15, SSTF+15]. Dans un tel cadre, générer une base d'images d'apprentissage s'avère assez simple. Pour cela, nous pourrons utiliser des détecteurs et descripteurs connus et robustes. De plus, gérer la complexité du réseau (et donc des calculs) utilisés pour décrire un point permettra de trouver un équilibre entre précision, robustesse et temps de calcul.

- **Estimation de l'illumination dans une scène pour y insérer des objets de façon naturelle**. Le fait d'insérer un objet de manière naturelle dans une scène est essentiel en réalité augmentée. Pour cela, des méthodes telles que l'inpainting [BRB09], sont envisagées. Dans notre approche, nous comptons plutôt estimer l'illumination globale de la scène afin de l'appliquer à la texture de l'objet que nous désirons insérer à la manière de [Deb08].

Liste des publications

[1] Youssef El Rhabi, Loic Simon, and Luc Brun. Estimation de la pose d'une caméra à partir d'un flux vidéo en s' approchant du temps réel. In *Journées francophones des jeunes chercheurs en vision par ordinateur*, 2015.

[2] Youssef El Rhabi, Loic Simon, Luc Brun, Josep Llados Canet, and Felipe Lumbreras. Information theoretic rotationwise robust binary descriptor learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 368–378. Springer, 2016.

References

[AB86] Haruo Asada and Michael Brady. The curvature primal sketch. *IEEE transactions on pattern analysis and machine intelligence*, 1(1) :2–14, 1986.

[AKB08] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure : Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[AM+13] Sameer Agarwal, Keir Mierle, et al. Ceres solver, 2013.

[ANB13] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *BMVC*, 2013.

[AOV12] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak : Fast retina keypoint. In *CVPR*, pages 510–517. IEEE, 2012.

[BBB+93] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04) :669–688, 1993.

[BK12] L BRUN and WALTER KROPATSCH. Hierarchical graph encodings. *Image processing and analysis with graphs : theory and practice*, 2012.

[BL07] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74(1) :59–73, 2007.

[BPS05] Gabriele Bleser, Yulian Pastarmov, and Didier Stricker. Real-time 3d camera tracking for industrial augmented reality applications. 2005.

[BRB09] Derek Bradley, Gerhard Roth, and Prosenjit Bose. Augmented reality on cloth with realistic illumination. *Machine Vision and Applications*, 20(2) :85–92, 2009.

[Bry75] Arthur Earl Bryson. *Applied optimal control : optimization, estimation and control*. CRC Press, 1975.

[BTM15] Vassileios Balntas, Lilian Tang, and Krystian Mikolajczyk. Bold-binary online learned descriptor for efficient image matching. In *CVPR*, pages 2367–2375, 2015.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006.

[CC12] Changhyun Choi and Henrik I Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group : A combined approach of keypoint and edge features. *The International Journal of Robotics Research*, 31(4) :498–519, 2012.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[CKF11] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7 : A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[CL85] Raja Chatila and Jean-Paul Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985.

[CLSF10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief : Binary robust independent elementary features. *ICCV*, pages 778–792, 2010.

[CMNT99] Jose A Castellanos, JMM Montiel, José Neira, and Juan D Tardós. The spmap : A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5) :948–952, 1999.

[CMPC06] Andrew I Comport, Eric Marchand, Muriel Pressigout, and Francois Chaumette. Real-time markerless tracking for augmented reality : the virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics*, 12(4) :615–628, 2006.

[Dav03] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, volume 3, pages 1403–1410, 2003.

[DC02] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7) :932–946, 2002.

[DD92] Daniel F DeMenthon and Larry S Davis. Model-based object pose in 25 lines of code. In *European conference on computer vision*, pages 335–343. Springer, 1992.

[Deb08] Paul Debevec. Rendering synthetic objects into real scenes : Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 classes*, page 32. ACM, 2008.

[Der93] Rachid Deriche. *Recursively implementating the Gaussian and its derivatives*. PhD thesis, INRIA, 1993.

[DG93] Rachid Deriche and Gerard Giraudon. A computational approach for corner and vertex detection. *International journal of computer vision*, 10(2) :101–124, 1993.

[DMM07] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. *From gestalt theory to image analysis : a probabilistic approach*, volume 34. Springer Science & Business Media, 2007.

[DNC+01] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3) :229–241, 2001.

[Dor99] Klaus Dorfmüller. Robust tracking for augmented reality using retroreflective markers. *Computers & Graphics*, 23(6) :795–800, 1999.

[ED06] Ethan Eade and Tom Drummond. Scalable monocular slam. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 469–476. IEEE Computer Society, 2006.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[ERSB15] Youssef El Rhabi, Loic Simon, and Luc Brun. Estimation de la pose d'une caméra à partir d'un flux vidéo en s' approchant du temps réel. In *Journées francophones des jeunes chercheurs en vision par ordinateur*, 2015.

[ERSB+16] Youssef El Rhabi, Loic Simon, Luc Brun, Josep Llados Canet, and Felipe Lumbreras. Information theoretic rotationwise robust binary descriptor learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 368–378. Springer, 2016.

[Fau93] Olivier Faugeras. *Three-dimensional computer vision : a geometric viewpoint*. MIT press, 1993.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395, 1981.

[FKS+16] Bin Fan, Qingqun Kong, Wei Sui, Zhiheng Wang, Xinchao Wang, Shiming Xiang, Chunhong Pan, and Pascal Fua. Do we need binary features for 3d reconstruction ? *arXiv :1602.04502*, 2016.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GGBW11] Pascal Gwosdek, Sven Grewenig, Andrés Bruhn, and Joachim Weickert. Theoretical foundations of gaussian convolution by extended box filtering. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 447–458. Springer, 2011.

[HBW07] Gang Hua, Matthew Brown, and Simon Winder. Discriminant embedding for local image descriptors. In *ICCV*, pages 1–8. IEEE, 2007.

[HCG+96] Gentaro Hirota, David T Chen, William F Garrett, Mark A Livingston, et al. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438. ACM, 1996.

[HJ12] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[HLJ+15] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet : unifying feature and metric learning for patch-based matching. In *CVPR*, pages 3279–3286, 2015.

[HLON94] Bert M Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International journal of computer vision*, 13(3) :331–356, 1994.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors : towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[JY08] Bouguet Jean-Yves. Calibration camera matlab stereo toolbox. In *Calibration camera matlab stereo toolbox*, 2008.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, pages 85–94. IEEE, 1999.

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[KMPK14] Daniel Kurz, Peter Georg Meier, Alexander Plopski, and Gudrun Klinker. Absolute spatial context-aware visual feature descriptors for outdoor handheld camera localization overcoming visual repetitiveness in urban environments. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 2, pages 56–67. IEEE, 2014.

[KS04] Yan Ke and Rahul Sukthankar. Pca-sift : A more distinctive representation for local image descriptors. In *CVPR*, volume 2, pages II–506. IEEE, 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KTSP14] Kalin Kolev, Petri Tanskanen, Pablo Speciale, and Marc Pollefeys. Turning mobile phones into 3d scanners. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3946–3953. IEEE, 2014.

[KVD14] Deval Keralia, K Vyas, and Khushali Deulkar. Google project tango-a convenient 3d modeling device. *International Journal of Current Engineering and Technology*, 4(5) :3139–3142, 2014.

[L+91] David G Lowe et al. Fitting parameterized three-dimensional models to images. *IEEE transactions on pattern analysis and machine intelligence*, 13(5) :441–450, 1991.

[L+97] Mark A Livingston et al. Magnetic tracker calibration for improved augmented reality registration. *Presence : Teleoperators and Virtual Environments*, 6(5) :532–546, 1997.

[LA09] Manolis IA Lourakis and Antonis A Argyros. Sba : A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1) :2, 2009.

[LB00] Vincent Lepetit and M-O Berger. A semi-automatic method for resolving occlusion in augmented reality. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 225–230. IEEE, 2000.

[LBJL07] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based slam : Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3) :343–364, 2007.

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk : Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555. IEEE, 2011.

[LCTHS88] Yann Le Cun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *The Connectionist Models Summer School*, volume 1, pages 21–28, 1988.

[Lev44] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2) :164–168, 1944.

[Lin93] Tony Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch : a method for focus-of-attention. *International Journal of Computer Vision*, 11(3) :283–318, 1993.

[Lin98] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2) :79–116, 1998.

[LKPB02] Peter Lang, Albert Kusej, Axel Pinz, and Georg Brasseur. Inertial tracking for mobile augmented reality. In *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, volume 2, pages 1583–1587. IEEE, 2002.

[LLF05] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 775–781. IEEE, 2005.

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166, 2009.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157. IEEE, 1999.

[LR87] Annick M Leroy and Peter J Rousseeuw. Robust regression and outlier detection. *Wiley Series in Probability and Mathematical Statistics, New York : Wiley, 1987*, 1, 1987.

[Mar63] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2) :431–441, 1963.

[MCUP04] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10) :761–767, 2004.

[MHB+10] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European conference on Computer vision*, pages 183–196. Springer, 2010.

[ML09] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340) :2, 2009.

[ML12] Marius Muja and David G Lowe. Fast matching of binary features. In *CRV*. IEEE, 2012.

[ML14] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11) :2227–2240, 2014.

[MLD+06] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370. IEEE, 2006.

[MMHM14] Pierre Martin, Eric Marchand, Pascal Houlier, and Isabelle Marchal. Mapping and re-localization for mobile augmented reality. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3352–3356. IEEE, 2014.

[MMM12a] Lionel Moisan, Pierre Moulon, and Pascal Monasse. Automatic homographic registration of a pair of images, with a contrario elimination of outliers. *Image Processing On Line*, 2 :56–73, 2012.

[MMM12b] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision*, pages 257–270. Springer, 2012.

[MMM13a] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *ICCV*, pages 3248–3255, 2013.

[MMM13b] Pierre Moulon, Pascal Monasse, and Renaud Marlet. La bibliothèque openmvg : open source multiple view geometry. In *Orasis, Congrès des jeunes chercheurs en vision par ordinateur*, 2013.

[Mor78] Jorge J Moré. The levenberg-marquardt algorithm : implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.

[MS69] Minsky Marvin and Papert Seymour. Perceptrons, 1969.

[MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60(1) :63–86, 2004.

[MS05] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10) :1615–1630, 2005.

[MTS+05] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2) :43–72, 2005.

[Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6) :756–770, 2004.

[NNB04] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

[OCLF10] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 32(3) :448–461, 2010.

[ODD96] Denis Oberkampf, Daniel F DeMenthon, and Larry S Davis. Iterative pose estimation using coplanar feature points. *Computer Vision and Image Understanding*, 63(3) :495–511, 1996.

[OKI15] Peter Ondrúška, Pushmeet Kohli, and Shahram Izadi. Mobilefusion : Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, 21(11) :1251–1258, 2015.

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1), 1996.

[Per08] Simon Perry. Wikitude : Android app with augmented reality : Mind blowing. *digital-lifestyles. info*, 23(10), 2008.

[PLD05] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *PAMI*, 27(8), 2005.

[PMK12] Antoine Petit, Eric Marchand, and Keyvan Kanani. Tracking complex targets for space rendezvous and debris removal applications. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4483–4488. IEEE, 2012.

[PSACMN13] Adrian Penate-Sanchez, Juan Andrade-Cetto, and Francesc Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE transactions on pattern analysis and machine intelligence*, 35(10) :2387–2400, 2013.

[QL99] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Transactions on pattern analysis and machine intelligence*, 21(8) :774–780, 1999.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[RBSJ79] Frederick H Raab, Ernest B Blood, Terry O Steiner, and Herbert R Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic systems*, (5) :709–718, 1979.

[RCC15] Datta Ramadasan, Marc Chevaldonné, and Thierry Chateau. Dcslam : un slam temps réel à contraintes dynamiques. In *Journées francophones des jeunes chercheurs en vision par ordinateur*, 2015.

[RD06] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[RDGM10] Julien Rabin, Julie Delon, Yann Gousseau, and Lionel Moisan. Mac-ransac : a robust algorithm for the recognition of multiple objects. In *Fifth International Symposium on 3D Data Processing, Visualization and Transmission (3DPTV 2010)*, page 051, 2010.

[RLdTR11] Antonio L Rodriguez, Pedro E López-de Teruel, and Alberto Ruiz. Gea optimization for live structureless motion estimation. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 715–718. IEEE, 2011.

[RMG+88] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.

[RODM15] Ives Rey-Otero, Mauricio Delbracio, and Jean-Michel Morel. Comparing feature detectors : A bias in the repeatability criteria. In *ICIP*, pages 3024–3028. IEEE, 2015.

[Ros99] Paul L Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2) :291–307, 1999.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb : an efficient alternative to sift or surf. In *ICCV*, pages 2564–2571. IEEE, 2011.

[SB91] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1) :11–32, 1991.

[SLL05] Stephen Se, David G Lowe, and James J Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on robotics*, 21(3) :364–375, 2005.

[SMD10] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam : Why filter ? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010.

[SNB14] Konstantinos Sechidis, Nikolaos Nikolaou, and Gavin Brown. Information theoretic feature selection in multi-label data through composite likelihood. In *S+SSPR*, pages 143–152. 2014.

[SRASC14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf : an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[SSS06] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism : exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[SSS08] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2) :189–210, 2008.

[SSTF+15] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, pages 118–126, 2015.

[Sut68] Ivan E Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM, 1968.

[Sut74] Ivan E Sutherland. Three-dimensional data input by tablet. *Proceedings of the IEEE*, 62(4) :453–461, 1974.

[SvHVG+08] Christoph Strecha, Wolfgang von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. Ieee, 2008.

[SZS+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*, 2013.

[TCFL13] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *CVPR*, pages 2874–2881, 2013.

[TGBC+11] Mohamed Tamaazousti, Vincent Gay-Bellile, Sylvie Naudet Collette, Steve Bourgeois, and Michel Dhome. Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3073–3080. IEEE, 2011.

[TNNL02] Juan D Tardós, José Neira, Paul M Newman, and John J Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4) :311–330, 2002.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface : Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[VH12] Jonathan Ventura and Tobias Höllerer. Wide-area scene mapping for mobile visual tracking. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 3–12. IEEE, 2012.

[VLF04] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera

tracking. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 48–57. IEEE Computer Society, 2004.

[Vuf13] SDK Vuforia. Vuforia developer portal, 2013.

[W+11] Changchang Wu et al. Visualsfm : A visual structure from motion system. 2011.

[Wel86] William M Wells. Efficient synthesis of gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2) :234–239, 1986.

[Wel95] Hybrid Self-Tracker Welch. An inertial/optical hybrid three-dimensional tracking system, university of north carolina chapel hill department of computer science. Technical report, TR 95-048, 1995.

[WF02] Greg Welch and Eric Foxlin. Motion tracking : No silver bullet, but a respectable arsenal. *IEEE Computer graphics and Applications*, 22(6) :24–38, 2002.

[WFW11] Zhenhua Wang, Bin Fan, and Fuchao Wu. Local intensity order pattern for feature description. In *ICCV*, pages 603–610. IEEE, 2011.

[WRM+08] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.

[Wu07] Changchang Wu. Siftgpu : A gpu implementation of scale invariant feature transform (sift). 2007.

[Wu13] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[Xu11] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv :1107.2490*, 2011.

[YC12] Xin Yang and Kwang-Ting Cheng. Ldb : An ultra-fast feature for scalable augmented reality on mobile devices. In *ISMAR*, pages 49–57. IEEE, 2012.

[YNA99] Suya You, Ulrich Neumann, and Ronald Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 260–267. IEEE, 1999.

[ZF12] Zhengyou Zhang and Olivier Faugeras. *3D dynamic scene analysis : a stereo based approach*, volume 27. Springer Science & Business Media, 2012.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11) :1330–1334, 2000.

[ZK15] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, pages 4353–4361, 2015.

[ZL15] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[ZW94] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, pages 151–158. Springer, 1994.

Table des figures