Stage long

1er août 2005 - 31 juillet 2006

États-Unis

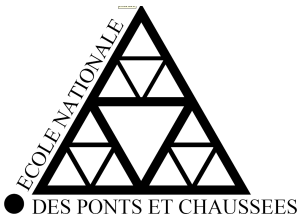
# Travail de recherche en vision par ordinateur

*Détection et mise en correspondance de points d'intérêt en temps réel notamment grâce à des arbres de classification*


Aurélien Boffy

Élève-ingénieur
Département « Ingénierie Mathématique et Informatique »

Tuteur de stage : M. Yanghai Tsin




École Nationale des Ponts et Chaussées
6-8 avenue Blaise Pascal
77455 Champs-sur-Marne
France
+33 1 64 15 30 00

SIEMENS

Siemens Corporate Research Inc.
755 College Road East
Princeton, NJ 08540
USA
+1 609 734 6500

# Fiche de synthèse

| | |
|---|---|
| Type de stage | Stage long |
| Année | 2006 |
| Auteur | Aurélien Boffy |
| Formation | IMI |
| Titre du rapport | **Travail de recherche en vision par ordinateur.** Détection et mise en correspondance de points d'intérêt en temps réel, notamment grâce à des arbres de classification |
| Organisme d'accueil | Siemens Corporate Research |
| Pays d'accueil | États-Unis (Princeton, NJ) |
| Responsable de stage | Yanghai Tsin |
| Mots clés | vision par ordinateur, mise en correspondance de points d'intérêt, arbre de classification, tracking |
| Thèmes | Informatique, Mathématiques appliquées, Recherche |

**Résumé**

Le problème abordé dans ce rapport est la mise en correspondance de points d'intérêt en temps réel. Plus particulièrement, nous nous intéressons à l'appariement de points entre deux images séparées par un angle de vue important. Ce problème est aujourd'hui encore l'un des plus fondamentaux dans la recherche en vision par ordinateur, car les applications sont nombreuses, de la détection d'objets à la reconstruction d'un environnement 3D.

Récemment, des progrès significatifs ont été apportés par l'équipe de Vincent Lepetit. Ils considèrent l'appariement de points d'intérêt comme un problème de classification et proposent d'utiliser des arbres de décision. Leur système est particulièrement robuste et fonctionne en temps réel (25 images par seconde).

Bien que cette méthode soit très prometteuse, les résultats obtenus restent relativement sensibles aux changements d'angle de vue et aux variations des conditions d'illumination. Nous proposons deux principales approches pour contourner ces difficultés :

1. Mise à jour en temps réel des arbres de classification, afin qu'ils s'adaptent aux conditions de l'environnement.
2. Spécialisation des arbres : chacun d'eux modélise plus spécifiquement l'apparence de l'objet vu depuis un certain volume de l'espace.

Le résultat est une méthode d'appariement de points d'intérêt dont la zone de couverture est plus étendue, et qui est plus robuste aux changements des conditions d'illumination (ombres, réflexions spéculaires, etc.).

Ces approches ont été décrites dans un article qui a été accepté pour une présentation orale lors de la conférence internationale de vision par ordinateur BMVC 2006.

**Mots-clés** Vision par ordinateur, mise en correspondance de points d'intérêt, arbre de classification, détection, tracking

**Abstract**

We address the real-time feature matching problem. We are concerned with matching features observed from two views separated by a wide baseline. This problem remains one of the most fundamental issues in computer vision research. A solution to it has many applications, such as object detection, recognition, tracking and 3D reconstruction.

Recently, significant progress was made by Lepetit and colleagues. They treat the issue of feature matching as a classification problem by using randomized classification trees. Their system shows real-time (25 frames per second) feature matching very robustly.

Though extremely promising, their actual results can vary depending on the viewpoint and illumination conditions. We combine two approaches to alleviate its limitations :

1. We update the trees at run-time, adapting them to the actual viewing conditions.
2. We spatially distribute the trees, so that each of them models a certain viewing volume more precisely.

The result is a more stable matching method that significantly extends detectable range and is much more robust to illumination changes, such as cast shadows or reflections.

We described these approaches in a paper, which was accepted for an oral presentation at the BMVC'06 conference.

**Keywords** Computer vision, feature matching, classification tree, detection, tracking

# Table des matières

# Table des figures

# Première partie

# L'entreprise d'accueil

# Chapitre 1

# Siemens AG

## 1.1 De l'atelier au conglomérat

En 1847, Werner von Siemens révolutionne la télégraphie dans un petit atelier de Berlin et fonde avec Johann Georg Halske sa petite entreprise : *Telegraphenbau-Anstalt von Siemens & Halske*. Presque 160 ans plus tard, Siemens AG est l'un des plus gros conglomérats mondiaux, spécialisé dans les équipements électroniques et informatiques. L'entreprise allemande, dont les sièges sont à Berlin et à Munich, emploie 461 000 personnes, et a un chiffre d'affaires de 75 milliards d'euros (chiffres de 2005).

## 1.2 Activités-clés

L'entreprise est particulièrement active dans six domaines d'activité (voir la figure 1.1 pour la part de chacune de ces activités dans le chiffre d'affaires) :

**Information et communication :** Cette activité regroupe l'installation de réseaux et le développement de produits utilisés dans les nouveaux modes de communication. Siemens est notamment le premier fournisseur mondial de modems Internet rapides.

**Automatismes et contrôle :** Les automatismes et le contrôle font de Siemens le premier fournisseur mondial de systèmes, solutions et services pour l'industrie et le bâtiment.

**Énergie :** Siemens est présent sur tous les métiers du cycle de l'énergie, de la production à la distribution de l'électricité, en passant par son transport.

**Transports :** Siemens est à la fois un équipementier automobile sur les marchés du premier équipement et de la rechange, et un ensemblier ferroviaire majeur.

**Médical :** Siemens est particulièrement actif dans l'imagerie médicale (scanners, échographie, etc.), et dans les réseaux informatiques hospitaliers.

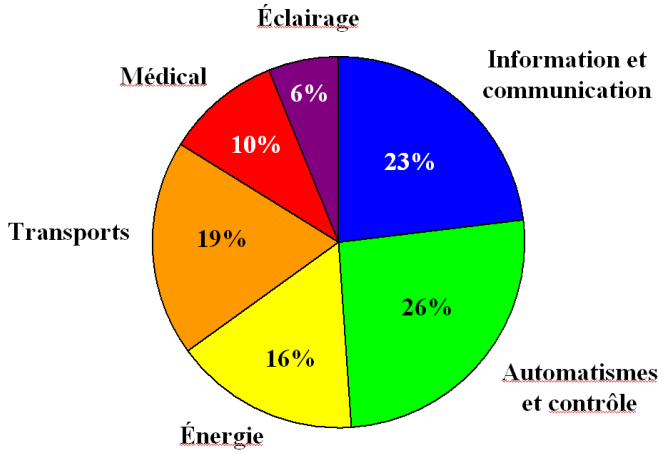



FIG. 1.1 – *Part des principales activités de Siemens AG dans le chiffre d'affaires en 2005*

C'est aussi le numéro un de l'aide auditive.

**Éclairage :** Siemens commercialise de l'éclairage destiné aux particuliers, mais aussi au domaine urbain, automobile et industriel.

Il convient d'ajouter à ces six branches principales, de nombreux autres domaines d'activité, comme les ordinateurs avec *Fujitsu Siemens Computer*), les semi-conducteurs avec *Infineon technologies AG* et l'électroménager.

## 1.3 Siemens et son marché

### 1.3.1 Siemens dans le monde

Siemens est aujourd'hui une véritable entreprise mondiale qui a une présence industrielle et commerciale dans la plupart des régions du globe (voir figure 1.2). Le pays d'origine de l'entreprise, l'Allemagne, représente, avec le reste de l'Europe, à peine la moitié du chiffre d'affaires et de l'effectif de la société.

### 1.3.2 Implantation et image de la société aux États-Unis

Les États-Unis représentent le premier marché de Siemens (21,5% du chiffre d'affaires mondial en 2005), juste devant l'Allemagne (21%). L'entreprise emploie plus de 70 000 personnes, dispose de 700 sites dans les 50 états, et continue à investir afin de poursuivre son expansion dans le pays : au cours des 18 derniers mois, Siemens a dépensé plus de 9 milliards d'euros

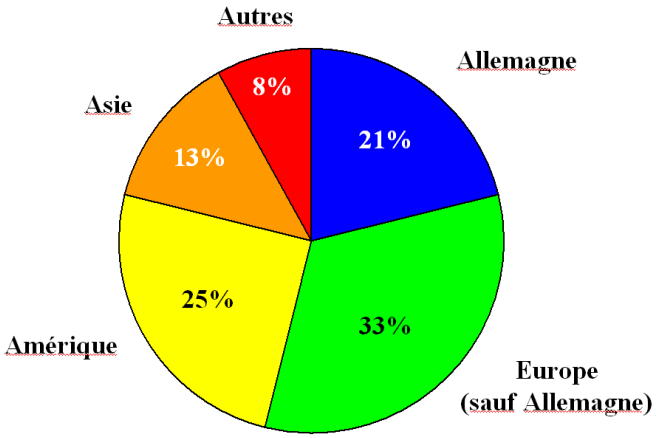



FIG. 1.2 – *Répartition spatiale du chiffre d'affaires de Siemens AG (en 2005)*

pour acquérir une trentaine de sociétés américaines.

Siemens est très présent dans la vie des Américains : à titre d'exemples, l'entreprise fournit des systèmes automatiques utilisés pour traiter 90% du trafic postal, des équipements pour produire plus d'un tiers de l'électricité du pays, et les systèmes de sécurité utilisés dans tous les aéroports américains. Cependant, bien que Siemens emploie plus de personnes aux États-Unis que *General Electric*, l'entreprise est moins bien connue. Même si Siemens n'a pas besoin de vendre ses produits au grand public, la société souhaite se construire une véritable identité dans le pays. C'est pourquoi elle s'est associée à Disney et sponsorise des événements populaires comme les courses de Nascar.

### 1.3.3 Concurrence

Comme Siemens est présent dans de multiples secteurs, il est difficile d'établir une liste exhaustive de ses concurrents. Ceux qui sont le plus souvent cités sont par exemple le conglomérat américain *General Electric*, le groupe néerlandais *Philips*, la société française *Alcatel*, ou encore l'entreprise japonaise *Hitachi*, qui proposent toutes une vaste gamme de produits.

La figure 1.3 présente le chiffre d'affaires des 15 plus grosses entreprises dans le domaine de l'ingénierie électrique et électronique.

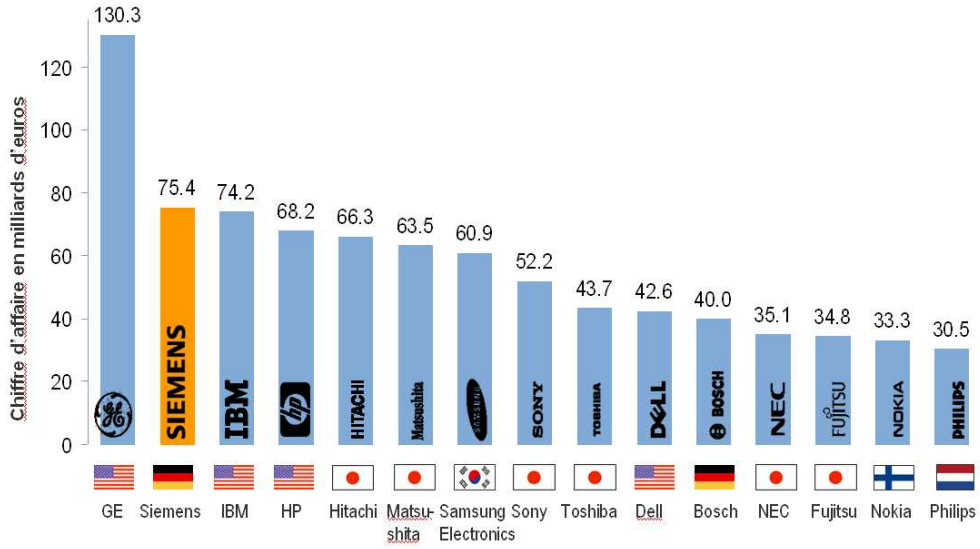



FIG. 1.3 – *Les 15 plus grosses entreprises de l'ingénierie électrique et électronique (chiffres de 2005)*

# Chapitre 2

# Siemens Corporate Research

## 2.1 Recherche et Développement à Siemens

En 1848, la société a construit la première ligne de télégraphe longue distance en Europe (500 km de Berlin à Francfort). Aujourd'hui, elle a conservé cette tradition de recherche et d'innovation : 75% du chiffre d'affaires est réalisé grâce à des produits et à des services développés dans les cinq dernières années.

Afin de rester compétitif, Siemens utilise énormément de ressources pour pour la recherche et le développement. 47 000 employés travaillent dans la R&D dans 150 centres de développement (voir figure 2.1) et Siemens y a consacré 5,2 milliards d'euros en 2005, soit 6,8% du chiffre d'affaires. Dans le secteur de l'ingénierie électrique, c'est ainsi la société dont les dépenses en R&D sont les plus importantes, devant IBM, Matsushita, Nokia, Sony et Samsung.

Cet intérêt pour la recherche et le développement est aussi visible lorsqu'on examine le nombre de brevets déposés par Siemens. Avec 53 000 brevets actifs, l'entreprise se classe première en Allemagne, deuxième en Europe derrière Philips, et neuvième dans le monde.

## 2.2 SCR : la R&D de Siemens aux États-Unis

Suite à l'expansion de Siemens aux États-Unis dans les années 1970, Siemens Corporate Research (SCR) a été fondée en 1977. C'est une des cinq filiales de Siemens AG destinées à la recherche et au développement, et la seule basée en dehors d'Europe, le siège étant près de l'université de Princeton, dans le New Jersey. C'est aussi en ce lieu qu'est situé le principal centre de recherche de SCR, où travaillent près de 220 chercheurs de plus de 20 nationalités différentes.

SCR est divisé en six départements, qui mènent chacun leurs propres projets et qui sont relativement indépendants :

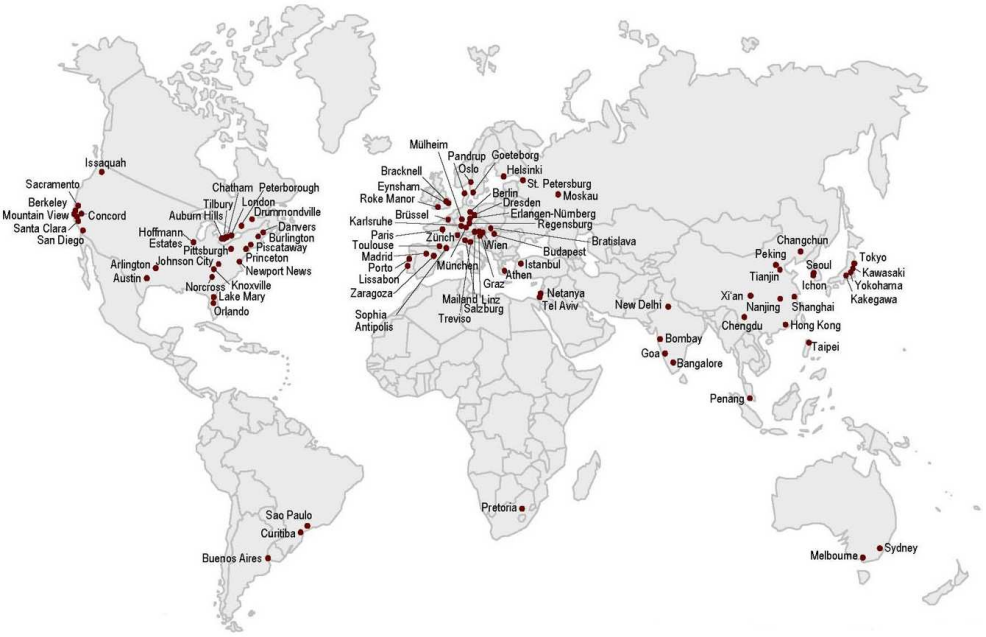



FIG. 2.1 – *R&D à Siemens : plus de 47 000 employés dans 150 centres de développement*

**Automatismes et contrôle** (*Automation and Control*) : logiciels et outils pour les automatismes et le contrôle dans l'industrie, ainsi que pour les technologies de construction.

**Imagerie et visualisation** (*Imaging and Visualization)* : traitement d'images et vision par ordinateur, appliqués à la médecine.

**Vision en temps-réel et modélisation** (*Real-Time Vision and Modeling*) : vision temps-réel pour la sécurité, la surveillance, les transports, la maintenance, etc.

**Systèmes de données intégrées** (*Integrated Data Systems*) : extraction de données, intégration, modélisation.

**Vision intelligente et raisonnement** (*Intelligent Vision and Reasoning*) : diagnostics assistés par ordinateur pour des applications médicales et industrielles.

**Génie logiciel** (*Software Engineering*) : architecture et qualité pour une efficacité accrue.

J'étais stagiaire dans le département *Real-Time Vision and Modeling*, où les différents programmes sont les suivants :

– Caméras intelligentes et vision par ordinateur (*Smart Cameras and Machine Vision*)

- Vision 3D et réalité augmentée (*3D Vision and Augmented Reality*)
- Méthodes statistiques pour la vision (*Statistical Methods for Vision*)
- Traitement du son et du signal, communications sans-fil (*Audio, Signal Processing & Wireless Communication*)

Mon projet s'intégrait au programme « Vision 3D et réalité augmentée », qui a pour but la mise au point de méthodes robustes de détection, de suivi et de reconnaissance, pour des systèmes de maintenance et d'inspection.

# Deuxième partie

# Contexte et enjeux du stage

# Chapitre 3

# Environnement de travail

## 3.1 De très nombreux stagiaires

Au laboratoire de recherche de Princeton, le nombre de stagiaires est assez surprenant. Selon la période de l'année, ils sont entre 70 et 120, soit presque autant que les travailleurs permanents. Une majorité d'entre eux sont allemands, mais il y a aussi beaucoup de Français, de Chinois et d'Indiens. Les stagiaires américains sont très peu nombreux.

Cette singularité du laboratoire m'a particulièrement intrigué et j'ai cherché à savoir ce qui poussait l'entreprise à faire venir tant de jeunes stagiaires. Le PDG de Siemens Corporate Research a donné un jour une conférence au sein du laboratoire et a répondu que les stagiaires apportaient du dynamisme et une certaine fraîcheur. Notons par ailleurs qu'ils sont particulièrement bon marché...

Du fait de cette importante proportion de jeunes étudiants, l'ambiance est particulièrement décontractée. Ceci est certainement aussi une spécificité des laboratoires de recherche situés aux États-Unis. Aucun code vestimentaire n'est de rigueur, les employés eux-mêmes sont souvent en « jean / t-shirt » et les horaires de travail sont très flexibles.

## 3.2 Un travail trop individuel

Malgré ce nombre important de stagiaires, une de mes plus grosses déceptions concernant ce stage est le fait d'avoir trop souvent travaillé seul. Il n'y a pas eu de réel travail d'équipe avec d'autres stagiaires ou employés et cela se réduisait à demander conseil au « spécialiste en C++ », à « l'expert en OpenGL » ou à l'employé de l'équipe le plus compétent en apprentissage automatique.

Travailler uniquement avec mon tuteur contribuait néanmoins à me donner plus de responsabilités car je maîtrisais vraiment l'évolution de mon projet, et j'ai pu mettre en œuvre toutes les idées auxquelles j'ai pensé.

Néanmoins, confronter mes idées avec celles d'autres collègues et bénéficier d'autres retours que ceux de mon tuteur aurait pu être utile et fructueux.

De ce point de vue, je n'étais pas une exception parmi les stagiaires de Siemens, car rares étaient ceux qui travaillaient sur un projet avec des collègues autres que leur tuteur.

# Chapitre 4

# Un véritable travail de recherche

## 4.1 La recherche à SCR

Il y a encore quelques années, les chercheurs de Siemens Corporate Research étaient très libres quant à leurs travaux de recherche. Dans les dernières années, les obligations de transferts technologiques sont devenues de plus en plus systématiques, mais SCR reste encore aujourd'hui, du point de vue d'un Français, plus proche d'un établissement public de recherche que d'un département Recherche & Développement d'une entreprise. Les publications scientifiques sont toujours très nombreuses, et il y a généralement plusieurs chercheurs du laboratoire représentant SCR dans les principaux congrès internationaux de vision par ordinateur.

## 4.2 Le problème des stagiaires-programmeurs

Je suis très satisfait de cette expérience, car j'ai notamment pu apprécier en quoi consiste le travail d'un chercheur. Je craignais en effet avant le début du stage que Siemens ne m'ait engagé que pour faire un travail, que d'aucuns qualifieraient de rébarbatif, d'implémentation de méthodes existantes. Mes craintes furent au demeurant amplifiées lorsqu'à mon arrivée au laboratoire, et après avoir discuté avec quelques collègues, je me suis aperçu que c'était finalement le rôle d'une grande partie des stagiaires. Pour n'évoquer que les Français, il est bien sûr compréhensible que des élèves venus d'écoles comme l'ÉPITA se voient confier des tâches d'implémentation qui leur plaisaient, mais j'ai aussi connu des étudiants de l'école Polytechnique qui avaient passé la quasi-totalité de leur stage à programmer des méthodes existantes, voire à « nettoyer » du code plus ancien.

## 4.3 Une vraie liberté d'initiative

Pour ma part, mon tuteur m'a laissé la possibilité de travailler de façon plus autonome et de prendre de véritables initiatives. Lorsque je suis arrivé, le sujet même du stage n'était pas clairement fixé. Le thème était certes défini, mais c'est ensemble que nous avons décidé des voies que nous allions suivre et des méthodes que nous allions utiliser. Au fil du temps, mon tuteur me laissait choisir les publications scientifiques que je souhaitais lire et les idées que je désirais développer.

## 4.4 Un travail efficace ?

Lors de ce stage, c'est le plus souvent en concertation avec mon tuteur que nous décidions de ce que nous allions faire par la suite. Ainsi, tout au long de l'année, j'ai réfléchi à de multiples solutions et testé de nombreux algorithmes différents pour résoudre un même problème. Comme souvent en recherche, la plupart des pistes que nous avons suivies se sont révélées infructueuses et cela m'a conduit à me poser des questions quant à l'efficacité de mon travail. Plusieurs fois, nous avons en effet dû abandonner des idées et un travail sur lequel j'avais travaillé de longues semaines. Ceci est légèrement frustrant au début car on a l'impression d'avoir travaillé en vain. Ainsi, une partie non négligeable de mon travail effectué à Siemens n'a pas abouti à un résultat concret, et le contenu présenté dans la publication que j'ai écrite (voir section 10 page 77) ne représente finalement qu'une partie limitée de ce que j'ai fait pendant l'année.

Néanmoins, cela aura été d'une part très formateur, car essayer diverses techniques est un excellent moyen de progresser, et d'autre part, réfléchir au problème posé en testant différentes méthodes permet de s'approprier le sujet et de mieux en maîtriser les enjeux et difficultés.

## 4.5 Clauses de confidentialité

Comme je l'ai évoqué plus haut, j'ai fait plusieurs projets lors de ce stage, car nous avons abordé différents problèmes de vision par ordinateur, étudié plusieurs pistes pour chaque question, eu différentes idées et testé diverses techniques.

Cependant, je ne peux dans ce rapport évoquer la totalité du travail que j'ai fourni pendant l'année pour des raisons de protection intellectuelle. J'ai signé en arrivant à Siemens plusieurs clauses de confidentialité et mon tuteur m'a rappelé plusieurs fois que je n'étais pas autorisé à parler de certains points.

Ainsi, je décrirai surtout ici le travail relatif à l'article scientifique que j'ai écrit, qui est désormais public. J'évoquerai aussi quelques autres tâches que

j'ai effectuées (comme l'implémentation de méthodes aujourd'hui classiques, telles que RANSAC ou SIFT), mais je ne pourrai aborder la totalité de ce qui m'a occupé durant cette année de stage.

# Chapitre 5

# La réalité augmentée

Mon stage s'inscrivait dans le cadre d'un programme du département *Real-Time Vision and Modeling* concernant la réalité augmentée. Par cette expression, on entend un système permettant de superposer à l'image d'une scène réelle, des éléments virtuels, ceci en temps-réel et de façon la plus « réaliste » possible.

Différentes problématiques se posent et sont résumées dans la figure 5.1 :

**Alignement des caméras réelle et virtuelle :** Pour obtenir un rendu réaliste, la perspective de l'objet virtuel doit correspondre avec celle de la scène réelle. Pour ce faire, il faut retrouver les propriétés de la caméra réelle ayant donné lieu à l'observation (réglages internes et point de vue par rapport à la scène), et créer les éléments synthétiques en utilisant une caméra virtuelle reprenant ces propriétés. Ainsi, contrairement à l'image (a) où la voiture rouge virtuelle a été insérée de façon arbitraire, sa position dans l'image (b) respecte la perspective réelle.

**Cohérence spatio-temporelle :** L'image (b) n'est cependant pas encore réaliste, notamment car la partie arrière du véhicule devrait être occultée par le bâtiment photographié. Le problème de la cohérence spatio-temporelle concerne ce genre de difficultés, et le résoudre aboutit à l'image (c).

**Cohérence photométrique :** Ce problème concerne la prise en compte des inter-réflexions lumineuses (ombres, reflets) entre les scènes réelle et virtuelle. L'image (d) montre le résultat obtenu en tenant compte de la cohérence photométrique.

Pour Siemens Corporate Research et plus particulièrement pour le programme du département dont je faisais partie, le problème de l'alignement des caméras réelle et virtuelle constitue le principal enjeu. En effet, le poids relatif de chacune de ces problématiques dépend du contexte dans lequel la méthode de réalité augmentée est employée. En effet, les applications sont multiples (médecine, militaire, marketing, robotique, etc.), et on comprend

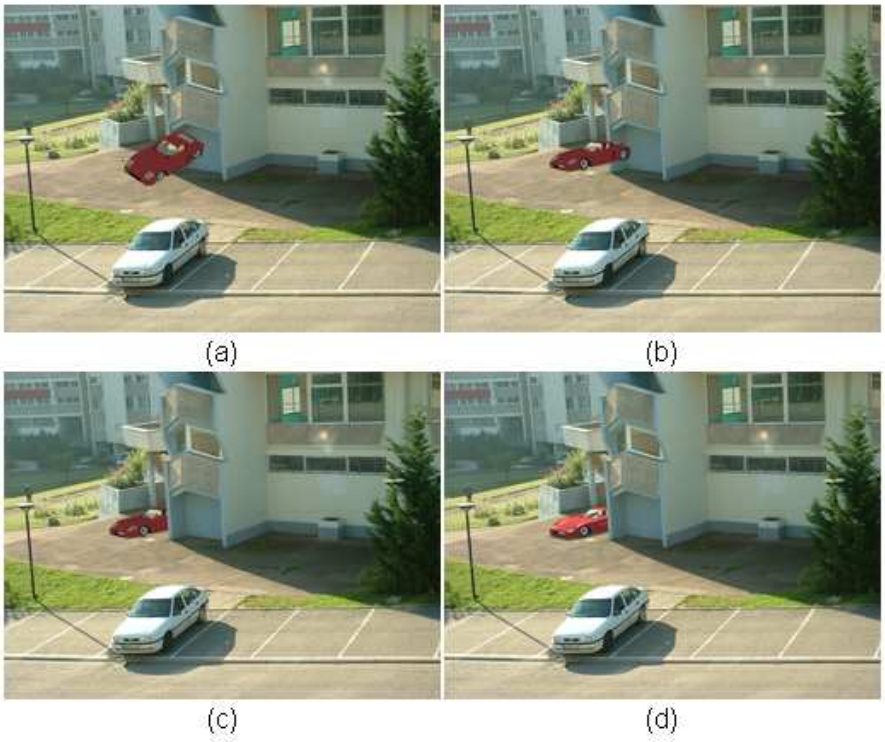



FIG. 5.1 – *Problématiques en réalité augmentée*

FIG. 5.2 – *Utilisation de la réalité augmentée pour l'assistance dans des opérations de maintenance*

par exemple que la cohérence photométrique soit plus importante pour des applications dans le cinéma ou dans les jeux-vidéo qu'ailleurs.

Siemens est particulièrement intéressé par les applications dans l'industrie. La réalité augmentée est alors utilisée afin de guider un opérateur effectuant des opérations d'assemblage ou de maintenance sur des machines compliquées. L'opérateur est équipé de lunettes spécifiques permettant d'ajouter à ce qu'il voit naturellement des informations graphiques et des instructions virtuelles, afin par exemple de le guider à faire sa tâche pas à pas sans avoir à consulter un manuel. Les photos de la figure 5.2 montre un opérateur équipé de ce genre de lunettes en train de travailler sur une photocopieuse. Évidemment, la réalité augmentée est quelque peu superflue pour la maintenance d'un objet si simple, et les clients de Siemens Corporate Research pour ce type de produits sont plutôt Boeing, BMW, ou d'autres branches du groupe Siemens lui-même.

# Troisième partie

# Travail effectué

# Chapitre 6

# Évaluation de l'état de l'art

## 6.1 Préliminaires

### 6.1.1 Une particularité des stagiaires en année de césure ?

Durant les premières semaines, j'ai passé la majeure partie de mon temps à lire des articles scientifiques que mon tuteur sélectionnait pour moi. Mes connaissances en vision par ordinateur et en traitement d'images étaient en effet assez limitées au début de ce stage, même si j'avais suivi les cours d'initiation de Monsieur Keriven et de Monsieur Paragios pendant ma deuxième année à l'école des Ponts. Ceci met certainement en relief une particularité des étudiants en année de césure : les autres stagiaires étaient généralement déjà spécialisés en vision artificielle ou en traitement du signal car ils étaient souvent déjà diplômés ou en stage de fin d'études. Néanmoins, comme nous travaillions tous sur des sujets à la pointe de la recherche, cette phase de lecture d'articles destinée à appréhender l'état de l'art relatif au sujet du stage est plutôt courante à SCR.

### 6.1.2 Une lecture active

Pour chaque publication, j'ai écrit un compte-rendu, où, après avoir rapidement résumé l'apport de l'article et les méthodes présentées, j'essayais de mettre ces dernières en relation avec mes précédentes lectures et d'avoir un esprit critique quant à la validité de ce que les auteurs prétendent. Évidemment, ce n'était pas aisé pour les premiers articles faute de recul, mais après quelques semaines, je prenais finalement plaisir à comparer les résultats de diverses méthodes ayant le même objectif, ou à observer comment un même problème pouvait être résolu par des approches radicalement différentes. De plus, il était aussi intellectuellement satisfaisant de s'apercevoir que ma maîtrise du sujet progressait et que j'acquérais une connaissance suffisamment large du problème pour être capable de mettre en relation les différentes méthodes.

Avoir eu à écrire une critique sur chaque article m'a aussi aidé à reconnaître les caractéristiques d'une bonne publication, ce qui m'a été particulièrement utile lorsque j'ai eu à écrire moi-même un article sur une partie du travail que j'ai effectué durant ce stage (voir section 10 page 77). En effet, lorsqu'un chercheur soumet un article à un congrès scientifique, sa publication est préalablement relue par un comité de lecture. Un article peut ainsi être relu par une demi-douzaine d'autres scientifiques qui effectuent un travail critique sur la publication soumise, afin de déterminer si elle peut être acceptée pour la conférence ou le journal. Plusieurs points sont examinés par le comité de lecture, et ce sont généralement les mêmes pour toutes les conférences :

- clarté des explications
- apport scientifique de la publication
- qualité de la validation expérimentale

C'est avec les mêmes angles de lecture que mon tuteur m'a demandé de juger les publications que j'ai eu à lire, et ceci m'a donc solidement préparé pour l'écriture de mon propre article scientifique.

### 6.1.3 Familiarisation avec les articles scientifiques

Cette première phase m'a bien sûr aidé à acquérir les connaissances nécessaires pour pouvoir travailler correctement et plus efficacement durant la suite du stage. Mais au delà de cet objectif initial, elle m'a permis de me familiariser avec la lecture d'articles scientifiques. Ceux-ci ont en effet généralement une structure relativement standardisée :

- Titre
- Liste des auteurs
- Résumé
- Introduction
- Méthodes
- Résultats expérimentaux
- Discussions
- Bibliographie

Au fil du temps, j'ai appris à adapter ma lecture afin d'extraire de plus en plus rapidement et efficacement les informations importantes. Par la suite, et tout au long de l'année, j'ai évidemment continué à lire de multiples articles et je pense que ceci a été très formateur et m'aidera pour ma vie professionnelle future, certes plus particulièrement si je me destine à une carrière dans le milieu de la recherche.

## 6.2 L'invariance aux changements de point de vue

### 6.2.1 Présentation du problème de l'extraction de primitives images

Un des problèmes fondamentaux en vision par ordinateur et en traitement d'images concerne la détection de primitives (*features* en anglais). C'est une étape préalable à de nombreuses applications. Le principe est d'extraire d'une image des régions d'intérêt qui ont une certaine singularité et sont particulièrement porteuses d'information. Les primitives que l'on cherche à extraire peuvent être :

- des points d'intérêt (aussi appelés *points anguleux* ou *points saillants*) comme des coins ou des jonctions en T
- des contours (bordures des objets par exemple)
- des régions de l'image
- etc.

Ces primitives correspondent bien sûr à des objets présents dans la scène qui a été photographiée ou filmée (le mot *objet* est à prendre au sens large : ce peut être des personnes ou encore des lettres manuscrites dans le cas de la reconnaissance de caractères).

La difficulté du problème consiste à mettre au point une méthode qui soit *robuste* : en appliquant le même algorithme de détection de primitives à une autre image où figure le même objet, on souhaite maximiser la probabilité que les primitives détectées correspondent aux mêmes points de l'objet, et que les points détectés dans la première image le soient aussi dans la deuxième. On parle aussi souvent de la *répétabilité* du détecteur. Divers facteurs peuvent en effet modifier l'apparence de l'objet et donc perturber les résultats du détecteur de primitives :

- le bruit de l'image
- les conditions d'illumination
- la position de la caméra ou de l'appareil photo par rapport à l'objet

### 6.2.2 Le détecteur de Harris

Pour comprendre plus précisément le principe de fonctionnement d'un détecteur de points d'intérêt et les difficultés mises en jeu, nous allons rapidement décrire la méthode du détecteur de Harris. Avec Stephens, Harris a proposé en 1988 un algorithme de détection qui est certainement aujourd'hui le plus connu et qui est toujours très utilisé car il donne de très bons résultats et est assez simple à mettre en œuvre [10].

Les points d'intérêt recherchés sont des points au voisinage desquels l'image varie significativement dans plusieurs directions. Ce peut être des coins, des jonctions en T, des jonctions en Y, etc. Comme souvent pour les

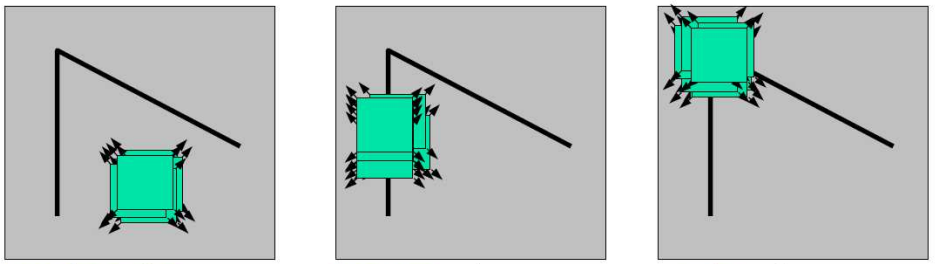



FIG. 6.1 – *Détecter les coins sans retenir les contours.*

algorithmes de détections de primitive, il faut éviter de retenir des points situés sur un contour d'un objet pour deux principales raisons :

1. Lorsque l'objet est observé d'un point de vue différent, le bord de l'objet sur l'image ne correspond plus à la même zone physique dans la scène.
2. Un point situé sur un contour n'est généralement pas stable car les points du même bord situés dans son voisinage ont souvent une apparence similaire.

### Le détecteur de Moravec

Le principe du détecteur de Harris est plus facilement accessible si l'on rappelle le fonctionnement du détecteur de Moravec, décrit en 1981 [23] : considérons une petite fenêtre de $N \times N$ pixels que l'on centre sur un point $(u, v)$ de l'image. Lorsque l'on décale légèrement cette fenêtre dans différentes directions et que l'on calcule la moyenne des différences d'intensité entre les pixels correspondants, plusieurs cas sont à envisager (voir la figure 6.1) :

- Si l'intensité est globalement uniforme autour du point $(u, v)$, les écarts d'intensités calculés seront tous assez faibles.
- Si la fenêtre est située au niveau d'un contour qui passe par le point $(u, v)$, un décalage le long de ce bord engendrera de faibles écarts d'intensité, tandis qu'un décalage de la fenêtre perpendiculairement à la direction de ce bord provoquera des différences plus importantes.
- Si le point $(u, v)$ est un coin, tout décalage de la fenêtre engendrera des écarts d'intensité importants.

Étant donné un déplacement $(\Delta x, \Delta y)$, la fonction d'auto-corrélation au point $(u, v)$ est définie comme suit :

$$c_{(u,v)}(\Delta x, \Delta y) = \sum_{(x_i, y_i) \in W} \left| I(x_i + \Delta x, y_i + \Delta y) - I(x_i, y_i) \right|^2$$

où :
- $I(x, y)$ désigne l'intensité du pixel $(x, y)$
- $W$ est la fenêtre centrée au point $(u, v)$

Ainsi, la méthode décrite ci-dessus revient formellement à retenir les maxima locaux dans l'image de cette quantité :

$$D(u, v) = \min_{(\Delta x, \Delta y) \in E} c_{(\Delta x, \Delta y)}(u, v)$$

où : $E = \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$ désigne les différents déplacements considérés.

### Les améliorations proposées par Harris et Stephens

Le détecteur décrit ci-dessus souffre de trois principaux défauts que Harris et Stephens ont corrigés dans [10] en 1988.

**1. Un opérateur anisotropique** Comme l'ensemble $E$ des décalages utilisés est discret, la réponse est anisotropique. En utilisant une approximation du premier ordre, on obtient :

$$I(x_i + \Delta x, y_i + \Delta y) \approx I(x_i, y_i) + \begin{pmatrix} I_x(x_i, y_i) & I_y(x_i, y_i) \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

où $I_x = \frac{\partial I}{\partial x}$ désigne la dérivée partielle en x de la fonction intensité.

Ainsi :

$$\begin{aligned} c_{(u,v)}(\Delta x, \Delta y) &\approx \sum_{(x_i, y_i) \in W} \left[ \begin{pmatrix} I_x(x_i, y_i) & I_y(x_i, y_i) \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right]^2 \\ &\approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} \underbrace{\begin{pmatrix} \sum_W (I_x(x_i, y_i))^2 & \sum_W I_x(x_i, y_i) I_y(x_i, y_i) \\ \sum_W I_x(x_i, y_i) I_y(x_i, y_i) & \sum_W (I_y(x_i, y_i))^2 \end{pmatrix}}_{=M \text{ : matrice d'autocorrélation}} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \end{aligned}$$

**2. Une réponse trop bruitée** Utiliser une fenêtre rectangulaire et binaire tend à renvoyer des résultats assez bruités. Harris et Stephens proposent plutôt de pondérer les valeurs dans la somme avec une fonction gaussienne 2D $g_\sigma$, d'écart-type $\sigma$, centrée au point $(u, v)$. Ainsi, on réécrit la matrice d'autocorrélation de la sorte :

$$M = \begin{pmatrix} \sum g_\sigma(x_i, y_i)(I_x(x_i, y_i))^2 & \sum g_\sigma(x_i, y_i) I_x(x_i, y_i) I_y(x_i, y_i) \\ \sum g_\sigma(x_i, y_i) I_x(x_i, y_i) I_y(x_i, y_i) & \sum g_\sigma(x_i, y_i)(I_y(x_i, y_i))^2 \end{pmatrix}$$

où pour chaque somme, $(x_i, y_i)$ parcourt toute l'image (en pratique, on continue à ne prendre en compte qu'une région limitée autour du point $(u, v)$, où le poids de la fonction gaussienne n'est pas négligeable).

Ceci peut être réécrit plus simplement grâce à l'opérateur de convolution :

$$M = g_\sigma * \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

**3. Un algorithme trop sensible aux points situés sur des contours** Dans la méthode originale de Moravec, seul le minimum de $c_{(\Delta x,\Delta y)}(u,v)$ pour les 4 déplacements considérés est pris en compte. On comprend facilement que l'algorithme retient souvent des points situés sur des contours alors que nous avons expliqué que l'on cherche à les éviter.

Harris et Stephens utilisent une méthode plus subtile grâce à la *matrice d'autocorrélation* (aussi appelée *matrice des moment du second ordre*) qui représente les variations locales de l'image au point considéré. Comme nous avons vu que $c_{(u,v)}(\Delta x, \Delta y)$ est grand lorsque l'intensité varie fortement dans la direction perpendiculaire au déplacement $(\Delta x, \Delta y)$, on déduit que les points d'intérêt sont les points $(u,v)$ pour lesquels la matrice d'autocorrélation a deux valeurs propres grandes. Cela correspond aux points pour lesquels il existe localement une base de vecteurs propres décrivant des variations locales importantes de l'image.

Harris et Stephens ont proposé de calculer la fonction d'intérêt suivante :

$$\Theta(u,v) = \det(M) - \alpha \operatorname{trace}(M)$$

Le premier terme correspond au produit des valeurs propres, alors que le second pénalise les points de contour qui n'ont qu'une seule forte valeur propre (typiquement, on choisit $\alpha = 0,04$).

Les points d'intérêt retenus sont ainsi ceux qui correspondent aux maxima locaux de cette fonction $\Theta(\cdot,\cdot)$ et qui sont au delà d'un certain seuil.

### Mise en œuvre du détecteur de Harris

1. Construction des images correspondant aux dérivées premières de l'image initiale. Pour ce faire, on peut convoluer l'image par les filtres de dérivation basiques, tels que $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$, mais on peut aussi utiliser une convolution avec la dérivée première d'une gaussienne.
2. Évaluation de la matrice d'autocorrélation en tout point de l'image par calcul de moyennes pondérées par une gaussienne des valeurs issues des images calculées en 1.
3. Calcul de la fonction d'intérêt de Harris $\Theta$ en tout point de l'image.
4. Recherche des maxima locaux de $\Theta$ supérieurs à un seuil fixé.

La figure 6.2 représente les points de Harris détectés sur deux images avec cette méthode.

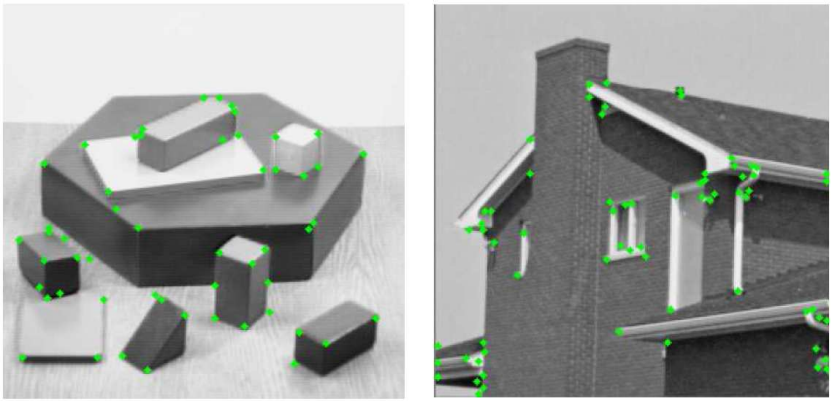


FIG. 6.2 – *Points d'intérêt extraits par le détecteur de Harris.*

### 6.2.3 La mise en correspondance des points d'intérêt

#### Présentation du problème

Une fois les points d'intérêt extraits sur différentes images, le processus le plus courant concerne leurs *mises en correspondance*, ou *appariement*, comme représenté sur la figure 6.3. Supposons être en présence de deux photos d'un même objet. Après avoir détecté les points d'intérêt sur chacune de ces images, nous souhaitons apparier ceux qui correspondent au même point physique de l'objet. Plusieurs difficultés apparaissent :

**Occlusion :** Si la position de la caméra par rapport à l'objet est différente, ou si un autre élément de la scène occulte une partie de l'objet qui nous intéresse, certains points détectés sur la première image peuvent ne pas être visibles sur la deuxième et inversement.

**Répétabilité du détecteur :** En admettant que le point physique soit apparent sur les deux images, comme son apparence est différente, il n'est pas nécessairement retenu par le détecteur de points d'intérêt sur chaque image. C'est la raison pour laquelle nous avons expliqué que le détecteur devait être le plus *robuste* possible.

**Appariement :** Enfin, même si le point a bien été détecté sur les deux images, les mettre en correspondance n'est pas toujours aisé.

#### La somme des différences d'intensité au carré

La plupart des méthodes utilisent des *scores* afin de déterminer pour un point d'intérêt de la première image, celui de la seconde qui lui « ressemble » le plus. Les scores permettent en outre de trier les pixels candidats dans l'ordre de préférence, ce qui rend possible des stratégies globales d'appariement des points des deux images.

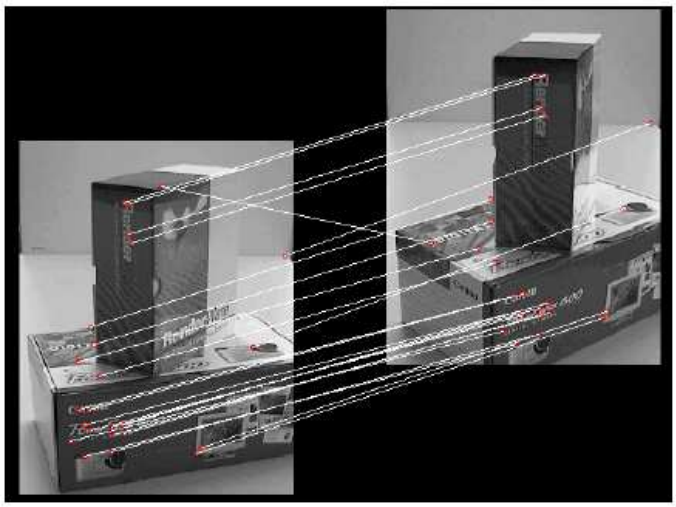


FIG. 6.3 – *Appariement de points d'intérêt.*

Il existe plusieurs méthodes de calcul de *scores*. Ils sont généralement calculés en examinant le voisinage des points d'intérêt.

La méthode la plus simple utilise la SSD (*Sum of Square intensity Difference*), c'est-à-dire la somme des différences d'intensité lumineuse des pixels situés au voisinage de chaque point au carré. Pour un point $M_1$ de la première image et un point $M_2$ de la seconde, cette quantité se calcule donc ainsi :

$$SSD = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \Big[ I_1\big(M_1 + (i,j)\big) - I_2\big(M_2 + (i,j)\big) \Big]^2$$

où :

- $N$ caractérise la taille des voisinages autour des points que l'on considère.
- $I_1$ et $I_2$ représente les fonctions intensités des deux images

Pour trouver le point homologue d'un point $M_1$, la méthode la plus basique consiste à calculer cette quantité pour tous les points d'intérêt $M_2$ détectés sur la seconde image, et à retenir celui qui minimise la SSD.

### La corrélation croisée normalisée

Si la position de l'appareil n'est pas la même pour les deux photos, ou si les conditions lumineuses ont été modifiées, il apparaît que cette quantité est loin d'être optimale. Le simple fait que la seconde image soit plus sombre suffit à fausser totalement les résultats obtenus.

Il existe de multiples mesures de corrélation et nous pouvons citer celle qui utilise la *corrélation croisée normalisée*. Introduisons les opérateurs suivants :

- La moyenne d'intensité au voisinage du point $M$ de l'image d'intensité $I$ :
$$I(\bar{M}) = \frac{1}{N^2} \sum_{i=-N}^{N} \sum_{j=-N}^{N} I(M + (i, j))$$
- Le « produit scalaire » :
$$\langle I_1(M_1), I_2(M_2) \rangle = \frac{1}{N^2} \sum_{i=-N}^{N} \sum_{j=-N}^{N} \big(I_1(M_1+(i,j)) - \bar{I_1}\big)\big(I_2(M_2)(i,j) - \bar{I_2}\big)$$
- La norme :
$$|I(M)| = \sqrt{\langle I(M), I(M) \rangle}$$

La corrélation croisée normalisée calculée entre le voisinage du point $M_1$ de l'image d'intensité $I_1$ et le point $M_2$ de l'image d'intensité $I_2$ est alors cette quantité :

$$\rho = \frac{\langle I_1(M_1), I_2(M_2) \rangle}{|I_1(M_1)||I_2(M_2)|}$$

Cette valeur est comprise entre $-1$ et 1. Plus elle est proche de 1, plus les points $M_1$ et $M_2$ (et leur voisinage) se ressemblent. Un score proche de 0 indique que les patchs ne sont pas corrélés, alors qu'un score de $-1$ signifie que les patchs sont les négatifs l'un de l'autre.

### Limites

Ces méthodes d'appariement fondées sur des mesures de corrélations trouvent assez rapidement leurs limites, ceci pour deux principales raisons :

1. Si l'on considère des voisinages classiques de $64 \times 64$ pixels, chaque point d'intérêt est en fait caractérisé par $64 \times 64 = 4096$ entiers compris entre 0 et 255 (pour des images typiques en niveaux de gris). Avec la méthode basique présentée ci-dessus où l'on propose de calculer des distances basées sur ces représentations pour chaque paire de points d'intérêt détectés, on comprend que le temps d'exécution soit particulièrement important. Même s'il existe différents algorithmes plus sophistiqués, comme *kd-tree* ou *Best Bin First*, les calculs restent relativement lents si l'on travaille dans de telles dimensions. Or le critère du temps d'exécution est particulièrement important pour mon projet où l'on a pour objectif un programme capable de traiter les images en temps-réel, telles qu'elles sont enregistrées par la caméra, soit 25 images par seconde. Il y a bien sûr une certaine redondance d'information lorsque l'on considère ces patchs autour de chaque point d'intérêt.

De nombreux *descripteurs* ont été proposés pour « décrire » le voisinage de l'objet par des vecteurs de quelques dizaines d'entiers : par exemple, le descripteur *SIFT* (détaillé dans la section suivante), qui est l'un des plus utilisé aujourd'hui, consiste en un vecteur de 128 entiers. La ressemblance entre des descripteurs peut alors être mesurée à l'aide de différentes distances, comme la *distance euclidienne* classique ou la *distance de Mahalanobis* qui permet d'accorder un point moins important aux composantes des descripteurs les plus bruitées.

2. Les premières applications de ces méthodes utilisant des mesures de corrélation ont été la *stéréovision* et le *tracking*. Dans les deux cas, les deux images où l'on souhaite mettre en correspondance les points d'intérêt sont très similaires et le problème est ainsi simplifié. Zhang *et al.* [27] ont montré en 1995 qu'il était possible d'apparier des points de Harris en utilisant des fenêtres de corrélation entre deux images prises sous des angles de vue très différents. Cependant, même si la position de l'appareil photo par rapport à l'objet est identique sur les deux images, il suffit que l'image ait subi une rotation pour que l'appariement soit complètement perturbé. Notamment depuis le travail de Schmid et Mohr [25] de 1997, la plupart des descripteurs des voisinages des points d'intérêt sont invariants à la rotation.

### 6.2.4 Scale Invariant Feature Transform (SIFT)

Afin de mieux comprendre comment peut être défini un descripteur de points d'intérêt, nous allons décrire la méthode *SIFT*, établie en 1999 par David G. Lowe [17, 18]. C'est encore aujourd'hui l'un des descripteurs les plus performants.

Au moment de calculer le descripteur, le point d'intérêt s'est déjà vu attribué une position, une orientation et même une échelle (nous verrons plus loin comment l'échelle peut être déterminée. Nous préférons dans un premier temps expliquer le principe d'un descripteur). Reste donc à définir le descripteur de son voisinage, qui doit être aussi invariant que possible aux variations restantes, telles que les changements d'illuminations ou de point de vue.

Lowe s'est inspiré d'études sur le système visuel biologique, notamment sur les neurones complexes du cortex visuel primaire. Plus que les intensités lumineuses des pixels du voisinage, ce sont les gradients qui sont pris en compte. De plus, de petits décalages de la position du point d'intérêt sont tolérés.

La figure 9.1 illustre le principe de fonctionnement de l'algorithme. Dans un premier temps, en utilisant l'image $L(\cdot, \cdot, \sigma)$ de l'espace d'échelle correspondant à l'échelle $\sigma$ déterminée préalablement (voir plus loin), on calcule la norme $m$ et l'orientation $\theta$ des gradients d'intensité en chacun des pixels $(x, y)$ du voisinage de taille $16 \times 16$ autour du point d'intérêt $(u, v)$ :

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

$$\theta(x,y) = \tan^{-1} \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

Ces calculs sont symbolisés par les petites flèches dans la partie gauche de la figure. Afin d'assurer l'invariance à la rotation, les orientations des gradients sont ajustées pour prendre en compte l'orientation du point d'intérêt, attribuée précédemment.

Le descripteur est symbolisé dans la partie droite de la figure. Il correspond à un ensemble de plusieurs histogrammes des orientations de gradient, chacun représentant une zone de $4 \times 4$ pixels. Comme le montre la figure, chaque histogramme est composée des huit classes.

En fait, afin de ne pas trop pénaliser un petit décalage dans la localisation du point d'intérêt et pour éviter les effets de bords, la contribution de chaque gradient — initialement égale à sa norme — est pondérée selon trois critères :

- chaque gradient calculé peut avoir un impact dans quatre histogrammes différents, car un gradient est pris en compte pour un histogramme si la distance entre le point où il est calculé et le centre de la zone de l'histogramme est inférieure à la « largeur » de quatre pixels. Sa contribution est pondérée en prenant en compte cette distance.
- De même, chaque gradient est pris en compte dans deux classes contiguës de chaque histogramme, correspondant aux deux orientations (parmi les huit) qui sont les plus proches de l'orientation du gradient étudié. Sa participation à chacune des deux classes est une nouvelle fois pondérée pour tenir compte de l'écart entre l'orientation du centre de la classe et la véritable orientation du gradient.
- La contribution des gradients dans les histogrammes est enfin pondérée selon une fonction gaussienne symbolisée par le cercle gras de la partie gauche de la figure 9.1, afin d'attribuer une importance moindre aux pixels situés plus loin du point d'intérêt détecté.

Le vecteur du descripteur est alors formé en concaténant les valeurs des entrées de chaque histogramme, correspondant aux longueurs des flèches sur la figure. Celle-ci ne montre qu'un tableau de $2 \times 2$ histogrammes calculés à partir d'un voisinage de $8 \times 8$ pixels alors que la méthode imaginée par Lowe propose en fait de construire $4 \times 4$ histogrammes en considérant un voisinage de $16 \times 16$ pixels. Comme chaque histogramme est constitué de huit classes, le vecteur caractérisant un point d'intérêt est donc composé de $4 \times 4 \times 8 = 128$ valeurs.

Dans une dernière phase, ce vecteur est modifié afin de réduire les effets des variations d'illumination :

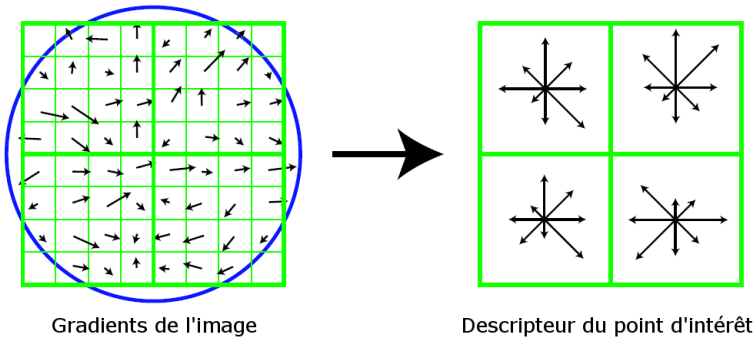



FIG. 6.4 – *Principe du descripteur* SIFT

- Il est d'abord normalisé à l'unité : une modification du contraste de l'image — qui multiplie la valeur de chaque pixel par une constante — multiplierait tous les gradients par la même constante et n'aurait donc aucun impact sur le vecteur si celui-ci est normalisé. Notons qu'un changement de luminosité — qui ajoute une constante à chaque pixel de l'image — ne modifie pas la valeur des gradients qui sont calculés comme une différence d'intensité entre les pixels voisins. Ainsi, le descripteur est invariant à tout changement affine de l'illumination.
- Des modifications non linéaires des intensités peuvent aussi se produire (saturation, modification des conditions d'illumination de la scène, etc.). Généralement, les effets sont plus importants pour les normes des gradients que pour leurs orientations. L'influence des gradients dont la norme est trop importante est minimisée en seuillant le vecteur normé par une valeur de $0,2$ puis en renormalisant le vecteur descripteur. Ceci revient à donner plus d'importance à la distribution des orientations des gradients et à restreindre l'impact des gradients dont la norme est élevée.

### 6.2.5 Des patchs invariants aux changements de point de vue

#### Limites des voisinages de forme fixe

Pour comparer deux points d'intérêt, nous considérons toujours leur voisinage, que ce soit par l'intermédiaire de la corrélation croisée normalisée ou d'un descripteur comme le descripteur SIFT que nous venons d'étudier. Sans information supplémentaire sur le point d'intérêt, nous avons jusqu'ici proposé de prendre en compte des régions de forme fixe autour des points, ce qui pose évidemment des problèmes lorsque les images sont prises avec des points de vue très différents. Dans la figure 6.5, la position de l'appareil photo par rapport au livre a été modifiée entre les deux images (a) et (b). Comme le montrent les images (d) et (e), il est évident que considérer des

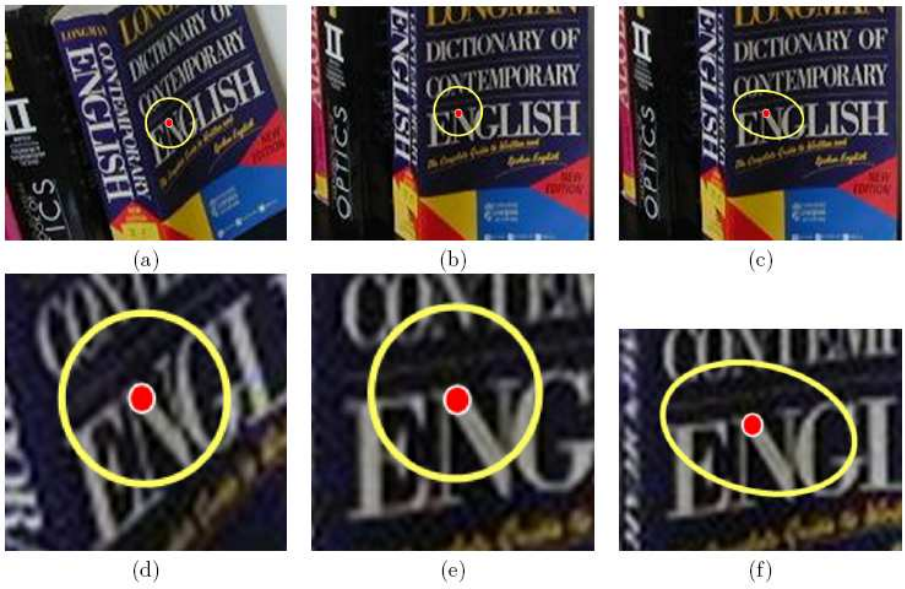



FIG. 6.5 – *Nécessité d'utiliser des voisinages dont la forme n'est pas fixe*

voisinages de forme fixe (ici circulaire) ne permet pas de gérer les déformations géométriques résultant du changement de point de vue, car la région étudiée ne correspond pas à la même zone physique de l'objet (et ce même en normalisant par rapport à la rotation comme nous l'avons vu plus haut). L'objectif est d'utiliser des voisinages de forme variable, qui, pour des points d'intérêt homologues, doivent correspondre à la projection de la même région de l'espace. Ainsi, en utilisant un patch elliptique comme sur l'image (c), on note sur l'image (f) que cela permet de recouvrir exactement la même zone de l'objet que sur l'image (d), et donc facilite grandement l'appariement.

### Points d'intérêt invariants à l'échelle

Les premières transformations géométriques étudiées dans ce cadre ont été les changements d'échelle. Généralement, on étudie ainsi deux photos prises avec des zooms différents. En considérant un voisinage de taille fixe autour des points d'intérêt, on comprend aisément qu'il soit quasiment impossible d'apparier les deux images.

Plusieurs approches ont été décrites pour obtenir l'invariance aux changements d'échelle. En 1998, les propriétés d'échelle caractéristiques ont été étudiées par Lindeberg [15]. Il propose d'utiliser ce que l'on appelle un *espace d'échelle*, qui est en fait un ensemble d'images d'une scène représentée à plusieurs niveaux de résolution. Pour les générer, on convolue l'image initiale avec des gaussiennes ayant des écarts-type différents et un espace d'échelle

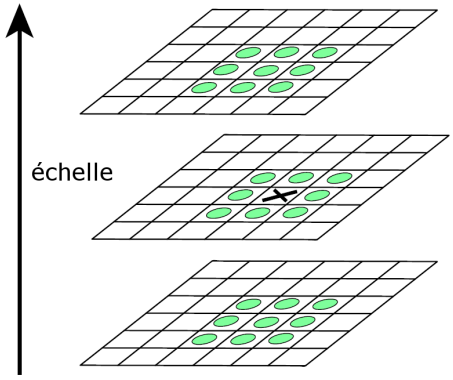



FIG. 6.6 – *Extrema dans l'espace d'échelle.*

est donc défini comme une fonction $L$ :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

où :
- $\sigma$ est l'écart-type de la gaussienne $G$, que l'on fait varier
- $*$ est l'opérateur de convolution
- $I$ est la fonction intensité de l'image

Le principe est alors de calculer la réponse d'une fonction $F$ donnée en chaque point de l'image et pour chaque niveau de résolution de cet espace d'échelle. Les échelles caractéristiques des structures locales sont alors indiquées par des extrema dans la représentation tri-dimensionnelle : les deux dimensions classiques de l'image et la dimension correspondant aux différentes échelles testées. Un pixel est ainsi comparé à ses 26 voisins (Figure 6.6).

Différentes fonctions $F$ ont été proposées. En voici deux exemples :
- Comme le détecteur de Harris décrit plus haut est l'un des plus fiables et des plus répétables, on a d'abord essayé de se servir de la fonction qu'il utilise pour trouver des extrema dans l'image, afin de trouver ceux dans l'espace tridimensionnel. Cependant, la fonction atteint rarement des extrema dans l'espace d'échelle et ne convient donc pas. La méthode dite de *Harris-Laplacien* propose d'utiliser la fonction de Harris pour la localisation spatiale du point d'intérêt et l'opérateur Laplacien pour déterminer son échelle caractéristique.
- Pour le détecteur *SIFT*, Lowe propose en 1999 d'utiliser une fonction $F$ égale à la différence de deux gaussiennes [17]. Ceci présente notamment l'intérêt de réutiliser les images calculées pour créer l'espace d'échelle, puisqu'il suffit de les soustraire deux à deux.

La figure 6.7 donne un exemple d'échelle caractéristique sur deux photos prises avec un zoom différent. On a ici calculé la réponse au Laplacien à diffé-

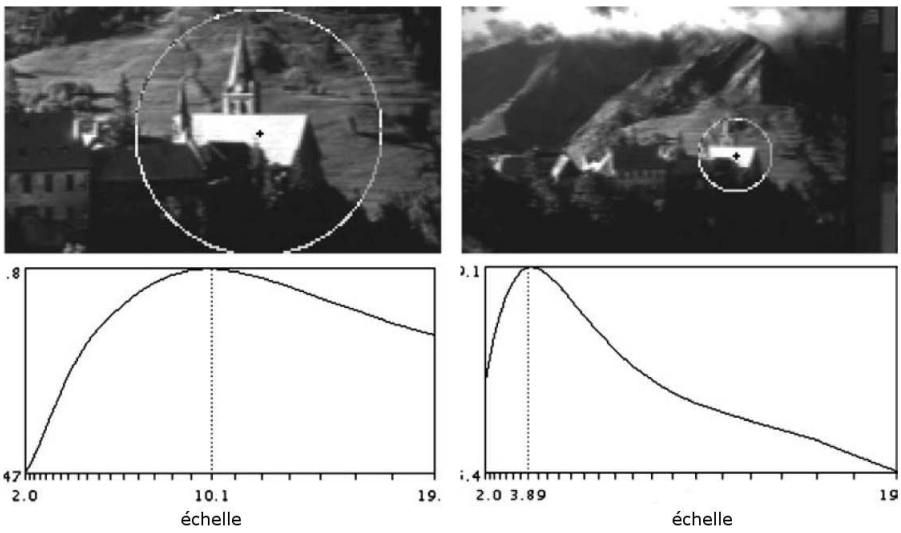



FIG. 6.7 – *Exemple d'échelle caractéristique.*

rentes échelles et sélectionné les maxima. Le rapport des échelles correspond au facteur d'échelle entre les deux images.

### Importance des transformations affines

Si l'on généralise et considère un mouvement quelconque de l'appareil photo par rapport à l'objet (et non plus seulement un éloignement ou un rapprochement), il faut prendre en compte des transformations géométriques plus complexes.

Le plus souvent, les régions de l'objet autour des points d'intérêt peuvent être considérées comme localement planes (ceci est notamment faux si le point est situé sur une arête de l'objet). En *géométrie projective*, deux images d'un même plan sont liées par ce que l'on appelle une *homographie* ou encore une *transformation projective* (Figure 6.8). Cependant, une homographie peut être localement approchée par une transformation affine car l'on peut généralement ignorer les effets de perspectives.

### Points d'intérêt invariants aux transformations affines

Idéalement, on cherche donc des méthodes capables d'extraire des points d'intérêt accompagnés de paramètres permettant de définir la forme du voisinage à considérer (souvent les paramètres d'une ellipse, comme dans l'exemple de la figure 6.5).

Les formes des régions sont déterminées indépendamment pour chaque point d'intérêt. Pour pouvoir comparer ces formes variables entre elles et calculer des indices de ressemblance, elles sont normalisées dans une phase

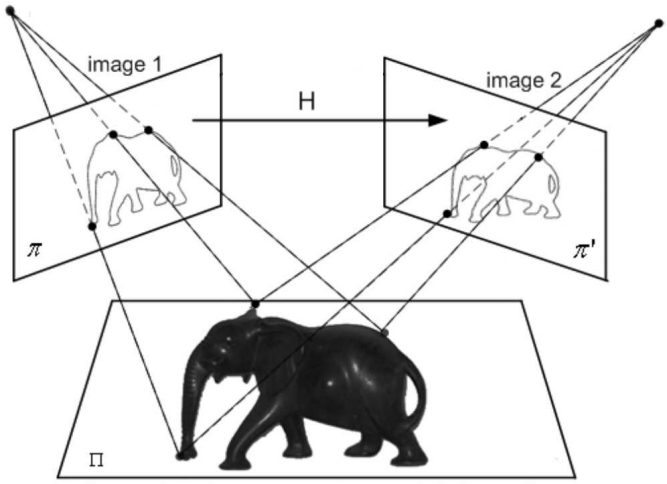



FIG. 6.8 – *Deux images d'un même plan sont liées par une homographie.*

préliminaire. Ainsi, toutes les ellipses sont « transformées » en cercles de même taille et on peut calculer des descripteurs à partir de ces voisinages normalisés comme on le faisait auparavant lorsque les voisinages avaient une forme fixe. Ce processus est représenté sur la figure 6.9 : à partir de deux vues de la même partie d'un objet (a), on détecte les points d'intérêt et la forme de leur voisinage (b). La troisième colonne est un agrandissement des régions détectées. On normalise alors les ellipses pour obtenir des cercles de même taille (d). Enfin, comme précédemment, on a la possibilité de normaliser une deuxième fois les patchs pour obtenir une invariance à la rotation et à d'éventuelles déformations photométriques (e).

Différents algorithmes ont été proposés ces dernières années afin de détecter de tels points d'intérêt ainsi que leur voisinage, de façon invariante aux transformations affines. Un article de 2005 dont les auteurs sont huit chercheurs qui ont particulièrement participé à la recherche dans ce domaine dresse un état de l'art [22] en comparant plusieurs méthodes qui sont certainement les plus utilisées aujourd'hui.

**Le détecteur *Harris-Affine* [21], 2002.** Les points d'intérêt sont extraits avec le détecteur de Harris et l'échelle est déterminée avec la méthode décrite plus haut qui utilise les extrema du Laplacien dans l'espace d'échelle. La forme elliptique du voisinage est calculée grâce à une méthode décrite dans un article de Lindeberg et Gårding de 1997 [16] qui utilise aussi la *matrice d'autocorrélation* introduite dans la section 6.2.2 page 26.

$$M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

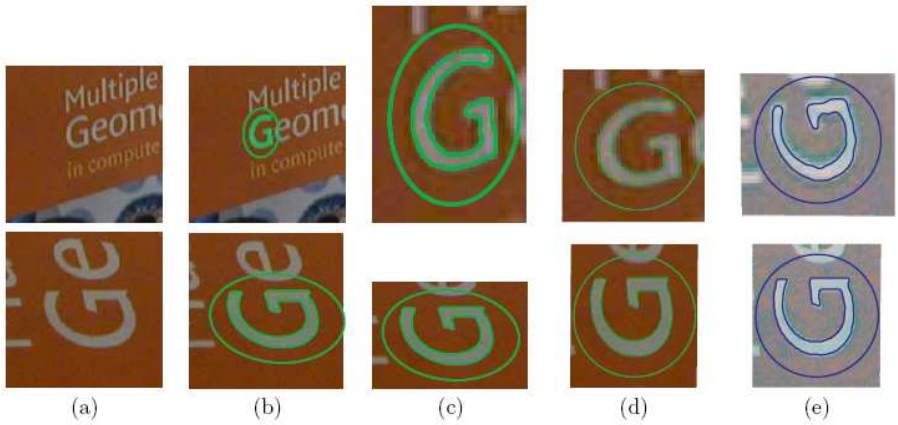



FIG. 6.9 – *Régions invariantes aux transformations affines et normalisations.*

L'idée est de rechercher de façon itérative la transformation qui, appliquée au point d'intérêt et à son voisinage, permettrait d'obtenir une *matrice d'autocorrélation* qui aurait ses deux valeurs propres égales.

**Le détecteur *Hessian-Affine* [21], 2002.** Le principe est le même que pour le détecteur Harris-Affine, sauf que les points d'intérêt sont détectés en utilisant la matrice Hessienne et non grâce à la matrice d'autocorrélation.

$$H = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix}$$

où $I_{xx}$, $I_{yy}$ et $I_{xy}$ représentent les dérivées secondes de la fonction intensité.

**Le détecteur *EBR* (*Edge Based Region detector*) [26], 2002.** Cette méthode part du principe que les contours sont généralement plus stables que les points d'intérêt car ils peuvent être détectés malgré des changements importants d'échelle, de point de vue ou d'illumination. Les points d'intérêt sont une fois de plus sélectionnés grâce au détecteur de Harris. Les contours correspondant à chaque coin de Harris détecté sont extraits grâce au célèbre *algorithme de Canny* [5] qui date de 1986. Pour chaque coin détecté $\boldsymbol{p}$ (voir figure 6.10), on fait alors se déplacer deux points $\boldsymbol{p_1}(l)$ et $\boldsymbol{p_2}(l)$ qui s'éloignent dans les deux directions en suivant le contour et avec des vitesses paramétrées couplées. Ces points définissent ainsi une famille de parallélogrammes et on sélectionne celui qui permet de maximiser une certaine fonction qui dépend des paramètres photométriques de la zone de l'image située à l'intérieur. Ainsi, la forme du voisinage est dans ce cas un parallélogramme et non une ellipse.

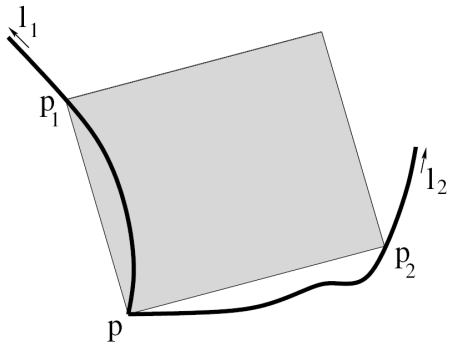



FIG. 6.10 – *Principe du détecteur* EBR

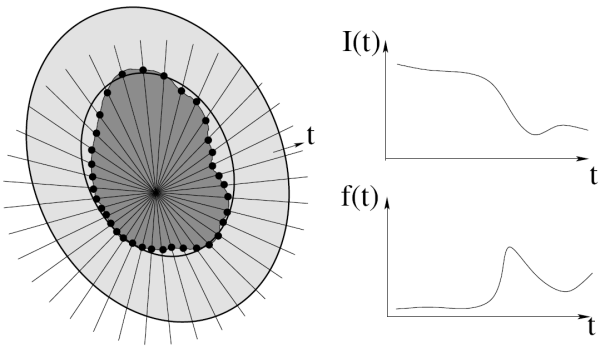



FIG. 6.11 – *Principe du détecteur* IBR

**Le détecteur *IBR* (*Intensity extrema Based Region detector*) [26], 2000.** Les points d'intérêt utilisés par cette méthode sont tout simplement les extrema locaux de la fonction intensité de l'image. Une fonction $f$ dépendant de l'intensité $I$ est étudiée le long de rayons émanant de chaque point d'intérêt (voir figure 6.11). Pour chaque rayon, le point où cette fonction $f$ passe par un extremum est sélectionné. Un tel point correspond généralement à une position où l'intensité augmente ou diminue rapidement. On relie alors ces points pour obtenir une région supposée être invariante aux transformations affines.

**Le détecteur *MSER* (*Maximally Stable Extremal Region*) [20], 2002.** Les *MSER* sont des régions de l'image qui sont des composantes connexes « stables » de l'image seuillée. Plus exactement, ces régions sont déterminées en appliquant un seuillage à l'image (c'est-à-dire que tous les pixels ayant une intensité inférieure à un seuil donné deviennent noirs et les autres deviennent blancs) avec une valeur pour le seuil qui part de zéro et augmente continûment. Durant le processus, on étudie l'évolution de la taille de

chaque région « noire ». Lorsque le grossissement de cette taille passe par un minimum local (en fonction de l'augmentation du seuil), la composante connexe est dite « au plus stable ». On a alors obtenu une *MSER* (le mot « extremal » évoque la propriété selon laquelle les pixels de la *MSER* sont tous plus foncés — ou tous plus clairs — que les pixels situés aux frontières de la région).

**Des détecteurs complémentaires** Aucune de ces différentes méthodes ne surpasse les autres pour tous les types d'images et tous les types de transformations. Les résultats sont en effet souvent très variables pour un même algorithme, selon qu'on l'utilise avec des images contenant des contours distincts (bâtiments, dessins animés, etc.) ou avec des images ayant de nombreuses textures (scènes naturelles).

Pour résumer, on peut néanmoins affirmer que la méthode *MSER*, suivie de l'approche *Harris-Affine*, donne généralement les meilleurs résultats. Cependant, les détecteurs sont complémentaires car ils ont des propriétés différentes.

### Applications

Savoir extraire et mettre en correspondance des points d'intérêt à partir d'images prises sous des points de vue très différents grâce à des régions invariantes aux transformations affines a de multiples application en vision par ordinateur.

- Reconstruction tridimensionnelle d'une scène photographiée sous différents points de vue
- Détection d'objets (par exemple un véhicule pour un logiciel d'aide à la conduite)
- Catégorisation d'images
- Reconnaissance d'objets définis pour lesquels l'algorithme a été entraîné
- Recherche d'images par le contenu dans une grosse base de données ou dans un film
- Construction de panoramas [4]

# Chapitre 7

# Appariement de points d'intérêt en utilisant des arbres de classification

## 7.1 L'article de Lepetit *et al.*

J'ai été particulièrement intéressé par la méthode décrite dans l'article « Randomized Trees for Real-Time Keypoint Recognition » [13] (En Français : « Arbres aléatoires pour la reconnaissance de points d'intérêt en temps réel ») qui a été publié quelques mois avant mon arrivée à Siemens dans une des conférences majeures en vision par ordinateur : « Conference on Computer Vision and Pattern Recognition » (*CVPR'05*). Elle est l'œuvre de chercheurs de l'École Polytechnique Fédérale de Lausanne (EPFL) : Vincent Lepetit, Pascal Lagger et Pascal Fua. Ils ont publié en septembre 2006 dans le journal PAMI (*Pattern Analysis and Machine Intelligence*) [12], ce qui est une confirmation que l'idée est particulièrement intéressante.

## 7.2 Temps de calcul

Le principal avantage de cette méthode par rapport à la plupart des techniques évoquées plus haut concerne la vitesse d'exécution. Les auteurs affirment obtenir des résultats temps-réel, c'est-à-dire qu'ils parviennent à traiter au moins 25 images par seconde. C'est ce que mon tuteur souhaite pour pouvoir mettre en œuvre un algorithme de réalité augmentée en temps-réel.

Les détecteurs de points d'intérêt décrits ci-dessus sont en effet généralement assez lents. De plus, comme nous l'avons expliqué, la façon la plus simple de trouver les correspondances entre deux images est de comparer tous les points d'intérêt d'une image avec tous ceux de l'autre, en utilisant des descripteurs locaux comme SIFT. Malheureusement, cette méthode est

quadratique par rapport au nombre de points d'intérêt, ce qui la rend inutilisable dans le cas d'un algorithme temps-réel. Des algorithmes d'appariement utilisant différentes sortes de *schémas d'indexation*, qui ne garantissent pas l'identification des plus proches voisins exacts, mais qui sont bien plus efficaces, ont bien sûr été proposées [2], mais le temps de calcul reste toujours trop important.

L'idée de Vincent Lepetit et de son équipe est d'utiliser une phase d'apprentissage pendant laquelle on peut se permettre de prendre un peu de temps pour entraîner le système, afin de rendre la phase de mise en correspondance, qui est la plus cruciale du point de vue du temps d'exécution, plus rapide.

Une telle approche n'est évidemment pas adaptée à la mise en correspondance d'images quelconques, étant donné que le système nécessite une phase d'apprentissage. Cependant, ceci ne constituait pas un véritable problème par rapport aux attentes de Siemens, mon tuteur étant plutôt intéressé par la reconnaissance, la détection ou le suivi d'objets donnés et connus à l'avance (comme un moteur d'avion, voir section 5.1 page 21).

## 7.3 L'appariement de points d'intérêt vu comme un problème de classification

Une unique image de l'objet d'intérêt que l'on souhaite reconnaître est utilisée pendant la phase d'apprentissage. Nous appellerons cette image l'*image de référence*.

La première étape de la phase d'apprentissage consiste à sélectionner un ensemble $\mathbf{K} = \{\mathbf{k}_1 \dots \mathbf{k}_N\}$ de $N$ points d'intérêt sur cette image de référence (en pratique, on choisira souvent $N = 200$). Dès lors, pendant la phase d'appariement qui doit s'accomplir en temps réel, les points d'intérêt sont dans un premier temps extraits de l'image courante, puis on attribue à chaque patch $\mathbf{p}(\mathbf{k}^{courant})$, centré sur un point d'intérêt $\mathbf{k}^{courant}$ de l'image courante, une étiquette $Y\big(\mathbf{p}(\mathbf{k}^{courant})\big)$ correspondant à une des classes de l'ensemble $\mathbf{C} = \{-1, 1, 2, \dots, N\}$, tel que :

- Si $Y\big(\mathbf{p}(\mathbf{k}^{courant})\big) = i$ où $i \in \{1 \dots N\}$, le point $\mathbf{k}^{courant}$ est classé avec le point d'intérêt $\mathbf{k}_i$ de l'image de référence.
- Si $Y\big(\mathbf{p}(\mathbf{k}^{courant})\big) = -1$, le point $\mathbf{k}^{courant}$ est classé avec tous les points qui ne font partie d'aucune des classes de l'ensemble $\mathbf{K}$.

Pour mieux comprendre cette classification, il faut considérer qu'une classe $i$, où $i \in \{1 \dots N\}$, correspond à l'ensemble de toutes les apparences possibles que peut prendre le voisinage du point d'intérêt $\mathbf{k}_i$ de l'image de référence, lorsqu'il est vu sous différents points de vue ou avec différentes conditions d'illumination, de bruit, etc. Ainsi, si $Y\big(\mathbf{p}(\mathbf{k}^{courant})\big) = i$, cela signifie que les points $\mathbf{k}^{courant}$ de l'image courante, et $\mathbf{k}_i$ de l'image de référence, correspondent en fait au même point physique de l'objet d'intérêt.

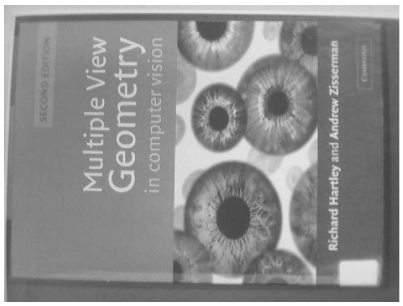




FIG. 7.1 – *Exemples d'images de référence de l'objet d'intérêt*

Construire $Y$ est bien sûr théoriquement impossible. Le but du problème est de construire un classifieur $\hat{Y}$ telle que la probabilité $\mathbf{P}\big(\hat{Y}(\mathbf{p}) \neq Y(\mathbf{p})\big)$ qu'un patch soit mal classé, soit la plus faible possible.

## 7.4 La base de données d'apprentissage

Pour certaines tâches, comme la détection de caractères ou de visages, on peut utiliser des bases de données contenant de multiples images pour construire le classifieur. Pour le problème qui nous concerne, nous voulons cependant n'utiliser qu'une unique image de l'objet d'intérêt en entrée. La figure 7.1 montre deux exemples d'images de référence utilisées pour entraîner le système.

### 7.4.1 Changements de point de vue

La base de données d'apprentissage contient en fait des images synthétiques que les auteurs proposent de générer en déformant l'image de référence par des transformations affines. Ainsi, comme nous l'avons vu dans la section 6.2.5 page 38, si les patchs locaux peuvent être considérés comme étant localement plans, on obtient l'apparence qu'ils auraient depuis différents points de vue.

Une transformation affine $\mathbf{A}$ peut être décomposée en $\mathbf{A} = \mathbf{R}_\theta \mathbf{R}_\theta \mathbf{R}_\phi^{-1} \mathbf{S} \mathbf{R}_\phi$ où $\mathbf{R}_\theta$ et $\mathbf{R}_\phi$ sont des matrices de rotation et où la matrice diagonale $\mathbf{S} = diag[\lambda_1, \lambda_2]$ est une matrice de changement d'échelle.

Les angles $\theta$ et $\phi$, ainsi que les facteurs d'échelle $\lambda_1$ et $\lambda_2$ définissent la transformation affine. Pour la construction de la base de données d'apprentissage, Lepetit *et al.* proposent que $\theta$ et $\phi$ varient dans l'intervalle $[-\pi; \pi]$, et les facteurs d'échelle $\lambda_1$ et $\lambda_2$ dans l'intervalle $[0,3; 5]$. Évidemment, plus ces intervalles sont petits, plus la classification est facile (car, comme les patchs ont tendance à avoir une apparence plus similaire, la variance intra-classe diminue), mais la région autour de l'objet d'intérêt dans laquelle le programme sera susceptible de fonctionner correctement sera aussi plus petite.

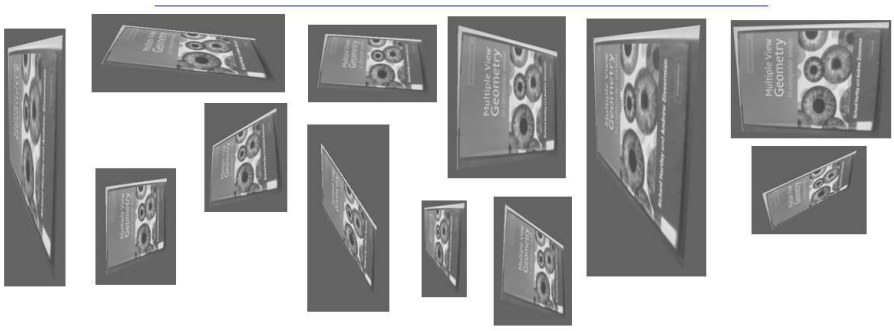


FIG. 7.2 – *Exemples d'images synthétiques utilisées pour la phase d'apprentissage*

La figure 7.2 présente différentes images qui ont été créées à partir de la première photo de la figure 7.1, afin de construire le classifieur.

### 7.4.2 Variation des conditions d'illumination

Aucune transformation de l'intensité des pixels n'est opérée pendant la génération des images de la base de données d'apprentissage. Comme nous le verrons plus loin, les tests utilisés pour la classification consistent uniquement en des comparaisons binaires des intensités (« tel pixel du patch entourant le point d'intérêt est-il plus sombre ou plus clair que tel autre pixel ? »). Ainsi, la classification est invariante aux changements monotones d'intensité. Pour les autres variations d'illuminations, telles que les ombres, la réflexion spéculaire ou les effets de saturation, les auteurs comptent uniquement sur la robustesse de la méthode de classification.

## 7.5 Extraction des points d'intérêt

La méthode utilisée pour la détection des points d'intérêt n'est pas déterminante dans l'approche présentée dans l'article. La plupart des méthodes connues pourraient donner des résultats intéressants. Néanmoins, la contrainte temps-réel impose un algorithme très rapide. Le détecteur de Harris présenté dans la section 6.2 page 30, est par exemple trop lent. Lepetit *et al.* ont proposé leur propre algorithme, qui est très simple et qui permet d'attribuer très facilement une orientation à chaque point d'intérêt.

Comme illustré sur la figure 7.3.a, l'idée de base est de parcourir l'image en considérant les intensités des pixels situés sur des cercles centrés en chaque point. Si deux pixels diamétralement opposés sur ce cercle ont une intensité proche de celle du pixel du centre, on décide que celui-ci n'est pas un point d'intérêt. En effet, pour un point situé dans une zone uniforme ou le long d'un contour (c'est-à-dire pour les points que l'on souhaite éviter), on peut

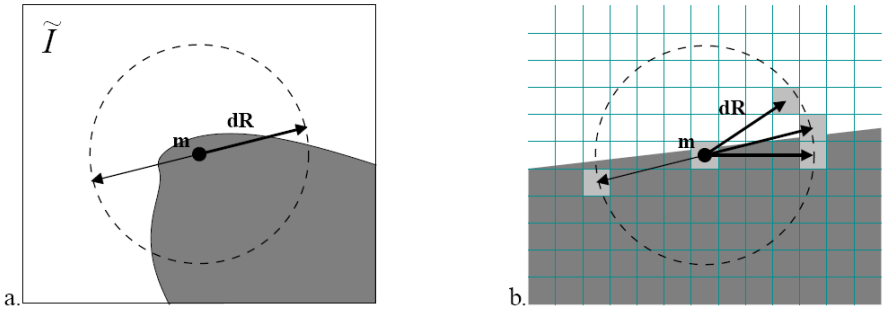



FIG. 7.3 – *Principe de l'algorithme proposé par Lepetit* et al. *pour la détection des points d'intérêt*

toujours trouver une telle paire de pixels diamétralement opposés.

Ainsi, autour de chaque point $\mathbf{m}$ de l'image, on parcourt le cercle en effectuant des tests du type :

$$\text{Si} \quad |\tilde{I}(\mathbf{m}) - \tilde{I}(\mathbf{m} + \mathbf{dR}_\alpha)| \leq \tau \quad \text{et} \quad |\tilde{I}(\mathbf{m}) - \tilde{I}(\mathbf{m} - \mathbf{dR}_\alpha)| \leq \tau$$
$$\text{Alors} \quad \mathbf{m} \text{ n'est pas un point d'intérêt.}$$

où :
- $\tilde{I}$ est l'image convoluée par une fonction gaussienne, afin de réduire la sensibilité au bruit du détecteur de points d'intérêt
- $\mathbf{dR}_\alpha = (Rcos\alpha; Rsin\alpha)$, $R$ étant le rayon du cercle et $\alpha$ variant dans l'intervalle $[0; \pi]$

Généralement, la plupart des pixels de l'image sont rejetés très rapidement après le premier test, sans avoir à parcourir tout le cercle.

Comme nous l'avons expliqué dans la section 6.2 page 30, on cherche le plus souvent à éviter les points situés sur des contours car ils ne sont pas stables. Ainsi, comme nous travaillons toujours avec des images discrétisées, nous ne comparons pas uniquement les pixels diamétralement opposés, mais aussi leurs voisins, comme illustré sur la figure 7.3.b.

Comme pour la plupart des détecteurs de points d'intérêt, plusieurs réponses positives autour de la zone d'intérêt sont détectées. Afin d'éviter de tels amas de points d'intérêt, un score est attribué à chaque point d'intérêt détecté, et seuls les maxima locaux sont conservés. Ce score, qui est calculé pendant le parcours du cercle, est le suivant :

$$S(\mathbf{m}) = \sum_{\alpha \in [0;\pi[} \tilde{I}(\mathbf{m} - \mathbf{dR}_\alpha) - 2\tilde{I}(\mathbf{m}) + \tilde{I}(\mathbf{m} + \mathbf{dR}_\alpha)$$

## 7.6 Attribution d'une orientation

L'apparence du voisinage d'un point d'intérêt peut évidemment varier énormément lorsque son orientation par rapport à la caméra évolue, même si la position de celle-ci reste constante. Comme l'invariance à l'orientation est généralement assez facile à obtenir, Lepetit *et al.* propose de l'utiliser afin de diminuer la variabilité de l'apparence des patchs d'une même classe, et ainsi simplifier le travail du classifieur.

Pour attribuer une orientation à chaque patch détecté, la méthode utilisée est celle présentée dans l'article de Lowe qui introduit le détecteur SIFT [17, 18]. En effet, nous avons évoqué le fait, dans la section 9.1 page 59, que le descripteur SIFT d'un point d'intérêt est calculé en prenant en compte le voisinage déjà normalisé par rapport à l'orientation.

L'orientation affectée à un point d'intérêt correspond à l'orientation majoritaire des gradients d'intensité dans un voisinage de ce point.

## 7.7 Sélection des points d'intérêt de l'image de référence

Les $N$ points d'intérêt de l'ensemble $\mathbf{K}$ (cf. section 7.3), sélectionnés sur l'image de référence, sont utilisés pendant la phase de détection pour établir des correspondances avec les points de l'image courante. Pour maximiser le nombre de correspondances que l'on va pouvoir établir, il faut que les points physiques de l'objet d'intérêt correspondant aux points d'intérêt de l'ensemble $\mathbf{K}$ soient tels que leurs projections sur les images traitées pendant la phase de détection aient la probabilité la plus élevée possible d'être détectées comme des points d'intérêt, malgré la perspective, les variations d'illumination et le bruit.

Pour sélectionner les points d'intérêt les plus stables, Lepetit *et al.* proposent de synthétiser de nombreuses images en appliquant des transformations affines à l'image de référence. Les points d'intérêt sont extraits sur chacune de ces images avec le détecteur décrit dans la section précédente, et comme la transformation affine est connue, on peut connaître les vraies correspondances entre les images, et ainsi calculer la détectabilité des points d'intérêt, c'est-à-dire la proportion des images où ils sont effectivement retenus par le détecteur.

La figure 7.4 illustre ce processus. Chaque ligne correspond à l'apparence que prend le voisinage d'un point donné de l'objet sous différents points de vue. Le patch est vert lorsqu'un point d'intérêt est effectivement détecté et est rouge dans le cas contraire. Ainsi, on note que le point d'intérêt de la première ligne n'est pas très stable car il n'est vraiment détecté comme tel que lorsque la « caméra virtuelle » est située près de la couverture du livre. Il ne pourra donc être utilisé pour établir des correspondances pendant la

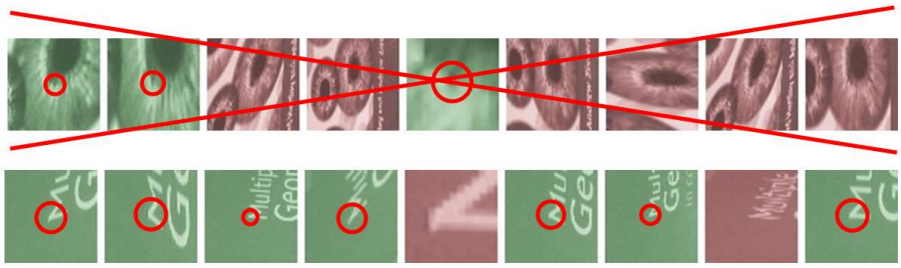


FIG. 7.4 – *Sélection des points d'intérêt les plus stables*


FIG. 7.5 – *Image de référence et points d'intérêt sélectionnés*

phase de détection que trop rarement. À l'inverse, le coin de la lettre « M » est un point de l'objet assez stable et il est sélectionné parmi les $N$ points d'intérêt de l'ensemble $\mathbf{K}$.

Les points sélectionnés par cette méthode sur la première image de la figure 7.1 sont identifiés par des points rouges sur la figure 7.5.

## 7.8 Principe des arbres de classification

Il existe plusieurs algorithmes pour implémenter le classifieur $\hat{Y}$ introduit dans la section 7.3. On peut par exemple citer l'algorithme des K plus proches voisins (*K-Nearest Neighbor*), les machines à vecteur de support (*Support Vector Machines* ou *SVM*), ou encore les réseaux de neurones (*neural networks*).

Lepetit *et al.* proposent plutôt d'utiliser des *arbres de classification aléatoires*, qui ont été présentés par Y. Amit et D. Geman en 1997 [1]. Ce choix tient notamment au fait que cette méthode de classification est très rapide

et donc adaptée à un programme temps-réel, tout en restant suffisamment robuste.

Rappelons la fonction de ces arbres : étant donné une nouvelle image (le plus souvent filmée par une caméra), la première étape consiste à en extraire les points d'intérêt avec la méthode présentée dans la section 7.5. Le voisinage de chaque point d'intérêt est alors considéré afin d'éventuellement apparier ce point avec un des points d'intérêt de l'image de référence. Plus précisément, on souhaite attribuer à ce voisinage une étiquette correspondant à une des classes de l'ensemble $\mathbf{C} = \{-1, 1, 2, \ldots, N\}$ (voir détails dans la section 7.3 page 77).

Pour ce faire, un patch $\mathbf{p}(\mathbf{k})$ (par exemple de taille $32 \times 32$ pixels), centré sur un point d'intérêt $\mathbf{k}$ de l'image courante, est lâché dans l'arbre, comme l'illustre la figure 7.6. À chaque nœud interne, un test élémentaire sur ce patch est utilisé pour décider si le patch doit continuer dans la branche gauche ou dans la branche droite. Lepetit *et al.* propose d'utiliser un test très simple : 2 pixels du patch sont comparés et selon que le premier est plus clair ou plus foncé que le second, le patch est envoyé d'un côté ou de l'autre.

La feuille $\eta$ atteinte par le patch $\mathbf{p}(\mathbf{k})$ contient une estimation des $N$ probabilités conditionnelles que le patch considéré appartienne à chacune des $N$ classes considérée, sachant qu'il a atteint cette feuille :

$$\mathbf{P}\big(\ \hat{Y}(\mathbf{p}(\mathbf{k})) = i \quad | \quad \mathbf{p}(\mathbf{k}) \text{ a atteint la feuille } \eta\ \big) \quad \text{où } i \in \{1, 2, \ldots, N\}$$

Afin de réduire le bruit et la variance des résultats, Lepetit *et al.* proposent d'utiliser $L$ arbres (généralement $L = 20$) : chaque patch est lâché dans chacun des arbres, et la distribution moyenne des $L$ feuilles atteintes est calculée. Le patch est alors classé en conservant la probabilité la plus élevée. Plus formellement, en notant $\mathbf{P}_l$ les probabilités de la feuille atteinte par le patch $\mathbf{p}$ dans l'arbre $l \in \{1 \ldots L\}$, l'étiquette du patch est attribuée ainsi :

$$\hat{Y}(\mathbf{p}) = \underset{c=1\ldots N}{\operatorname{argmax}}\, d^c(\mathbf{p}) = \underset{c=1\ldots N}{\operatorname{argmax}}\, \frac{1}{L} \sum_{l=1\ldots L} \mathbf{P}_l(\hat{Y}(\mathbf{p}) = c)$$

où $d^c(\mathbf{p})$ désigne la moyenne des probabilités de la classe $c$ dans les feuilles atteintes, et constitue une mesure de la confiance qui peut être attribuée à la classification. Ainsi, si $\max_c(d^c(\mathbf{p}))$ est trop faible, on considère que le patch ne correspond à aucun des $N$ points d'intérêt sélectionnés sur l'image de référence.

Finalement, si l'on note $h$ la hauteur (aussi appelée « profondeur ») moyenne des arbres de classification, la classification d'un patch ne requiert que $L \times h$ comparaisons entre intensités lumineuses et quelques additions pour le calcul de la distribution moyenne. Généralement, $L$ est choisi égal à 20 et $h$ vaut entre 10 et 30. De plus, aucune normalisation de l'intensité des pixels du patch n'est nécessaire car la classification dépend uniquement

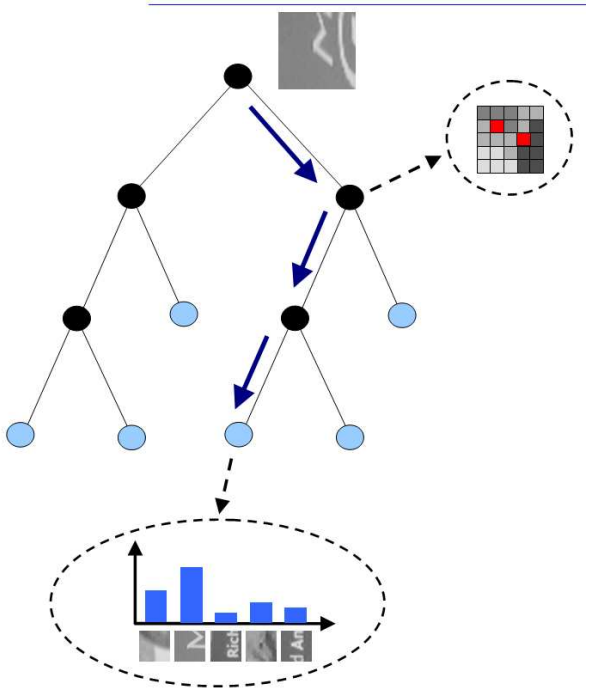


FIG. 7.6 – *Schéma illustrant le fonctionnant d'un arbre de classification*

de comparaisons binaires entre ces intensités. On gagne ainsi du temps par rapport à de nombreuses autres techniques qui requièrent une normalisation, par exemple en posant la norme $L_2$ des intensités égale à un. On comprend ainsi pourquoi la méthode proposée est si rapide.

## 7.9 Construction des arbres de classification

Nous abordons dans cette partie la problématique de la construction de ces arbres de classification, qui a lieu pendant la phase d'apprentissage, en utilisant une unique image de l'objet d'intérêt.

### 7.9.1 Méthode classique

Dans l'article original [13], Lepetit *et al.* propose de construire les arbres en suivant la méthode classique, de façon descendante.

Une base de données d'apprentissage est dans un premier temps construite, en synthétisant de nombreux patchs grâce à des transformations affines appliquées aux voisinages des points d'intérêt sélectionnés sur l'image de référence pendant la première étape de la phase d'apprentissage (voir section 7.7).

Tous ces patchs, qui sont étiquetés par le numéro de la classe à laquelle ils appartiennent (les auteurs proposent de synthétiser 100 patchs pour chacun des $N$ points d'intérêt sélectionnés), sont lâchés dans l'arbre qui ne contient initialement qu'un unique nœud. Le test associé à ce nœud est alors choisi par un algorithme glouton, de façon à « séparer au mieux » cet ensemble de patchs. Un bon critère pour évaluer l'efficacité de cette « partition » est le gain d'information : séparer un ensemble $S$ de patchs en plusieurs sous-ensembles $S_i$, grâce à un test donné, engendre le gain d'information suivant :

$$\Delta E = -\sum_i \frac{|S_i|}{|S|} E(S_i)$$

où :

- $E(s)$ désigne l'entropie $-\sum_{j=1}^{N} p_j log_2(p_j)$ avec $p_j$ la proportion de patchs de $s$ appartenant à la classe $j$.
- $|\cdot|$ représente le cardinal de l'ensemble.

En fait, utiliser un véritable algorithme glouton où tous les tests possibles seraient essayés afin de savoir lequel permettrait la meilleure partition des patchs serait relativement lent. On considère plutôt une batterie de $n$ tests où l'on choisit les 2 pixels à comparer aléatoirement, et l'on conserve celui qui engendre le gain d'information le plus élevé.

L'ensemble des patchs de la base de données initiale est ainsi divisé en deux sous-ensembles, selon que les patchs aient été dirigés dans la branche gauche ou dans la branche droite de l'arbre suite au test de ce premier nœud. De chaque côté, on répète alors la même procédure pour déterminer un test pseudo-optimal, en ne considérant que les patchs qui ont atteint le nœud étudié.

On continue ce processus de façon récursive, en décidant qu'un nœud est terminal si on a atteint une profondeur maximale donnée (par exemple 15), si le nombre de patchs atteignant ce nœud est trop faible (par exemple 10), ou si tous ces patchs appartiennent déjà à la même classe (auquel cas une partition supplémentaire serait évidemment superflue).

### 7.9.2 Méthode simpliste mais efficace

Dans l'article publié dans le journal PAMI [12], Lepetit *et al.* proposent une autre approche beaucoup plus simple : la structure de l'arbre est fixée (par exemple un arbre complet de profondeur 10), et les 2 pixels à comparer à chaque nœud interne sont choisis aléatoirement. Les expériences menées par les auteurs montrent que le classifieur obtenu est à peine moins efficace (les performances sont très comparables), alors que le gain en temps de calcul est évidemment considérable.

### 7.9.3 Estimation des probabilités des feuilles de l'arbre

Après avoir établi la structure de l'arbre et choisi les tests de chaque nœud grâce à une des deux méthodes présentées ci-dessus, il reste à estimer les distributions de probabilités des feuilles de l'arbre. Pour ce faire, on lâche dans l'arbre un nouvel ensemble de patchs synthétisés par transformations affines (environ 1000 patchs par point d'intérêt), et on calcule tout simplement la proportion des patchs de chaque classe qui sont parvenus dans chacune des feuilles.

Construire une vingtaine d'arbres indépendamment en suivant la méthode classique nécessite environ 15 minutes, alors que quelques secondes suffisent lorsque l'on choisit les tests de chaque nœud de façon aléatoire.

## 7.10 Détection et suivi

Dans les problèmes de réalité augmentée évoquée au début de ce rapport, on distingue généralement deux phases. Dans un premier temps, un *détecteur* a pour objectif de déterminer la position de la caméra par rapport à l'objet d'intérêt lorsque celui-ci passe dans le champ de vision de la caméra. Une fois l'objet détecté, on peut alors utiliser des algorithmes de *suivi* (*tracking* en anglais) qui utilisent la cohérence spatio-temporelle pour faciliter l'estimation de la position de l'objet. En effet, en supposant que la caméra a un mouvement fluide avec une vitesse modérée, on peut utiliser le fait que les points d'intérêt se sont peu déplacés entre deux images consécutives de la vidéo pour faciliter leur détection et leur mise en correspondance. Il arrive néanmoins que l'algorithme de suivi « perde » l'objet, par exemple si le mouvement de la caméra est trop brusque ou si l'objet part du champ de la caméra puis y revient. Dans ce cas, le détecteur est de nouveau mis à contribution pour tenter de « retrouver » la position de l'objet.

On comprend facilement que les caractéristiques d'un algorithme de détection et d'un algorithme de suivi ne sont pas les mêmes. Par exemple, Le premier se doit d'être particulièrement robuste aux conditions d'illumination et aux changements de points de vue, alors que le second connaît déjà l'apparence des points d'intérêt dans l'image précédente.

L'algorithme de *Lepetit et al.* considère le problème de la mise en correspondance de points d'intérêt entre deux images séparées par un angle de vue important (« *wide-baseline feature matching* » en anglais), qui correspond bien sûr plutôt à un problème de détection, car deux images successives d'une vidéo sont généralement très ressemblantes (On parle alors de « short-baseline » et de l'hypothèse des petits déplacements). De plus, il ne tire pas profit des informations du temps $t-1$ pour estimer la position de la caméra au temps $t$. Cependant, comme les performances de cette méthode sont relativement impressionnantes et qu'elle fonctionne en temps-réel, on peut aussi l'utiliser pour effectuer le suivi de l'objet et obtenir de très bons résultats.

# Chapitre 8

# Estimation de la position de la caméra

## 8.1 L'algorithme des trois points

Après cette étape de mise en correspondance, il convient de calculer la position relative de la caméra par rapport à l'objet d'intérêt, et dans le même temps de déterminer si les correspondances trouvées sont justes ou fausses.

Nous ne rentrerons pas trop en détail sur les techniques géométriques utilisées car, même si j'ai passé un temps non négligeable à lire des articles et à implémenter quelques méthodes, mon travail de recherche à proprement parler ne portait pas sur ce point.

Le problème posé est le suivant : comment calculer la position de la caméra par rapport à l'objet de référence à partir des correspondances établies entre des points d'intérêt sélectionnés de l'image de référence, et d'autres points détectés sur l'image courante ? On suppose connues les coordonnées 3D dans un repère donnée des points de l'objet correspondant aux $N$ points d'intérêt de l'image de référence qui ont été sélectionnés pendant la phase d'apprentissage. Ainsi, nous avons plus précisément des correspondances 3D-2D entre des points de la scène et des points de l'image courante. La figure 8.1 illustre le principe de cette étape de l'algorithme : le schéma de gauche montre les correspondances 3D-2D issues de la phase d'appariement, et sur le schéma de droite, on a réussi à estimer la position de la caméra et dans le même temps à distinguer les correspondances qui étaient en fait des erreurs.

Ceci est un problème classique de géométrie en vision par ordinateur, particulièrement dans la branche qui se préoccupe de problèmes où l'objectif est de reconstruire la structure d'une scène à partir du mouvement de la caméra (*Structure from Motion*). Différentes techniques permettent de trouver la position de la caméra à partir de trois correspondances (on parle d' « algorithmes des trois points »), par la résolution d'une équation po-

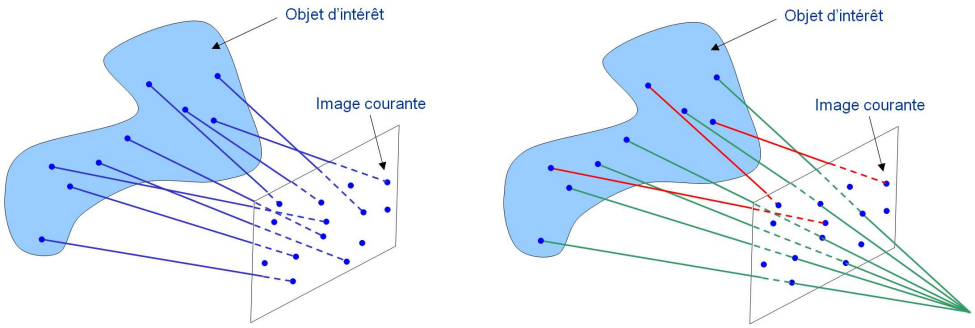



FIG. 8.1 – *Détection des fausses correspondances et estimation de la position de la caméra*

lynômiale. Celle-ci a été proposée dès 1841 par le mathématicien allemand Grunert [8]. Sa méthode et celles d'autres chercheurs ont été comparées dans un article de 1991 [9].

Notons que si l'on ne dispose pas des coordonnées 3D des $N$ points de l'objet correspondants aux points d'intérêt sélectionnés durant la phase d'apprentissage, il est aussi possible d'utiliser la *géométrie épipolaire* et la *matrice fondamentale* pour distinguer les correspondances justes et les erreurs. Cependant, je n'ai pas utilisé et implémenté ces méthodes car il est généralement assez facile de calculer une estimation précise de la position 3D de ces points à partir de correspondances, et le laboratoire disposait des outils nécessaires.

## 8.2 RANSAC

### 8.2.1 Principe

Malheureusement, toutes les correspondances trouvées pendant la phase d'appariement ne sont pas correctes. La méthode pour calculer la position de la caméra doit donc être robuste, en ce sens qu'elle doit pouvoir gérer les mauvaises correspondances.

En effet, utiliser trois correspondances choisies aléatoirement pour calculer la position de la caméra par rapport à l'objet n'est pas une solution envisageable. Si l'une de ces trois correspondances est fausse, le résultat sera complètement aberrant. Quand bien même les trois correspondances seraient justes, leur précision peut ne pas être suffisante. La méthode la plus souvent utilisée pour gérer ce genre de difficultés a été proposée par Fischler et Bolles en 1981 et est connue sous le nom de *RANSAC*, abréviation de *RANdom SAmple Consensus* [7].

Nous allons expliquer le principe de RANSAC grâce à un problème plus facile à représenter : l'estimation d'une droite à partir d'un ensemble de points, présentés sur la figure 8.2. On remarque que deux points semblent

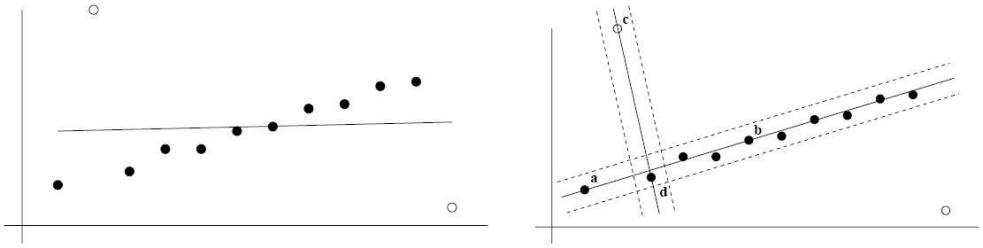



FIG. 8.2 – *Estimation d'une droite avec RANSAC*

ne pas appartenir à la droite, ce sont des données aberrantes, *outliers* en anglais, que l'on distingue des *inliers*. Dans notre cas, les *outliers* sont les faux appariements. La droite de la figure de gauche représente ce que l'on obtiendrait en utilisant la méthode classique des moindres carrés : la réponse est bien trop affectée par les points aberrants.

L'idée de RANSAC est très simple : pour estimer la droite correspondant à cet ensemble de données, deux points sont choisis aléatoirement. Ils définissent une droite dont le *support* est par définition le nombre de points qui se situent à sa proximité. Ce processus est répété plusieurs fois en choisissant aléatoirement d'autres paires de points (appelés « échantillons »), et la droite qui a le support contenant le plus de points est sélectionnée. Si un *outlier* est utilisé pour définir la droite, le support de celle-ci sera très restreint, comme l'illustre la droite $(\mathbf{c}, \mathbf{d})$ de la figure 8.2. La précision des points est aussi importante : le support de la droite $(\mathbf{a}, \mathbf{d})$ ne contient que 4 *inliers*, alors que celui de la droite $(\mathbf{a}, \mathbf{b})$ en contient 10. Ainsi, même si la droite $(\mathbf{a}, \mathbf{d})$ a aussi été construite à partir de deux *inliers*, c'est la droite $(\mathbf{a}, \mathbf{b})$ qui sera sélectionnée. On répète le processus jusqu'à avoir trouvé une estimation pour laquelle le pourcentage d'*inliers* est suffisant, ou après avoir testé un nombre donné d'échantillons.

Dans notre cas, ce sont trois appariements qui sont sélectionnés pour déterminer la position de la caméra par rapport à l'objet de référence. Les autres points de cet objet qui ont été mis en correspondance sont alors projetés sur l'image courante en utilisant cette estimation de la position de la caméra, et une correspondance est considérée comme correcte (*inlier*) si la distance entre cette projection et le point apparié est inférieure à un seuil donné.

Trois paramètres apparaissent importants dans l'implémentation de RANSAC :

- La distance-seuil qui détermine si une donnée est un *inlier* ou un *outlier* (elle est le plus souvent choisie de façon empirique)
- Le nombre $N$ d'échantillons testés avant de choisir celui qui a abouti à la meilleure estimation

– La taille de support $S$ considérée comme acceptable, qui est le deuxième critère de terminaison de l'algorithme

Lorsqu'on est capable d'estimer le pourcentage d'outliers $\epsilon$, les paramètres $N$ et $S$ peuvent être choisis de façon à ce que la probabilité $p$ qu'au moins un échantillon testé ne contiennent que des *inliers* soit de $0,99$ : si la taille de l'échantillon est $s$ ($s = 2$ dans le cas de l'estimation d'une droite, et $s = 3$ dans le cas de l'algorithme des trois points), alors la probabilité qu'un échantillon contienne au moins un *outlier* est $1 - (1 - \epsilon)^s$. Lorsqu'on tire au sort $N$ échantillons, la probabilité qu'au moins un ne contienne aucun *outlier* est donc $p = 1 - \left[1 - (1 - \epsilon)^s\right]^N$, d'où :

$$N = \frac{log(1 - p)}{log(1 - (1 - \epsilon)^s)} \tag{8.1}$$

où l'on choisit généralement $p = 0,99$.

### 8.2.2 RANSAC adaptatif

Dans notre cas, nous n'avons pas d'estimation $\epsilon$ du pourcentage d'outliers (expérimentalement, il peut varier entre 5% si la caméra est située dans une position proche de celle utilisée pour photographier l'image de référence, à 80% si l'angle de vue est très différent). Nous utilisons donc une variante de l'algorithme original, présentée dans le livre de Hartley et Zisserman, et appelée « RANSAC adaptatif » [11]. L'idée est d'initialiser $\epsilon$ avec la pire estimation du nombre de valeurs aberrantes ($\epsilon = 1$), puis de mettre à jour cette estimation au fur et à mesure des tests lorsque des supports de plus en plus grands sont détectés. Par exemple, si le premier échantillon aboutit à un support comprenant 60% des données, $\epsilon$ est mis à jour à $0,4$, et donc la quantité $N$ d'échantillons à tester diminue. L'algorithme se termine lorsque $N$ échantillons ont été testés. Il peut arriver qu'une mise-à-jour de $\epsilon$ conduise à une nouvelle valeur de $N$ plus petite que le nombre d'échantillons déjà testés, auquel cas l'algorithme se termine également.

### 8.2.3 Estimation finale

Une dernière étape consiste à affiner l'estimation de la position de la caméra en utilisant cette fois-ci tous les *inliers* présents dans le plus gros support obtenu. En effet, RANSAC est très utile pour séparer les *inliers* et les *outliers*, mais la position calculée est souvent trop grossière pour des applications en réalité augmentée. N'importe quel algorithme de minimisation non linéaire peut être utilisé pour cette phase, et nous avons choisi l'algorithme de Levenberg-Marquardt, présenté en 1944 par Levenberg et redécouvert par Marquardt en 1963.

### 8.2.4 Pseudo-code de la procédure d'estimation de la position de la caméra

Finalement, le pseudo-code que nous avons utilisé est le suivant :

- $N = \infty$, $nb\_echantillons = 0$
- Tant que $nb\_echantillons < N$
  - sélection aléatoire de 3 correspondances
  - estimation de la position de la caméra par l'algorithme des 3 points
  - calcul de $\epsilon$, la proportion d'outliers
  - Mise-à-jour de $N$ avec la formule 8.1, en prenant $p = 0,99$
  - incrémentation de $N$.
- Estimation optimale en utilisant l'algorithme de Levenberg-Marquardt à partir de tous les inliers issus de l'estimation précédente qui a conduit au support le plus important.

Avec cette méthode, il est apparu expérimentalement que nous arrivions à calculer la position de la caméra même si le pourcentage d'*outliers* était très important (jusque 80%).

# Chapitre 9

# Propositions d'améliorations de la méthode

## 9.1 Limites de la méthode de Lepetit *et al.*

L'approche proposée par Lepetit *et al.* est très prometteuse et donne des performances d'appariement bien meilleures que les méthodes existantes. Le détecteur/descripeur SIFT (dont la partie *descripteur* est décrite dans la section 9.1 page 59) est souvent considéré comme l'algorithme le plus efficace, mais ses performances sont assez largement inférieures à celles obtenues avec l'approche de Lepetit *et al.*, que ce soit en termes de nombre et de précision des correspondances ou du point de vue de la rapidité d'exécution. Rappelons toutefois que ces deux méthodes n'ont pas les mêmes objectifs et les mêmes applications, notamment car l'algorithme SIFT n'a pas besoin de phase d'apprentissage et permet de mettre en correspondance des images quelconques.

Néanmoins, la méthode des arbres de classification présente quand même quelques limitations. Les figures 9.1 et 9.2 mettent en effet en évidence des situations où l'appariement des points d'intérêt n'est pas satisfaisant et ne suffit pas pour pouvoir obtenir une estimation correcte de la position de l'objet. Les correspondances fausses figurent en rouge alors que les traits verts symbolisent des appariements corrects. En fait, ces situations sont des situations « limites » car si la caméra descend encore un peu dans la figure 9.1 ou si la lumière est encore plus forte dans la figure 9.2, le nombre de correspondances correctes deviendra vraiment insuffisant pour pouvoir estimer la position de la caméra et dans le même temps distinguer les *inliers* des *outliers*.

La figure 9.1 met particulièrement en évidence la sensibilité de l'algorithme aux changements d'angle de vue. Quand la position de la caméra par rapport à l'objet est trop différente de celle utilisée pour prendre en photo l'image de référence qui sert à construire les arbres, les performances

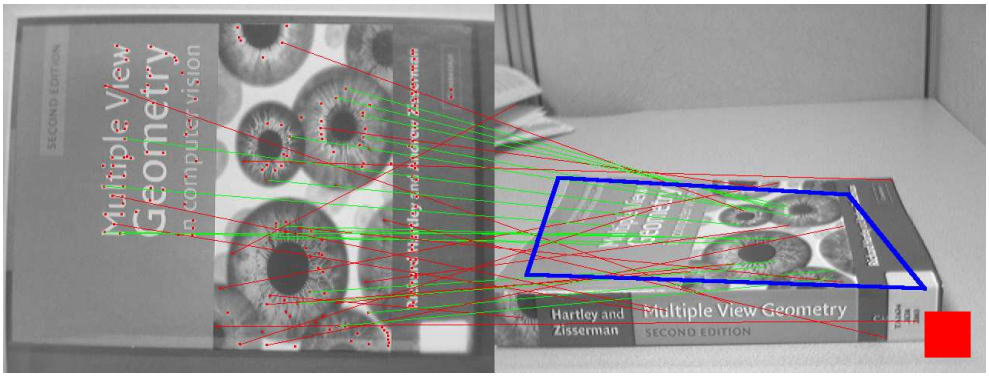



FIG. 9.1 – *Mauvais résultats dus à un angle de vue trop différent par rapport à l'image de référence*

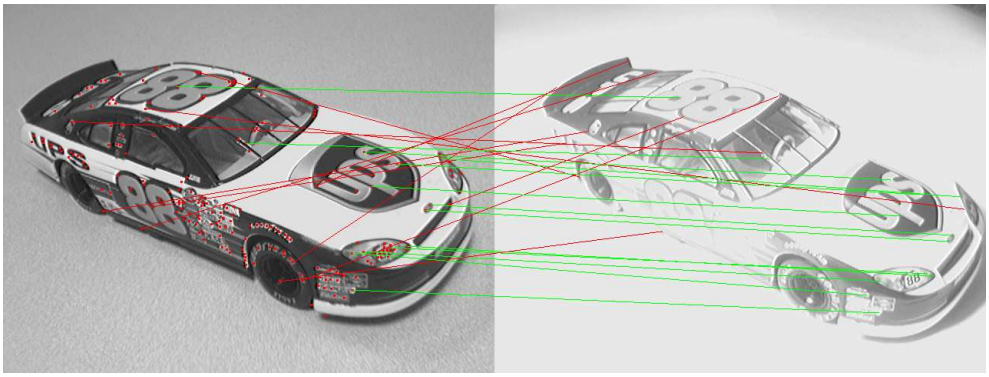


FIG. 9.2 – *Sensibilité de l'algorithme aux conditions d'illumination*

peuvent être considérablement affectées.

La figure 9.2 souligne quant à elle l'importance des conditions d'illumination. Nous avons expliqué dans la section 7.4.2 page 46 que la classification avec les arbres était invariante aux changements monotones d'intensité lumineuse. Cependant, d'autres variations par rapport à l'image de référence peuvent avoir lieu, comme des onbres, de la réflexion spéculaire ou des effets de saturation, et la méthode est alors mise en défaut.

## 9.2 Utilisation d'arbres de classification adaptatifs

### 9.2.1 Mise à jour des arbres en temps réel

L'idée de cette amélioration est en quelque sorte de poursuivre la phase d'apprentissage en analysant la phase d'appariement en temps réel. Plus précisément, nous proposons d'améliorer les arbres de classification en prenant en compte les correspondances qui ont été établies en temps réel avec

les images en provenance de la caméra.

Nous distinguons deux types de mise à jour.

### Renforcement des *inliers*

Nous avons expliqué dans la section 8 page 54, que des méthodes géométriques indépendantes couplées à l'algorithme RANSAC étaient utilisées pour calculer la position de la caméra par rapport à l'objet d'intérêt, et pour déterminer si les correspondances trouvées étaient jutes (on parle d'*inliers*) ou fausses (*outliers*).

Dans un premier temps, nous essayons de renforcer les inliers, afin qu'ils deviennent de plus en plus stables dans le futur. Rappelons que dans notre cas, un inlier est en fait une correspondance 3D-2D entre un point de l'objet et un point d'intérêt de l'image courante. Cette correspondance a été établie en lâchant dans les arbres de classification un patch centré autour de ce point d'intérêt.

Nous examinons alors les feuilles que ce patch a atteint (une feuille par arbre). Nous savons qu'en calculant la moyenne des distributions de ces feuilles, la classe dont la probabilité est la plus élevée est associée au point de l'objet correspondant au point d'intérêt de l'image. Nous proposons simplement d'augmenter la probabilité de cette classe dans chacune des feuilles et de diminuer la probabilité des autres classes, comme illustré sur la figure 9.3. Afin de pouvoir faire ce genre d'opérations, notons que les probabilités conditionnelles d'une feuille sont stockées sous la forme de fréquences, c'est-à-dire par l'intermédiaire de la quantité $K$ de patchs qui ont atteint la feuille, et des quantités $K_i$ correspondant au nombre de patchs qui appartenaient à la classe $i$ (les probabilités sont alors naturellement les valeurs $\frac{K_i}{K}$). La mise à jour des feuilles correspond simplement à l'incrémentation des entiers $K$ et $K_i$.

En fait, on ne fait ici que continuer la phase d'apprentissage, mais au lieu d'utiliser des images synthétiques générées à partir d'une unique image grâce à des transformations affines, nous utilisons des images réelles.

### Recouvrement des *outliers*

Nous nous proposons aussi d'améliorer les arbres en regardant pourquoi un point d'intérêt a été incorrectement apparié, et en essayant de modifier les arbres pour diminuer la probabilité qu'une telle erreur se reproduise à l'avenir. Notre but est donc de recouvrer les correspondances fausses et d'apparier les points de l'objet dont la mise en correspondance a échoué, avec les point de l'image courantes qui leur correspondent.

Comme nous connaissons la position de la caméra par rapport à l'objet, nous sommes capable de projeter les points 3D de cet objet sur l'image courante. Nous effectuons cette projection pour les points de l'objet qui n'ont

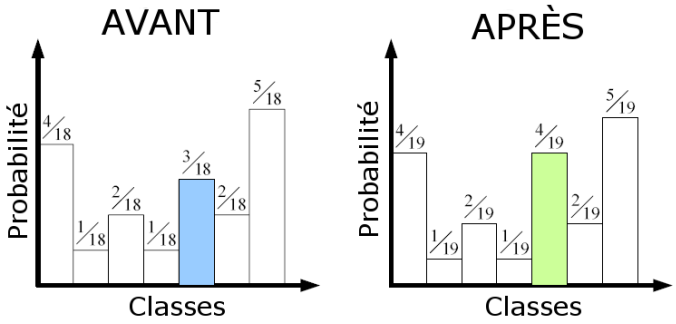



FIG. 9.3 – *Mise à jour des probabilités d'une feuille de l'arbre*

pas été appariés ou dont l'appariement s'est révélé faux selon l'algorithme RANSAC.

Deux cas peuvent alors se produire lorsque l'on projette un point de l'objet sur l'image courante, illustrés par les deux schémas de la figure 9.4.

1. On considère la fausse correspondance (en rouge) du schéma de gauche. On projette le point 3D qui a été incorrectement apparié (trait vert) et on examine un petit voisinage autour de la projection. On constate ici qu'aucun point de l'image courante n'a été détecté en tant que point d'intérêt dans ce voisinage. Ceci signifie que les arbres n'ont pas été mis en défaut, mais que le problème vient simplement du fait que le point de l'objet, dans ces conditions spécifiques d'angle de vue et d'illumination, n'apparaît pas comme un point saillant sur l'image courante. Il est dans ce cas inutile de mettre à jour les arbres de classification, car ce point de l'objet ne sera de toute façon jamais détecté par le détecteur de point d'intérêt dans le futur si on se retrouve dans les mêmes conditions.
2. Dans le schéma de droite, on considère une autre fausse correspondance (trait rouge d'en haut). On projette une fois de plus le point de l'objet sur l'image courante (trait vert), mais on constate cette fois-ci qu'un point d'intérêt a été détecté dans le voisinage de la projection mais qu'il a été incorrectement apparié (trait rouge d'en bas) à un autre point de l'objet. On se propose alors de modifier les arbres de décision pour que la classification de ce point d'intérêt ait plus de chances de réussir dans le futur. Pour ce faire, on identifie une nouvelle fois les feuilles des arbres que le patch autour du point a atteint, et pour chacune d'elle, on augmente la probabilité de la classe qu'on sait maintenant être correcte.

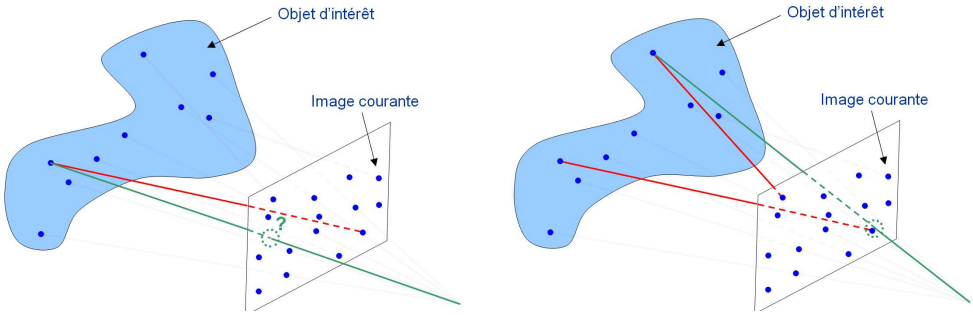



FIG. 9.4 – *Projection des points 3D mal appariés sur l'image courante*

### 9.2.2 Comment ces mises à jour permettent-elles d'améliorer les performances ?

Dans cette partie, nous allons voir pourquoi et comment ces mises à jour des probabilités des feuilles permettent de contourner les difficultés de la méthode originale de Lepetit *et al.*

#### Correction des imperfections géométriques

Nous avons vu dans la section 7.4 page 45, qu'une seule image de l'objet d'intérêt est utilisée dans la phase d'apprentissage pour construire les arbres de classification.

La base de données d'apprentissage est alors créée en synthétisant de nouvelles images grâce à des transformations affines appliquées à l'image initiale. Nous avons expliqué que ceci permettait d'obtenir l'apparence que prendraient les voisinages des points d'intérêt s'ils étaient vu depuis différents points de vue. Ceci suppose néanmoins, comme nous l'avons précisé, que ces voisinages puissent être considérés comme localement plans. Notons qu'on ne fait pas l'hypothèse que l'objet en lui-même soit plan, mais uniquement que les voisinages des points d'intérêt le soient.

Cependant, les voisinages des points considérés sont assez grands (généralement $32 \times 32$ pixels), et pour un objet 3D complexe, les points d'intérêt correspondent souvent à des structures 3D locales. Il est dans ce cas impossible de retrouver l'apparence de ces structures sous différents angles de vue en utilisant uniquement des transformations 2D. Les arbres sont donc construits en utilisant des informations erronées.

Lepetit et son équipe évoquent la possibilité d'utiliser des modèles 3D denses de l'objet, mais de tels modèles peuvent ne pas être disponibles et être compliqués à construire.

En utilisant notre approche et en mettant à jour les arbres de classification grâce aux images réelles de l'objet, nous avons accès à des connaissances

sur l'apparences que prennent les voisinages des $N$ points d'intérêt sélectionnés pendant la phase d'apprentissage, lorsqu'ils sont vus sous différents angle de vue. Ces informations n'étaient pas disponible lorsque nous avons initialement construit les arbres, mais nous en tirons désormais profit pour les mettre à jour. Ainsi, si l'on met à jour les arbres en utilisant un nombre suffisant d'observations de l'apparence réelle des points d'intérêt, l'erreur de modélisation initiale pourra être corrigée.

En conséquence, même pour un objet 3D complexe, utiliser des arbres adaptatifs nous permet de continuer à n'utiliser qu'une seule image de l'objet d'intérêt pour construire les arbres. Nous avons conduit avec succès nos différentes expériences en suivant cette approche. Nous évitons ainsi d'avoir à construire un modèle 3D dense de l'objet mais nous avons toujours besoin de connaître les coordonnées 3D des $N$ points de l'objet correspondant aux points d'intérêt sélectionnés dans la phase d'apprentissage. Cependant, comme nous l'avons expliqué dans la section 8.1 page 54, on peut même se passer d'avoir ces coordonnées 3D en entrée, si :

- Supposer l'objet plan permet quand même d'obtenir un nombre suffisant de correspondances correctes à partir desquelles les coordonnées 3D de la plupart des points d'intérêt peuvent être estimées grâce à des algorithmes classiques de *Structure From Motion*
- La géométrie épipolaire seule permet de séparer les *inliers* des *outliers*

**Compensation d'autres imperfections du modèle**

De façon similaires, d'autres erreurs de mises en correspondances dues aux imperfection du modèle peuvent être compensées en utilisant les arbres adaptatifs. Les images utilisées pendant la phase d'apprentissage doivent en effet être aussi ressemblantes que possible à celles qui sont filmées par la caméra pendant la phase d'exécution. Cependant, les images synthétiques ne peuvent être parfaites pour de multiples raisons.

Par exemple, la résolution de l'image initiale est évidemment limitée. Or, pendant la phase en temps réel, on peut imaginer que la caméra puisse s'approcher très près de l'objet d'intérêt. De nouvelles informations vont alors apparaître qui n'étaient pas visibles pendant la phase d'apprentissage et qui donc ne sont pas prises en compte dans les arbres de classification.

Les conditions d'illumination sont aussi un problème important pour la méthode originale. Si de nouvelles ombres ou de la réflexion spéculaire perturbent l'apparence de l'objet, nous avons vu que les résultats peuvent être très mauvais car ces nouvelles apparences des patchs n'ont pas été envisagées pendant la phase d'apprentissage où toutes les images de la base de données proviennent d'une unique image de l'objet, et donc ne modélisent que des conditions d'illuminations très particulières.

Cependant, si l'on utilise suffisamment d'observations de l'apparence de l'objet, les arbres vont finir par modéliser plus précisément son apparence dans différentes conditions géométriques et photométriques. Ainsi, nous sommes finalement capable d'augmenter significativement la zone de couverture des arbres, et ceux-ci ont des résultats satisfaisants dans des gammes de conditions d'illumination bien plus grandes.

### 9.2.3 Discussions

#### Une condition nécessaire

Bien évidemment, la méthode des arbres adaptatifs ne peut permettre d'améliorer les performances que si l'estimation de la position de la caméra a réussie. En effet, nous projetons les points 3D sur l'image courante donc si la position estimée de la caméra n'est pas correcte, nous allons au contraire introduire des fausses informations dans les arbres. Nous proposons tout simplement de ne mettre à jour les arbres que si l'objet est détecté avec suffisamment d'assurance, par exemple si le nombre d'*inliers* dépasse un seuil donné.

#### Le problème de l'occlusion

La deuxième étape de la mise à jour des arbres, à savoir la tentative de recouvrement des *outliers* peut aussi avoir un effet négatif si on est en présence d'un phénomène d'occlusion. En effet, lorsqu'on projette un point 3D $P$ de l'objet qui a été mal apparié, si un point d'intérêt $p$ a été détecté dans le voisinage de la projection, on va modifier les arbres pour que ce point d'intérêt $p$ ait tendance à être apparié avec $P$ dans les futurs images. Cependant, dans le cas d'un phénomène d'occlusion, $p$ peut ne pas être la projection du point $P$ mais celle d'un point $Q$ qui bloque la visibilité du point $P$ car situé sur le segment $[p\ P]$. Notons que pour qu'une telle situation se produise, il ne suffit pas qu'un objet non modélisé soit situé entre la caméra et l'objet d'intérêt : il faut aussi que la projection du point visible de cet objet situé sur le segment $[p\ P]$ soit détecté comme un point d'intérêt.

Expérimentalement, de telles situations accidentelles sont relativement rares, et si la caméra ou les objets en causes se déplacent, les éventuelles mises à jour erronées n'auront généralement pas lieu à plusieurs reprises dans la même feuille et pour la même classe.

De plus, si la vraie correspondances est de nouveau visible, l'arbre se ré-adaptera dans le bon sens.

#### Le problème du surapprentissage

Un risque avec cette approche concerne le surapprentissage (*overfitting* en anglais). Si nous mettons à jour les arbres alors que la caméra reste dans la

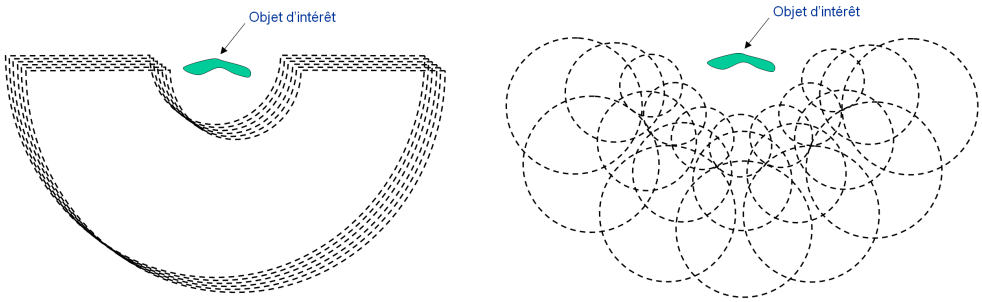



FIG. 9.5 – *Distribution spatiale des arbres de classification*

même zone pendant une longue période, ils vont finir par être excessivement adaptés à cette position de la caméra, et les performances pourraient chuter si celle-ci se déplaçait rapidement vers une autre région. Cependant, notre amélioration suivante permet d'éviter ce problème.

## 9.3 Distribution spatiale des arbres de classification

### 9.3.1 Spécialisation des arbres

L'apparence d'un point d'intérêt peut évidemment varier énormément selon le point de vue. Dans la méthode originale de Lepetit *et al.*, chaque arbre de classification doit modéliser l'apparence « globale » des points d'intérêt.

Le schéma de gauche de la figure 9.5 illustre ce principe : les régions autour de l'objet d'intérêt symbolisent les zones de couverture des arbres. Elles sont les mêmes pour tous les arbres de classification et dépendent en fait uniquement des intervalles dans lesquelles sont choisis les paramètres $\theta$, $\phi$, $\lambda_1$ et $\lambda_2$ des transformations affines qui sont utilisées pour synthétiser les images de la base de données d'apprentissage (voir section 7.4.1 page 45). Ainsi, si la caméra est située dans ces zones de couverture, les arbres devraient être capables de détecter l'objet d'intérêt. Il est évident qu'il existe donc un compromis entre le taille de ces zones et les performances des arbres, car la variabilité intra-classe augmente avec le volume de la zone de couverture.

Nous proposons de spécialiser les arbres pour que chacun d'entre eux modélise plus spécifiquement l'apparence de l'objet vu depuis un certain volume de l'espace. Chaque arbre est donc entraîné pour pouvoir détecter et suivre l'objet lorsque la caméra se trouve dans cette région. Le schéma de droite de la figure 9.5 illustre la disposition des arbres, les cercles correspondant aux zones de couverture.

Une grille de positions régulière (uniforme dans la direction angulaire,

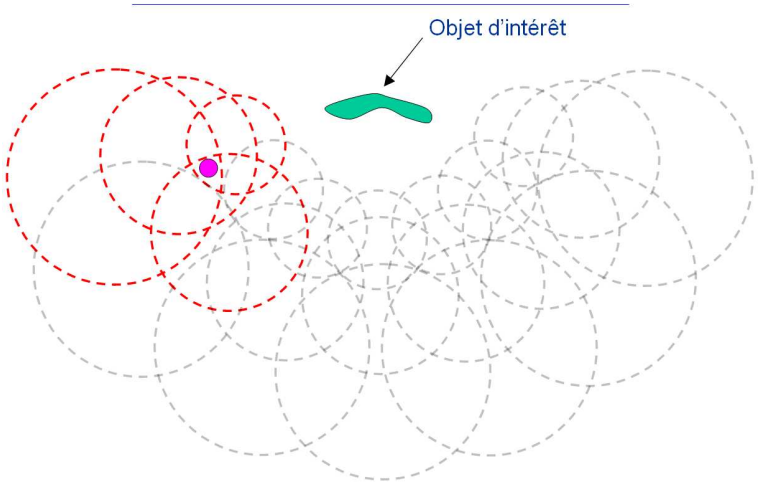



FIG. 9.6 – *Sélection des arbres utilisés pour la phase d'appariement, et éventuellement mis à jour*

et logarithmique dans la direction radiale) est définie autour de l'objet d'intérêt. Une position de la grille, combiné avec un rayon définit une zone de couverture sphérique. Ces régions se chevauchent et les zones de recouvrement sont en fait bien plus importantes que sur le schéma.

Comme l'illustre la figure 9.6, on sélectionne, pendant la phase d'exécution, les arbres qui couvrent la position de la caméra calculée pour la frame précédente de la vidéo (rond violet sur le schéma). On suppose ainsi que le mouvement de la caméra entre deux images consécutives est petit comparé à la taille des zones de couverture des arbres. Ces arbres sélectionnés (en rouge sur le schéma) sont utilisés pour trouver les correspondances et sont éventuellement mis à jour. Lors du processus de classification, la moyenne des distributions des feuilles atteintes est maintenant pondérée en utilisant une fonction décroissante de la distance entre la position de la caméra et le centre des zones de couverture sphériques.

### 9.3.2 Avantages

Comme les arbres sont plus spécialisés par rapport à la méthode originale de Lepetit *et al.*, les variances intra-classes sont plus faibles et les arbres sont donc plus efficaces. Ceci permet d'utiliser moins d'arbres, et des arbres plus petits, tout en obtenant les mêmes performances de mise en correspondance. Le temps d'exécution peut ainsi être réduit.

De plus, utilisée en combinaison avec les arbres adaptatifs, cette approche permet de ne mettre à jour qu'un ensemble local d'arbres, et ainsi d'éviter le surapprentissage des arbres à une échelle globale.

Enfin, un dernier avantage concerne la sélection des points d'intérêt qui

est faite pendant la phase d'apprentissage (section 7.7 page 48). Dans la méthode initiale, on cherche des points qui sont globalement stables. De tels points peuvent être assez rares et volatiles. Utiliser des arbres répartis dans l'espace permet de sélectionner différents points d'intérêt selon la position de la caméra. En effet, certains points peuvent être particulièrement stables s'ils sont vus depuis une région limitée de l'espace, mais non-détectables ailleurs. De tels points peuvent être retenus dans le cadre d'une approche où les arbres sont répartis dans l'espace, alors qu'ils seraient certainement ignorés par l'algorithme proposé par Lepetit *et al.*. Pendant la phase d'exécution, on utilise donc uniquement les points d'intérêt que l'on sait être stables et particulièrement saillants lorsqu'ils sont observés depuis la zone où se situe la caméra.

### 9.3.3 Limites de cette approche

#### Nécessité d'un modèle 3D

Pour générer les images de la base de données d'apprentissage, nous avons besoin de connaître l'apparence de l'objet depuis une position 3D donnée. Nous avons pour cela besoin de créer un modèle texturé de l'objet d'intérêt. La nécessité d'un tel modèle est un défaut par rapport à l'approche considérée jusqu'ici. Néanmoins, si un tel modèle n'est pas disponible, les deux améliorations proposées étant relativement indépendantes, on peut n'utiliser que les arbres adaptatifs, sans répartir ces derniers dans l'espace.

#### Une méthode de *tracking*

Cette approche correspond plutôt à une approche de type *suivi* que de type *détection*, car on a besoin de la position de la caméra estimée dans l'image précédente pour choisir les arbres qui seront utilisés pour l'image courante. Si l'objet n'a pas été détecté dans la *frame* précédente, nous proposons d'utiliser tous les arbres de classification. Les performances sont alors similaires à celles de la méthode de Lepetit *et al.*, mais le temps de calcul est un peu plus important car plus d'arbres sont utilisés. Nous pourrions aussi décider de conserver, en plus de ces arbres répartis dans l'espace, d'autres arbres définis comme dans la méthode originale.

## 9.4 Détecteur FAST

La méthode utilisée pour détecter les points d'intérêt est décrite dans la section 7.5 page 46. Une fois les points détectés, une orientation leur ai attribué en suivant la méthode de SIFT, comme expliqué dans la section 7.6.

J'ai testé plusieurs autres détecteurs et je propose finalement d'utiliser le détecteur FAST (*Features from Accelerated Segment Test*), proposé en 2005

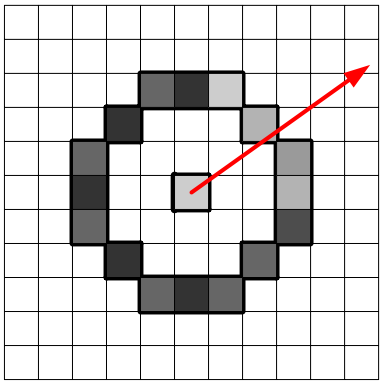


FIG. 9.7 – *Principe de fonctionnement du détecteur* FAST

par E. Rosten et T. Drummond [24], et que j'ai amélioré de façon très simple pour qu'il fournisse aussi une orientation à chaque point détecté.

Comme nous l'avons déjà précisé, le détecteur doit être particulièrement rapide, et c'est bien sûr le cas du détecteur FAST, comme son nom l'indique...Comme dans le cas de la méthode proposée par Lepetit *et al.*, on parcourt l'image en considérant les intensités des pixels situés sur des cercles, cette fois de 16 pixels (plus précisément les cercle dits « de Bresenham »), centrés en chaque point (voir schéma 9.7). Le centre **m** du cercle est considéré comme un point d'intérêt si les intensités de 12 points consécutifs de ce cercle sont toutes significativement supérieures ou toutes significativement inférieures à l'intensité du pixel **m** (*significativement* signifiant ici que la différence d'intensité doit être supérieure a un seuil fixé). On peut optimiser la recherche de tels points en n'examinant dans un premier temps que les pixels 1, 9, 5 et 13 qui sont situés sur les 4 « côtés ». En effet, **m** ne peut être un point d'intérêt que si 3 de ces 4 pixels remplissent la condition ci-dessus. Avec cette optimisation, il a été montré que sur une vidéo donnée, uniquement $3,8$ pixels du cercle sont testés en moyenne pour décider si un point est un point d'intérêt.

Si 12 pixels au moins ont une intensité relativement différente de celle du centre, 4 ou moins ont une intensité plus proche. On comprend que les points d'intérêt correspondent à des sortes de « pointes ». Je propose d'utiliser le centre de cette « pointe » pour définir l'orientation du point d'intérêt. Comme le cercle de Bresenham est défini par 16 pixels et que le nombre de pixels dont l'intensité est significativement différente de celle du pixel du centre peut être pair ou impair, $16 \times 2 = 32$ orientations différentes peuvent être attribuées à un point d'intérêt.

En pratique, le détecteur obtenu est plus rapide et plus stable que celui proposé par Lepetit *et al.*. L'attribution de l'orientation est quasi immédiate

et on gagne donc beaucoup de temps par rapport à la méthode de SIFT, où il faut constituer l'histogramme des directions des gradients autour des points d'intérêt. L'orientation affectée est légèrement moins stable mais la variance intra-classe des arbres est de toute façon surtout liée aux variations d'apparences dus aux changements d'angle de vue.

Finalement, la méthode proposée dans cette section de détection de points d'intérêt et d'attribution d'orientation donne de meilleurs résultats et ce beaucoup plus rapidement.

## 9.5 Résultats expérimentaux

### 9.5.1 Préliminaires

Dans cette section, nous montrons à quel point les différentes approches que nous avons proposées permettent d'améliorer les performances par rapport à la méthode initiale proposée par Lepetit *et al.* [13]. Les vidéos sont filmées en utilisant une simple web-cam classique (*Unibrain Fire-I*).

Les gains de performances des deux principales améliorations (arbres adaptatifs et distribution spatiale des arbres) sont évalués séparément. En effet, nous avons expliqué, dans la section 9.3.3 page 68, qu'un modèle 3D de l'objet est nécessaire pour pouvoir répartir les arbres dans l'espace, alors qu'une unique photo de l'objet est utilisée pour la méthode des arbres adaptatifs.

Ainsi, afin de pouvoir établir des comparaisons justes et équitables, un modèle 3D texturé de l'objet est utilisé à la fois pour la méthode originale et pour notre approche lorsque nous souhaitons observer à quel point la distribution spatiale des arbres permet d'améliorer les résultats, alors qu'une unique image de l'objet est utilisée pour les deux méthodes lorsque nous souhaitons évaluer les performances des arbres adaptatifs.

### 9.5.2 Efficacité des arbres adaptatifs

Dans cette partie, nous utilisons uniquement la mise à jour des feuilles des arbres, mais nous ne les répartissons pas dans l'espace. Une forêt de 20 arbres est utilisée, et 200 points d'intérêt sont sélectionnés pendant la phase d'apprentissage.

Dans un premier temps, nous pouvons simplement observer visuellement les gains de performances par rapport aux deux images 9.1 et 9.2 de la section 9.1 page 59 qui évoquent les limites de la méthode originale. En utilisant notre approche, nous obtenons les résultats présentés sur les figures 9.8 et 9.9. Il est clair que le nombre de correspondances correctes est bien plus élevé et donc que la position de la caméra par rapport à l'objet peut être estimé plus facilement et plus précisément.

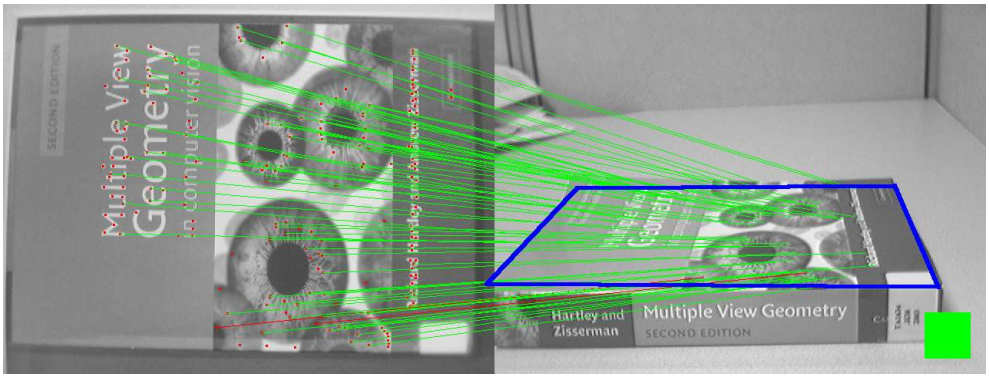



FIG. 9.8 – *Performances malgré un angle de vue difficile*

FIG. 9.9 – *Performances malgré des conditions d'illuminations particulières*

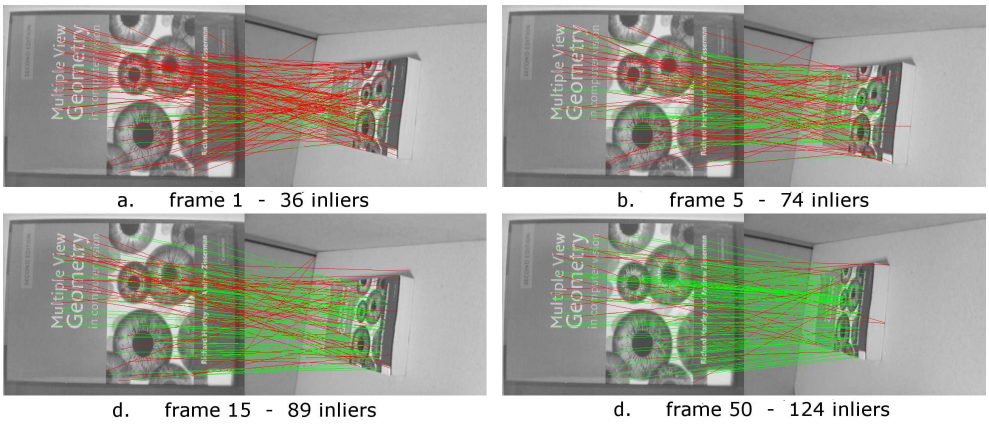



FIG. 9.10 – *Évolution du nombre d'inliers en adaptant les arbres en temps réel*

### Évolution du nombre d'*inliers*

La figure 9.10 illustre l'efficacité de la méthode des arbres adaptatifs sur une séquence vidéo. Au début de la vidéo, les arbres n'ont pas encore été mis à jour et sont donc ceux construits par la méthode originale. On remarque que la couverture du livre est alors très difficilement détectée, car la plupart des correspondances sont fausses. En démarrant le processus de mise à jour des arbres dès la première frame, on note que le nombre de correspondances correctes (ou *inliers*) augmente très rapidement puisqu'il est passe de 36 à 124 en l'espace de 50 frames, soit 2 secondes de vidéo. Il est a noté que la caméra n'est pas immobile pendant ces 2 secondes

### Réponse aux changements brusques des conditions d'illumination

Des vidéos contenant une petite voiture devant un arrière-plan encombré ont été utilisées pour les deux expériences suivantes. La première consiste à allumer et à éteindre une forte lumière qui provoque de la saturation et de la réflexion sur la petite voiture. La figure 9.11 montre l'évolution du pourcentage d'*inliers* en fonction du temps.

Au début de la vidéo, les conditions d'illumination sont « normales ». On remarque qu'en adaptant les arbres de classification, les performances s'améliorent dès la première image pour atteindre un taux d'*inliers* proches de 90%. Au niveau de la frame 150, la forte lumière est allumée. Les deux méthodes sont très affectées par ce changement d'illumination et le pourcentage d'*inliers* tombe aux environs de 30 %. Cependant, en adaptant les arbres en temps réel à ces nouvelles conditions d'illumination, les performances remontent en l'espace de quelques secondes. Un exemple d'appariements aux environs de la frame 300 est présenté sur la figure 9.12 : de nombreuses correspondances sont trouvées, bien que les conditions lumineuses entre l'image

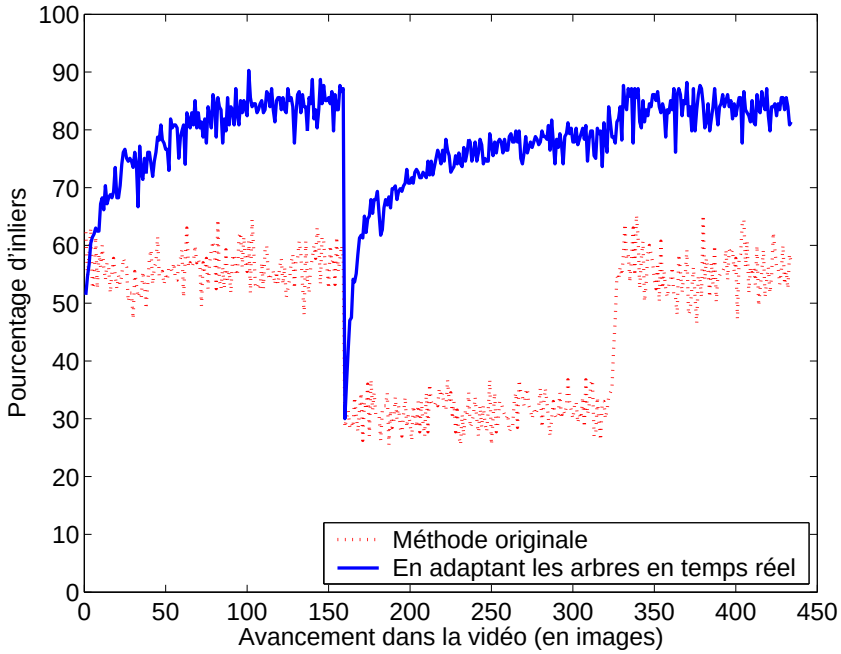



FIG. 9.11 – *Adaptation aux changements brusques des conditions d'illumination*

de référence et l'image courante soient très différentes.

Lorsqu'on éteint cette forte lumière, on remarque aussi que les performances ne diminuent pas et le pourcentage d'*inliers* retrouve directement le niveau qu'il avait juste avant cette perturbation. Ceci signifie donc qu'il n'y a pas eu de problème de surapprentissage (*overfitting*), les arbres ne sont pas « sur-adaptés » à une lumière particulière.

Des résultats similaires peuvent être observés dans le cas d'autres variations d'illumination, comme les ombres par exemple.

### Deux modes de fonctionnement différents

On peut utiliser le principe des arbres adaptatifs de deux façons différentes. Après la construction initiale des arbres décrites dans l'article de Lepetit *et al.*, on peut les améliorer soit *offline*, soit *online*.

La première possibilité revient en fait à continuer la phase d'apprentissage en utilisant des vidéos de l'objet d'intérêt pour améliorer la performance des arbres. Par exemple, on peut filmer l'objet dans différentes conditions d'illumination et d'angles de vue, et utiliser ces séquences vidéos pour mettre à jour les arbres afin qu'ils fonctionnent mieux pendant la véritable phase d'exécution.

L'adaptation *online* réfère quant à elle à la mise à jour « classique » des arbres en utilisant la vidéo en temps réel.

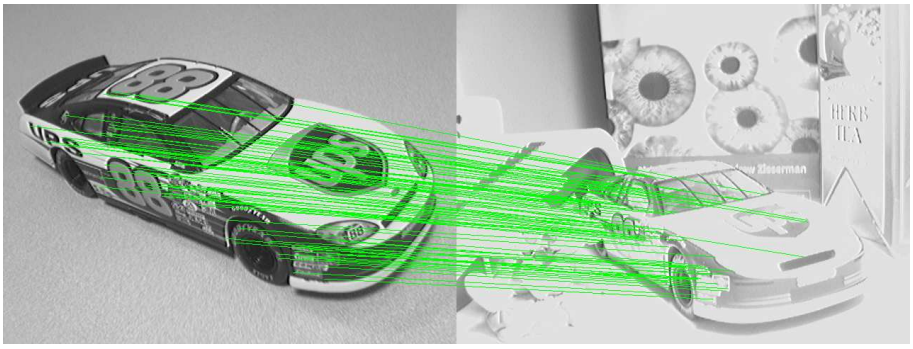


FIG. 9.12 – *Appariement dans des conditions d'illumination extrêmes*

Il est bien sûr évidemment possible de coupler ces deux modes de fonctionnement.

La figure 9.13 permet de comprendre l'influence de ces deux façons d'utiliser la méthode adaptative. Les courbes représentent le pourcentage de correspondances justes (*inliers*) en fonction de la distance entre la caméra et l'objet de référence, la distance unité étant celle correspondant à l'image de référence. Ainsi, par exemple, les performances diminuent lorsque la caméra s'éloigne trop de l'objet d'intérêt.

La courbe du bas est obtenue avec la méthode originale. Les autres courbes correspondent aux performances des arbres lorsque ceux-ci sont mis à jour en utilisant la méthode décrite dans la partie 9.2. Pour l'adaptation *offline*, les arbres sont améliorés grâce à une séquence de 350 frames (14 secondes), alors que la séquence *online* compte 440 frames. Ainsi, on note que les performances augmentent significativement même lorsque l'on n'utilise qu'une très courte vidéo *offline* et que l'on arrête la mise à jour des arbres par la suite. Ceci montre clairement l'efficacité de la méthode adaptative.

### 9.5.3 Utilité de la répartition dans l'espace des arbres de classification

L'intérêt de la répartition des arbres de classification dans l'espace est illustré par les courbes de la figure 9.14. Ce graphique montre le compromis que nous avons évoqué dans la section 9.3.1 page 66 entre le volume des zones de couverture des arbres et leurs performances. La méthode de Lepetit *et al.* est utilisée pour construire plusieurs forêts en utilisant différents intervalles pour les paramètres d'échelle $\lambda_1$ et $\lambda_2$ (introduis dans la section 7.4.1 page 45). Dans chaque cas, une forêt comporte 20 arbres. Par exemple, lorsque les intervalles choisis sont $[0,3 \quad 2,5]$ ou $[2,0 \quad 5,0]$, il n'est pas surprenant de voir que le pourcentage d'*inliers* est élevé lorsque la caméra se trouve à l'intérieur de ces zones, mais diminue fortement dès qu'elle s'en éloigne. Si l'on essaie alors de couvrir une zone plus importante, comme

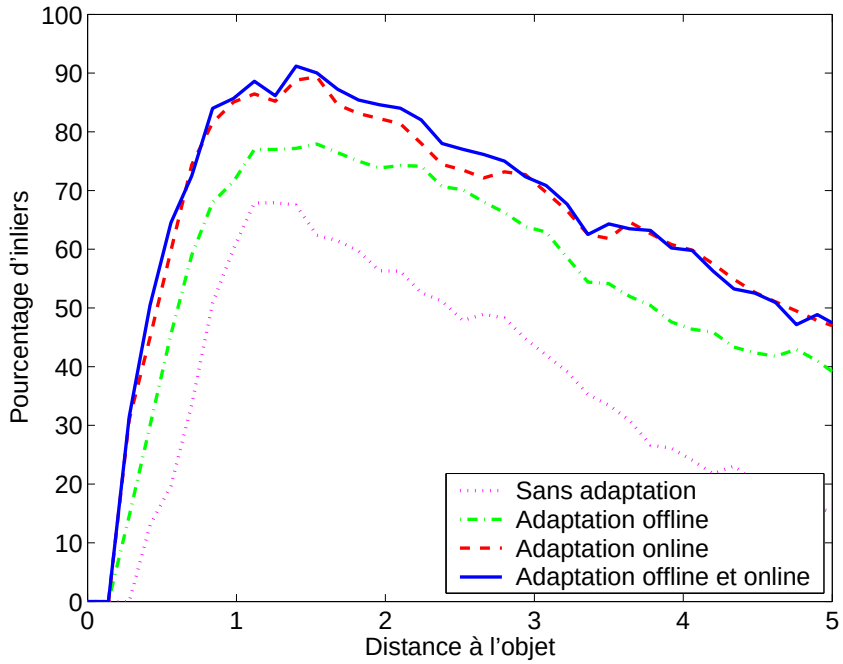



FIG. 9.13 – *Deux modes de fonctionnement différents des arbres adaptatifs*

$[0,3 \quad 5,0]$, on constate que les performances diminuent de façon globale.

Répartir les arbres dans l'espace permet clairement d'éviter ce *trade-off* (ou « compromis »), car la zone de couverture peut être étendue sans perte de performance globale. Pour obtenir la courbe de la figure 9.14, 40 arbres ont été construits, mais uniquement entre 15 ou 20 arbres (sur ces 40) sont utilisés pour une image donnée.

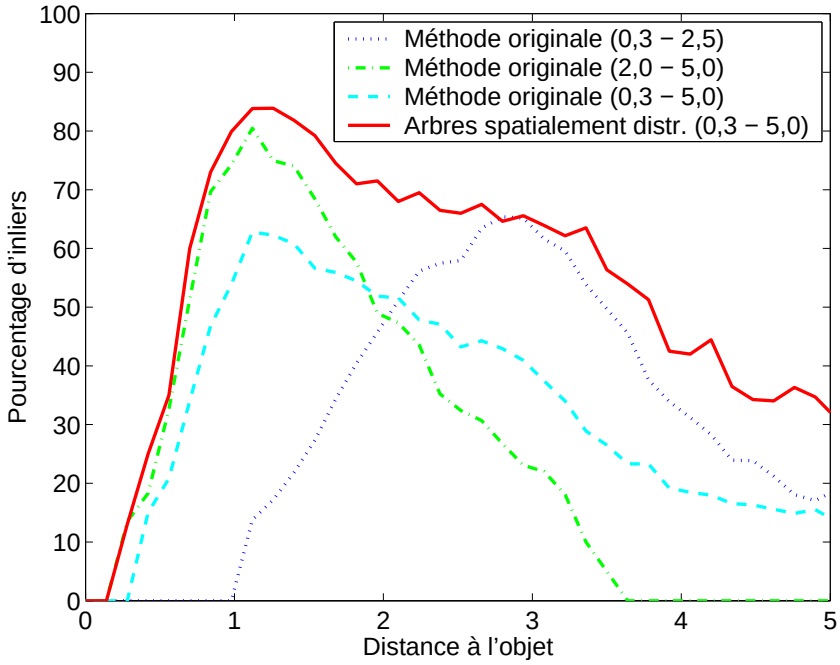



FIG. 9.14 – *Amélioration de la précision de l'appariement et de la zone de couverture des arbres grâce à leur répartition dans l'espace*

# Chapitre 10

# De l'article au congrès scientifique

## 10.1 Écriture et soumission d'une publication

Mon tuteur m'a proposé d'écrire un article scientifique décrivant les différentes idées que j'ai eues pour améliorer la méthode de Lepetit *et al.*, notamment car les résultats expérimentaux étaient très concluants. Le fait d'avoir lu et écrit des critiques sur de nombreux articles tout au long de mon stage m'a particulièrement servi pour écrire ma propre publication. J'ai donc rédigé seul une première mouture de l'article et mon tuteur m'a ensuite aidé à améliorer sa structure. J'ai aussi reçu l'aide d'Américains qui ont relu la version finale pour reformuler certaines tournures.

Nous avons décidé de soumettre l'article à une conférence internationale de vision par ordinateur appelée BMVC (*British Machine Vision Conference*). Il existe des dizaines de conférences dans ce domaine, les plus petites ayant une envergure nationale, voire régionale. Il est difficile d'établir un classement des conférences, mais il semble que l'on considère aujourd'hui trois conférences comme étant d'un niveau supérieur aux autres :

- CVPR (*Conference on Computer Vision and Pattern Recognition*)
- ICCV (*International Conference on Computer Vision*)
- ECCV (*European Conference on Computer Vision*)

Derrière ces trois congrès majeurs, on retrouve BMVC, ainsi que d'autres conférences importantes comme ICPR (*International Conference on Pattern Recognition*). En fait, ce genre de classement est évidemment très arbitraire et l'envergure des conférences évolue aussi très vite. Par exemple, le nombres de publications soumis à la conférence BMVC a quasiment doublé entre 2005 et 2006, ce qui montre que ce congrès est de plus en plus reconnu.

Près de 450 articles ont été soumis à BMVC en 2006. Chaque publication est lue par un comité de lecture qui n'a pas connaissance des auteurs de l'article. Plus précisément, un papier soumis est relu par trois experts et la

décision d'accepter l'article pour le congrès est prise par un « Area Chair ».

Les auteurs des articles considérés par le comité de lecture comme étant les plus intéressants ont la possibilité de donner une présentation orale de 20 minutes pour présenter leur travail. Les auteurs des autres articles acceptés pour le congrès doivent résumer leur contribution sur un poster au format A1.

Environ 30% des articles soumis ont été acceptés pour la conférence BMVC 2006, dont 42 pour une présentation orale (moins de 10% du nombre total d'article soumis).

J'ai soumis ma publication dont le titre est « Real-Time Feature Matching using Adaptive and Spatially Distributed Classification Trees » le 19 avril 2006 et j'ai reçu le compte-rendu du comité de lecture le 15 juin. La conclusion de l'*Area Chair* était « A clear oral » et j'ai donc eu la chance de faire partie des 10% des articles acceptés pour une présentation orale.

## 10.2 Préparation de la présentation orale

Comme je n'avais jamais assisté à une conférence, je ne savais pas trop quelle devait être la forme de ma présentation orale. À Siemens Corporate Research, la plupart des chercheurs sont cependant habitués à assister aux principaux congrès de vision par ordinateur, et j'ai pu obtenir des conseils de la part de différents collègues. Ainsi, j'ai notamment essayé de préparer un *PowerPoint* avec le moins de texte possible et de nombreuses animations. Plusieurs vidéos ont été intégrées à ma présentation, ce qui permet d'appréhender le gain d'efficacité par rapport à la méthode originale beaucoup plus facilement qu'avec des photos ou des graphiques. J'ai aussi essayé d'anticipé les questions qui pourraient m'être posées après ma présentation, et j'ai donc prévu plusieurs diapositives supplémentaires à ne montrer qu'en cas de questions précises. En effet, le temps qui nous est imparti étant limité à 15 minutes de présentation pour 5 minutes de question, on ne peut évidemment évoquer tous les éléments de la méthode et il faut sélectionner ce qui permet de comprendre le plus rapidement le principe général.

La conférence a eu lieu du 4 au 7 septembre 2006 alors que mon stage s'est terminé le 31 juillet. Cependant, la présentation était bien sûr au point avant mon départ de Siemens, ce qui m'a permis de m'entraîner plusieurs fois : je l'ai présentée une première fois devant mon tuteur et quelques collègues, qui m'ont donné quelques conseils pour l'améliorer. Une seconde fois, tout le département *Real-Time Vision and Modeling* a été invité, et j'ai pu profiter du *feedback* de nombreux autres chercheurs.

## 10.3 Déroulement de la conférence

### 10.3.1 Premières impressions

La conférence a eu lieu à Édimbourg, en Écosse, entre le 4 et le 7 septembre. C'est l'âge des participants qui m'a d'abord surpris. Même s'il y avait bien sûr de nombreux chercheurs assez âgés, une majorité des participants étaient en fait de jeunes étudiants préparant leur thèse ou leur PhD. Je pense néanmoins que j'étais le plus jeune, car les autres participants avaient je pense tous, au moins déjà un master (ou équivalent). Il semble en effet que le système du stage long soit assez unique, et il est certainement assez difficile d'arriver à obtenir une publication dans une conférence internationale dans le cadre d'un stage de quelques mois.

De multiples nationalités étaient représentées : de nombreux participants étaient anglais et américains, mais il y avait aussi beaucoup de Chinois, d'Indiens, d'Allemands et de Français. J'ai notamment rencontré plusieurs personnes qui avaient aussi fait un stage pour Siemens Corporate Research à Princeton.

L'atmosphère décontractée m'a aussi étonné. Il n'était pas nécessaire d'emmener un costume, la plupart des participants étant en « jean / t-shirt », et ce même pendant le « dîner de conférence », organisé dans la bibliothèque du *Old College* d'Édimbourg dans un cadre très « chic »...

### 10.3.2 De nombreuses rencontres

Les conférences internationales représentent des opportunités uniques de pouvoir rencontrer de très nombreux chercheurs travaillant dans le même domaine que soi. Pendant les séances de posters bien sûr, mais aussi pendant les pauses ou les repas en commun, les discussions permettent aux chercheurs de confronter leurs idées avec d'autres personnes aussi passionnées qu'eux, et j'avais vraiment l'impression d'être dans un environnement particulièrement propice à l'éclosion de nouvelles idées, ou à la naissance de partenariats entre différents laboratoires. Pour ma part, j'ai ainsi eu, en l'espace de trois jours, plusieurs propositions de projets de fin d'étude (que je vais devoir faire à la fin de mon master à partir du mois d'avril prochain), voire des propositions de thèses dans le cas où je décide de me lancer dans cette voie.

Voir que son travail intéresse de nombreux autres chercheurs est aussi quelque chose de particulièrement appréciable.

Enfin, te telles conférences permettent de rencontrer des chercheurs très connus dans le domaine. Dans le cadre du projet que j'ai exposé dans ce rapport, j'ai par exemple pu croisé Chris Harris, le « Monsieur des points de Harris » que j'ai présentés dans la section 6.2.2 page 26.

J'ai aussi et surtout eu la possibilité de discuter longuement avec Vincent Lepetit, qui est bien sûr l'auteur principal de l'article sur lequel j'ai travaillé dans le cadre du projet évoqué dans ce rapport. Nous avons notamment

discuté des idées que je décris dans ma publication : il était satisfait que d'autres chercheurs utilisent et améliorent la méthode qu'il a proposée et était impressionné par les résultats expérimentaux obtenus. Il m'a aussi indiqué quelles étaient ses futures pistes de recherche concernant les arbres de classification.

De nombreux autres chercheurs renommés dans le domaine de la vision par ordinateur étaient aussi présents. Nous pouvons par exemple citer A. Fitzgibbon, A. Zisserman, D. Kriegman, ou encore Nikos Paragios, qui a donné des cours à l'ENPC en 2004, et que je retrouverai pour le master recherche « Mathématiques, Vision, Apprentissage » que je vais faire l'année prochaine à l'École Normale Supérieure de Cachan.

# Chapitre 11

# Extension : utilisation d'un vocabulaire visuel

Comme je l'ai expliqué dans la section 4.5 page 18, je me suis engagé à ne parler dans ce rapport que de la partie de mon stage en rapport à l'article qui a été publié. Comme ceci ne représente qu'une part restreinte du travail que j'ai fourni, mon tuteur m'a aussi permis d'évoquer sans trop entrer dans les détails un autre projet sur lequel j'ai travaillé en lien direct avec le sujet que j'ai décrit jusqu'ici. Ce travail a notamment abouti à une « déclaration d'invention » (« Invention disclosure » en anglais), qui est une étape nécessaire en vue de l'obtention d'un brevet, et qui contient une courte description de l'invention en vue d'étudier sa brevetabilité.

## 11.1 Objectif

Nous cherchons dans ce projet à établir une méthode permettant une fois de plus d'établir des correspondances entre des points d'intérêt de deux images d'une même scène. Cependant, contrairement à ce qui précède, les deux images correspondent à une scène arbitraire, non étudiée auparavant. Ainsi, même si une phase d'apprentissage est toujours mise à profit, celle-ci utilise une base de données contenant de multiples images diverses, et non plus une photo de la scène précisé que l'on souhaite reconnaître et détecter dans la phase d'exécution.

Une telle méthode *générique* permettrait de réutiliser le même classifieur dans différentes situations, sans avoir à le ré-entraîner pour chaque nouvelle scène.

## 11.2 Vocabulaire visuel

Nous faisons l'hypothèse qu'il existe un ensemble de « catégories de points d'intérêt » qui permettent de décrire une scène, et nous nous propo-

sons de construire une sorte de *vocabulaire visuel* où les mots correspondent à ces catégories. Cette approche est notamment inspirée des méthodes utilisant les *textons* [19] et les *sacs de mots*, aussi appelés « bags of words » en anglais, voire plus spécifiquement dans le domaine de la vision par ordinateur : « bags of keypoints » [6]. Ces approches, d'abord utilisées dans le contexte de l'analyse de textes, notamment par les moteurs de recherche comme Google, sont depuis quelques années utilisées pour la classification d'images.

Un *mot visuel* du *vocabulaire* correspond à une classe de points d'intérêt qui ont une particularité commune (par exemple, ils correspondent à une distribution spécifique des gradients dans leur voisinage). Dans le cas des *sacs de mots*, une image est alors décrite par un histogramme des occurrences de ces mots, et cet histogramme est utilisé pour classer les images dans différentes catégories.

## 11.3 Création du vocabulaire et du classifieur

Le vocabulaire est créé pendant une phase d'apprentissage. De multiples images quelconques sont utilisées, issues de bases de données existantes. Des méthodes rapides d'extraction de points d'intérêt sont appliquées sur ces images, et les voisinages des points détectés sont normalisés pour obtenir :

- dans un premier temps une invariance à l'échelle en utilisant la théorie de l'espace d'échelle déjà évoquée dans la section 6.2.5 page 36,
- dans un second temps des patchs invariants aux transformations affines en utilisant une des méthodes décrites dans la section 6.2.5 page 38.

Les patchs normalisés sont alors regroupés en grappes (« *to cluster* » en anglais) en utilisant une méthode classique comme les nuées dynamiques (*k-mean*) ou des mélanges de gaussiennes (*Gaussian mixture*). Le *clustering* peut se faire à partir des intensités de ces patchs, ou à partir de descripteurs comme SIFT, décrit dans la section 9.1 page 59. La figure 11.1 illustre le principe du clustering dans le cas d'un descripteur de 3 dimensions. Chaque *grappe*, ou *cluster*, correspond à un *mot visuel* du *vocabulaire* utilisé pour décrire une image. Pour des algorithmes comme *K-Means*, le nombre de mots doit être défini à l'avance, et il s'avère que ce paramètre est très important pour le bon fonctionnement de la méthode.

Pendant la phase de mise en correspondance qui doit être très rapide, nous ne pouvons pas utiliser les méthodes généralement lentes qui permettent de normaliser les patchs par rapport aux changements de points de vue. C'est pourquoi nous proposons de synthétiser de nouveaux patchs en appliquant des transformations affines à ceux qui viennent d'être *clusterisés* et qui sont donc étiquetés. Cette approche rappelle celle utilisée par Lepetit *et al.* dans le projet décrit précédemment, où ils proposent de géné-

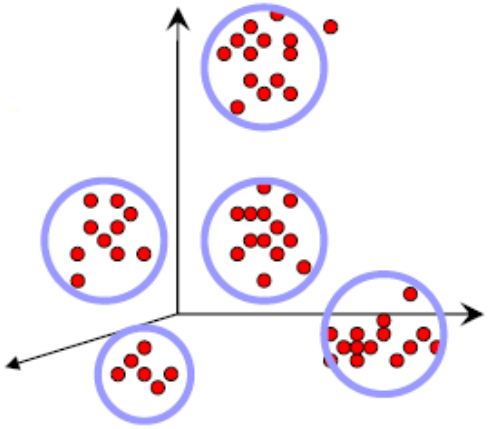


FIG. 11.1 – Clustering *des patchs et obtention des* mots visuels

rer de nouvelles vues de l'objet d'intérêt grâce à des transformations affines, et de les utiliser pour construire le classifieur, ceci afin de pallier l'absence d'un véritable détecteur de points d'intérêt invariant aux transformations affines pendant la phase temps-réel.

Comme dans la section 7.9 page 51, nous construisons alors une forêt d'arbres de classification, à partir des patchs extraits de la base de données qui ont été normalisés, et de ceux qui ont été synthétisés par transformations affines.

## 11.4 Mise en correspondance

Pour établir des correspondances entre deux images, celles-ci sont d'abord traitées indépendamment : on extrait les points d'intérêt avec la même méthode que pendant la phase d'apprentissage, à ceci près que les patchs ne sont pas normalisés par rapport au changement de point de vue, car cette étape est trop gourmande en temps de calcul et nous venons d'expliquer que les arbres ont été construits pour gérer cette difficulté.

La deuxième étape concerne la classification des patchs extraits, grâce aux arbres de classification, qui revient à associer chaque point d'intérêt à un mot visuel du vocabulaire, ceci pour les deux images entre lesquelles on souhaite établir des correspondances.

Reste enfin à apparier ces mots visuels. Si le vocabulaire est très complet et discriminant, il peut arriver qu'il n'y ait qu'un point détecté dans chaque image faisant partie d'une classe donnée, et donc une correspondance unique. Cependant, la mise en correspondance n'est généralement pas aussi triviale, car plusieurs occurrences des même mots sont présentes dans

les deux images. Dans ce cas, nous nous aidons des relations géométriques entre les mots visuels. Différentes approches existent dans la littérature, et j'ai notamment pu m'inspirer de la notion de *shape-context* introduite par Belongie *et al.* [3] et d'un article datant de 2005 dont mon tuteur est l'un des co-auteurs, qui utilise les contraintes spatiales pour la détection d'objet [14]. Je n'ai néanmoins pas l'autorisation d'entrer plus dans les détails de la méthode que nous utilisons.

# Chapitre 12

# Travail d'implémentation

## 12.1 Préliminaire

Même si mon stage était très orienté « recherche », j'ai évidemment quand même souvent dû exercer des tâches d'implémentation, ceci n'étant pas du tout pour me déplaire, car j'ai toujours été passionné par la programmation.

J'ai utilisé différents langages durant ce stage :

**C++ :** Dans le département de vision en temps réel, ce langage est le plus utilisé, car il permet souvent d'obtenir des programmes dont l'exécution est plus rapide qu'avec du code en *Java*. J'utilisais notamment la librairie *IPP* d'Intel pour les opérations courantes de traitements d'images, et le logiciel *Purify* pour détecter les fuites de mémoire.

**Matlab :** J'ai surtout utilisé *Matlab* pendant les phases d'expérimentation car il est très facile d'obtenir rapidement des résultats avec quelques lignes de code.

**OpenGL :** J'ai utilisé *OpenGL*, et plus particulièrement la bibliothèque *GLUT* pour la gestion des fenêtres et l'interaction avec l'utilisateur, car je travaillais toujours à l'aide d'une caméra branchée à l'ordinateur, qui me permettait d'analyser très rapidement les conséquences des modifications que j'apportais au code.

**VRML :** Je me suis servi du langage *VRML* pour créer des modèles 3D texturés.

## 12.2 La contrainte temps-réel

J'ai eu la possibilité d'améliorer significativement mes compétences dans l'utilisation de ces outils, notamment grâce à la contrainte temps-réel, qui oblige à programmer des algorithmes *C++* efficaces. En effet, dans une application temps-réel, nous disposons uniquement de 40 millisecondes pour

traiter chaque image (d'une résolution de $640 \times 480$ pixels). Je décomposais le traitement d'une image en différentes étapes, et essayais de modifier le code pour gagner quelques dixièmes de milliseconde sur chacune d'elles.

En ce qui concerne le projet évoqué jusqu'ici dans ce rapport, une exécution-type donne ce genre de résultats :

| **Étape** | **Durée** |
|---|---|
| Extraction des points d'intérêt | 10 ms |
| Attribution des orientations | 7 ms |
| Classification avec les arbres | 12 ms |
| Estimation de la position de la caméra avec RANSAC | 7 ms |
| Estimation finale avec Levenberg-Marquardt | 6 ms |

Ma première implémentation que l'on pourrait qualifier de « naïve » de l'algorithme, ne permettait de traiter qu'environ cinq images par seconde, mais j'ai utilisé différentes astuces pour optimiser le code. Celles-ci allaient par exemple de l'utilisation de la commande `memset` pour l'initialisation d'un tableau, à une refonte totale du système de gestion des arbres de classification, en passant par l'emploi de *tables de conversion* (aussi appelées « *tables de correspondances* », ou « *look-up tables* » en anglais) pour le calcul de la fonction arctangente...

Une autre difficulté dans ce projet concerne la quantité de mémoire utilisée pendant la phase d'apprentissage. Il convient ici aussi d'optimiser le code car une implémentation « naïve » pour la construction des arbres de classification nécessite beaucoup trop de mémoire.

## 12.3 Amélioration du *portfolio* de Siemens Corporate Research

Selon les différents projets et tâches que j'ai accomplis lors de ce stage, j'ai parfois dû m'approprier un code déjà existant et l'améliorer ou lui rajouter des éléments, et j'ai d'autres fois eu la possibilité de partir d'un code vierge. Ce deuxième cas est souvent le plus agréable, car la compréhension d'un code écrit par d'autres personnes est souvent assez difficile et fastidieuse, surtout lorsqu'il n'est pas suffisamment commenté.

Pour le projet que je peux évoquer dans ce rapport et dont j'ai parlé jusqu'ici, à savoir l'estimation de la position de la caméra par rapport à un objet donné en utilisant des correspondances entre points d'intérêt établies grâce à des arbres de classification, je suis parti d'un code vierge. J'ai implémenté seul les étapes de détection des points d'intérêt et d'appariement, j'ai travaillé en collaboration avec un autre stagiaire (Nicolas Gogin) pour la partie géométrique avec l'algorithme des trois points, et j'ai utilisé un code

existant pour l'algorithme d'optimisation Levenberg-Marquardt.

J'ai été surpris par le manque d'organisation au sein du laboratoire concernant la gestion des différents programmes et algorithmes disponibles. Ainsi, en commençant mon programme, j'ai tout naturellement demandé à mon tuteur quelle classe C++ je devais utiliser pour la gestion des images (création d'une nouvelle image, accès à l'intensité lumineuse d'un pixel, et autres opérations de base). Il m'a répondu que de nombreuses classes étaient employées dans les différents groupes et pour les différents projets. Ceci est un exemple parmi beaucoup d'autre qui souligne le manque d'homogénéisation et de mise en commun du travail effectué au sein du département.

De même, j'ai parfois eu à utiliser des méthodes aujourd'hui très classiques, et qui sont très utilisées en vision par ordinateur, comme RANSAC (décrit dans la section 8.2 page 55), ou l'algorithme des nuées dynamiques (*K-Means* en anglais). Je m'attendais alors à pouvoir réutiliser un code déjà existant, car de nombreux chercheurs du laboratoire devaient certainement utiliser ces méthodes, mais mon tuteur m'a confié qu'il serait plus rapide de les reprogrammer que de trouver quelqu'un disposé à retrouver son code et à le partager...

### 12.3.1 Programmation générique

À mon niveau, j'ai tenté de faire changer un peu les choses, et nous avons notamment décidé avec mon tuteur de programmer des versions génériques et réutilisables des algorithmes classiques comme RANSAC ou les nuées dynamiques. Pour ce faire, j'ai implémenté des codes les plus propres et le plus documentés possible, en utilisant ce que l'on appelle en C++ les « *templates* » qui permettent d'obtenir un code invariant au type de données.

À titre d'exemple, la réutilisation de l'algorithme RANSAC, pour une application autre qu'estimer la position de la caméra à partir de correspondances 3D-2D, ne nécessite désormais que la création une nouvelle classe fille qui hérite de la classe mère que j'ai créée, et qui se contente de définir

- la fonction d'estimation des paramètres du modèle à partir d'un échantillon (l'équivalent de l'estimation de la position de la caméra à partir de trois correspondances avec l'algorithme des trois points),
- et la fonction qui détermine si une observation appartient au support de ce modèle (l'équivalent de la fonction qui calcule la distance entre un point d'intérêt et la projection sur l'image du point de l'objet qui lui est apparié).

Ce type de programmation est évidemment courant en C++ et n'a rien de compliqué, mais était malheureusement très peu utilisé au sein du département. Mes implémentations génériques des méthodes classiques comme RANSAC ou K-Means sont déjà utilisées dans différents projets et il semble qu'un début d'homogénéisation des différents codes est en train de se pro-

duire.

### 12.3.2 Utilisation d'un gestionnaire de versions

Un autre problème que j'ai souvent rencontré au début de mon stage concerne les conflits de versions. Comme je l'ai expliqué à titre d'exemple, il existe différentes bibliothèques de gestion d'images qui circulent au sein même du département. Lorsque j'ai eu à travailler en collaboration avec d'autres chercheurs, même si par chance nous utilisions la même bibliothèque, la probabilité restait élevée que des conflits se produisent car nous n'avions pas la même version. En effet, si par exemple quelqu'un corrige un bug dans une fonction, il n'aura pas pour autant la possibilité de prévenir tous les autres chercheurs qui utilisent la même bibliothèque...

Des outils comme *CVS* ou *SourceSafe*, appelés « gestionnaires de versions », existent pour éviter ce genre de désagréments, mais ils n'étaient étonnamment pas du tout utilisés dans le département. À mon arrivée, j'ai proposé à mon tuteur de les mettre en place, et nous avons commencé à notre petit niveau à les utiliser : tout au long de l'année, nous avons pu travailler tous les deux sur différents projets sans aucun conflit. Au fur et à mesure de l'année, le système s'est un peu développé, et à mon départ, une partie non négligeable du département utilisait *SourceSafe*. Par exemple, si une optimisation de mon implémentation générique des nuées dynamiques est un jour proposée par un chercheur, elle pourra directement profiter à tout ceux qui utilisent aussi le code.

# Conclusion

Comme je l'ai expliqué en début de document, ce rapport ne peut exposer l'intégralité de ce que j'ai effectué pendant ce stage. Il aborde plutôt un sujet précis sur lequel j'ai travaillé, en envisageant toutes les étapes classiques du travail d'un chercheur :

- Lecture d'articles scientifiques afin d'appréhender l'état de l'art
- Implémentation de différentes méthodes pour mieux comprendre les enjeux et les difficultés du problème
- Travail de réflexion pour imaginer de nouvelles méthodes, et trouver diverses idées pour améliorer les algorithmes existants
- Écriture d'un article scientifique
- Préparation d'une présentation orale pour une conférence internationale (BMVC 2006)

En ce qui concerne le projet décrit dans ce rapport, l'objectif était de proposer des approches permettant d'améliorer les résultats d'une méthode dont les performances sont déjà au sommet de l'état de l'art. Dans ce cas précis, les idées que j'ai proposées sont finalement conceptuellement très simples. Cependant, les résultats expérimentaux sont particulièrement concluants. Il en est en fait de même pour la méthode proposée par l'équipe de Lepetit : l'article original ne repose pas sur des résultats mathématiques compliqués mais parvient finalement, en utilisant un détecteur de points d'intérêt basique, et des arbres de classification, qui correspondent à une des méthodes d'apprentissage les plus élémentaires, à obtenir des résultats meilleurs que toutes les méthodes proposées jusqu'ici.

En utilisant les informations nouvelles concernant l'apparence réelle de l'objet d'intérêt lorsqu'il est vu sous différents angles de vue et dans des conditions d'illuminations nouvelles, nous améliorons les arbres de classification en temps réel. Ceci permet de pouvoir gérer des objets 3D plus complexes, des angles de vue plus extrêmes, des matériaux spéculaires, etc.

Le fait de répartir les arbres dans l'espace et de les spécialiser, permet de diminuer leur variance intra-classe et donc d'améliorer leurs performances. Ainsi, on peut grossir la zone de couverture de la forêt et rendre l'ensemble du processus de mise en correspondance encore plus rapide.

Comme nous l'avons évoqué, la possibilité de pouvoir se passer d'une phase d'apprentissage spécifique à un objet donné aurait un intérêt certain. C'est une des pistes de recherche future, que nous avons déjà abordée dans ce rapport avec l'utilisation d'un vocabulaire visuel.

Le fait qu'il existe de nombreux autres axes de recherche dans cette direction rend la problématique particulièrement passionnante. Par exemple, l'ajout et le retrait dynamiques dans les arbres de classification de nouvelles classes correspondant à d'autres points d'intérêt est une autre voie de recherche qui aurait elle aussi de multiples applications.

# Bilan personnel

Par rapport à mes différentes interrogations concernant ma formation d'ingénieur, ce stage au laboratoire de Recherche & Développement de Siemens aura été particulièrement intéressant, ceci pour deux principales raisons.

Je ne connais le domaine de la vision par ordinateur que depuis peu de temps. C'est en fait le séminaire de département du début de la deuxième année, à Sophia Antipolis, qui m'a permis de découvrir différentes problématiques et applications de cette discipline. J'ai rapidement été séduit et, après avoir suivi quelques cours d'introduction à l'École des Ponts en deuxième année, j'ai choisi de faire un stage long dans le domaine afin d'avoir une véritable expérience avant de poursuivre éventuellement mes études dans cette voie. Comme le travail que j'ai eu à effectuer pendant le stage m'a passionné et que les différents enjeux et problématiques de la vision par ordinateur que j'ai découverts tout au long de l'année, que ce soit au sein du laboratoire en discutant avec des collègues ou pendant la conférence à Édimbourg, m'ont généralement particulièrement intéressés, cette année m'a finalement conforté dans ma décision de continuer mes études dans le domaine de la vision par ordinateur et du traitement d'images.

L'autre question, plus difficile, concerne le choix entre me lancer dans la recherche ou dans une carrière dans l'industrie. Le fait d'avoir travaillé pour un département de R&D est de ce point de vue particulièrement intéressant car j'ai pu goûter aux différents aspects de l'un et de l'autre. Cependant, il semble que ce genre de laboratoires où la recherche est particulièrement encouragée ne soient pas légion en France et je pense en fait que j'ai fait mon stage dans une structure assez unique. J'ai malgré tout été en relation toute l'année avec de nombreux étudiants en PhD ou en post-doc et j'ai aujourd'hui une bien meilleure connaissance du milieu de la recherche comme de celui de l'industrie, ce qui me permettra le jour venu de faire un choix plus réfléchi. Je dois cependant avouer que je n'ai toujours pas décidé si je souhaite faire une thèse après mon master-recherche.

Par rapport à la suite de mon cursus, le projet abordé dans ce rapport constitue également une introduction idéale. J'ai en effet choisi de suivre l'année prochaine le master de l'ENS Cachan intitulé « Mathématiques, Vi-

sion, Apprentissage ». Or la méthode d'appariement de points d'intérêt en utilisant des arbres de classification se situe exactement à l'interface entre *vision* et *apprentissage*. De plus, j'ai abordé pendant ce stage plusieurs autres problématiques de vision, et j'ai étudié d'autres méthodes d'apprentissage comme les machines à vecteurs de support (en anglais *Support Vector Machine* ou *SVM*).

Les problématiques de l'apprentissage et plus généralement de l'intelligence artificielle m'attirent et me fascinent de plus en plus et j'essaierai de prendre le plus de cours possibles dans cette direction.

Enfin, pour conclure ce bilan personnel, il est évident qu'une année à Princeton, entre New York et Philadelphie, est aussi une très bonne expérience et une opportunité unique de découvrir une culture que je ne connaissais finalement pas aussi bien que ce que je pensais.

# Bibliographie


[1] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7) :1545–1588, 1997.

[2] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE PAMI*, 24(4) :509–522, 2002.

[4] Matthew Brown and David G. Lowe. Recognising panoramas. In *International Conference on Computer Vision (ICCV)*, pages 1218–25, Oct. 2003.

[5] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8 :679–698, 1986.

[6] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

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

[8] J. Grunert. Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie. *Grunerts Archiv für Mathematik und Physik*, pages 331–356, 1841.

[9] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Conference on Computer Vision and Pattern Recognition*, 1991.

[10] C. Harris and M. Stephens. A combined corner and edge detector. In *4th ALVEY Vision Conference*, 1988, pages =.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second edition, 2004.

[12] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[13] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Conference on Computer Vision and Pattern Recognition, San Diego, CA*, June 2005.

[14] Yan Li, Yanghai Tsin, Yakup Genc, and Takeo Kanade. Object detection using 2d spatial ordering constraints. In *CVPR (2)*, page 1188, 2005.

[15] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2) :77–116, 1998.

[16] Tony Lindeberg and Jonas Gårding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image Vision Comput.*, 15(6) :415–434, 1997.

[17] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece, September 1999.

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[19] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, 7(5), 1990.

[20] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the 13th British Machine Vision Conference*, pages 384–393, September 2002.

[21] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002. Copenhagen.

[22] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2) :43–72, 2005.

[23] Hans P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report Report No. STAN-CS-80-813, Department of Computer Science, Stanford University, 1980. Memo AIM-340.

[24] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005.

[25] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5) :530–534, May 1997.

[26] Tinne Tuytelaars and Luc J. Van Gool. Content-based image retrieval based on local affinely invariant regions. In *VISUAL '99 : Proceedings of the Third International Conference on Visual Information and Information Systems*, pages 493–500, London, UK, 1999. Springer-Verlag.

[27] Z. Zhang, R. Deriche, O. Faugeras, and Q.T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence Journal*, 78 :87–119, October 1995.

# Index