Image processing applications in object detection and graph matching : from Matlab development to GPU framework

Beibei Cui

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ PRÉPARÉE À L'UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

École doctorale n°37
Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

Image processing applications in object detection and graph matching: from Matlab development to GPU framework

Official Documentation

par

Beibei CUI

Connaissance et Intelligence Artificielle Distribuées,
Univ. Bourgogne Franche-Comté, UTBM, France

Thèse présentée et soutenue à Belfort, le 27 mars 2020 devant le jury ccomposé de:

Abderrafiaa KOUKAM	**Président**, Professeur des universités, *CIAD, Univ. Bourgogne Franche-Comté*
Cyril FONLUPT	**Rapporteur**, Professeur des universités, *Université du Littoral - Côte d'Opale*
Adnan YASSINE	**Rapporteur**, Professeur des universités, *ISEL, Université Le Havre Normandie*
Lhassane IDOUMGHAR	**Examinateur**, Professeur des universités, *IRIMAS, Univ. Haute Alsace*
Yassine RUICHEK	**Examinateur**, Professeur des universités, *CIAD, Univ. Bourgogne Franche-Comté*
Jean-Charles CRÉPUT	**Directeur de thèse**, Maître de Conférences HDR, *CIAD, Univ. Bourgogne Franche-Comté*

N° | 2020 | UBFC | A020

Abstract

Automatically finding correspondences between object features in images is of main interest for several applications, as object detection and tracking, flow velocity estimation, identification, registration, and many derived tasks. In this thesis, we address feature correspondence within the general framework of graph matching optimization and with the principal aim to contribute, at a final step, to the design of new and parallel algorithms and their implementation on GPU (Graphics Processing Unit) systems. Graph matching problems can have many declinations, depending on the assumptions of the application at hand. We observed a gap between applications based on local cost objective functions, and those applications with higher-order cost functions, that evaluate similarity between edges of the graphs, or hyperedges when considering hypergraphs. The former class provides convolution-based algorithms already having parallel GPU implementations. Whereas, the latter class puts the emphasis on geometric inter-feature relationships, transforming the correspondence problem to a purely geometric problem stated in a high dimensional space, generally modeled as an integer quadratic programming, for which we did not find GPU implementations available yet.

Two complementary approaches were adopted in order to contribute to addressing higher-order geometric graph matching on GPU. Firstly, we study different declinations of feature correspondence problems by the use of the Matlab platform, in order to reuse and provide state-of-the-art solution methods, as well as experimental protocols and input data necessary for a GPU platform with evaluation and comparison tools against existing sequential algorithms, most of the time developed in Matlab framework. Then, the first part of this work concerns three contributions, respectively, to background and frame difference application, to feature extraction problem from images for local correspondences, and to the general graph matching problem, all based on the combination of methods derived from Matlab environment. Secondly, and based on the results of Matlab developments, we propose a new GPU framework written in CUDA C++ specifically dedicated to geometric graph matching but providing new parallel algorithms, with lower computational complexity, as the self-organizing map in the plane, derived parallel clustering algorithms, and distributed local search method. These parallel algorithms are then evaluated and compared to the state-of-the-art methods available for graph matching and following the same experimental protocol. This GPU platform constitutes our final and main proposal to contribute to bridging the gap between GPU development and higher-order graph matching.

Résumé

Déterminer des mises en correspondance d'objet, ou de caractéristiques d'objet, dans des images présente un grand intérêt pour beaucoup d'applications telles que la détection et le suivi de cible, l'estimation du flot optique, l'identification, et d'autres tâches dérivées. Dans cette thèse, nous abordons le problème de mise en correspondance dans le cadre général de l'optimisation de l'appariement de graphe, dans le but de contribuer, comme résultat final, au développement de nouveaux algorithmes parallèles implémentés sur plateforme GPU (Graphics Processing Unit). Le problème d'appariement de graphe peut être décliné de diverses manières suivant l'application considérée. Nous observons un fossé entre les applications basées sur des fonctions de coût locales et les applications avec des fonctions de coût d'ordre supérieur, évaluant la similarité entre les arêtes du graphe, ou les hyperliens lorsqu'il s'agit d'un hypergraphe. La première classe d'applications comporte des algorithmes de résolution basés sur des calculs de convolution et possède déjà des implémentations parallèles sur GPU. La deuxième classe d'applications met l'accent sur les relations géométriques entre caractéristiques extraites de l'image, transformant le problème de mise en correspondance en un programme quadratique en nombre entiers avec contraintes, pour lequel nous n'avons pas trouvé de solution GPU accessible actuellement.

Deux types d'approche ont été adoptées pour contribuer à la problématique d'appariement de graphe sur GPU. Premièrement, nous étudions différentes déclinaisons de cette problématique via l'utilisation de la plateforme Matlab afin de pouvoir réutiliser et fournir des solutions représentatives de l'état de l'art, ainsi que des protocoles d'expérimentation et des données d'entrée nécessaires pour une plateforme GPU dédiée à l'évaluation et la comparaison avec les algorithmes séquentiels sur Matlab. Ainsi, une première partie du travail concerne trois contributions respectivement, aux techniques de soustraction d'arrière-plan et de différence d'image pour la détection, au problème d'extraction de caractéristiques pour la mise en correspondance, et au problème général d'appariement de graphe, toutes basées sur la combinaison de méthodes issues de l'environnement Matlab. Deuxièmement, nous proposons une infrastructure logicielle GPU nouvelle, écrite en CUDA C++, spécifiquement adaptée au problème d'appariement de graphe géométrique, proposant de nouveaux algorithmes parallèles de complexité calculatoire plus réduite, tels que les cartes auto-organisatrices dans le plan, des algorithmes de cluster qui en sont dérivés, et des recherches locales distribuées. Ces algorithmes parallèles sont évalués et comparés aux approches de l'état de l'art pour le problème d'appariement de graphe, en suivant un protocole d'expérimentation identique. Cette plateforme GPU constitue notre principale proposition pour contribuer à combler le fossé entre développement GPU et son application au problème général d'appariement de graphe.

Acknowledgements

I would like to take this opportunity to acknowledge people for their help. They led me and insisted on my doctoral thesis work.

First of all, I would like to thank my supervisor, Professor Jean-Charles CRÉPUT, for the opportunity to study this interesting topic, for his wise supervision, patience and support. I would also like to thank his valuable guidance and helpful suggestions. I admire his passion for science, and I am grateful that he use his epistemology to inspire people around.

In addition, I would like to thank the colleagues of the Computer Vision team for their pleasant working atmosphere and many interesting moments in the team. Especially, I'd like to express my gratitude to Abdelkhalek MANSOURI for his advices and discussions regarding some details of this thesis.

Finally, I want to thank my family for their endless trust in me, and I'd like to acknowledge the China Scholarship Council (CSC) for providing me financial support during my PhD study.

Thank you!

Contents

Chapter 1

Introduction

1.1 Context

Graph optimization techniques can be applied in almost all areas of image processing and computer vision. They can be applied to an extensive range of problems, from simple object detection and tracking based on graph differences to more complex and challenging problems such as graph matching (GM) with higher-order potential functions. Graph matching is an essential problem in computer science. It can be applied to a variety of issues such as pattern recognition, machine learning, and computer vision.

Establishing the correspondence between two feature sets is a fundamental issue of image matching. Two sets of feature points are respectively extracted from two images that are to be matched. Then, the main task is to find the corresponding feature point pairs between the reference image and query image, while maintaining some invariant relationships between features. This classic problem is challenging to handle in obtaining accurate solutions. The recent revival of the combinatorial optimization methods for feature matching changed this situation. Graph matching is mostly expressed as an integer quadratic programming (IQP) problem, for which obtaining an exact solution is computationally tricky to handle. Therefore the graph matching problem is confirmed to be a Non-deterministic Polynomial (NP-hard) problem, which needs to find an approximate solution to the problem. Here, this thesis aims to cover set core problems and solutions that arise in the context of graph matching optimization in general, study their complexity and their implementation on sequential and parallels devices, and provide some new state-of-the-art solutions.

1.2 Problematic

We are interested in real-time algorithms for detection and tracking tasks by using graph optimization techniques. The frame difference method (FD) and background subtraction method (BS) methods are ones of the most basic and simple techniques for object detection. They can realize real-time detection with $O(N)$ complexity, with N the problem size, both on memory and time, since they are based on a strong assumption of background stability and brightness hypothesis. Basic processing operations tools with $O(N)$ time complexity have straightforward, fast implementations and are generally compatible with the real-time context of the application in tracking. In addition, their parallel implementation in GPU systems is now a matter of current reality. We propose a new combination of these tools for object detection.

The assumption of background stability and brightness hypothesis can sometimes be relaxed, and a matching method should instead take care of the geometric relationships between feature descriptors to discover correspondences between images. Graph matching algorithm is aimed to find the corresponding map between two sets of features to preserve the geometric relationships of two graphs as much as possible. Existing graph matching methods are generally divided into exact and inexact graph matching [LNZ+19]. The graph isomorphism, or subgraph isomorphism, are cases of exact graph matching. Inexact (approximate) graph matching has a broader range of applications. It refers to matching problems where it is impossible to match exactly, and where a strict decision is replaced by a score function to maximize. Besides, it is not robust to variation or interference, so we focus on the problem of forming graph matching in an inexact way.

The main goal of graph matching is to find the geometric relationship between pairs (two-order GM) or k-tuple (k-order) of features. Due to the higher-order potentials and the quadratic nature of the function cost, most approaches employ tensor-based or matrix-based representation on high dimensional space. For example, an $O(N^2)$ assignation matrix represents the solution, whereas the affinity matrix will have dimension $O\left(N^4\right)$, where N is the number of features. The problem increases in complexity when the objective score function contains higher-order potentials, from first-order, second-order, and high-order terms. Solving graph matching presupposes solving the different versions of matching problems. Here, we choose to contribute to the development of CPU and GPU tools for two-order graph matching and its related sub-problems, such as feature point extraction problematic. We provide proposals considering IQP formulation with the aim to provide state-of-the-art methods that could serve as a basis for further parallel GPU development of GM techniques, as a final step of this thesis. We propose to model

GM in the Euclidean plane, avoiding the requirement of a large affinity matrix like in IQP models.

1.3 Objectives and Contribution

The general goal is to develop state-of-the-art real-time algorithms in the context of graph matching applications, by either the use of a general environment such as Matlab platform, and with the goal to elaborate strategies for parallel execution et resolution on GPU systems, whenever this requirement should appear promising and effective. Since development on GPU are more fastidious, and specialized, we applied the effort on GPU development only on the main problem of two-order graph matching, which is representative of the gap between local correspondence methods, which have many GPU versions, and higher-order correspondence IQP problem, which are poorly implemented on GPU. Other developments were done in Matlab environment, since it allows fast development and allows to provide many original codes and data benchmarks that are necessary tools for evaluation and comparison with GPU systems. An objective of this thesis is also to customize a soon GPU platform with shared benchmark data to support comparison to the state-of-the-art methods most often available in the Matlab platform.

Main objectives and effective contribution of this thesis can be dividing into the following contributions:

- A systematic algorithm for moving object detection with application in real-time surveillance.
- Using entropy and Marr wavelets to automatic feature detection for first-order image matching.
- Affinity-preserving integer projected fixed point under spectral technique for second-order graph matching.
- GPU framework for second-order graph matching in the plane and related parallel algorithms.

Firstly, we propose a systematic tracking algorithm based on the combination of Laplace filter, frame difference method, background subtraction method and Canny edge detector. Laplace filter was used to strengthen image information and improve the detection effect. Canny detector is highly correlated with the edge contour, and as usual, it is more helpful for dividing the pixels into foreground and background. The morphological post-processing operation is expressed as a combination of erosion and dilation. Here, the

closing operation will be employed to fill the small apertures and connect the small gaps. Three frame difference and background difference operations are performed separately, while threshold binarization and Canny edge detection are performed to identify and extract edge information. Then, the combination of these two main methods undergoes a logical OR operation followed by a morphological operation for obtaining the final moving objects. This systematic tracking algorithm was tested on standard datasets with the evaluation criteria of accuracy, recall, precision, and F-measure. All of their programmatic be assembled representative through the Graphical User Interface to make the whole complicated process intuitive and straightforward. Another main contribution of the proposed application is the achievement of a systematic detection algorithm for multi-target tracking in real-time surveillance.

Secondly, a first-order graph matching algorithm using entropy and response based on Marr wavelets within the scale-interaction method is proposed. First, feature detection is performed through Marr wavelets within the scale-interaction method to obtain a feature vector set of the reference image. Second, feature extraction is executed under the mesh division strategy and entropy algorithm, accompanied by the assessment of the distribution criterion. The image is meshed by $n \times n$ to obtain n^2 sub-regions, then the detected feature points are mapped into the respective sub-areas and sorted according to their deviation value Dev in each sub-area to which they belong. Meanwhile, every local information entropy H_i of each sub-region and the average information entropy $\bar{H}$ of the all are calculated. The entropy-based selection method dramatically reduces computation time. Finally, feature points matching and error elimination are performed. Image matching is achieved by the nearest neighbor search with normalized cross-correlation similarity measurement to perform coarse matching on feature points set. As to the matching points filtering part, the RANSAC procedure removes outlier correspondences.

Thirdly, we propose a combination of state-of-the-art power iteration based methods for graph matching, and also a customized environment in Matlab for sharing data with a GPU platform on this problem. A combination variant of second-order graph matching method using the candidate assignment affinity-preserving algorithm and integer projected fixed-point algorithm improved by spectral technique is presented to realize one to one correspondence. It can reflect the geometric similarity relationship between the pairwise matching features and retaining as many attributes as possible. Affinity-preserving can realize normalization by constructing the same maximum degree for preserving the relative affinity relations between reference and query graph. The spectral matching algorithm splits the set of candidate assignments into correct assignments and rejected assignments during the discriminating loop. Then, the improved integer projected fixed-point algorithm realizes the main matching iteration loop. It is

a series of linearly assigned problem where the next solution x_{k+1} is found by using the previous solution x_k, which means the quadratic score S^* is getting closer and closer to the optimal discrete solution since the binary solution returned will never be worse than the initial solution. This whole algorithm effectively achieves a stable optimal solution x^*. The greedy algorithm will be typically used as the final post discretization.

Fourthly, we propose a GPU platform for graph matching. A version of graph matching in the plane is defined, and parallel algorithms in the plane are customized, applied and studied. These algorithms are new versions of a parallel self-organizing map and a parallel local search method. They are compared to state-of-the-art IQP methods. We choose to reuse and customized parallel techniques and heuristics that were already applied to graph minimization problems [Zha13, Wan15, Qia18]. Based on a cellular decomposition of the plane for fast closest-point search, they were applied to mesh generation [WZC+15], fast stereo correspondence problems [WZC+16], traveling salesman problem [WZC17], and super-pixel segmentation [WMC17]. We adapt few of them to graph matching in the plane.

1.4 Plan of the thesis

In chapter 2, we briefly introduce background on object detection and graph matching algorithms, including definitions and related work about basic notions of graph theory, taxonomy of graph matching in first, second and high order, power iteration, CNN for graph matching, and GPU for graph matching.

In chapter 3, we present our proposed solution for moving object detection. It includes two main approaches, namely the frame difference algorithm and background subtraction algorithm. The proposed systematic multi-object detection algorithm is based on the Laplace filter and Canny detector, which can enhance detection since they are less independent of external noise. We applied the method to standard benchmark image sequences and developed a graphical user interface for evaluation and visualization.

In chapter 4, we present a first-order graph matching application that uses our proposed local entropy based method on Marr wavelets within scale-interaction to improve the accuracy of automatic feature detection in the context of image matching. We test the proposed technique for a cross-correlation first-order graph matching problem.

In chapter 5, we present a methodology for two-order graph matching. We present our improved IQP algorithm based on the combination of standard algorithms, and provide experiments of the different approaches. Various data sets are tested for comparison of

matching accuracy, objective score, and time computation in the presence of deformation noise and outliers.

In chapter 6, we provide a two-order graph matching problem stated in the euclidean plane, and a set of GPU tools customized for graph matching evaluation and comparison. Experiments are conducted to evaluate our proposed customized parallel algorithms.

In chapter 7, we conclude this thesis and discuss some future works.

Chapter 2

Background

2.1 Introduction

The goal of this chapter is to present an overview of detection and matching problems in general, to review on which conditions fast and local graph-optimization methods can be applied. Moving object detection and tracking from video sequences is a relevant research field since it can be used in many applications. Currently, ones of the main detection algorithms include frame difference method (FD), background subtraction method (BS), and optical flow method. We first studied this most standard methods in the domain of image matching, generally based on (local) first order cost functions. Also, they require strong assumption on background or/and brightness constancy.

Graph matching allows to alleviate dependence on brightness constancy assumption, by focalizing on feature points and their pairwise geometric relationships, instead of only local information. Establishing the pairwise geometric correspondence relationship between two feature point sets from a reference and a query graph is of main interest in image processing for identifying and matching a target. The general problem of feature correspondence problem involving pairwise constraints can be modeled into integer quadratic programming problems (IQP). The IQP is NP-hard. Usually an exact optimal solution can only be obtained for very small graphs, not real-case application. Therefore, the focus of IQP research is to design more accurate and faster algorithms to approximate it. Most of the approaches are heuristics, since they do not guaranty optimality. So the parallel implementation on GPU CUDA platform is a good choice for resolving these challenges.

Since we are interested in the three types of image processing problems associated with moving object detection and tracking, one or two order graph matching, and parallel Euclidean matching approach on GPU CUDA platform, we will present detail background

information about the algorithms that address these issues in the following sections. In this thesis, we provide an implementation of object detection and tracking by a new combination of background subtraction and frame difference algorithm. We propose an efficient solution to extract more effective feature points by using Marr-wavelets scale interaction algorithm and local entropy. We propose a combination of state-of-the-art power iteration based methods for graph matching, and also a customized environment in Matlab for sharing data with a GPU algorithm. Finally, we propose a GPU platform for graph matching in geometric plane, that allows testing new proposed parallel algorithms.

Section 2.2 presents existing research methods about detection and target tracking. Mainly used algorithm are background subtraction, frame difference, and optical flow. Section 2.3 presents some basic definitions and classical algorithms about feature points extraction from the literature, such as Harris, SUSAN, SURF, SIFT. Section 2.4 presents general definition of the standard GM problem and the definition of high-order graph matching problem. Section 2.5 introduces the different kinds of applications on power iteration for explaining the current state-of-the-art kernel algorithms in graph matching. Section 2.6 presents the research of convolutional neural network for graph matching. Section 2.7 introduces the usage of the GPU CUDA platform in this field. It reviews the knowledge of some related topics to parallelism processing. It presents some GPU parallel implementations for graph matching from the literature. Section 2.8 concludes this chapter.

2.2 Dense correspondences methods for detection and tracking

We are mainly interested in dense correspondence problem since they represent the most basic expression of graph matching problematic, the image themselves being graph data structures. Moving object detection and tracking [LXM+14, HCC+15, YB18] is an image processing process used to extract moving objects in a sequence of images, usually based on image features such as edges, colors, and textures. Figure 2.1 shows different visual representations of a detection method: rectangle box, shape, and silhouette. For real-time intelligent surveillance [BDYR18, LM18], automated vehicle tracking, personnel tracking, and many other applications, it is undoubtedly an indispensable area of research, not only in 2D motion observed but also in 3D scenes [HCW+19]. Globally, the objective of multi-target tracking is to jointly estimate, at each observation time,

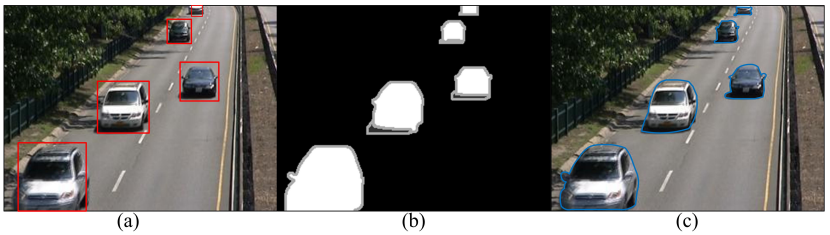


FIGURE 2.1: Different visual representations of the detection: (a) Rectangle box representation approach, (b) Shape representation approach, (c) Silhouette representation approach.

the number of targets and their trajectories from noisy sensor measurements. According to the recent review [LXM+14], multiple object tracking methods could be classified roughly within two classes of Detection-Based Tracking (DBT) and Detection-Free Tracking (DFT). The former includes a detection step of the objects before estimate their trajectories. The latter focuses on the tracking process exclusively, given a predefined initialization.

It is worth noting that DBT allows objects to appear and disappear and has more general applications, whereas its behavior mainly depends on the quality of the detection procedure, which provides observations for the tracking operations as trajectory computation. In this thesis, we are interested in the detection phase only. Globally, some of the most popular methods for shape detection are optical flow method [TXZ+19], background subtraction (BS) method [ZL10, ZZL+17], and frame differential (FD) method [ZWQ12]. They can be considered as the most simple and straightforward methods generally used for real-time detection. Among them, the shape representation method can most accurately reflect the shape of the object, delimit the outline and completely fill the interior of the object.

Optical flow method [TXZ+19] estimates the displacement field between two images, so it not only needs to locate the position of each pixel accurately but also needs to find the corresponding points between two input images [DFI+15]. That is to say, optical flow method has a relatively high computational complexity. Therefore it spends more time than other methods, so it is more complicated since it computes a dense optical flow field. As a widely used target extraction technique, the background subtraction method can extract objects with a relatively simple algorithm. Although it is relatively easy to implement, it is sensitive to the changes of the light [NJ08]. The frame difference method is still one of the fundamental techniques in computer vision. Frame difference method has the advantage of a small amount of calculation, but it is sensitive to the noise [ZDX+07] and sometimes it seems to appear the empty phenomenon, that consists of some small apertures and gaps, so its results are not accurate enough. Although there are numerous difficulties with frame difference and background subtraction, these

problems are under addressing by some improved methods in recent pieces of literature on the field.

A new inter-frame difference algorithm combined with background subtraction for object tracking is put forward by Muyun Weng [WHD10], this algorithm not only has a low cost-time but also has stronger validity and more extensive flexibility. In 2013, Liu Gang et al. [GSY+13] proposes an algorithm based on the traditional three-frame differential method combined with the Canny edge detection algorithm. In 2014, Lavanya [Lav14] comes up with a method to detect motion. A video monitoring and a more robust detection system was thus developed with the adjustment of camera movements. Similar to this, Hongkun Liu [LDW+16] demonstrates an approach combining background subtraction and three-frame difference to applied to underwater robots to execute underwater missions and detect a moving object by using underwater video, but without affected by the change of lighting condition and the sensitive scenes.

In this thesis, we propose a new combination of background subtraction and frame difference methods. We evaluate its complexity and show that the methods are suitable for real-time applications. Computation is based on local filtering and generally, these methods already have GPU parallel implementations available. However, one key assumption is constancy of the background (fixed camera) and also brightness constancy hypothesis for optical flow algorithm. Here, we are also looking to more general graph correspondence problem, relaxing these assumptions with cluttered context, luminosity change, and large deformations, but rather searching for sparse correspondences.

2.3 Problematic of feature point extraction

Image matching is used to determine the geometric alignment of two or more images of the same scene taken by the same or different sensors from different viewpoints at the same or different times. We can distinguish dense correspondence, that determines correspondences at the pixel level, and sparse correspondence, that determines correspondences between a sparse set of higher level features being first extracting from images. Most of the time such features represent invariant information at some location in the image, as corners, edges, gradients. Since we are interested on sparse correspondence, we study the standard methods for automatic extraction of the feature point sets from images. This becomes a critical step since it should avoid the presence of outliers and should allow to easily discriminate objects. In this thesis, we propose a method for feature extraction step suitable to first-order matching.

Local feature descriptors, that is, providing detail feature detection and feature description information, play a fundamental and vital role in the process of feature correspondence, directly affecting the accuracy and objective score of graph matching. High-quality local feature descriptors describe key points with uniqueness, repeatability, accuracy, compactness, and effective representation. These key points can keep robust and constant in terms of scaling, rotation, affine transformation, illumination, and occlusion [LZL+18]. This section focuses on the theoretical and mathematical descriptions of various local feature descriptors.

Over the years, many researchers have made outstanding contributions to local feature descriptors in areas such as image matching and recognition. Below we display some of the critical extraction algorithms for feature descriptors:

- *Harris*: Harris and Stephens [HS88] propose a combined corner and edge detector based on the local auto-correlation function. Although Harris detector is the simplest and commonly used method for feature point detection, it cannot meet the requirements for extracting more qualified vital points in terms of eligibility and repeatability for images with scale changes and large rotations. This is due to the lack of sufficient consideration of discriminative information for key points detected by the Harris detector.

- *Harris-Laplace*: Mikolajczyk [MS04] presented a scale-invariant Harris-Laplace detector that combines the Harris detector with the Laplacian-based scale selection. It uses the scale-adapted Harris function to localize points in scale-space and selects the points to attain a maximum over scale by using the Laplacian-of-Gaussian function. This algorithm can automatically adjust the position, scale, and shape of point neighborhood to acquire affine invariant points with better repeatability and accuracy.

- *SUSAN*: Smith and Brady presented the Smallest Univalue Segment Assimilating Nucleus (SUSAN) detector [SB97] for low-level image processing. This method can quickly and accurately find the edges, lines, corners, and junctions of the image, what more, SUSAN has an excellent noise suppression ability than other methods.

- *SIFT*: One of the most widely used descriptors Scale Invariant Feature Transform (SIFT) was developed by Lowe [Low99]. This method converts the image into a large number of local feature vectors. Each local feature vector is invariant to image translation, scaling, and rotation, and is invariant to illumination changes and affine or 3D projection parts. SIFT algorithm mainly includes four steps: (a)

TABLE 2.1: Summary table of feature detectors.

Feature detector	Invariance			Qualities			
	Rotation	Scale	Affine	Repeatability	Localization	Robustness	Efficiency
Harris	Yes			+++	+++	+++	++
Harris-Laplace	Yes	Yes		+++	+++	++	+
SUSAN	Yes			++	++	++	+++
SIFT	Yes	Yes		++	+++	+++	++
SURF	Yes	Yes		++	+++	++	+++

Use the Difference of Gaussian (DoG) to determine the extreme value of the scale-space; (b) Keypoints localization; (c) Direction distribution of key points based on local image gradient; (d) The keypoint descriptor is a local feature descriptor.

- *SURF*: A fast and performant scale and rotation invariant interest point detector Speeded-Up Robust Features (SURF) was developed by Herbert Bay [BETVG08]. Based on sums of Haar wavelet components, SURF outperforms some state-of-the-art algorithms, especially in terms of speed, it can almost realize real-time computation without loss in performance. SURF is an effective implementation of SIFT, which calculates image derivatives and quantifies gradient directions in a small number of histogram tiles by applying integral images.

A summary of these feature detectors described above is given in Table 2.1. It includes two assessment aspects: invariance and qualities. Globally, all these methods have efficient implementations, we assume that they already have efficient GPU parallel implementations. Then, once the feature point sets available, we are now interested in the matching process itself to find the true correspondences between the two images. This process is modeled as the graph matching problem.

2.4 General formulation of graph matching problem

This section introduces the general representation of traditional graph matching and the definition of the high-order graph matching problem. The main purpose is on finding the correspondence one-to-one mapping between two feature sets from two image sources. The goal is to maximize a function score among the set of correspondence pairs. In first order matching, only local attribute descriptors are considered and evaluated, whereas in the general case of graph matching, two order potentials between pairs (edges) of features must also be maximized, to established the similarity between edges of features. On the other hand, the high-order GM method considers the invariant geometric information by considering relationship between tuples of feature points. The input feature graph becomes a hyper-graph, where edges are replaced by hyper-edges, that is subsets of k points, with the order $k \geq 2$, rather than only considering couples of points.

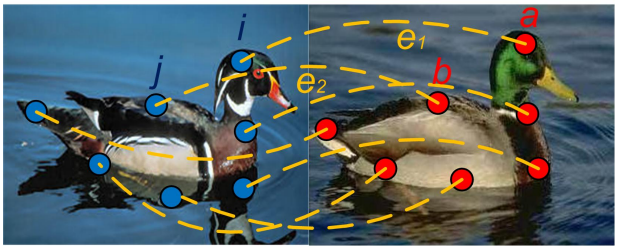


FIGURE 2.2: Feature point correspondence mapping.

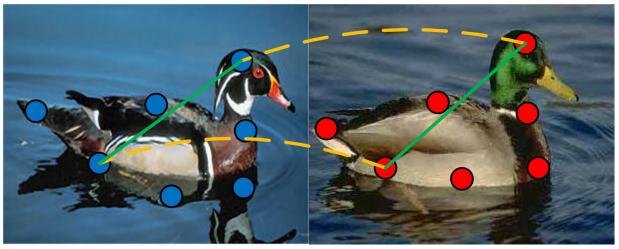


FIGURE 2.3: Euclidean distance as two order similarity measure.

2.4.1 Definition of standard graph matching problem

Supposed it is given a pair of graphs $G_P = (P, E^P)$ with N_P feature points for the reference graph G_P, and $G_Q = (Q, E^Q)$ with N_Q feature points for the query graph G_Q. P and Q are the two sets of feature points, and E^P and E^Q denote edge sets. We note $i, j \in P$ and $a, b \in Q$ as representing feature points. Therefore, the main problem is to find a suitable one-to-one mapping from one feature set to the other feature set as illustrated in the Figure 2.2. The pictures in Figure 2.2 are from PF-WILLOW dataset[1].

Finding a mapping form P to Q can be equivalent to find an $N_P \times N_Q$ assignment matrix X, such that $X_{ia} = 1$ when point i is assigned to point a, and $X_{ia} = 0$ otherwise. Therefore, a one-to-one admissible solution must verify the following constraints in 2.1, that requires a binary solution, and 2.2 and 2.3, that express the two-ways constraints of a one-to-one mapping.

$$X \in \{0,1\}^{N_P \times N_Q} \tag{2.1}$$

$$\forall i \sum_{a=1}^{N_P} X_{ia} \leq 1 \tag{2.2}$$

$$\forall a \sum_{i=1}^{N_Q} X_{ia} \leq 1 \tag{2.3}$$

[1] *https://www.di.ens.fr/willow/research/proposalflow/*

Then, the problem of graph matching can be formulated as the maximization of the following general objective score function 2.4:

$$score(X) = \sum_{ia,jb} H_{ia,jb} X_{ia} X_{jb}, \tag{2.4}$$

where $H_{ia,jb}$ means the similarity or affinity measurement corresponding to the tuple of feature points i, j and a, b. The higher is the score $H_{ia,jb}$, the higher are the similarities between the two corresponding edges (i, j) and (a, b). The product $X_{ia}X_{jb}$ is equal to 1 if and only if points i, j are respectively mapped to points a, b.

Then, we need to know how to compute such a positive and symmetric similarity matrix H. Many cost functions may be used to compute affinity matrices for first-order and second-order GM. Note that $H_{ia,ia}$ represents first-order similarity term, between local attributes of points $i \in P$ and $a \in Q$. For example, the authors in [Yan10, SS13] use the normalized cross-correlation (NCC) cost function, as we have used to validate our feature point extraction method in this thesis. But any point-to-point distance function can be used, as euclidean distance between SIFT descriptors, or sum of squared error data terms.

Duchenne et al. in [DBKP11] propose a general formula to compute the second-order affinity term $H_{ia,jb}$ as shown in 2.5, where f is a feature vector associated to each edge.

$$\forall ia, jb \; H_{ia,jb} = exp(-\gamma \| f_{i,j} - f_{a,b} \|^2) \tag{2.5}$$

Leordeanu et al. in [LH05], and as most often encountered, computes the Euclidean distance between the corresponding candidate point pairs i, j and a, b, to build the affinity term $H_{ia,jb}$, as shown in equation 2.6. Here, σ_d is the sensitive controller of the deformation.

$$H(ia, jb) = \begin{cases} 4.5 - \dfrac{(d_{ij} - d_{ab})^2}{2\sigma_d^2} & if |d_{ij} - d_{ab}| < 3\sigma_d \\ 0 & otherwise \end{cases} \tag{2.6}$$

2.4.2 Definition of high-order graph matching problem

We generalize the previous notations. The purpose of high order graph matching problem [LCL11] is to establish a mapping between the nodes of two hyper-graphs $G_P = (\mathcal{P}, \mathcal{E}^{\mathcal{P}})$ and $G_Q = (\mathcal{Q}, \mathcal{E}^{\mathcal{Q}})$, where $\mathcal{E}$ represents k order hyper-edges, which are k-tuples of feature points, as for example $(i_1, i_2, ..., i_k) \in P^k$ and $(j_1, j_2, ..., j_k) \in Q^k$ are two hyper-edges

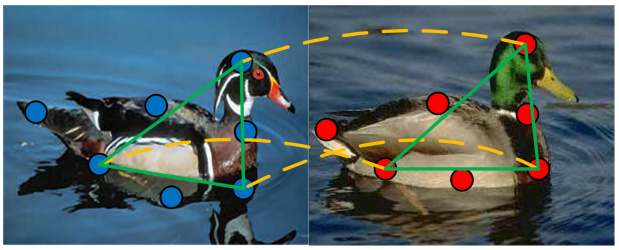


FIGURE 2.4: Similarity measure between triangles in high-order GM.

of $\mathcal{E}^{\mathcal{P}}$ and $\mathcal{E}^{\mathcal{Q}}$ respectively. As previously N_P and N_Q stand for feature point sizes. Figure 2.4 shows an illustration of third-order ($k = 3$) graph matching.

The goal of hyper-graph matching is to find a binary assignment matrix $X \in \{0,1\}^{N_P \times N_Q}$ that maximizes the following k-order score function:

$$score(X) = \sum_{i_1 j_1, \ldots, i_k j_k} H_{i_1 j_1, \ldots, i_k j_k} X_{i_1 j_1} \ldots X_{i_k j_k}, \tag{2.7}$$

$$s.t.\ X \in \{0,1\}^{N_P \times N_Q}, \tag{2.8}$$

$$\forall i \sum_{j=1}^{N_P} X_{i,j} \leq 1, \forall j \sum_{i=1}^{N_Q} X_{i,j} \leq 1. \tag{2.9}$$

The binary constraint and the two-way constraint formulas are expressed in 2.8 and 2.9 for one-to-one mapping from G_P to G_Q.

In practice third-order graph matching is addressed where $k = 3$. Then, for the construction of three-order affinity term $H_{ia,jb,kc}$, we can refer to Duchenne [DBKP11] in which a generic truncated Gaussian kernel is constructed as follows:

$$\forall ia, jb, kc\ H_{ia,jb,kc} = exp(-\gamma \| f_{i,j,k} - f_{a,b,c} \|^2) \tag{2.10}$$

where $f_{i,j,k}$ is the feature vector describing the tuple (i, j, k).

The third order term $H_{ia,jb,kc}$ is often based on angles among triplets as demonstrated in [LCL11]. Figure 2.5 shows a sample of two triangles which are compared by their respective angles as in the equation 2.11 :

$$H_{\omega_1,\omega_2,\omega_3} = exp\{-\frac{1}{\sigma\, s} \Sigma_{k=1}^{3} |sin(\theta_{\omega_k}^{G_P}) - sin(\theta_{\omega_k}^{G_Q})|\}, \tag{2.11}$$

where $\theta_{\omega_k}^{G_P}$ and $\theta_{\omega_k}^{G_Q}$ represent the angles of the respective triangles with index ω_k between the query graph G_P and the reference graph G_Q, respectively.

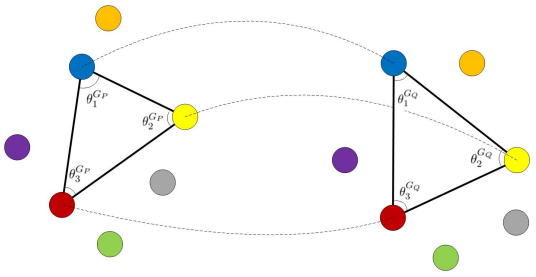


FIGURE 2.5: Two triangles compared by their angles.

2.5 State-of-the-art methods to graph matching

2.5.1 Power iteration and spectral matching

Since the graph matching based on IQP is an NP-hard problem, various approximate solutions are used to attempt to solve the pairwise similarity corresponding mapping problem. Leordeanu and Hebert provided a spectral matching (SM) algorithm [LH05] based on the main strength cluster of the adjacency matrix by finding its principal eigenvector. Cour et al. [CSS07] present a new spectral relaxation technique named spectral matching with affine constraint (SMAC). It includes a normalization procedure which matches the scoring capabilities of existing graphics to improve matching accuracy significantly. Zass and Shashua [ZS08] interpreted hyper-graph matching (HGM) algorithm, where a hypergraph represents the complex relationship. Leordeanu et al. [LHS09] solved the matching problem by using integer projected fixed point (IPFP) algorithm, which found a discrete solution with climbing and convergence properties. Reweighted random walks for graph matching (RRWM) algorithm was introduced by Cho et al. [CLL10], and it combined mapping constraints with re-weighted jumping schemes.

Al of these methods more or less implement an iteration loop to compute a principal eigenvector. The basic technique being power iteration method. Although this method cannot guarantee to achieve the final global optimal solution during the operation, it can converge to the fixed point of the tensor [RK00]. The power iteration idea has already been used in different graph matching algorithm, such as SM [LH05], RRWM [CLL10], Tensor based high-order GM [DBKP11], and be extented by Max-pooling strategy for GM [CSDP14]. The pseudo-code is given in in Algorithm 2.1.

In [LH05], Leordeanu presented spectral matching (SM) for correspondence problems, which is one of the most basic methods for GM. The SM approach first compute the principal eigenvector by power iteration. Then, it applies a greedy procedure to transform

Algorithm 2.1 Power iteration for eigenvalue problem

Input: matrix H
Output: V main eigenvector of H
1: initialize V randomly;
2: **repeat**
3: $V \leftarrow HV$;
4: $V \leftarrow \frac{1}{\|V\|_2} V$;
5: **until** convergence

the real value vector solution to a binary vector that satisfies the one-to-one constraint mapping. The detail steps of this procedure are shown in Algorithm 2.2.

Algorithm 2.2 Spectral technique for correspondence problem (SM)

1: Build the affinity matrix M;
2: Set x^* be the principal eigenvector of M; Initialize the solution vector x with the $n \times 1$ zero vector, and set all candidate assignments in L;
3: Find $e^* = argmax_{e \subset L}(x^*(e))$, if $x^*(e^*) = 0$, stop and return the solution x. Otherwise set $x(e^*) = 1$ and remove e^* from L;
4: Based on one-to-one corresponding constraints, remove from L all potential assignments in conflict with $e^* = (i, a)$;
5: If L is empty return the solution x. Otherwise go back to step 3.

2.5.2 Reweighted random walk matching

In [CLL10], Cho et al. proposed a reweighted random walk graph matching algorithm (RRWM), which can efficiently calculate quasi-stationary distribution using an adaptation of the power iteration method as outlined in Algorithm 2.3, where M is the affinity matrix, α is a reweighting factor, and β is the inflation factor. This algorithm proposed an affinity-preserving transformation of M (steps 1 to 3), an inflation step (step 8), and a bistochastic normalization (steps 9 to 12). The main operation is $x^T = \alpha \bar{x}^T + (1-\alpha) y^T$ at step 15. Note that final step 18 greedily generates the final binary vector satisfying the one-to-one mapping constraints. The algorithm mainly operates in continuous space, whereas discretization appears as a final step only.

2.5.3 Max-pooling matching

In [CSDP14], Cho et al. proposed a max-pooling graph matching strategy, that has been improved very recently by the work of [Ref on the recent 2018 paper]. Max pooling consists in updating the affinity of a correspondence pair by only considering the neighboring correspondence pairs with maximum affinity, instead of considering a summation, or average, of the neighboring correspondence pairs as in power iteration method. By only

Algorithm 2.3 Reweighted Random Walk Graph Matching (RRWM)

Input: weight matrix M, the reweight factor α, and the inflation factor β
Output: solution x

1: Prevent conflicting walks by setting $M_{ia;jb} = 0$ for all conflicting match pairs
2: Set the maximum degree $d_{max} = max_{ia}\Sigma_{jb}M_{ia;jb}$
3: Initialize the transition matrix $P = M/d_{max}$
4: **repeat**
5: (Affinity-preserving random walking by edges)
6: $\bar{x}^T = x^T P$;
7: (Reweighting with two-way constraints)
8: $y^T = \exp(\beta\bar{x}/max\bar{x})$
9: **repeat**
10: normalize across rows by $y_{ai}/\Sigma_{i=1}^{I} y_{ai}$
11: normalize across columns by $y_{ai}/\Sigma_{a=1}^{A} y_{ai}$
12: **until** y converges
13: $y = y/\Sigma y_{ai}$;
14: (Affinity-preserving random walking with reweighted jumps)
15: $x^T = \alpha\bar{x}^T + (1-\alpha)y^T$;
16: $x = x/\Sigma x_{ai}$;
17: **until** x converges
18: Discretize x by the matching constraints

considering maximum affinity neighbor pairs, this allows to limit the influence of outlier pairs. The algorithm is given by its pseudo-code in Algorithm 2.4. The algorithm computes the max-pooling product $M \circledast x = x_{ia}M_{ia;ia} + \Sigma_{j\in\mathcal{N}_i} max_{b\in\mathcal{N}_a} x_{jb}M_{ia;jb}$(as shown in step 4) instead of the sum-pooling product Mx, since it is conjectured that sum-pooling will often results in bad local minimum. The main loop is similar to power iteration, but max-pooling implies specific implementation for computing the max-pooling product. The entire power iteration continues until the solution x converges.

Algorithm 2.4 Max-pooling matching (MPM)

Input: affinity matrix M
Output: soft-assignment x

1: Initialize the starting assignment x as uniform;
2: **repeat**
3: **for** each candidate match (i, a) **do**
4: $x_{ia} \leftarrow x_{ia}M_{ia;ia} + \Sigma_{j\in\mathcal{N}_i} max_{b\in\mathcal{N}_a} x_{jb}M_{ia;jb}$;
5: $x \leftarrow \frac{1}{\|x\|_2}x$;
6: **end for**
7: **until** x convergence;
8: Discretize the solution x if needed.

Algorithm 2.5 Integer projected fixed point method for GM (IPFP)

Input: the number of iterations t, any initial solution x_0

$k = 0$;
$x^* = x_0$;
$S^* = (x^*)^T M x^*$, where $x_i \geqslant 0$ and $x \neq 0$;
for $k \leq t$ **do**
    $x_k \leftarrow x^*$;
    $b_{k+1} = P_d(Mx_k)$; // The projection step
    $C = x_k^T M(b_{k+1} - x_k)$;
    $D = (b_{k+1} - x_k)^T M(b_{k+1} - x_k)$;
    $r = min\{-C/D, 1\}$; // The discriminant condition
    $x_{k+1} = x_k + r(b_{k+1} - x_k)$; // Maximize the original quadratic score
    if $b_{k+1}^T M b_{k+1} \geq S^*$ **then**
        $S^* = b_{k+1}^T M y b_{k+1}$;
        $x^* = b_{k+1}$; // Update the optimal solution
    end if
    if $x_{k+1} = x_k$ **then**
        return x^*;
    end if
    $k = k + 1$;
end for
return x^*

2.5.4 Integer projected fixed point method

Leordeanu [LHS09] proposed an integer projected fixed point (IPFP) algorithm for GM. Most of the algorithms seek a good approximate solution by relaxing the integer one-to-one constraint, in order to be able to find an optimal global value of the new problem effectively. However, IPFP is mainly based on searching for discrete solutions and maximize the quadratic score in the discrete domain.

The iterative algorithm can take any continuous or discrete solution given by any other method as input, thereby continuously improving it. Each iteration includes two phases related to the Frank-Wolfe (FW) algorithm [JCC Ref]. The first phase in the discrete domain maximizes the linear approximation of the quadratic function around the current solution, which is equivalent to giving a direction. The second phase maximizes the original quadratic score in the continuous domain along this direction. Even if a non-discrete solution is found in the second phase, due to the influence of the first phase, the optimization direction is finally towards the integer solution. The core of the algorithm is always improving the quadratic score in the continuous domain to converge to the maximum value eventually. In the case of convex quadratic functions, the solution for each iteration is always discrete, and the algorithm converges within a limited number of steps. For non-convex quadratic functions, this method tends to approximate the discrete solution and returns the best discrete solution encountered along the path.

The detail of the IPFP can be found in Algorithm 2.5. First, it takes any initial solution x_0 (continuous or discrete) as input x^*, and set the initial quadratic score as $S^* = x_0^T M x_0$. Then the projection P_d is used to find a discrete solution b_{k+1} around the current solution x_k. This discrete solution b_{k+1} can not only maximize the dot product of Mx_k in the continuous domain but also provide the largest possible increase direction in a first order approximation. The effect of step 3 is to ensure that the quadratic score will increase with each iteration. Step 4 ensures that the returned binary solution b_{k+1} is not worse than the initial solution x^* under the condition $b_{k+1}^T M b_{k+1} \geqslant S^*$.

2.6 Convolutional neural network for graph matching

Convolutional neural network was mainly used for digital and document recognition [LBB+98] and small-scale target recognition in previous research [KH+09]. Recently, some of the authors come up with GM algorithms by using CNN. Lin Ma proposed semantic matching for images and sentences based on multimodal CNN [MLSL15], which took the image and sentence as input. The algorithm relies on a convolutional structure to combine different semantic segments of a sentence, and learns the different degrees of interaction between images and constituent segments so that the matching relationship between modalities can be fully utilized. In 2017, Ufer et al. [UO17] presented another semantic feature matching algorithm with pre-trained CNN features. CNN is mainly used as an activation guided feature selection based on convolution feature pyramids. The same year, a geometric matching based on CNN architecture was proposed by Rocco et al. [RAS17]. They use a standard CNN architecture in the first step of feature extraction. For the matching phase, a similar process is also applied to the matching layer, where the correlation layer is used, and the normalization step followed. The final matching network is passed through a regression network, which outputs the parameters of the geometric transformation. The output matching network map went through a regression network for exporting geometric transformation parameters. The whole architecture mimics the standard matching process by basing on the traditional approach. Convolutional Neural Networks (CNN) and Fully Convolutional Networks (FCN) become more and more competitive [WGY17], with a large scope of applications, but they necessitate supervised learning using a huge amount of ground-truth information to learn the network. In this thesis, we address graph matching by optimization techniques only.

2.7 GPU CUDA platform and graph matching

2.7.1 Generality on parallel processing

From the perspective of multiple processors, parallel processing is an operation where each processor executes a single task on it data in parallel. Parallel operation includes task-based and data-based parallel processing.

- *Task-based parallel processing* This parallel mode divides the computing task into several small but different tasks. Some computing units are responsible for fetching, and some computing units are responsible for computing. Such a humongous task can form a pipeline.
- *Data-based parallel processing* This parallel mode decomposes the data into multiple parts, and allows multiple arithmetic units to calculate these small pieces of data, and then aggregate them.

Generally speaking, multi-threaded programming of the CPU favors the first parallel mode, and parallel programming model of the GPU favors the second. Many parallel approaches have been introduced, such as [Nak12], [SPS12], [GHOI14], and [JDJ19]. The advantages of parallel operations are: Threads have less overhead than processes; Resources can be shared between threads; Make full use of server hardware resources; Improve service throughput and reduce response time; Contention and scalability of resources. Multi-threaded concurrent memory consumption is relatively small.

2.7.2 GPU CUDA platform

In this thesis, parallel image matching operation will be implemented based on a GPU platform. Nowadays, GPU provides excellent parallelism. A parallel process means that multiple processors perform the same function across the elements of a data set at the same time. NVIDIA offers a family of sustainable GPU products that transparently extend the parallelism of applications to take advantage of increasing processor cores. This automatic scalability is achieved through the Compute Unified Device Architecture (CUDA) scalable programming model. CUDA is defined as a universal parallel computing platform and programming model that utilizes the parallel computing engine in NVIDIA GPUs. NVIDIA GPUs provide a highly parallel, multi-threaded, and multi-core environment for parallel computing architecture called SIMT (single instruction, multi-threaded), similar to SIMD (single instruction, multi-data). When a CUDA

program on the host CPU calls a kernel function with an appropriate kernel configuration according to the GPU hardware, each multiprocessor on the GPU will get one or more thread blocks to execute.

CUDA is a development environment for GPU computing. It is a new software and hardware architecture. It can treat the GPU as a parallel data computing device and allocate and manage the calculations performed. In the CUDA architecture, these calculations no longer have to map calculations to graphics APIs (OpenGL and Direct 3D) like the so-called GPGPU architecture in the past, so for developers, the threshold for CUDA development has been dramatically reduced. CUDA's GPU programming language is based on the standard C language, so any user with a C language foundation can develop CUDA applications. In addition to a parallel computing architecture, CUDA is a universal language for the coordinated work of CPU and GPU. CUDA can be divided into a host and a device. The host is the CPU or a computer with a GPU, it reads and writes files to configure memory, or calls GPU resources. The device is the GPU and has independent computing resources. The following Figure 2.6 is the architecture of the CUDA program. The host needs to transfer data to the memory on the device before it can be processed in the device, and the program executed on the device is the kernel function, then the same kernel is executed via the thread, as shown in the orange part below. In the figure, the thread is the smallest unit. Multiple threads can form a block, and multiple blocks can form a grid. Each 32 threads form a group of warps, and all threads in a group of warps execute the same instructions.

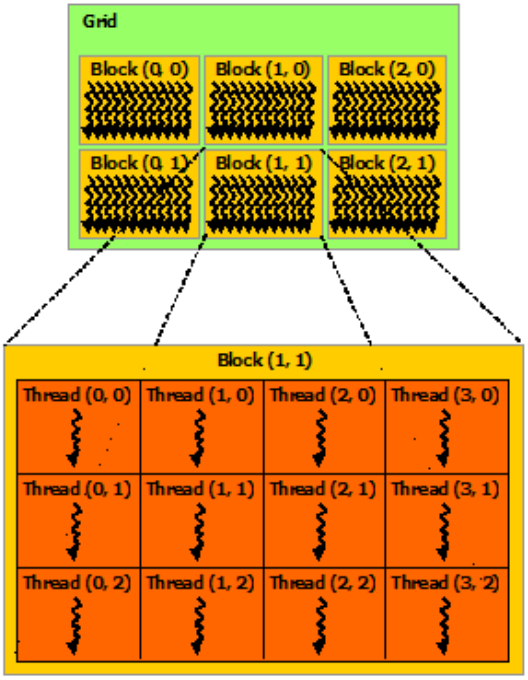


FIGURE 2.6: The architecture of the CUDA program.

Compared with CPU, GPU has relatively less on-chip memory. In terms of parallel computing, GPU and CPU have the following differences:

- **Number of tasks** The CPU is suitable for a relatively small number of tasks, while the GPU is suitable for a large number of tasks.
- **Task complexity** The CPU is suitable for logically complex tasks, while the GPU is suitable for logically relatively simple tasks (which can be described with fewer statements).
- **Thread support** Because the register set of threads in the CPU is standard, when the CPU switches threads, the contents of the thread's registers are saved in RAM, and when the thread starts again, data is restored from RAM to the registers. Each thread in the GPU has its own register set, so its switching speed will be much faster. Of course, the CPU is more powerful for a single thread.
- **Processor allocation principle** The CPU is generally based on the time slice rotation scheduling principle, and each thread executes a single time slice fixedly; the GPU's strategy is to swap in and out quickly when the thread is blocked.
- **Data throughput** Each stream processor in the GPU is equivalent to a CPU core. A GPU generally has often 16 stream processors, and each stream processor can calculate 32 numbers at a time.

2.7.3 Graph matching and related approaches on GPU

The Graphics Processing Units (GPUs) are a dedicated processing unit designed to address bottlenecks caused by graphics applications. Due to the higher transistor density and parallel hardware structure, current GPUs have been extensively studied in the graphics field and the field used as general-purpose processing equipment [KPC+09]. This technology is commonly referred to as GP-GPU (General Purpose Computing on GPU) and is used to ensure real-time performance of processing large amounts of data in applications related to video encoding or computer vision [OLG+07]. In this section, we would like to describe and analyse some implementations of the graph matching algorithms that run on GPU through some recent research literature.

Sinha [SFPG06] introduced a feature tracking and SIFT feature extraction algorithm for video analysis in real-time vision systems based on GPU. The computation was divided between CPU and GPU, the result shown that GPU implementation, with the advantage of raw processing power, is much faster than the optimized CPU version. A bipartite graph matching computation on GPUs had been proposed by Vasconcelos

[VR09]. Garcia [GDNB10] proposed fast GPU-based implementations for feature matching by using the naive brute-force k nearest neighbor (kNN) search algorithm based on API CUDA and CUBLAS. This kNN search algorithm is applied to high-dimensional SIFT matching. Each of the feature points extracted by SIFT in query image should found its corresponding k closest features in the reference image. Then a voting algorithm is used for decision on the GPU platform. This algorithm mainly focuses on SIFT feature extraction in GPU-based implementations but without a clear description of the matching parallelization. Auer [AB12] presented a fine-grained shared-memory parallel algorithm for greedy graph matching with an implementation on the GPU. Due to the GPU's excellent memory bandwidth, the proposed greedy algorithm's performance on GPU is better than in multi-core CPU. However, this approach has no implementation available.

Therefore, to sum up, we did not found GPU parallel algorithms addressing two-order graph matching problem specifically, based on geometric relationships between edges or tupples of feature points. Many GPU programs exist for point to point correspondence based on first order local cost function, whereas high order potentials look not be addressed yet in GPU.

2.8 Conclusion

In this chapter, we have reviewed definitions and literature background related to our proposed solutions to the object detection and tracking and graph matching problems. We have presented some state-of-the-art algorithms for the graph matching problem. Note that overall computation complexity depends of the affinity matrix size or its representation. In the general case, this matrix has $O\left((N_P \times N_Q)^2\right)$ size, that is very large. Hence, strategy must be adopting to reduce the dimensionality. Most of the time, a restricted list of candidate correspondence pairs is pre-computed that allows to reduce the affinity matrix size to $O\left((N_P \times K)^2\right)$, where K is a constant. Using specific data-structures should also becomes necessary to implement sparse graph structures. In this thesis, as a final application, we will address graph matching problematic by GPU implementation based on very low cost data structures with $O(N)$ complexity, N beeing the instance size, without computing a large size affinity matrix.

Chapter 3

Background subtraction and frame difference for multi-object detection

3.1 Introduction

This chapter presents our first application within Matlab environment. Nowadays, target detection and tracking have been extensively studied, and various motion detections can be performed on different objects. In this chapter, we mainly focus on frame difference (FD) and background subtraction (BS) algorithms for object detection based on video sequences. Each frame is a picture, and each video is a sequence of images. The detected objects, called foreground objects, can be moving or stationary objects such as people, birds, vehicles. On the other hand, the background objects are usually static fixed objects. The purpose of this chapter is to represent objects in a video sequence to explicitly perform object detection as a first step of real-time tracking.

Although there are many works on detection and tracking, there seems to be no systematic way to appear today. The overall problem remains an open field with methods having their qualities and limits. With all of this in mind, we restrict our attention to the detection phase, rather than tracking. We propose an improved algorithm that combines many of the standard tools encountered in this setting. The approach mainly integrates frame difference, background subtraction, Laplace filter, and Canny edge detector (called 3FDBS-LC). The method introduces a fusion of information from BS and FD processes and executes the FD based on three frames instead of two as usual. As presented in experiments, this new combination outperforms BS or DF separate implementations while preserving the potential for real-time execution.

The rest of this chapter is organized as follows: Section 3.2 gives detailed explanations of pre-processing and post-processing treatments. In section 3.3, the methodology and procedures for the main approach that we proposed are described. The experimental results are given in section 3.4. In section 3.5, the proposed method is applied to actual scenes with video rate processing. Finally, conclusions are presented and suggestions are made for further research.

3.2 Basic filters and definitions

Image pre-processing and post-processing play an essential role in this research. As we consider the detection phase of objects only, given a sequence of input images, the output of detection is represented as a binary image from which individual connected components directly represent detected objects. This information is the basis for further tracking operations and its quality should impact the rest of the tracking operations. As a result, this binary output must reflect the object shapes with the most fidelity, delimiting contours and adequately filling object interiors. The binary image serves as a result for ground truth evaluation and comparison of different methods, in a qualitative and quantitative ways, to compare the quality of the obtained shape.

Basic processing operations such as color conversion, image binarization, filtering processing, and edge detection are current basic operations in object tracking. Most of these basic tools have straightforward fast implementations and are generally compatible with a real-time context of application. Most of these filters have $O(N)$ time complexity, with N the number of pixels. Also, their parallel implementation in GPU (Graphic Processing Unit) system is now a matter of current fact. We detail the standard processing methods to be combined in the proposed object detection algorithm.

3.2.1 Color to gray scale conversion

RGB comes from the abbreviation of three primary colors red, green, and blue. It is a model in which these three colors are added together in various ways to reproduce a broad array of colors under different weights. On the other hand, the grayscale image is one in which the value of each pixel is a sample representing a kind of light, that is to say, it carries only intensity information, varying from black to white. Since color scale images typically carry much information, when dealing with image, computer needs to read all of it is data information, which will consume more computing time, so it is not conducive to image processing and calculation. Under this situation, it is necessary to convert the color scale image to a grayscale image to increase computational efficiency.

FIGURE 3.1: From the left to the right: (a) original image, (b) gray scale image and (c) binary image.

3.2.2 Binarization of image

Binary images are typically quantized to consist of two possible intensity values, usually 0 and 1, respectively, representing black and white. It is derived from the threshold division of the grayscale image: these pixels with a gray level above the specific threshold are set to 1, and the remaining pixels are set to 0. It means that it will produce an image with a white object on a black background (or vice versa, depending on the specific threshold values), usually used to separate a foreground image from the background image. The grayscale images have a grayscale value ranging from 0 to 255, where 0 is black and 255 is white, while the black and white image has only 0 and 1 values where 0 represents black, 1 represents white. The purpose of image binarization is to speed up the logical decision process when merging information. So binary images can improve recognition efficiency when performing computer recognition. Figure 3.1 displays the original image and its corresponding grayscale image and binary image.

3.2.3 Filtering process

Filter processing is the design and realization of a rejector that satisfies the requirements of image processing. Among different kinds of filters, the most commonly used are Mean filter, Median filter, Gaussian filter, and Laplace filter.

Mean filter is a common linear smoothing algorithm in image processing and noise reduction. The principle of mean filtering is simply like a spatial window sliding filter, which replaces the center value with the average value of all the neighbors' pixel values in the window. The window is usually squared to diminish the point where the pixel value varies significantly between pixels and pixels due to noise. Instead of using the mean value to replace all of the surrounding pixels, the Median filter replaces them with median values. Median filters can reduce not only noise but also protect the edge and other detail information of images. Since the median filter obtains a median value, but without considering the unrepresentative of the surrounding pixels, the median filter is more robust for preserving sharp edges. The calculation formula of the mean filter and

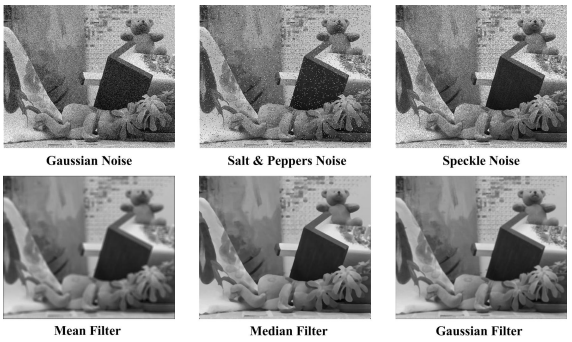


FIGURE 3.2: The first row presents a grayscale image disturbed by Gaussian noise, Salt and Peppers noise, and Speckle noise respectively; the second row presents the Salt and Peppers noise image through different filters: Gaussian filter, Median filter and Mean filter.

median filter are defined as below:

$$Mean(x, y) = \sum M(x, y)/n, \tag{3.1}$$

$$Median(x, y) = med(M_1, M_2, \cdots, M_n), \tag{3.2}$$

where n is the number of pixels, M is the value of each pixel. Gaussian filter is considered as an ideal time-domain filter whose impulse response is a Gaussian function. The effect of Gaussian smoothing is to blur the image like the mean filter. The Gaussian standard deviation determines the degree of smoothness. The higher standard deviation is, the larger convolution kernels will be. Gaussian outputs a weighted average of the neighborhood of each pixel, with the average weighting being more toward the value of the center pixel. This is in contrast to the uniform weighted average of the mean filter. Because of this, Gaussian provides milder smoothness and retains edges better than the mean filter of the same size. Gaussian operator is defined as

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.3}$$

where x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. Figure 3.2 shows a grayscale image disturbed by Gaussian noise, Salt and Peppers noise, and Speckle noise, respectively. Among them, we focus on the middle one to show how to remove salt and pepper noise from an image using the mean filter, median filter, and Gaussian filter. With mean filtering, even though the noise interference can be eliminated, it is not as good as a median filter in preserving edge information. Compared with these two filters, however, the Gaussian filter is better able to remove noise without improving the sharpness of the image.

Laplacian is the second-order derivative of the Gaussian equation. Compared with the first-order differential, the second-order differential has stronger edge localization capability and a better sharpening effect. Unlike a Gaussian filter that can blur an image, the effect of image sharpening is to enhance the gray contrast and make the blurred image clearer. Because Laplacian is a differential operator, its application can enhance the region of grayscale mutation in the image and weaken the slowly changing region of the grayscale. Therefore, the sharpening process may choose Laplacian to process the original image to generate an image that describes the abrupt grayscale change. Finally, the Laplacian image is superimposed with the original image to produce a sharpened image. The primary method of Laplacian sharpening can be expressed as:

$$\frac{\partial}{\partial x}G_\sigma(x,y) = \frac{\partial}{\partial x}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

$$\frac{\partial^2}{\partial x^2}G_\sigma(x,y) = \frac{x^2}{\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}} - \frac{1}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.5}$$

$$\nabla^2 G_\sigma(x,y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}, \tag{3.6}$$

where x, y are the pixel coordinates, and σ is the standard deviation of the Gaussian distribution. One proposal in this application is to integrate the Laplacian filter into the combined BS/FD tracking method. Laplacian filter will be adopted, which not only produces sharpening effects but also preserves background information. The gray value in the image can be preserved, and more details are highlighted.

3.2.4 Edge detection

Edge detection is an image processing technique used to find the boundaries of objects within an image. There are many different types of edge detection operations. Commonly used edge detection algorithms include the Sobel, Prewitt, Roberts, and Canny methods.

Sobel operator formed by a pair of 3×3 convolution kernels, one of the kernels is generated from 90° rotation of another. It is used on 2-D spatial gradient measurement to calculate the approximation of the gradient function for image intensity equation, acquiring the high spatial frequency domain of the corresponding edge. Prewitt operator has a very similar derivate mask as a Sobel operator, and it is formed by a pair of 3×3 convolution kernels. Prewitt operator can also be called as derivative operators or derivative masks. It is based on the convolution of the image with a small separable and

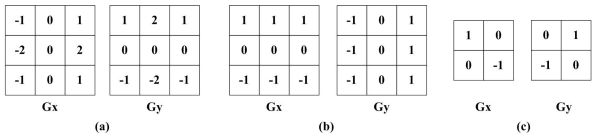


FIGURE 3.3: The horizontal and vertical convolution kernels of (a) Sobel operator, (b) Prewitt operator and (c) Roberts operator.

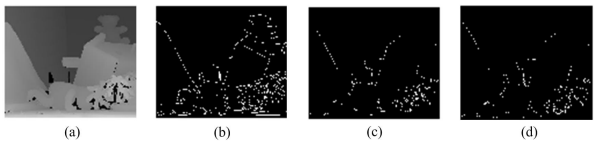


FIGURE 3.4: (a) Original image and its corresponding processed image: (b) Canny operator, (c) Prewitt operator and (d) Roberts operator.

integer-valued filter in the horizontal and vertical directions. These two operators can be used in vertical direction and horizontal direction. Nevertheless, the coefficient of the derivate mask of the Sobel operator can be adjusted flexibly according to algorithm requirements. Roberts's operator is fast and easy to implement. The operator formed by a pair of 2×2 convolution kernels. The principle of Roberts operator is achieved by computing the sum of the squares of the neighbor pixels to approximate the gradient value of an image. Figure 3.3 shows these three kinds of operators' horizontal and vertical convolution kernels.

Robert operator can locate the target accurately, but it is less sensitive to noise because it is not smooth. The Prewitt operator and the Sobel operator belong to the first-order differential operator, the former is the averaging filter, and the latter is the weighted averaging filter. They are good at detecting grayscale in low noise images, but they do not perform well with images under complex noise. Canny edge operator is more accurate than Sobel, Prewitt, and Roberts operators. From Figure 3.4, we can see that the Canny edge detector can more completely discover the edge information of the image, so it performs better than other operators. In this work, the Canny edge detector is adopted.

3.2.5 Morphological transform

Morphology processing is an operation which displays a specific structural element in an input image and generates the desired output image. The function of morphological processing is to eliminate interferences, fill small apertures, and smooth boundary. The most fundamental morphological operators are erosion and dilation. Erosion operation can eliminate holes or noise, but also reduce the size of the affected area. Dilation

operation can fill some of the gaps in the moving target area; it also can inflate the edge pixels of a moving object.

Opening operation is defined as carrying out the erosion operation first, then performing dilation operation by using the same structural element. It can be expressed by the following formula:

$$Dst1 = open\,(src, elem) = dilate\,(erode\,(src, elem))\,, \tag{3.7}$$

where $Dst1$ represents the result of the final operation, scr stands for the object X and $elem$ denotes the structural element S. Opening operations can eliminate tiny objects, separate the objects at subtle joints, smooth the boundaries of large objects, but without significantly altering the area of the object. On the contrary, the following closing operation

$$Dst2 = close\,(src, elem) = erode\,(dilate\,(src, elem)) \tag{3.8}$$

is defined as the dilation operation followed by the erosion operation. The closing operation can fill some of the small gaps in the moving target area, connect the objects closer to each other, smooth the boundary of the target, and keep the size of target unchanged. In this algorithm, the closing operation will be employed. Therefore, after performing such closing morphological processing on the binary image, the small apertures are filled, and the small gaps are connected.

3.3 Proposed object detection algorithm

After image conversion and Laplace filter processing, the most critical parts we propose now are the frame difference method, the background subtraction method, and a combination with edge detection. Standard approaches and our proposed new combination are presented in this section. Considering the disadvantages of frame difference and background subtraction, which are easy to disturb by the sensitivity of noise and brightness, adding edge detection occupies a significant role, because of its independence with the external influence. To get better precision on edge width, we use the Canny detector, which is one of the most accurate edge detection methods. Then, we respectively present the edge detector, the BF and BS methods separately, and our new combination method called 3FDBS-LC, in the following sections.

3.3.1 Canny edge detector

Canny edge detector is came up with John F. Canny in 1986. The Canny algorithm is designed to meet three main criteria: low error rate, good localization, and mark uniqueness. Owing to its optimality to meet with the three criteria, the canny operator experienced a multi-stage process:

a) Use a Gaussian filter to smooth the image and filter out the noise. In order to minimize the impact of noise on edge detection, noise must be filtered out to prevent false detection. The Gaussian convolution kernel H of size $(2k+1) \times (2k+1)$ is given below:

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp\{-\frac{(i-(k+1))^2 + ((j-(k+1))^2}{2\sigma^2}\}. \tag{3.1}$$

The size of the Gaussian convolution kernel can affect the performance of the Canny detector. The larger the size is, the lower the sensitivity of the detector to noise will be. General 5×5 is a relatively good trade-off.

b) Calculate the gradient intensity and direction of each pixel in the image. The edges in the image can point at all directions, so the Canny algorithm uses multiple operators to detect the image. The gradient intensity value G and direction θ are defined in

$$G_x = S_x * W_S \tag{3.2}$$

$$G_y = S_y * W_S \tag{3.3}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.4}$$

$$\theta = arctan(G_y/G_x), \tag{3.5}$$

where W_S is the size of window, S_x denotes the operator in the x direction for detecting the edge in the y direction, S_y denotes the operator in the y direction for detecting the edge in the x direction. G_x and G_y represent the gradient values in the x and y directions, respectively.

c) Apply non-maximum suppression to eliminate spurious response from edge detection. Non-maximum suppression is an edge-sparse technique that helps to suppress all gradient values outside the local maximum to zero. As shown below, the gradient is divided into eight directions, namely E, NE, N, NW, W, SW, S, SE. The gradient direction of the

pixel P is θ, then the gradient linear interpolation G_{P1} and G_{P2} of the pixels $P1$ and $P2$ are defined as follows:

$$tan\theta = G_y/G_x \tag{3.6}$$

$$G_{P1} = (1 - tan\theta) \times E + tan\theta \times NE \tag{3.7}$$

$$G_{P2} = (1 - tan\theta) \times W + tan\theta \times SW. \tag{3.8}$$

d) Use double-threshold detection to determine the true and potential edges.

e) Finish the edge detection by suppressing the isolated weak edges.

The detail pseudo-code description for the following three steps is presented in Algorithm 3.1.

Algorithm 3.1 Canny edge detection algorithm.

Input: The gradient linear interpolation G_P, G_{P1}, G_{P2}

function NON-MAXIMUM SUPPRESSION(G_P, G_{P1}, G_{P2})
  if $G_P \geq G_{P1}, G_P \geq G_{P2}$ **then**
    $G_P \rightarrow edge$
  else
    $G_P \rightarrow Suppressed$
  end if
end function
function DOUBLE-THRESHOLD(G_P)
  if $G_P \geq HighThreshold$ **then**
    $G_P \rightarrow StrongEdge$
  end if
  if $LowThreshold \leq G_P \leq HighThreshold$ **then**
    $G_P \rightarrow WeakEdge$
  end if
  if $G_P \leq LowThreshold$ **then**
    $G_P \rightarrow Suppressed$
  end if
end function
function SUPPRESS ISOLATED LOW THRESHOLD POINTS(G_P)
  if $G_P == LowThreshold$ **then**
    $G_P \rightarrow StrongEdge$
  else
    $G_P \rightarrow Suppressed$
  end if
end function

3.3.2 Frame differencing method

The frame difference method can be implemented on a series of consecutive images. Gray values and gradient vectors are used to determine information for moving objects.

The method calculates the difference between two consecutive images by comparing the point-by-point gray values to obtain a frame difference image. The formula for the difference between two frames can be written as

$$D_k(x, y) = |f_k(x, y) - f_{k-1}(x, y)|, \tag{3.9}$$

where the current frame image gray value is f_k, the adjacent frame image gray values is f_{k-1}, and D_k is image after difference between f_k and f_{k-1}. We define R_k as the binary conversion of the difference image. If $D_k(x, y) > T$, $R_k(x, y)$ belongs to foreground and set to 1, on the contrary, it belongs to the background and it will be set to 0, where T is a fixed empirical threshold.

The disadvantages of the difference between the two frames are the generation of foreground aperture and ghosting problems. In contrast, the three-frame difference method can better weaken this problem. This is achieved by subtracting the current frame image with the previous frame and the subsequent frame, respectively. After that, a logical OR operation is performed based on these results, as done by Yanzhu Zhang [ZWQ12]. Here, we will analyze it in detail and name it the traditional frame difference method (FD). Besides, when dealing with a complex scene, the edge information of the moving target is easily affected by the background scene. This edge information cannot be extracted entirely. Conversely, Canny edge detection is good at getting the edge information of an object. Therefore, three frame differences can be combined with Canny edge detection.

3.3.3 Background subtraction method

The principle of background subtraction is to subtract the background image from the current frame using difference computation. The process can be divided into the following two steps. First, the current frame image K_{th} and the background image are obtained from the video sequence. Second, a difference calculation is performed between the current frame image and the latest background image to obtain a frame difference image. Lijing Zhang [ZL10] uses this background subtraction (BS) method with morphological filtering and contour projection analysis as post-processing. However, due to noise, shadows, and light interference, the results of the difference may be irrelevant. The challenge is to propose a background optimization method that can filter these unavoidable disturbances while correctly detecting moving objects. Therefore, an improved background subtraction method has been proposed in which an accurate Canny detector is inserted.

3.3.4 Proposed 3FDBS-LC detection method

An outline of the entire processing flowchart of this 3FDBS-LC method is summarized in Figure 3.5, and in Figure 3.6. The pseudo-code description for the method is presented in Algorithm 3.2. First, after converting a color image into a gray scale image, the Laplace filter occupying the dominant character will sharpen the outline and detail of the gray scale target. Secondly, three-frame difference and background difference operations are performed separately. Then, threshold binarization and Canny edge detection are performed to identify and extract edge information. Finally, the combination of these two main methods undergoes a logical OR operation followed by a morphological operation for obtaining the final moving object shapes. Once all operations performed, the process enters the next cycle for real-time monitoring. Note that treatments are straightforward operations roughly executed within a $O(N)$ time complexity, with N the number of pixels, that make the global method a good candidate for real-time execution. Also, their intrinsic parallelism should allow efficient parallel implementation in GPU systems.

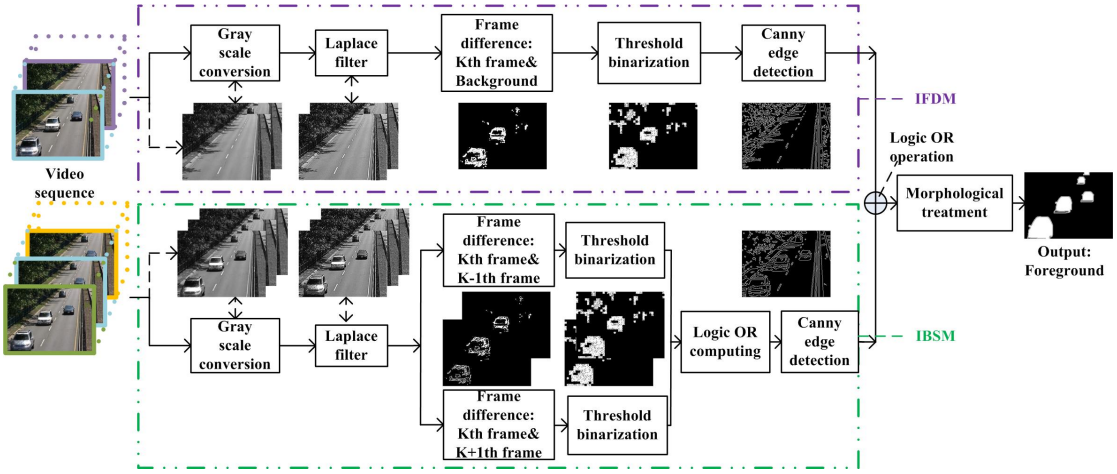


FIGURE 3.5: The framework of our improved algorithm.

3.4 Experiment and evaluation

3.4.1 Dataset

Here, three benchmarks, the SABS dataset, the Wallflower dataset, and the Multivision dataset are applied to experiments. They are used for visual demonstration, comparative experimentations, and numerical evaluation under different standard criteria.

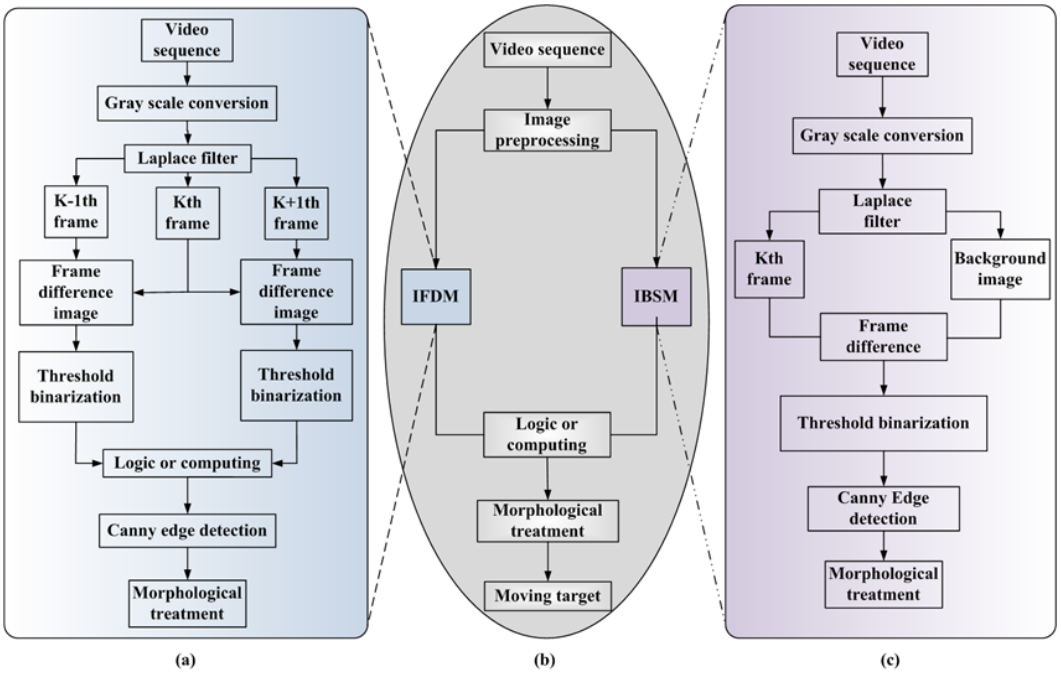


FIGURE 3.6: Flow chart of the proposed algorithm: (a) Frame difference component, (b) Main algorithm, (c) Background subtraction component.

The SABS (Stuttgart Artificial Background Subtraction) dataset[1] is an artificial benchmark for pixel evaluation of background models [BHH11]. SABS consists of video sequences with nine different background external changes for video surveillance. It has been added global illumination and Gaussian noise. Compared to other manually ground truth datasets, the SABS dataset does not so much suffer from imperfect labels. SABS contains ground-truth annotation and additional shadow annotation for tracking evaluation.

Wallflower dataset[2] consists of 7 test scenarios [TKBM99]. Each scenario represents a different, potentially problematic situation for background maintenance. When dealing with these image sequences, the output of the algorithm is divided into background image and foreground image, accompanying with their corresponding hand-segmented evaluation image. In order to deal with various problems that arise at the spatial scale, the evaluation image is segmented at pixels, regions, and frames levels. These training images, test images, and hand-segmented evaluation images are useful for training, evaluation, and comparison work.

Multivision dataset[3] is a database for evaluation of hardware/software real-time vision

[1] *http://www.vis.uni-stuttgart.de/en/research/information-visualisation-and-visual-analytics/visual-analytics-of-video-data/sabs.html*
[2] *https://www.microsoft.com/en-us/download/details.aspx?id=54965*
[3] *http://atcproyectos.ugr.es/mvision/*

Algorithm 3.2 Proposed 3FDBS-LC detection algorithm.

Input: Three adjacent frames m_{k-1}, m_k, m_{k+1} and background frame b_k

function IFDM(m_{k-1}, m_k, m_{k+1}) // Subfunction: frame difference algorithm
  $T_k \leftarrow T$ // To acquire adaptive threshold value ($\leftarrow$ means to become)
  for $Input : m_{k-1}, m_k, m_{k+1}$ **do**
    $f_{k-1} \leftarrow n_{k-1} \leftarrow m_{k-1}, f_k \leftarrow n_k \leftarrow m_k, f_k \leftarrow n_k \leftarrow m_k$
    // Image preprocessing for adjacent three frames: $f_{Laplace} \leftarrow n_{GrayScale} \leftarrow m_{ColorScale}$
    $D_k \leftarrow |f_k - f_{k-1}|$ // Frame difference operation
    $D_{k+1} \leftarrow |f_{k+1} - f_k|$ // Frame difference operation
    if $D_k or D_{k+1} < T_k$ **then**
      $R_k, R_{k+1} \leftarrow 0(background)$ // Binary operation to get background
    else
      $R_k, R_{k+1} \leftarrow 1(foreground)$ // Binary operation to get foreground
    end if
    $R_k \cap R_{k+1} \rightarrow FD$ // Logic OR computing ($\rightarrow$ means to get)
    $FD + Canny \rightarrow FD_c$ // Canny edge detection processing
    $FD_c \rightarrow FD_m$ // Morphology processing
  end for
  return FD_m
end function
function IBSM(m_k, b_k) // Subfunction: background subtraction algorithm
  $T_k \leftarrow T$ // To acquire adaptive threshold value
  for $Input : m_k, b_k$ **do**
    $f_k \leftarrow n_k \leftarrow m_k$ // Image preprocessing $b_{Laplace} \leftarrow b_{GrayScale} \leftarrow b_k$
    $D'_k \leftarrow |f_k - b_{Laplace}|$ // Difference operation
    if $D'_k < T_k$ **then**
      $R'_k \leftarrow 0(background)$ // Binary operation to get background
    else
      $R'_k \leftarrow 1(foreground)$ // Binary operation to get foreground
    end if
    $R'_k + Canny \rightarrow BS_c$ // Canny edge detection processing
    $BS_c \rightarrow BS_m$ // Morphology processing
  end for
  return BS_m
end function
function 3FDBS-LC(FD_m, BS_m) // Main function
  $FD_m \cap BS_m \rightarrow result$ // Logic OR operation to get the result
  return $result$
end function

systems based on multiple cameras [FSRDR14]. In a vision system, the goal is to translate the image into detailed information and to provide a visual solution that efficiently processes images taken from multiple cameras and complements the estimation reliably and robustly. The benchmark provides a dataset with ground truth segmentation, which enable to carry out objective evaluation of frame difference algorithms and background subtraction algorithms, as required in our study.

3.4.2 Evaluation criteria

Based on ground truth assessment, some evaluation criteria are defined to assess and compare the data results between different tracking methods. In pattern recognition and information retrieval, precision is an indicator for the relevance of the results, and recall

is a measure of the return of real relevant results. The experimental output quality is evaluated in this experiment by using accuracy, recall, precision, and F-measure.

Accuracy is defined as the number of true positives (TP) plus the number of true negatives (TN) over all of the samples. Formally,

$$Accuracy = (TP + TN) \div (TP + FP + TN + FN), \tag{3.1}$$

where TP is the number of foreground pixels that are correctly defined as foreground, TN is the number of background pixels that are correctly defined as background, FP is the number of background pixels that are mistakenly defined as foreground, and FN is the number of foreground pixels that are mistakenly defined as background.

Recall is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN). Then,

$$Recall = TP \div (TP + FN) \tag{3.2}$$

Precision is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP).

$$Precision = TP \div (TP + FP) \tag{3.3}$$

F-measure is defined as the harmonic mean of Precision and Recall.

$$F - measure = \frac{2Recall * Precision}{(Recall + Precision)} \tag{3.4}$$

High scores for F-measure show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

3.4.3 Qualitative evaluation of the sequence of treatments

Experiments presented in chis chapter were conducted on a CPU Intel(R) Core(TM) i5-4590 3.3 GHz. The SABS-Bootstrap sequences with 352×288 images are used to demonstrate the results after different treatments, as shown in Figure 3.7. On these images, we can find the results about the standard BS method, the FD method, their combination with or without the Laplace filter (as shown in (d) - (h)) and the proposed method (i). We can visually check that the proposed method (i) can more clearly point

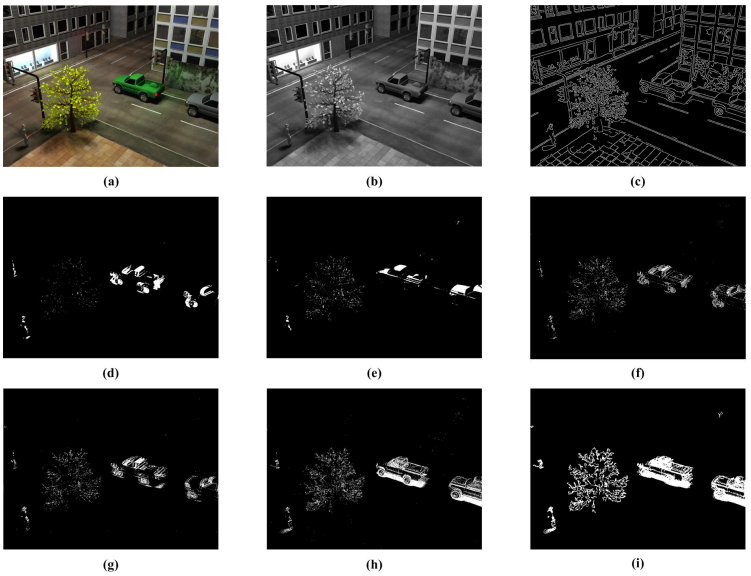


FIGURE 3.7: From the left to the right: (a) original color scale image, (b) grayscale image, (c) image processed by the Canny edge detector, (d) image extracted by standard three-frame difference, (e) image extracted by standard background difference, (f) the logic OR operation between (d) and (e), (g) the improved three-frame difference method after Laplace filter, (h) the improved background subtraction method after Laplace filter and (i) the improved 3FDBS-LC method.

out moving objects: running cars, walking pedestrians, and swinging trees that are blown by the wind.

3.4.4 Comparative evaluation

In the following comparative evaluation, all of the algorithm parameters were set as detailed in the previous sections and remained fixed for all the experiments. Based on Wallflower and Multivision datasets, ten image sequences are used: Camouflage, Foreground Aperture, Ground Truth Sequences, Chair Box, Hallway, Lab Door, LCD Screen, Wall, Crossing, and Suitcase. A visual presentation of the results obtained by two standard algorithms and by the proposed 3FDBS-LC method is given in Figure 3.8. The first column presents background images, the second column demonstrates every sample frame per sequences, ground truth images are shown in the third column, the fourth and fifth columns display the detected foreground under standard background subtraction and frame difference method respectively, moreover, the last column is the result of proposed 3FDBS-LC method. The improvement in shape detection should qualitatively be appreciated in the figure.

The quantitative numerical evaluations based on ground truth are reported in Table 3.1. A comparative evaluation of accuracy, precision, recall, and F-measure for three different tracking methods are included under the ten different image sequences. Figure 3.9 shows

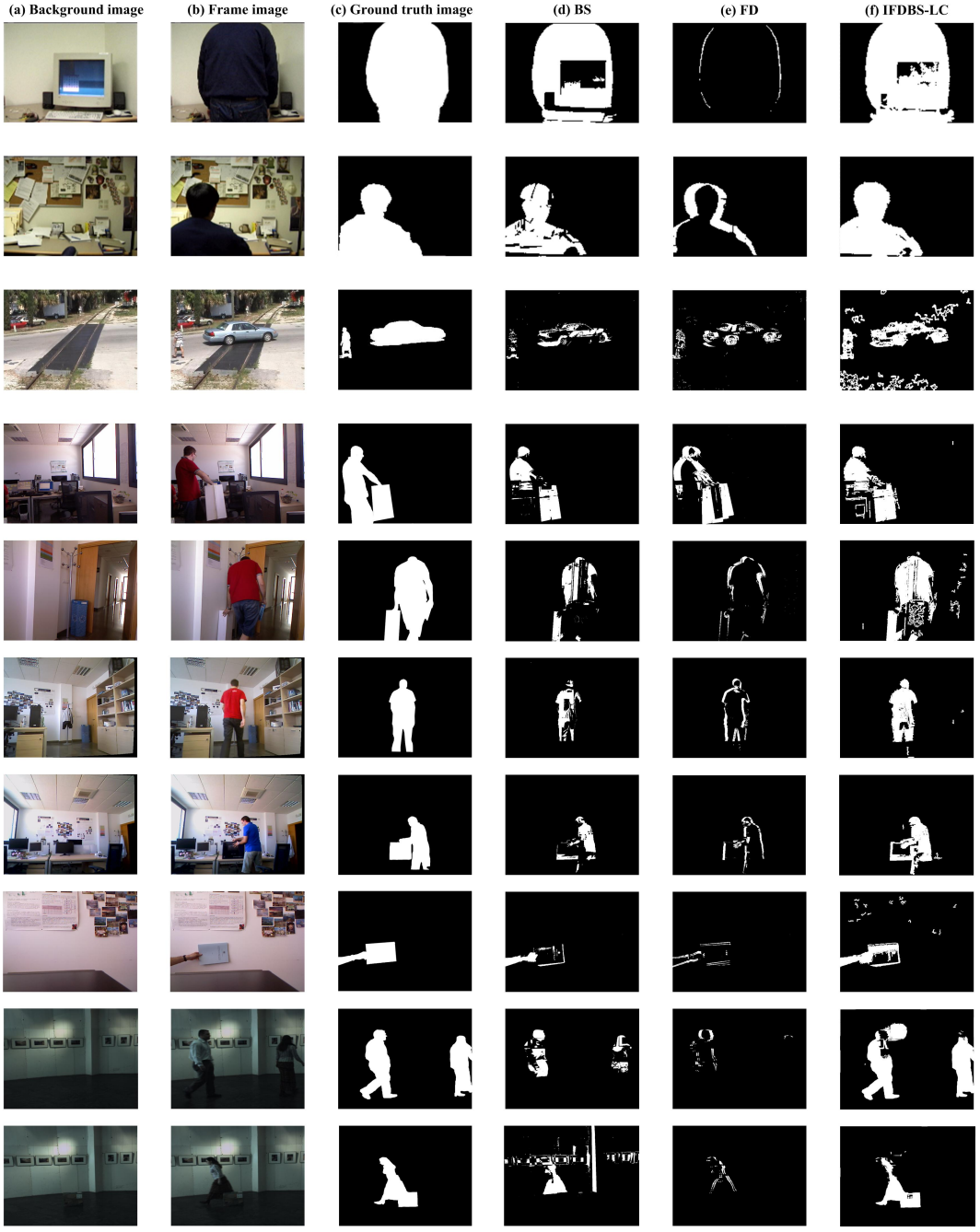


FIGURE 3.8: From the left to the right: (a) Background image, (b) Frame image, (c) Ground truth image, (d) Background subtraction method, (e) Frame difference method, and (f) The proposed 3FDBS-LC method.

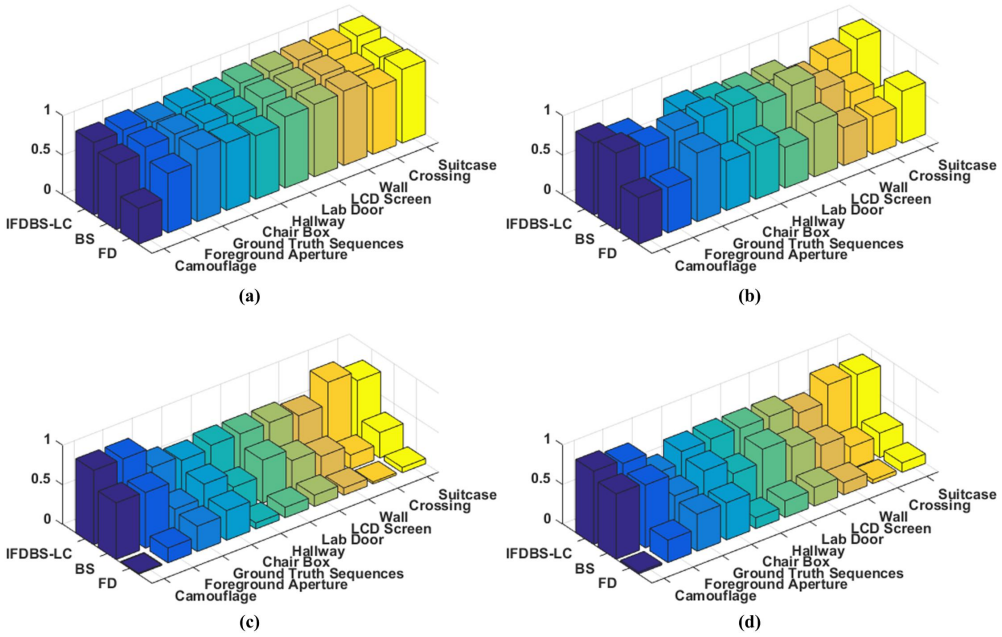


FIGURE 3.9: The comparison histograms of three different kinds of methods in (a)Accuracy, (b)Precision, (c)Recall, and (d)F-measure.

their corresponding histograms. From these results, it can be found that this proposed 3FDBS-LC algorithm can obtain good detection results superior to standard BS and FD methods.

3.5 Application to real-time video processing

In this section, we present implementation for real-time video monitoring. Our systematic tracking algorithm is realized as a set of MATLAB functions, embedded in a real-time video-rate driven loop, managed on a GUI (Graphical User Interface) platform for a convenient visualization and analysis, in a way similar to Andreatos [AZ09]. As can be seen from Figure 3.10, this GUI interface [CC18] mainly contains necessary image processing and experimental evaluation modules. Rectangle box and shape based representation methods mainly realize the actual scene view implementation.

We test the proposed method by using moving sequence for moving target tracking with application in real-time surveillance video. Our proposed method aims to detect all of the targets which are moving over an entire video sequence. This detection process is primarily shooting different consecutive images or frames at different time intervals. In our experiment, we use a CCTV (Closed-Circuit TeleVision) video sequence of automobile traffic presenting moving cars.

TABLE 3.1: Evaluation criteria results under different databases.

Algorithm	3FDBS-LC	BS	FD	3FDBS-LC	BS	FD
Dataset		Accuracy			Precision	
Camouflage	0.9153	0.8648	0.4522	0.9090	0.9480	0.5882
F-A	0.9319	0.9260	0.7525	0.8230	0.9095	0.5746
GT-S	0.8889	0.9420	0.9144	0.5207	0.9467	0.8702
Chair Box	0.9244	0.9301	0.8534	0.8980	0.9902	0.6264
Hallway	0.9119	0.8920	0.8022	0.8552	0.9921	0.6764
Lab Door	0.9547	0.9529	0.8996	0.8369	0.8651	0.5137
LCD Screen	0.9601	0.9571	0.9146	0.8484	0.9403	0.6844
Wall	0.9625	0.9639	0.9405	0.6904	0.8137	0.4795
Crossing	0.9579	0.8417	0.8311	0.8399	0.5977	0.4760
Suitcase	0.9822	0.8997	0.9318	0.9438	0.2978	0.6573
Dataset		Recall			F-measure	
Camouflage	0.9380	0.9480	0.0115	0.9232	0.8673	0.0226
F-A	0.9405	0.8021	0.1871	0.8778	0.8524	0.2823
GT-S	0.6922	0.5372	0.3195	0.5943	0.6854	0.4674
Chair Box	0.6363	0.6128	0.3736	0.7449	0.7571	0.4680
Hallway	0.6842	0.4866	0.0767	0.7602	0.6529	0.1379
Lab Door	0.6858	0.6320	0.1307	0.7539	0.7304	0.2084
LCD Screen	0.6864	0.5822	0.1417	0.7589	0.7191	0.2348
Wall	0.6313	0.4957	0.0908	0.6595	0.6161	0.1527
Crossing	0.9104	0.1773	0.0251	0.8737	0.2735	0.0477
Suitcase	0.7897	0.3194	0.0657	0.8599	0.3082	0.1195

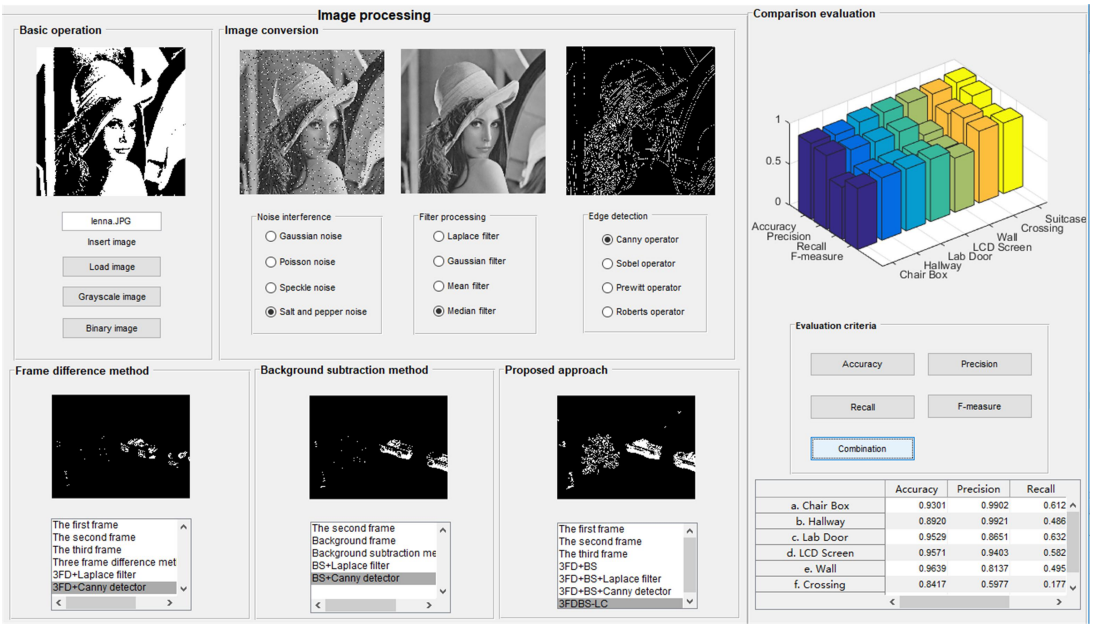


FIGURE 3.10: The display for object detection system.

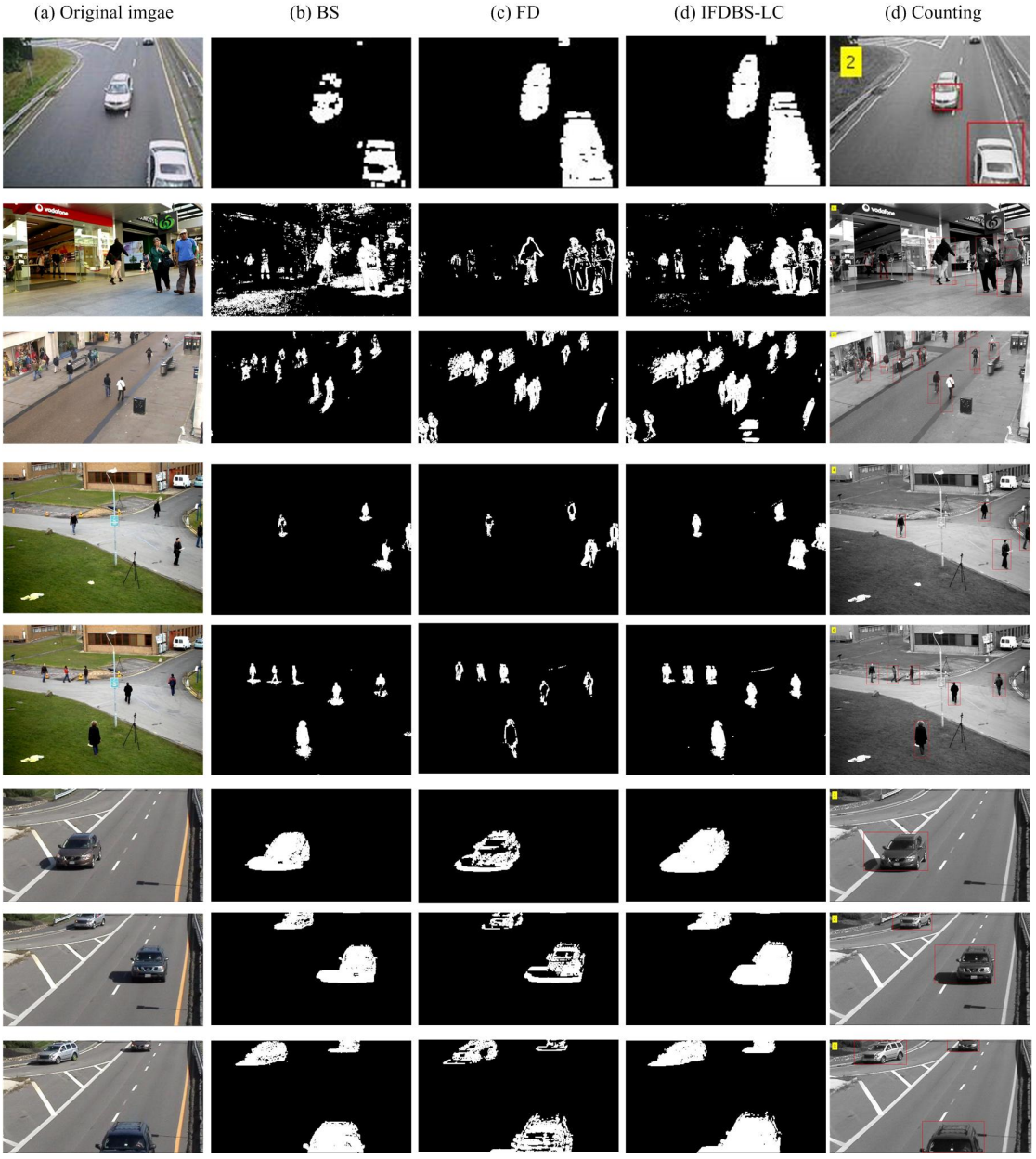


FIGURE 3.11: Applied to actual scene in real-time surveillance: (a) Original single frame, (b) BS, (c) FD, (d) IFDBS-LC (e) Realized by rectangle box.

The multiple Object Tracking Benchmark [4], and Active Vision Laboratory Benchmark [5] are used for qualitative visual evaluation and comparison. As depicted in the following Figure 3.11, we have demonstrated through qualitative evaluation that the system can provide accurate position estimation for a large number of vehicles or pedestrians in real-time. According to real-time validation, the actual implementation allows to deal with standard video-rate of 24 frames by second. Also, because of the parallel nature of most of the treatments, fastest video-rate processing is envisaged by GPU implementation.

[4] *https://motchallenge.net/vis/PETS09-S2L1*
[5] *http://www.robots.ox.ac.uk/ActiveVision/index.html*

This combined algorithm could then be used to track an unknown number of mobile topics with higher accuracy of the observed target shapes, and in real-time.

3.6 Conclusion

In this chapter, an improved object detection algorithm is proposed by systematically combining important features of background subtraction and frame difference methods usually employed in real-time surveillance tracking. The method mainly contains Laplace filter, frame difference method, background subtraction method, and Canny edge detector, which have real-time implementation available. The proposed algorithm was tested on standard datasets with the evaluation criteria of accuracy, recall, precision, and F-measure, and was compared, based on groundtruth evaluation, to the standard BS and FD methods. Results demonstrated an improvement in accuracy over the standard methods and computation time of the overall method remains compatible with standard video rate on a personal computer. Also, since these procedures are parallel by nature, the design of software in relation to GPU system is a matter of current investigation to further accelerate treatments.

Chapter 4

Using Marr-wavelets and entropy/response to automatic feature detection

4.1 Introduction

This chapter presents our second application within Matlab environment. Image matching, also refereed as feature point matching, is a fundamental issue in computer vision. In this chapter, we propose to use local entropy based on Marr wavelets within scale-interaction to improve the accuracy of automatic feature detection in the context of image matching. The goal is to improve the accuracy of the feature matching step while exhibiting a highly representative set of features of the objects within both images. To improve reliability, we propose to exploit local entropy under a mesh division strategy in combination with a sensitive feature selection stage.

Given a pair of images, how to detect and extract feature points is the first step of image matching. Automatic feature extraction is a central key-point to allow further pertinent image matching based on a feature to feature correspondence mapping. Given some standard nearest neighbor matching strategy, how to improve the reliability of feature sets is a question addressed here. To respond, we try to combine or enhance standard and easy to implement feature detection methods such that the resulting overall method, including both feature detection and matching, will be competitive both in computation time and quality of matching. Experimental results show that this algorithm can outperform some of the conventional feature extraction algorithms with higher subsequent matching recall rate of image matching.

A necessary graph matching procedure based on normalized cross-correlation similarity measure is applied to gauge the effectiveness of the approach in image matching. The quality evaluation is delegated to a RANSAC procedure that allows to eliminate wrong matching pairs and compute a recall rate of true matches. Experiments are conducting on standard image processing benchmarks. They illustrate how increasing feature set size and matching accuracy can be both achieved while preserving computation time.

This chapter is organized as follows. In section 4.2, the different steps of the proposed feature extraction and graph matching procedures are respectively presented. The evaluation of the proposed algorithm and its comparison with some conventional methods are exposed in section 4.3. Section 4.4 presents the conclusion.

4.2 Methodology

In this chapter, we present the implementation of the systematic image matching process includes feature points detection and extraction, feature points matching, and matching points filtering. This necessary process is shown in Figure 4.1. As to the input images, they are pretreated by Laplace filter firstly, because of its efficacy of sharpening and enhancement for the details of images. Then feature points are detected under Marr wavelets algorithm. Entropy and response are used to extract some of the points satisfying distribution criterion. The combination of normalization cross-correlation and nearest neighbor ratio realizes the implementation of the matching method. The last part is removing outliers by RANSAC.

4.2.1 Image pre-processing

Laplacian is a second-order derivative operator that detects the zero-crossing of the image intensity and usually produces more accurate edge detection results [TGY+10]. Laplace filter means a discrete approximation to the mathematical Laplace operator. Its second-order partial derivative in the orthogonal direction of continuous space and the approximation of its mathematical equivalent are defined below [vVYB89]:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{4.1}$$

$$\begin{aligned} \nabla^2 f(x,y) \cong \{ & f(x+1,y) + f(x-1,y) \\ & + f(x,y+1) + f(x,y-1) \} - 4f(x,y) \end{aligned} \tag{4.2}$$

$$I(X) = I(x,y) = f(x,y) - c \cdot \nabla^2 f(x,y) \tag{4.3}$$

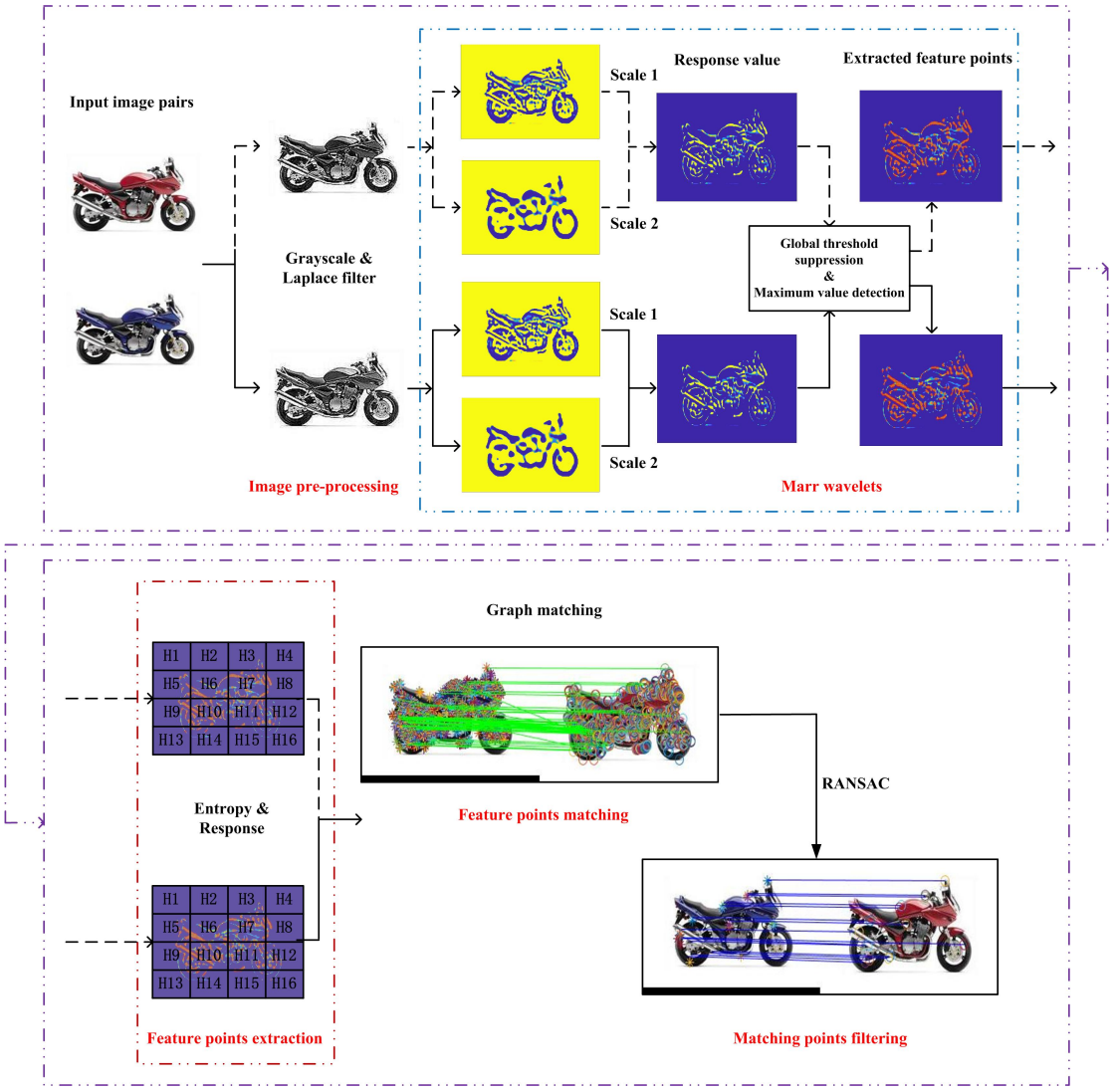


FIGURE 4.1: The basic flowchart of graph matching.

where $f(x, y)$ is the original image, $I(x, y)$ is the processed image, c is a constant.

From formula 4.2, the digital mask filter w can be viewed as the following 3×3 set of filter coefficients as shown in Figure 4.2 (a). The process of the Laplacian filter sharpening is essentially a convolution process. Suppose the origin pixel of f is located in the upper left corner of the image f, and set the middle value of mask w as the center of kernel. Let w move at all possible positions so that the center kernel of w can coincide with each of pixels of f. The convolution operation is essentially the sum of the products of the corresponding positions of the two functions. The convolution between f and its corresponding mask filter w is shown in Figure 4.2 (b). The definition of two-dimensional convolution is as:

$$I(x, y) = f * w = \sum_{k,l} f(x + k, y + l)w(k, l) \tag{4.4}$$

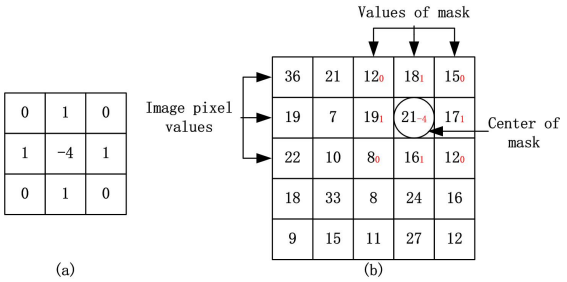


FIGURE 4.2: The mask filter of Laplace.

4.2.2 Marr wavelets within scale-interaction

Receptive field [Ste99] is used to describe the stimulation pattern of the retina. The receptive field of high-level neuronal cells in the visual pathway is synthesized from the low-level neuronal cell receptive field. Therefore, with the level increases, the range of receptive field becomes more considerable. Ultra-complex neuron cell models [DZC89] are capable of responding to sophisticated object features with powerful nonlinear processing capabilities, and most ultra-complex neuronal cells have end-stopping characteristics that are sensitive at the end of the line segments, corner points and line segments with high curvature.

In other words, the response of ultra-complex neuronal cells to light can be modeled by the response of the difference in spatial filters of different bandwidths to light. The response of the receptive field to light can be represented by a spatial filter function, such as a Gaussian difference function or a Gabor wavelet function.

The scale-interaction model for feature detection originate based on filtering using a class of self-similar Gabor functions or Gabor wavelets [MCvdM92, MSC+96], which can achieve the minimum possible joint resolution in space and frequency domain. It was came up with since its unique in attaining the minimum possible value of joint uncertainty [Dau85]. The function of the feature detection is defined as the following formula:

$$Q_{ij}(x, y, \theta) = f(W_i(x, y, \theta) - \gamma W_j(x, y, \theta)) \tag{4.5}$$

where γ is a normalizing factor, $W_i(x, y, \theta)$ and $W_j(x, y, \theta)$ are spatial filters. They go through the transformation of a nonlinear function f at location (x, y) with preferred orientation θ in two scales i and j respectively. If feature detection function Q_{ij} obtains a local maximum at the location (x, y), this location is considered to be a potential feature point position.

For further optimization, the Marr wavelets [BK98] was used instead of Gabor wavelets within the scale-interaction model, because of its isotropic [AM96, DLJZ10]. Two-dimensional Marr wavelets and its corresponding feature detection function are defined

as:

$$M_i(X) = \lambda_i(2 - \lambda_i^2 X^2)exp(-\frac{\lambda_i^2 X^2}{2}) \tag{4.6}$$

$$Q_{ij}(X) = |M_i(X) - \gamma M_j(X)| \tag{4.7}$$

$$R_{ij}(X) = I(X) * Q_{ij}(X) \tag{4.8}$$

where $X = (x, y)$, and $X^2 = (x^2 + y^2)$. $\lambda_i = 2^{-i}$ and i represents the scaling value of Marr wavelets. Convolve $Q_{ij}(X)$ with an grayscale image $I(X)$. If its response value $R_{ij}(X)$ obtains a maximum local value, then X is considered to be a potential feature point. Algorithm 4.1 is pseudo-code of Marr wavelets within scale-interaction. Figure 4.3 illustrates the process of extracting response value using Marr wavelets within scale-interaction. (b) and (c) are convolution results between the original picture and the mask filter. Then local maximum points are extracted on this basis.

Algorithm 4.1 Marr wavelets within scale-interaction algorithm

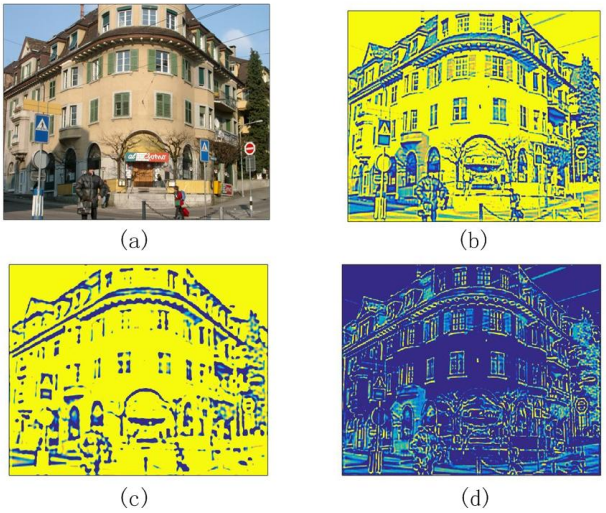
Input: $I(X), i, j$

Output: points

1: $I_i = MarrFilter(I(X), i)$
2: $I_j = MarrFilter(I(X), j)$
3: $I_{sub} \leftarrow |I_i - I_j|$
4: $local_{thr} \leftarrow max(max(I_{sub})) * r$
5: **if** $I_{sub}(i, j) < local_{thr}$ **then**
6: $\quad I_{localthr}(i, j) \leftarrow I_{sub}(i, j) \leftarrow 0$
7: **end if**
8: $points \leftarrow corner_{peaks} \leftarrow I_{localthr}$
9: **function** $MarrFilter(I(X), scale)$
10: $\quad \delta = 2^{scale}$
11: $\quad x = -(2 * fix(\delta)) : 1 : (2 * fix(\delta))$
12: $\quad y = -(2 * fix(2 * \delta)) : 1 : (2 * fix(2 * \delta))$
13: $\quad M_i(X) \leftarrow X \leftarrow meshgrid(x, y)$
14: $\quad I_{fil} \leftarrow I(X) * M_i(X)$
15: **end function**

4.2.3 Entropy and response

For the problem of uneven distribution or excessive number of feature points, accompanying with a long calculation time of matching, we propose a feature point extraction method based on image local entropy and feature point response, called entropy and response (ER) algorithm. In this section, three main parts are elaborated.



(a) (b) (c) (d)

FIGURE 4.3: From the left to the right: (a) Original image, (b) Filtered result when $i = 1$, (c) Filtered result when $i = 2$ and (d) Response image.

4.2.3.1 Entropy algorithm

In actual images, feature points often appear as sharp changes of gray values or inhomogeneity in grayscale distribution; that is, the local region of a feature point has a large amount of information. Entropy is a measure of information in an image, and local entropy is a measure of local area information of an image. Local entropy value under feature-rich region is much higher than the local entropy value under feature-poor region. Therefore, it is possible to determine which regions have more features by calculating the local information entropy of image, and then extract the feature points in these regions.

Information entropy [KSW85, LTRC11] is the amount that represents the overall characteristics of the source in a common sense. It is considered from the statistical properties of the entire source to measure the expected value of a random variable. An image is essentially a source of information that can be described by information entropy. Let the gray image G have m gray levels, mesh division is performed to obtain $n \times n$ subregions. The whole information entropy H_i and the average entropy $\bar{H}$ of the image are calculated as follows:

$$H_i = \sum_{i=1}^{m} p_i log_2 p_i \tag{4.9}$$

$$\bar{H} = \frac{1}{n^2} \sum_{j=1}^{n^2} H_j \tag{4.10}$$

where p_i is the probability that the i_{th} gray level appears, that is, the ratio of the number of pixels whose gray value is i to the total number of pixels of the image. So the local entropy is counted for the probability of occurrence of gray level in the sub-image. Since

the value of the information entropy is only related to the whole of the local gray-scale pixels, but independent with a single pixel, so it is not sensitive to the influence of noise and can improve the accuracy of the image authenticity description. Here, we use the local information entropy of the image to extract feature points. Under the meshing strategy, the image is divided into $n \times n$ sub-regions. Therefore, the sub-average entropy value of each sub-grid can be calculated. Figure 4.4 is a schematic diagram of mesh division, n is set to 40.

FIGURE 4.4: Schematic diagram of meshing: (a) Image divided by 4×4 sub-regions, (b) Image divided by 8×8 sub-regions.

4.2.3.2 Response algorithm

After mesh division and the computation of each local entropy, we will get $n \times n$ sub-regions for the whole image, then detected feature points are mapped into the respective sub-areas. In this case, if we compute and sort the entropy values of all of these feature points extracted, then the first N feature points with larger entropy values can be selected to describe the whole image. However, this method only utilizes the entropy values of feature points, without considering its distribution in the image. Finally, feature points with high entropy values may mostly appear in the same local area, which will cause aggregation. So a block division and response algorithm are proposed to deal with this problem.

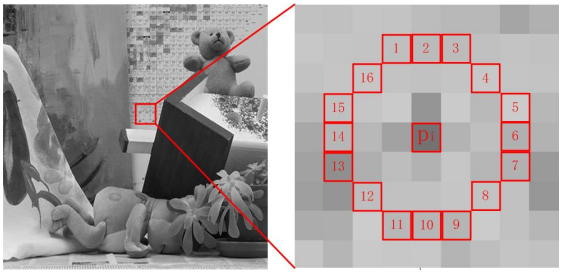


FIGURE 4.5: Bresenham discrete circle centered on pixel p_i.

As mentioned before, if a pixel presents a sharp change in its neighborhood, this pixel will have a stronger deviation value from the mean value. Based on the Bresenham

discrete circle [RPD10] with the pixel point p_i as the center and 3 pixels as the radius, 16-pixel points on the discrete circumference are considered in correspondence with the central pixel point p_i. This is shown in Figure 4.5. These 16 pixels are assigned to dark and bright area. The dividing criteria and deviation [KSK08] are respectively defined as the following:

$$S_{bright} = \{x | I_{p_i,x} > I_{p_i} + t\} \tag{4.11}$$

$$S_{dark} = \{x | I_{p_i,x} \leq I_{p_i} - t\} \tag{4.12}$$

$$Dev = max(\sum_{x \in S_{bright}} |I_{p_i,x} - I_{p_i}| - t, \sum_{x \in S_{dark}} |I_{p_i} - I_{p_i,x}| - t) \tag{4.13}$$

where S_{bright} indicates bright area, S_{dark} indicates dark area. I_{p_i} is the gray value of center point p_i, $I_{p_i,x}$ represents the gray value of a pixel labeled x on a discrete circumference centered at pixel p_i, t is the set threshold. Dev is the sum of the deviation value among the gray values of the pixel p_i and its corresponding neighboring pixels located in the bright or dark area.

4.2.3.3 Distribution criterion

After the calculation of the entropy and response, a distribution criterion is proposed to extract the corresponding points that meet the requirements. In the region where the local entropy is bigger than the average of entropy $\bar{H}$, the feature points are extracted with the ratio r. The only one strongest response point is extracted in each remaining region where the local entropy is smaller than $\bar{H}$. Here, we choose the unified ratio method for the selection of r. Assuming that there are m regions whose local entropy is greater than $\bar{H}$, then $r_i(1, 2, ..., m)$ is the ratio of extracting feature points in the i_{th} region. For example, when $r = 10\%$, that is, in the m_{th} region with large local entropy, the feature points with the top 10% responses given by Dev are extracted. The appropriate value of r is set empirically.

The detail of the entire ER method is outlined in Algorithm 4.2. Based on the mesh division strategy, we first compute each of the local average entropy value of all these sub-regions. Then the feature points are sorted according to the computation of their deviation values in each of their sub-region. Finally, the distribution criterion can not only effectively reduce some of the useless feature points, but also ensure the uniformity of feature point distribution. As we can see, the step entropy and response can both develop a custom algorithm to identify feature points. Entropy strategy is only to calculate the reflection degree of an individual pixel, but the response is based on the deviation value of a set pixels between the center point, and it is adjacent points based on the Bresenham discrete circle principle, so that the mutation of response is more

reflected for finding point with reliable contrast. Therefore, deviation is more capable of extracting more qualified feature points than entropy.

Algorithm 4.2 Entropy and response algorithms

Input: $I(X), points, m, n, t, r$

Output: $points_{select}$

for $j = 1 \rightarrow n^2$ **do**
  $points_j \leftarrow points$
  $H_j, \bar{H} \leftarrow Entropy(I(X), m, n)$
  $p_{sub(j)} \leftarrow Response(points_j, t)$
  if $H_j \geqslant \bar{H}$ **then**
    $points_{select} \leftarrow p_{(p_{sub(j)} * r)}$
  else
    $points_{select} \leftarrow p_{max}$
  end if
  $j = j + 1$
end for
return $points_{select}$
function $Entropy(I(X), m, n)$
  $p_{i(m*1)} \leftarrow Isub(i)_{(n*n)} \leftarrow I(X)$
  $H = \sum_{i=1}^{m} p_i log_2 p_i$
  $\bar{H} = \frac{1}{n^2} \sum_{j=1}^{n^2} H_j$
end function
function $Response(points, t)$
  $I_{p,x} \leftarrow circle_p \leftarrow points$
  $Dev = max(\sum_{x \in S_{bright}} |I_{p,x} - I_p| - t, \sum_{x \in S_{dark}} |I_p - I_{p,x}| - t)$
  $p_{sub} \leftarrow p_{(Dev_p)}$
end function

4.2.4 Graph matching problem

The purpose of graph matching [LCL10] is to determine the correct attribute correspondences $P = (V^P, E^P)$ and $Q = (V^Q, E^Q)$ between two graphs P and Q, where V means vertex, E represents egde. We customize corresponding mapping edges $e_1 = ij \in E^P$, $e_2 = ab \in E^Q$.

The objective of graph matching is to find the correct corresponding point pairs between two graphs P and Q among the feature points extracted. A unidirectional 'one-to-one' constraint is assumed, which requires one node in P to match at most one node in Q.

Cross-correlation is a standard method for estimating the degree of similarity between two sets of data [Bou96]. An essential application is the normalization cross-correlation (NCC), which has been used widely for many signal processing applications since its efficient and straightforward frequency domain expression, and it is less sensitive to linear variations in the amplitude of two comparison signals [YH09]. We use the NCC algorithm to measure the similarity between two feature point p in graph P and q in graph Q. These ratios $Ra(p,q)$ of the calculated correlation values represent the degree of matching between the two sets of corresponding images. The NCC algorithm used for finding similarity match between a window around feature point p and a window around feature point q is defined as:

$$Ra(p,q) = \frac{\sum_i [(Wp_i - \overline{Wp})(Wq_i - \overline{Wq})]}{\sqrt{\sum_i (Wp_i - \overline{Wp})^2}\sqrt{\sum_i (Wq_i - \overline{Wq})^2}} \tag{4.14}$$

where the summations are over all window coordinates, Wp_i and Wq_i are pixel intensity in windows for p and q respectively, each of the windows is sized as 5×5. Also, $\overline{Wp}$ and $\overline{Wq}$ are the corresponding mean of the window pixels. The coordinate of maximum values in this normalization cross-correlation are the positions of the best matches for reference images.

Based on NCC similarity measure, we use nearest neighbor ratio (NNR) method to perform coarse matching on feature points set. The screening and matching process of the NNR algorithm with unidirectional "one-to-one" constraint is described as below: Based on the sample feature point pairs in two images, first extract all of the maximum corresponding pairs by using NCC algorithm; Then compare these ratios with a fixed threshold set in advance, the case where NCC ratio is bigger than this fixed threshold value, it's corresponding point pairs are considered to be a match, otherwise this couple of points will be abandoned. This fixed threshold value is usually a constant of not more than 0.9. Since the correct matches have a stronger similarity than those wrong matches, it is a functional judging characterization for graph matching by the idea of the NNR. Figure 4.6 shows the flow chart of data processing. The detected feature points of two images are placed into two corresponding buffer, respectively. Each of the points p in graph P are used to calculate the ratio of NCC with all of the points q in graph Q, then sort all the ratios in descending order. Under the principle of NNR, some will be extracted as the best matching points, otherwise, others will be deserted.

4.2.5 Outlier elimination

In this application, we use a convolution with a Laplacian (unsharp masking) as a pre-processing step before the convolution with a Marr filter. Due to the linearity of these

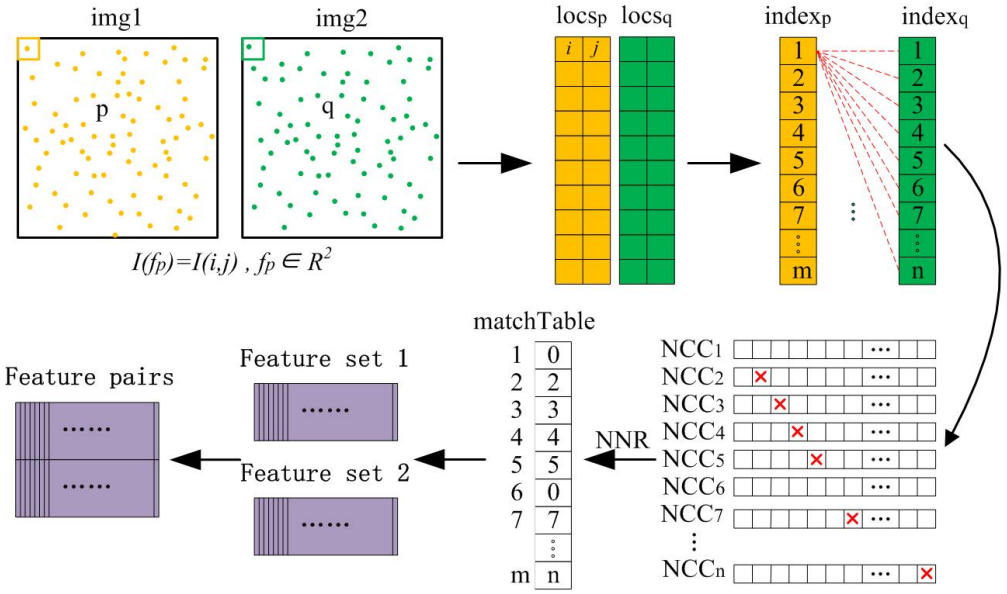


FIGURE 4.6: The basic flowchart of data processing.

two operations, this corresponds to a single convolution, which is a smoothed fourth-derivative filter. It is, therefore, that the number of feature points increases, as noise is amplified with each derivative. On the other hand, although the NNR method is easy to implement and sometimes well match, some points in these extracted feature points set are not matchable, so mismatched cleaning operation is necessary. Therefore, it is especially important to find a way to reduce the mismatch caused by interference.

RANSAC algorithm [RCP+13, Der10] which called the random sample consensus algorithm can eliminate the existence of mismatches very well. This algorithm has strong robustness and errata ability to the sample data set. The basic idea of the RANSAC algorithm is to extract the sample set from the model by using the iterative method. Look for an optimized parameter model that can contain more interior points in a data set, then use the residual set to test the extracted samples. The point in the algorithm that fits the data set model is called the inlier point; otherwise, it is called the outlier point or the wild point. So RANSAC algorithm can be used to find the optimal parameter model in a set of datasets containing outlier points by using an iterative algorithm. The detail implementation process of RANSAC is shown as follows:

- (a) Randomly extract a pairs of feature points that are not collinear from the data set ($a = 4$ in experiment), then calculate their transformation homography matrix H, and record it as model M.
- (b) Calculate projection error of each point in the dataset with model M_k. If the error is less than a predefined threshold τ, add it into the inner point set I_k.

- (c) If the current number of elements in inner point set I_k is greater than the number in optimal inner point set I_{best}, then update I_{best} and re-estimate the model M_{best}.
- (d) If the number of iteration is more than k, the operation will be exited; otherwise, the number of iterations is increased by 1, and the above steps are repeated.

The threshold τ is selected in accordance with n-dimensional chi-square distribution. χ is the cumulative chi-square distribution. It is assumed that the out-of-class point is Gaussian white noise with a mean of 0 and a variance of η. The number of iterations k is constantly updated rather than fixed until it is greater than the maximum number of iterations.

$$\tau^2 = \chi_n^{-1}(\mu)\eta^2 \tag{4.15}$$

$$k = \frac{log(1-p_c)}{log(1-\mu^a)} \tag{4.16}$$

where p_c is the confidence level, generally taking 0.95 to 0.99; μ is the ratio of inlier point; a is the minimum number of samples required to calculate for model. The pseudo-code of RANSAC is outlined in Algorithm 4.3.

Algorithm 4.3 RANSAC algorithm

Input: p_c, k_{max}, τ, a, m
Output: M_{best}, I_{best}
1: $k = 0, I_{max} = 0$
2: **for** $k < k_{max}$ **do**
3: $\tau^2 = X_n^{-1}(\mu)\eta^2$
4: Use randomly sampled subset a to estimate M_k and I_k.
5: **if** $|I_k| > I_{max}$ **then**
6: $M_{best} = M_k, I_{best} = I_k$
7: $\mu = |I_{best}|/m, k_{max} = log(1-p_c)/log(1-\mu^a)$
8: **end if**
9: $k = k + 1$
10: **end for**

4.2.6 Proposed algorithm

This section details the specific parameters used in the experiments. In feature points detection part, feature points are defined as local maxima inside the scale-interaction image (with $\gamma = 1$) by using Marr wavelets algorithm. The two scales we choose are $i = 1$ and $j = 2$. Then the mesh division and feature points extraction are processed, the image is meshed by $n \times n$ to obtain n^2 sub-regions. Here, $n = 40$ is selected. The detected feature points are mapped into the respective sub-areas and sorted according to their deviation value Dev in each sub-area to which they belong. Meanwhile, every local information entropy H_i of each sub-region and the average information entropy

$\bar{H}$ of the all are calculated. Assuming that there are k sub-regions with local information entropy greater than the average information entropy, then feature points with the top 30% responses are extracted in these k regions in our algorithm. In feature points matching part, NCC similarity measure algorithm and NNR method are used to perform coarse matching on feature points set. As to matching points filtering part, RANSAC can better achieve the deletion of mismatched points, finally realize a more accurate matching result. Its computational complexity is $O\left(max(N_P, N_Q)\right)$, with N_P the number of features in reference image, and N_Q the number of features query image. This algorithm is suitable for performing real-time global methods. In addition, it can also achieve efficient parallel implementation in GPU systems.

4.3 Experimental evaluation

Experiments were conducted on a CPU Intel(R) Core(TM) i5-4590 3.3 GHz. In this section, we evaluate the proposed method for feature points matching by using Visual Geometry Group dataset [1]. The following will be divided into two parts for experimental explanation: feature points extraction and feature matching respectively. As to quantitative evaluation, a set of experiments are conducted. The proposed algorithm based on Laplace filter and Marr wavelets under entropy method (L_Marr_E) was compared with other classic conventional methods, corner detector[Der04], Gilles[Gil98], Harris[HS88], LoG[Lin98] and SIFT [Low99] algorithms.

4.3.1 Feature points extraction

Based on the above experimental theory, the following experimental verification is performed. First of all, to verify the importance of the Laplacian filter algorithm in the feature point extraction phase. Figure 4.7 (b) and (c) show the comparison graph of before and after adding the Laplace filter. We can clearly see that the use of the Laplacian filter can greatly increase the number of feature points detected. Figure 4.8 shows feature points detection results among different kinds of algorithms: CD, Gilles, Harris, LoG, SIFT, and L_Marr_E. Correspondingly, Table 4.1 details the specific number of feature points extracted of Figure 4.8. The method proposed can extract more feature points than commonly used methods.

[1] *https://www.robots.ox.ac.uk/ vgg/data/*

FIGURE 4.7: Quantitative experimental analysis of feature point extraction: (a) Original image, (b) Feature point extraction without Laplace filter, (c) Feature point extraction with Laplace filter.

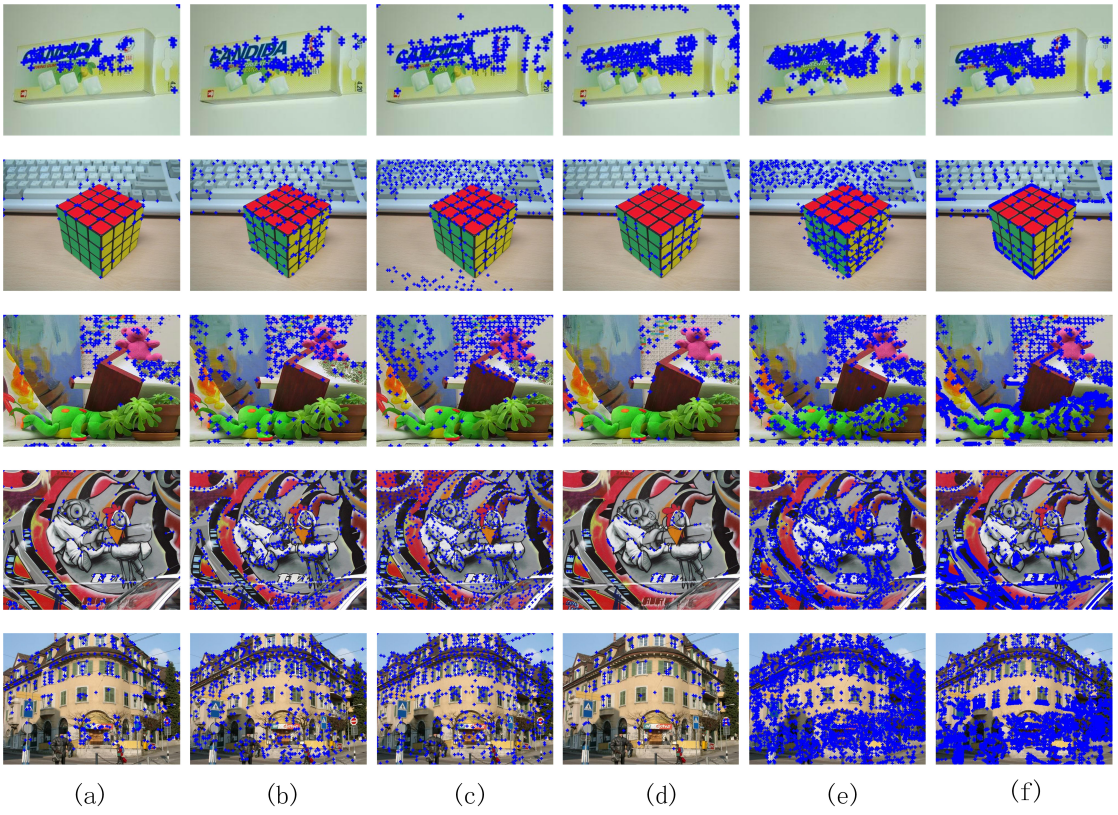


FIGURE 4.8: Feature points extraction under different algorithms: (a) CD (b) Gilles (c) Harris (d) LoG (e) SIFT (f) L_Marr_E.

TABLE 4.1: Feature point extraction results under different kinds of algorithms.

Method	CD	Gilles	Harris	LOG	SIFT	L_Marr_E
img.1	72	73	104	174	226	264
img.2	70	246	457	165	980	654
img.3	144	274	341	110	763	1576
img.4	203	844	1133	153	3533	8402
img.5	320	627	696	140	3366	7806

4.3.2 Feature points matching

The following experimental validation performs the entropy-based sparsification with application to feature matching. In combination with the susceptible feature selection stage, the ablation analysis provides some valuable intuition about the pipeline in feature matching phase. Then Figure 4.9 shows comparison results, we can find that RANSAC algorithm can implement very well on mismatched point clearing. Furthermore, this proposed method based on Laplace filter and Marr wavelets by using NCC algorithm (L_Marr) not only has high matching accuracy but also can significantly improve the correct feature matching number than the method without Laplace filter (Marr). Here, Recall, Time, and the number of image matching pairs are summarized in Table 4.2. The graph matching Recall [MM12] can be defined as follows:

$$Recall = N_{rm}/N_{tm} \tag{4.1}$$

where N_{rm} is defined as the number of detected true matches after RANSAC removes the mismatched points, and N_{tm} means the total number of correspondences. Although the recall of graph matching is increased, the matching time is significantly increased, as shown in the first two rows in Table 4.2. Fortunately, after L_Marr is processed by entropy (L_Marr_E), it has run time is significantly reduced as we expected.

TABLE 4.2: Experimental comparison results for quantitative analysis of feature point matching.

Methods	N_{rm}	N_{tm}	$Recall(\%)$	$Time(s)$
Marr	163	224	72.77	5.17
L_Marr	528	553	95.48	17.86
Marr_E	46	72	60.53	3.67
L_Marr_E	169	182	92.86	6.43

The longitudinal experiment assessment is carried out by taking from the test data of University of OXFORD's Object Categories Datasets, which consist of 13 different collections of images. Among them, seven sets of dataset were selected for comparative analysis in this section. Table 4.3 shows the mean values for all image pairs selected of the dataset. The best recall results are obtained by L_Marr_E, which produces the most

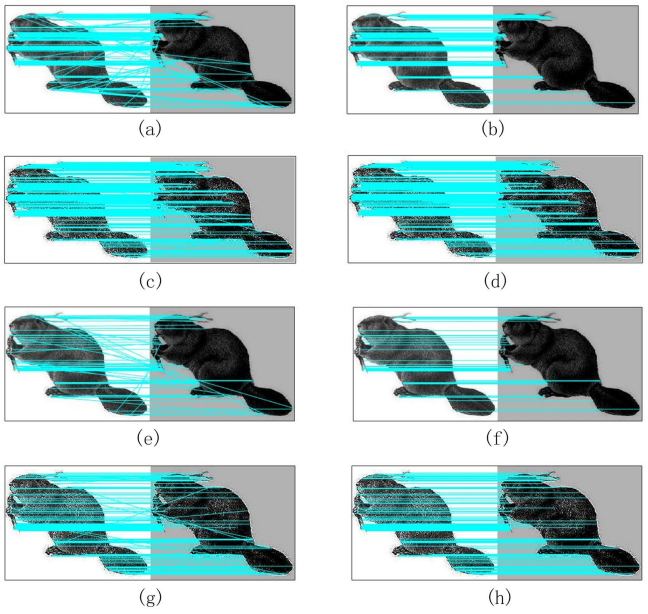


FIGURE 4.9: (a) and (b) show graph matching without Laplace filter before and after RANSAC optimization; (c) and (d) show graph matching under Laplace filter before and after RANSAC optimization; (e) and (f) show graph matching using entropy algorithm without Laplace filter before and after RANSAC optimization; (g) and (h) show graph matching using entropy algorithm under Laplace filter before and after RANSAC optimization.

significant number of feature points in two query images. Figure 4.10 demonstrates correspondence links between images under different algorithms: CD, Gilles, Harris, LoG, SIFT, and L_Marr_E. So the algorithm proposed can achieve a better matching effect. We also test our proposed algorithm on CMU house dataset as shown in Figure 4.11. However the recall in this case is only 0.1582, it's not satisfactory.

TABLE 4.3: Feature point matching results under different kinds of algorithms.

	Detector	$1^{st}Image$	$2^{nd}Image$	$Time(s)$	$Recall(\%)$
1	CD	51	53	1.76	0.36
2	Gilles	124	134	2.24	0.16
3	Harris	208	226	2.8	0.35
4	LoG	300	300	4.27	0.27
5	SIFT	1275	1258	5.49	0.48
6	L_Marr_E	1998	1929	7.22	0.52

4.4 Conclusion

While feature extraction methods are numerous, it is not straightforward that each could represent the objects to match from a query to a reference image adequately. In order to vary the feature set size while preserving a reasonable recall rate in graph matching, we have proposed a new combination of filters with an entropy-response based selection

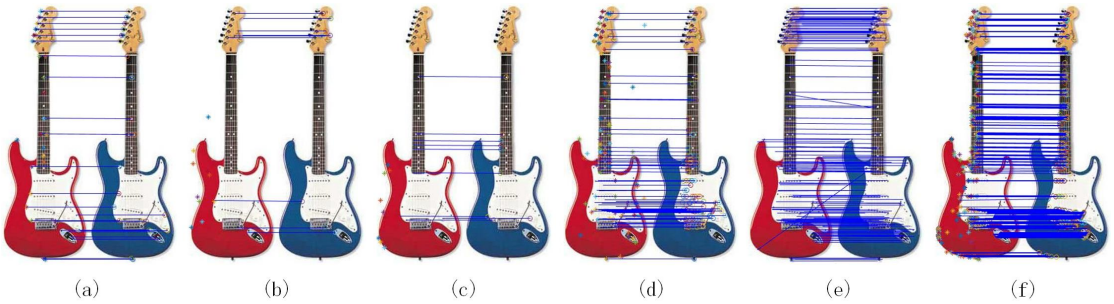


FIGURE 4.10: Feature points matching under different algorithms: (a) CD (b) Gilles (c) Harris (d) LoG (e) SIFT and (f) L_Marr_E.

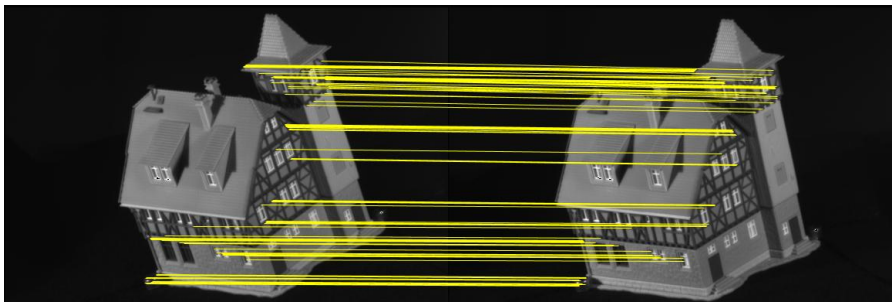


FIGURE 4.11: Feature points matching under CMU house dataset: (a) Before RANSAC (b) After RANSAC.

method. Laplace filter enhances the image's edge and details. Second, Marr wavelets embedded in scale-interaction is used to detect feature points. Then, entropy and brightness response are used to extract typical feature points. Most importantly, the entropy-based selection method dramatically reduces computation time. Image matching is achieved by nearest neighbor search with normalized cross-correlation similarity measure. Finally, RANSAC procedure removes outlier correspondences to achieve matching optimization. The comparison results show that our algorithm has a higher matching rate for reasonable computation time, despite the augmentation of the number of feature points under Laplace algorithm. In future work, we will improve implementation by exploiting natural parallelism of the method on the GPU platform.

Chapter 5

Affinity-preserving fixed point APRIP in Matlab framework for graph matching

5.1 Introduction

This chapter presents our third application within Matlab environment. We introduce in this chapter a new contribution of this thesis with an application to second-order graph matching in Matlab framework. The framework is based on the original Matlab application provided by Cho et al. [CLL10] and related to the reweighted random walks (RRWM) algorithm. It also provides other implemented algorithms such as the spectral matching (SM) algorithm [LH05] and integer projected fixed point (IPFP) algorithm [LHS09].

According to these starting approaches, we elaborate and propose a new combination to provide an improved algorithm using affinity preserving and reconstructed integer projected fixed point improved by spectral technique, called APRIP algorithm. More precisely the method combines the affinity matrix reweighting of RRWM into an updated IPFP loop, and starting with a solution provided by SM. All these approaches are based on IQP formulation. By considering the second-order term, these algorithms determine the mapping between two graphs that should reflect the geometric similarity relationship between the pairwise matching features.

All the algorithms are executed and compared based on a same experimental framework with common data from standard benchmarks in the domain. Then, this overall platform constitutes a second contribution of this chapter by providing all the data and interfaces

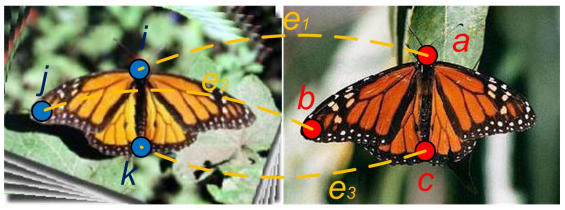


FIGURE 5.1: Graph matching assignment.

that will be adopted in the C++ GPU platform presented in the next chapter in order to evaluate new GPU proposals.

The chapter is organized as follows. In section 5.2, the problem formulation of the graph matching is given in detail. The proposed APRIP algorithm is demonstrated in section 5.3. The performance of the proposed algorithm is evaluated in section 5.4. Section 5.5 presents the conclusion.

5.2 Problem formulation

We set P and Q the two sets of features of query graph $G_P = (P, E^P)$ and reference graph $G_Q = (Q, E^Q)$ respectively. We note $i, j \in P$ and $a, b \in Q$ as feature points, $ij \in E^P$ and $ab \in E^Q$ as edges. Also, $e_1 = (i, a)$ and $e_2 = (j, b)$ represent, when needed, candidate assignments. The main task it to find a suitable one-to-one mapping between P and Q. A graph matching is shown in the Figure 5.1. The yellow lines are correct matches.

Affinity matrice M, also known as affinity tensor, is a basic statistical technique used to organize the mutual similarities between sets of feature points. The measurement of affinity can be interpreted as a product of a solution vector x, that represents the set of candidate correspondences, by the matrice. The solution variable $x \in \{0, 1\}^{N_P N_Q}$ is an indicator vector such that $x_{ia} = 1$ means feature $i \in P$ matches with feature $a \in Q$, $x_{ia} = 0$ means no correspondence, and where N_P and N_Q are the respective set sizes of P and Q.

A graph matching score S between edges can be defined by the following equation:

$$S = \sum_{ij \sim ab} f(ij, ab) = \sum_{ia,jb} M(ia, jb) x_{ia} x_{jb} = x^T M x, \tag{5.1}$$

where $ij \sim ab$ means (i, a) and (j, b) are correspondence pairs, and x is the indicator vector. Then, the purpose of the graph matching IQP problem is computing solution x^*

that maximizes the matching score as follows:

$$x^* = argmax(x^T M x), \tag{5.2}$$

$$s.t.\ x \in \{0,1\}^{N_P N_Q}, \tag{5.3}$$

$$\forall i \sum_{a=1}^{N_Q} x_{ia} \leq 1, \tag{5.4}$$

$$\forall a \sum_{i=1}^{N_P} x_{ia} \leq 1. \tag{5.5}$$

The binary constraint is expressed by equation 5.3, while 5.4 and 5.5 express the two-way constraints, that specify the solution to be a one-to-one mapping from P to Q. Note that by removing constraint 5.5, we obtain a many-to-one mapping, that is, a (partial) function from P to Q. In this chapter both constraints must be verified.

Affinity matrix M which consists of the relational similarity values between edges and nodes must is considered as an input of the problem. It can be noted that its size is defined by the total number of candidate assignment pairs considered. Then, the affinity matrix size may vary from $O\left((N_P N_Q)^2\right)$, in the case of full possible pairs, to $O\left((K \times N_P)^2\right)$ where K is some constant, in case of a restricted list of candidate pairs. Note that this list of candidate pairs must be added as part of the input to relate the entries of the affinity matrix to the feature points. The indicator variable x size varies also accordingly to the symmetric affinity matrix size. Its length corresponds to the column, of line, size of the matrix, and may vary from $N_P \times N_Q$ to $K \times N_P$ depending on the application.

Here, the matching score is completely retained as pairwise geometric only. The individual affinity $M(e_1, e_1)$ that represents first order affinity, is set to zero since there is no information about individual affinity. That is to say, all the diagonal values of the affinity matrix are zeros. The pairwise affinity $M(e_1, e_2) = M(ia, jb)$ between edges is given by:

$$M(ia, jb) = max(50 - d_{ia;jb}, 0), \tag{5.6}$$

where $d_{ia;jb}$ is the mutual projection error function used in [CLL09] between two candidate assignments (i, a) and (j, b), that includes euclidean distance evaluation d_{ij} between locations of features i, j. The table 5.1 summarizes notations and definitions used in this chapter.

TABLE 5.1: Summarization of notations.

Notation	Purpose
G_P	Reference graph
G_Q	Query graph
P	Set of features in G_P
Q	Set of features in G_Q
N_P	Total number of data featrues of G_P
N_Q	Total number of data featrues of G_Q
C	Mapping constrains
L	A set of candidate assignments
i, j	Feature points in G_P
a, b	Feature points in G_Q
$e_1 = (i, a), e_2 = (j, b)$	Candidate assignments
M	Affinity matrix
$M(e_1, e_2)$	Pairwise affinity
$M(e_1, e_1)$	Individual affinity
S	Graph matching score
x	Indicator vector
x^*	Optimal solution
d_{ij}	Euclidean distances between the point i and j

5.3 Algorithm

The proposed iterative APRIP algorithm is an improved version of the IPFP algorithm [LHS09] that appears to be more efficient than the original algorithm, providing higher accuracy and faster computation time. The algorithm mainly contains three elements that make it different from IPFP: affinity-preserving process applied to affinity matrix, use of an initial solution produced by spectral matching method, and a slight modification of integer projected fixed-point loop. The overall algorithm is presented in Algorithm 5.1. We describe the key elements below.

- *Affinity preserving transformation.* Drawing on the idea in PageRank [TFP06, SMPU13], and following the RRWM approach in [CLL10], we propose to transform the affinity matrix into a row stochastic matrix, preserving the relative strength of edge affinity, and transforming the power iteration loop to a random walk simulation where the indicator vector x, relaxed in continuous domain, should converge to a fixed point distribution of candidate assignment values. The affinity matrix

M is converted to a row stochastic matrix M' as follows:

$$M' = M/max_{ia} \sum_{jb} (M_{ia;jb} - min_{M_{ia;jb}}). \tag{5.1}$$

The new matrice can not only maintain the original affinity but also convert the affinity matrix to a random transition matrix yielding to better performance. Next, let M' denotes this new stochastic matrix. This transformation corresponds to step 1 in Algorithm 5.1.

- *First step construction by spectral matching.* Given the new affinity matrix M', a first step consists of constructing the initial solution vector that will be the input of the improved IPFP loop. We choose to use the spectral matching algorithm SM [LH05] that computes the principal eigenvector x' of M' by power iteration as a first step, and then generates a valid (admissible) binary solution as final step, by a greedy procedure. The details of this method are shown in Algorithm 5.1 from step 2 to step 13.

- *Improved integer projected fixed point loop.* The goal of this step is to improve the current solution using an improved version of IPFP. The IPFP can be seen as an adaptation of the Frank-Wolfe [Jag13] procedure for quadratic convex optimization applied in the context of graph matching IQP. Instead of generating the binary admissible solution as a final step as in SM and RRWM, IPFP integrates this discretization step following the power iteration step itself, by Hungarian algorithm. The pseudo-code of the algorithm is shown in Algorithm 5.1 from step 14 to step 32, with t the number of iterations, and k the current iteration. The projective operation P_d, by using Hungarian method, realizes the one-to-one constraint in the discrete domain and computes a discrete solution y_{k+1}. Then, the next operation addresses a linear sub-problem where the next solution x_{k+1} is found by displacement of the previous solution x_k along a suitable direction. The role of y_{k+1} is to provide a direction along which it is expected to maximize the original quadratic score further, as presented in steps 20 to 23. A modification we have introduced, concerns the moving step factor $r = min\{1, |C/D|\}$, used in our APRIP algorithm in step 22. The factor is different to the original factor $r = min\{1, -C/D\}$ provided by [LHS09]. We empirically verified that this new factor saves substantial running time and can yield more accurate solutions. The explanation of the improvement should come from the positivity of the factor, accelerating convergence. Finally, the whole algorithm returns a stable solution x^*.

The proposed APRIP combination algorithm appears to be more efficient than the original IPFP, and to be competitive with RRWM. The following experiments analyze the related performances.

Algorithm 5.1 APRIP algorithm

Input: affinity matrix M, set of candidate assignments L, k_{th} candidate assignment e_k, and the number of iterations t

Affinity preserving $M' = M/max_{ia} \sum_{jb}(M_{ia;jb} - min_{M_{ia;jb}})$;
Set x' be the principal eigenvector of M';
Initialize the solution vector x and the set of all candidate assignments L;
e_k is one of correct assignment among L;
for $L \neq \varnothing$ **do**
    $e'_k = argmax_{e_k \subset L}(x'(e_k))$;
    if $x'(e'_k) = 0$ **then**
        return x;
    else
        set $x(e'_k) = 1$ and remove e'_k from L;
        remove assignments in confilict with e'_k from L;
    end if
end for
$k = 0$;
$x^* = x$; // Give the output of SM to the following loop for input
$S^* = (x^*)^T M' x^*$;
for $k \leq t$ **do**
    $x_k \leftarrow x^*$;
    $y_{k+1} = P_d(M' x_k)$; // The projection step
    $C = x_k^T M'(y_{k+1} - x_k)$;
    $D = (y_{k+1} - x_k)^T M'(y_{k+1} - x_k)$;
    $r = min\{1, |C/D|\}$; // Translation factor
    $x_{k+1} = x_k + r(y_{k+1} - x_k)$; // Expect maximize the original quadratic score
    if $y_{k+1}^T M' y_{k+1} \geq S^*$ **then**
        $S^* = y_{k+1}^T M' y_{k+1}$;
        $x^* = y_{k+1}$; // Update the optimal solution
    end if
    if $x_{k+1} = x_k$ **then**
        return x^*;
    end if
    $k = k + 1$;
end for
return x^*

5.4 Experiment

5.4.1 Presentation

Performance evaluation can be done by computing the affinity score, but more importantly by evaluating accuracy according to a groundtruth set of true assignment pairs, that reflect the application requirement of graph matching. We also evaluate the computation time. Based on the above IQP, objective score can be obtained by formula 5.1. Accuracy can be got by dividing the actual correct number of matches detected by the maximum number of groundtruth pairs that can be returned, and the formula is as

follows:

$$Accuracy = x^* * V_{GTbool}/maxGT, \tag{5.1}$$

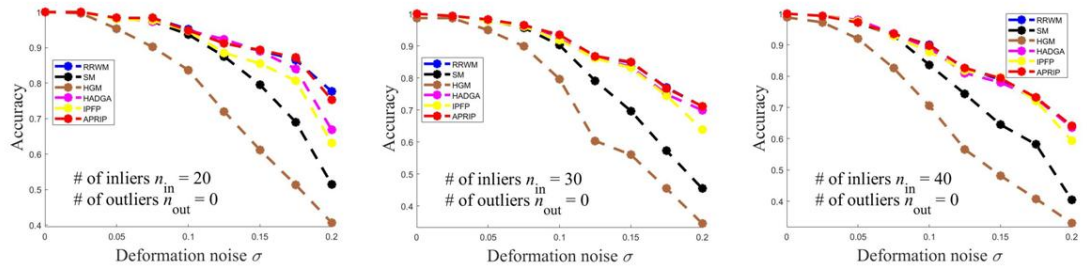
where x^* is the binary vector solution returned by the algorithm, $V_{GTbool} \in \{0,1\}^{N_P N_Q}$ is a binary vector representing the groundtruth pairs, and $maxGT$ is the maximum number of true assignments that could be returned when considering a many-to-one mapping, relaxing constraint formula 5.5. However, solutions returned by the algorithms must necessarily verify a one-to-one mapping in this chapter. They verify both constraints formula 5.4 and 5.5.

In this section, we perform experiment on synthetic data set, CMU house image data set, and real images data set. A last section presents computation times and discuss time complexity specifically. The proposed APRIP algorithm is compared with some state-of-the-art methods such as RRWM [CLL10], SM [LH05], IPFP [LHS09], HGM [ZS08], and HADGA [YZZ+15]. All of these different algorithms share the same images, feature points, and affinity matrices as input data. Each test set has a groundtruth solution for accuracy evaluation. Experiments were conducted on a CPU Intel(R) Core(TM) i5-4590 3.3 GHz.

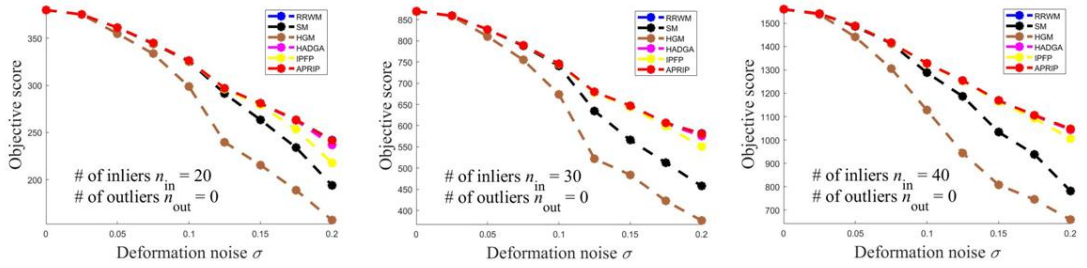
5.4.2 Synthetic image matching

In this experiment, we artificially constructed two set of feature points in some area in the plane which are related through some transformation in the plane, having n_{in} inlier points and n_{out} outlier points. This quantitative experiment is mainly divided into three types of experiments depending and their parameter setting: degree of deformation noise, number of outliers, and edge density.

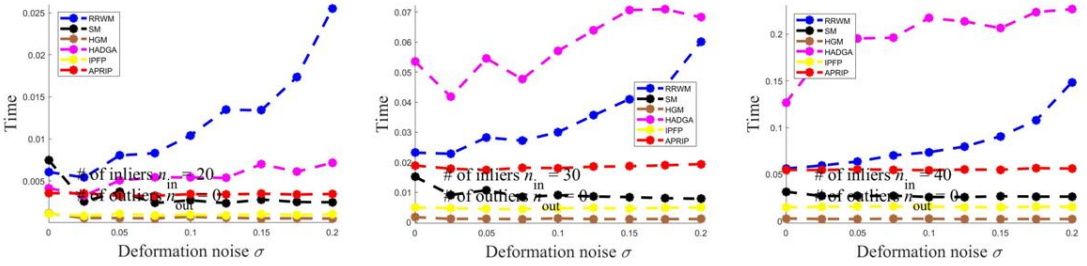
In the first type of experiment, the deformation noise is generated by using Gaussian noise distribution function $N(0, \sigma^2)$, and the deformation noise σ vary from 0 to 0.2 with the interval 0.02, while number of inliers n_{in} is set 20, 30, and 40 respectively, as shown in Figure 5.2. In the second experiment, the number of outliers varies from 0 to 20 while deformation noise σ is set to 0, 0.1, and 0.2, as shown in Figure 5.3. In the third experiment, the edge density of the reference image ρ varies from 0.1 to 1 with the interval 0.1, as shown in Figure 5.4. As we can see in these experiment results, APRIP algorithm and RRWM algorithm have almost identical objective score and accuracy matching results, but APRIP uses less computation time. Overall, the proposed method (red line) performs well with the change of inlier points and deformation noise, and it responds better in terms of time assessment relatively.



(a) Accuracy with 20, 30, and 40 inlier points (left to right).



(b) Objective score with 20, 30, and 40 inlier points (left to right).



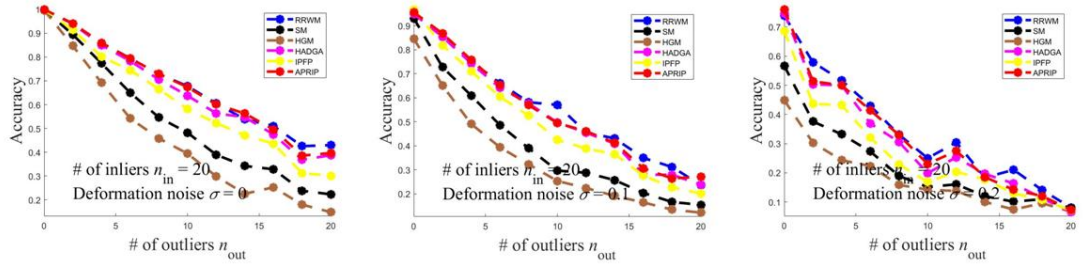
(c) Time with 20, 30, and 40 inlier points (left to right).

FIGURE 5.2: From top to bottom are the deformation noise tests according to the inlier points change under the evaluation of (a) Accuracy, (b) Objective score, and (c) Time.

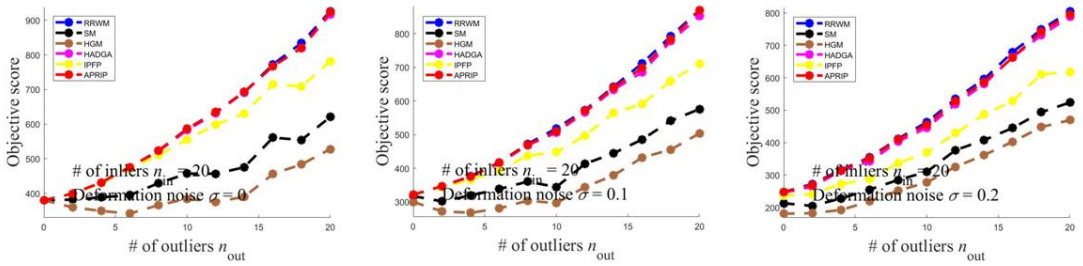
5.4.3 CMU house image matching

The experiment using CMU house sequence dataset[1] allows to evaluate matching on a sequence of increasing deformations of a same object. A total of 110 pictures in this dataset are divided into different sequence gaps (from 10 to 100 with an interval of 10). So finally we get ten sets of data pairs. Each set of image pairs consists of an initial fixed position picture (sequence 1) and of its varying transformation. To evaluate the matching accuracy, 30 iconic feature extraction points were manually tracked and marked on all frames as ground truth. In this typical and classical test, RRWM performs better than the other approaches. The experiment results are visualized in Figure 5.5, and the quantitative evaluation is reported in details in Table 5.2. For their detail results information of these 10 tests, we can refer to Table 5.3.

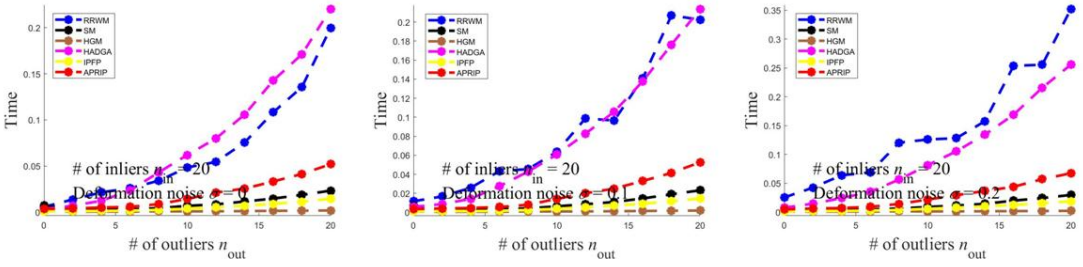
[1] *http://vasc.ri.cmu.edu/idb/html/motion/*



(a) Accuracy with 0, 0.1, and 0.2 deformation noise (left to right).

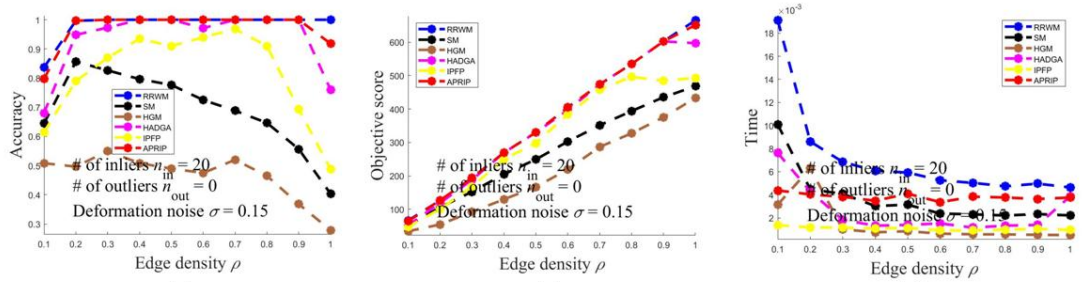


(b) Objective score with 0, 0.1, and 0.2 deformation noise (left to right).



(c) Time with 0, 0.1, and 0.2 deformation noise (left to right).

FIGURE 5.3: From top to bottom are the outliers tests according to the deformation noise change under the evaluation of (a) Accuracy, (b) Objective score, and (c) Time.



(a) (b) (c)

FIGURE 5.4: From left to right are the edge density tests under the evaluation of (a) Accuracy, (b) Objective score and, (c) Time.

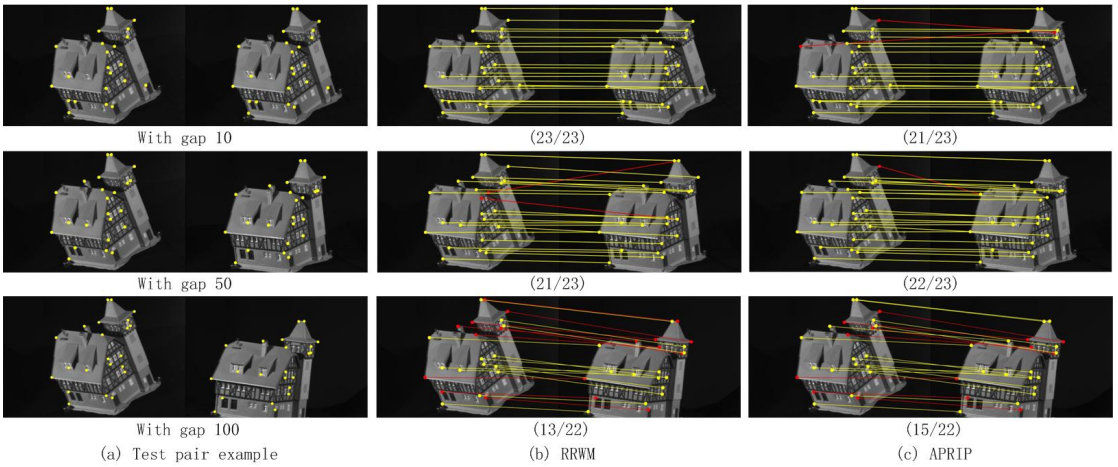


FIGURE 5.5: CMU house dataset matching result.

TABLE 5.2: Comparative evaluation on CMU database for RRWM, SM, IPFP, and APRIP.

	Methods	Accuracy	Score	Time(s)
1	RRWM	21.6/23.2	100.00	0.544
2	SM	20.9/23.2	94.51	0.194
3	IPFP	22/23.2	80.93	3.493
4	APRIP	21.4/23.2	79.52	0.490

5.4.4 Real image matching

In the experiment of real image matching, we use the CALTECH database [2] as customized in the Matlab environment by Cho et al. [CLL10], which consists of 30 different kinds of image pairs. All of the ground truths of these corresponding candidates are manually pre-labeled. Accuracy and objective score are the main judging criteria for matching. Table 5.4 shows the final average results for 30 pairs of pictures through RRWM, SM, IPFP, and APRIP algorithms. From this table, we can find that the algorithm APRIP performs much better than SM. Even if APRIP has similar accuracy and score as RRWM, it costs less computing time than RRWM when dealing with real images. APRIP was configured with less iterations IPFP, and it is much faster for an improved accuracy. Figure 5.6 shows the visual map of the feature point connections. We can see that APRIP can outperform RRWM and IPFP in some of datasets. In most cases, the matching results achieved by methods RRWM, SM, IPFP, and APRIP are comparable. The correct matches and wrong matches are marked in yellow lines and black lines, respectively. A detailed table of the results for these 30 test is given in Table A.1 of Appendix A.

[2] *https://cv.snu.ac.kr/research/ RRWM/*

FIGURE 5.6: From the left to the right: (a) RRWM algorithm, (b) SM algorithm, (c) IPFP algorithm and (d) APRIP algorithm for graph matching. (The yellow lines represent the correct matching pairs, and the black lines represent the wrong matches.)

TABLE 5.3: Detail experiment results for RRWM, SM, IPFP, and APRIP on 10 tests of CMU database.

	RRWM			SM		
	accuracy	score	time(s)	accuracy	score	time(s)
seq10	23/23	100.00	0.40	23/23	100.00	0.18
seq20	27/27	100.00	0.49	27/27	99.58	0.19
seq30	25/25	100.00	0.43	25/25	97.97	0.18
seq40	26/26	100.00	0.64	26/26	98.18	0.18
seq50	21/23	100.00	0.44	21/23	98.23	0.20
seq60	24/24	100.00	0.45	24/24	95.17	0.20
seq70	21/21	100.00	0.61	21/21	94.73	0.20
seq80	18/20	100.00	0.77	15/20	85.10	0.20
seq90	18/21	100.00	0.63	14/21	83.50	0.21
seq100	13/22	100.00	0.58	13/22	92.71	0.20
average	21.6/23.2	100.00	0.544	20.9/23.2	94.514	0.194
	IPFP			APRIP		
	accuracy	score	time(s)	accuracy	score	time(s)
seq10	21/23	82.38	1.90	21/23	82.37	0.07
seq20	27/27	79.06	1.56	27/27	79.00	0.19
seq30	25/25	83.45	2.43	25/25	83.22	0.25
seq40	26/26	78.94	1.67	26/26	78.66	0.27
seq50	23/23	81.89	4.25	22/23	81.25	0.76
seq60	24/24	86.30	2.08	24/24	85.24	0.30
seq70	21/21	73.32	5.18	21/21	72.57	0.98
seq80	19/20	78.16	5.21	18/20	76.54	0.21
seq90	17/21	81.11	5.36	15/21	77.30	0.95
seq100	17/22	84.69	5.29	15/22	79.00	0.92
average	22/23.2	80.93	3.493	21.4/23.2	79.515	0.49

TABLE 5.4: Comparative evaluation on CALTECH database for RRWM, SM, IPFP, and APRIP.

	Methods	Accuracy	Score	Time(s)
1	RRWM	12.5/21.5	96.36	0.15
2	SM	10.2/21.5	77.24	0.02
3	IPFP	11.9/21.5	97.81	0.64
4	APRIP	12.0/21.5	95.78	0.07

5.4.5 Discussion on time complexity

In the general case, when the total number of possible assignment pairs modeled in the affinity matrix is $N_P \times N_Q$, the time and memory complexity of the presented algorithms of this chapter is $O\left((N_P N_Q)^2\right)$ which is huge and limits the application to only very small instances.

We verify this point by the experiment reported in Figure 5.7 with changing the number of input feature points (inliers) of synthetic data. To compare the time consumptions, the inliers varies from 0 to 100 with the interval 10, i.e. $n_{in} = 10, 20, ..., 100$. Meanwhile, we set some fixed values: the number of outliers $n_{out} = 10$, deformation noise $\sigma = 0$, and the edge density $\rho = 1$. This experiment compares computation times of RRWM, IPFP, SM, and of our proposed APRIP algorithm.

In our Matlab environment, as shown in Figure 5.7, both computation time of RRWM and IPFP rise rapidly with instance size increasing when dealing with larger graph, but APRIP times rises relatively slowly. However, the simulation could not address larger size instances. We found that the platform could not execute instances with more than 100 feature points because of the memory complexity factor since the matrix size should be of 10^8 units, that is very large in a standard computer.

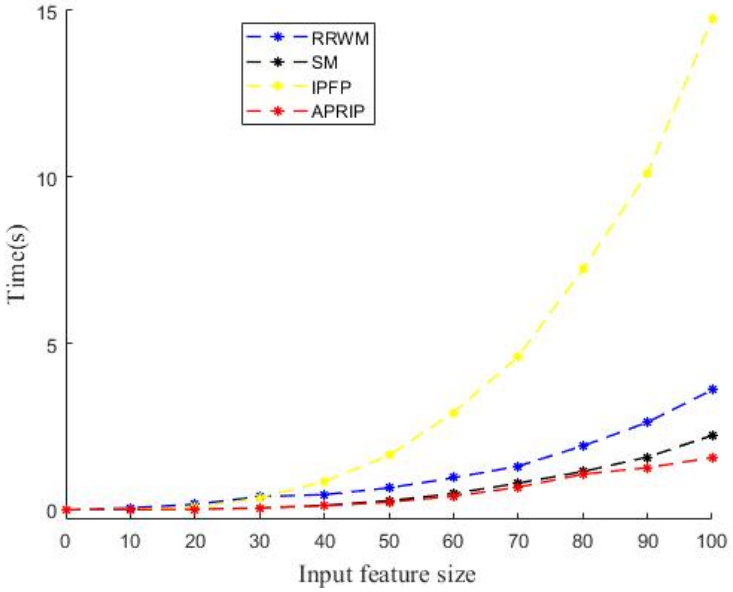


FIGURE 5.7: Time consumption for synthetic data with full affinity matrix size.

In order to address real image cases with hundred of feature points, it is necessary to pre-select a subset of candidate assignment pairs, in order to reduce the affinity matrix size. By choosing a small constant K for the maximum number of candidates corresponding points to a given point, the matrix size becomes $O\left((K \times N_P)^2\right)$. This is the case for the real images database in this experiment, for which a matching list is given as an input that defines the list of candidate assignment pairs and the matrix column size. The matching list allows to build the affinity matrix and relate its entries to the feature point indexes. As shown in Figure 5.8, computation time looks less correlated to size instance. It can be seen large variations in computation time, that looks to make behavior difficult to predict. Reducing the matrix size clearly allows faster computation time, but also requires a preliminary selection round of potential candidate pairs. This is generally done by a first-order matching round, such as K-nearest neighbor search, matching SIFT features to build a matching list, and furthermore build an affinity matrix. The memory remains in $O\left((K \times N_P)^2\right)$ and devising new algorithms with smaller memory occupations could be of interest when dealing with very large instances.

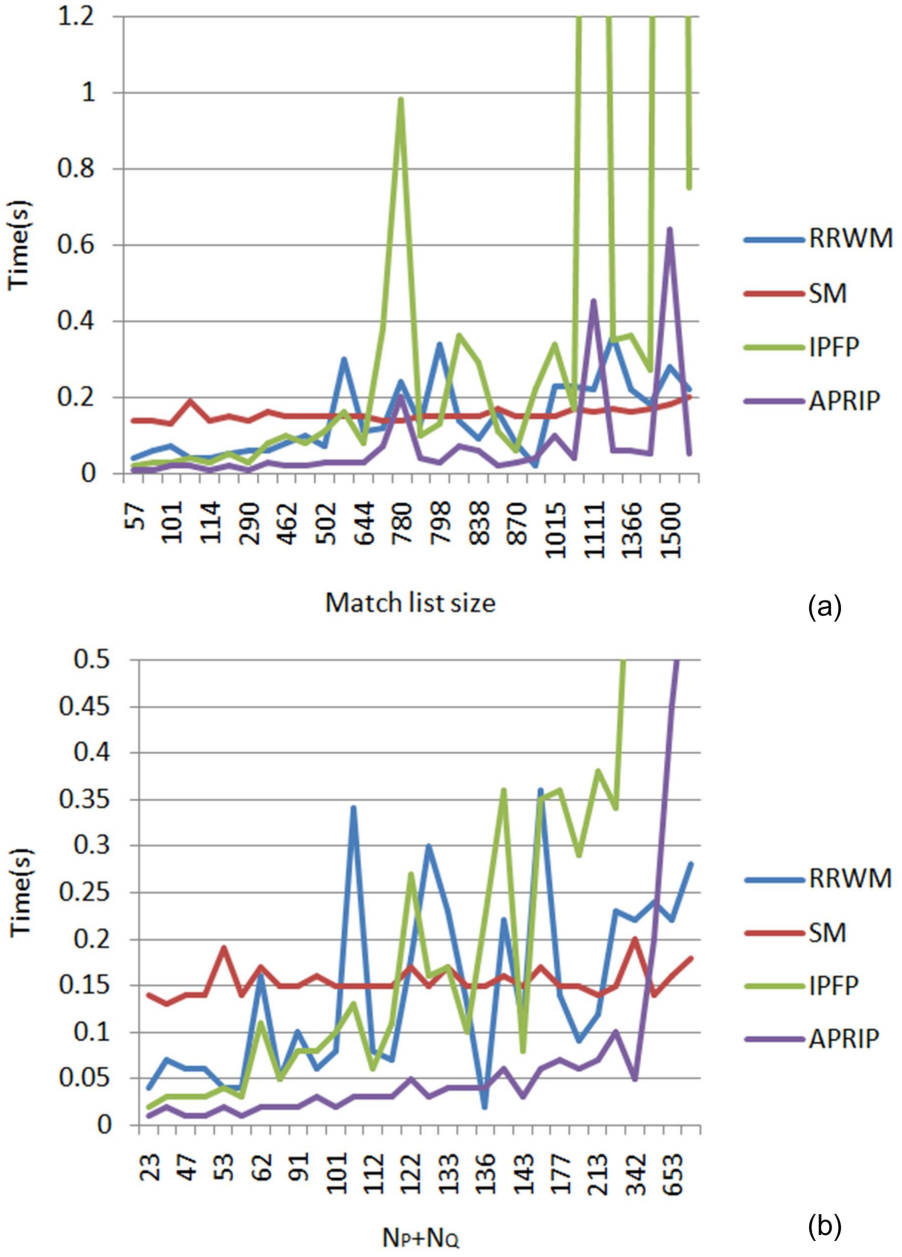


FIGURE 5.8: Time consumption for real data with customized affinity matrix.

5.5 Conclusion

We have presented how the Matlab platform allows to simulate and experiment graph matching algorithms based on standard IQP formulation, where the solution variable operates in the space of assignment pairs, of dimension $N_P \times N_Q$. Most of the approaches generally relax the binary constraints, and after following some power iteration procedure, generate the admissible solution as a final step, as in SM and RRWM. It can be the case that discretization is included in the main loop of the algorithm, as in IPFP and APRIP.

We presented a combination of the methods, called APRIP, that improves the original IPFP, and looks competitive inaccuracy, with using less computation time in most cases. All experiments were conducted on standard benchmarks and realized under a customized Matlab framework allowing comparative evaluation between the different algorithms. We observed that without restricting the matching list of candidate pairs, the algorithms require huge memory, in $O\left((N_P N_Q)^2\right)$, and could not address more than 100 input inliers. In practice, the affinity matrix must be sparse, and specific data structures must be introduced, as a matching list with the candidate pairs, to adapt the matrix computation operations to sparse graphs.

In real image cases, IQP methods are efficient sequential methods that should be carefully analyzed. In this study, we have customized a generic environment for test and evaluation of algorithms for graph matching. Based on this framework, it is now possible to extend the study to other type of development to address graph matching, as the use of GPU development, with the aim to reducing memory complexity to $O(N)$ in GPU, and avoiding the use of a large size affinity matrix by directly operating in the plane.

Chapter 6

Planar graph matching in GPU

6.1 Introduction

We have seen that the Matlab environment constitutes a useful framework for graph matching as an IQP problem. It offers useful mathematical abstractions, and it allows to develop and compare many algorithms based on a common evaluation platform, sharing input data, but also customizing affinity matrices and matching list of candidate solution pairs as input data. This allows to reuse these common data and context to start elaborate parallel GPU algorithms for graph matching applications. Since the actual development of GPU to graph matching is very sparse, we choose to reuse specific tools already developed in the context of optimization problems stated in the plane and that could present adequate properties in this context.

We reduce the dimensionality of data structures, we do not reuse the large affinity matrix. Only a matching list is considered. Feature points are points in the plane, and have coordinates in the plane, as usual, since we are interested in the geometric relationships between pairs. The indicator variable of IQP becomes a $O(N)$ mapping buffer defined by closest point search in the plane, N being feature point size, and the variable of the problem becomes now a moving graph in the plane. Euclidean distance is computing in real-time when needed to evaluate costs, to avoid a pre-computation of an affinity matrix. These standard parallel algorithms, that we adapt to GM, are respectively the Self-Organizing Map algorithm in the plane, called SOM, and a parallel local search algorithm, called Distributed Local Search (DLS). These tools can be classified as move making operations, since they consist in moving a graph, a copy of the first graph, in juxtaposition to the second graph by similarity, and evaluate the obtained assignment, through rigid or non-rigid transformations.

The DLS simply performs independent local search in parallel based on a partition of the feature set, by translation/rotation of independent rigid clusters. Each node of a cluster performs a search of a best displacement in the plane. The SOM is well known for its properties of density and topology preservation. It can be seen as an extension of the standard k-means algorithm, adding interaction between neighbor nodes of a graph. This property allows addressing elasticity constraints. The SOM will be used for clustering feature points and smoothly matching a graph of clusters to the second graph.

We present a first experiment of SOM and DLS into the context of graph matching. We first present a version of graph matching problem when stated in the plane, called here *planar graph matching*, in section 6.2. The SOM and DLS algorithms for graph matching are presented in 6.3. Finally, experiments onto the GM problem is presented in section 6.4. Comparative evaluations are performed. The last section concludes the chapter.

6.2 Definition of graph matching in Euclidean plane

6.2.1 Data instance and notations

We translate graph matching into a planar setting by considering the variable of the problem as a moving graph in the plane, that represents a copy of a first graph, that should move and match onto the feature point locations in the plane of a second graph. Notations are given such that to easily identify feature points to processors, on a one-to-one basis, such that N processors are required given a set of N points.

Main notations remain the same as previously for IQP model, except that now, variables are points in the plane of a moving graph. As previously, a pair of graphs $G_P = (P, E^P)$ and $G_Q = (Q, E^Q)$ is given, where P and Q are two sets of features with N_P and N_Q points, respectively. Without loss of generality, let $[N] = \{0, ..., N-1\}$, for any integer N, and identify feature point sets P and Q with their index set, that is, $P = [N_P]$ and $Q = [N_Q]$. This allows to identify a set of points, or a grid of points, to a grid of processors such that each point and its related processor share the same identifier index.

Since we do not compute an affinity matrix, we directly use feature point coordinates in the image plane as an input of the problem. We also reuse, as in IQP, the matching list of corresponding candidate pairs as input, instead of an affinity matrix. We note $ml \subset P \times Q$, the matching list. Recall that matching list has $N \times K$ size, for some constant K, whereas affinity matrix $(N \times K)^2$ size. Each feature point from either P

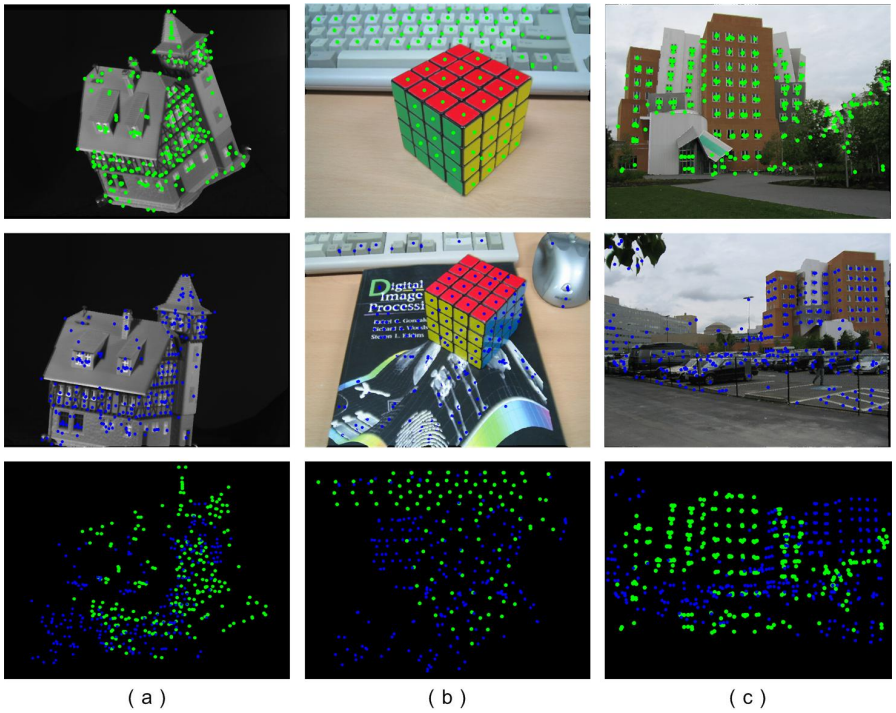


FIGURE 6.1: Feature point sets from three instances: (a) CMU, (b) Cube, and (c) Building. Bottom line present the merge points in image plane.

and Q has a location in the image plane. Let $p_i \in \mathbb{R}^2$ and $q_a \in \mathbb{R}^2$ be the points in the plane of features $i \in P$ and $a \in Q$, respectively. We shall note p and q the related vectors of point coordinates of respective sizes N_P, N_Q. The metric is the usual Euclidean distance in the plane $d(p_1, p_2)$ with $p_1, p_2 \in \mathbb{R}^2$. Three examples of feature point instances are presented in Figure 6.1. The bottom line in the figure shows the two sets of feature points merged into the image plane.

The variable becomes a vector p of feature point locations corresponding to the nodes in P. Here, we will note $p^0 = \left(p_1^0, ..., p_{N_P}^0\right)$ to represent the initial feature point locations, as given as input, and $p = (p_1, ..., p_{N_P})$, $p_i \in \mathbb{R}^2$, $i \in P$, to represent the variable. The variable has size N_P, instead of $N_P \times K$ or $N_P \times N_Q$ in IQP model.

Since the approach consists of finding a moving transformation of the first graph vertexes onto the second graph, we need to precisely define the one-to-one mapping obtained as a final solution to the problem. We give definitions for both one-to-one and many-to-one mapping, respectively, since both implementations can be customized using the same data structures. We define a one-to-one mapping as a partial function $m : P \to Q \cup \{-1\}$ by a double-checked closest point finding between the two graphs, defined as follows:

$$m(i) = \begin{cases} \arg\min_{a \in L_i}(d(p_i, q_a)) & \text{if } L_i \neq \emptyset \\ -1 \qquad \text{otherwise,} \end{cases} \tag{6.1}$$

with L_i being the Voronoï cluster of i conditioned by the matching list as defined in:

$$L_i = \{a \in Q | \, (i,a) \in ml, \forall j \in P, (j,a) \in ml, (j,a) \neq (i,a) \, , d(p_i, q_a) < d(p_j, q_a)\} \, , \quad (6.2)$$

where $ml \subset P \times Q$ is the matching list of candidate corresponding pairs. The Voronoï cluster L_i contains the second graph features in Q that are closest to $i \in P$ than any other $j \in P$. Then, a second step computes the closest point from each cluster.

The above mapping function clearly verifies a one-to-one constraint mapping, since a choice is executed according to a Voronoï partition of the second graph points. We also define a direct mapping $m_d : P \rightarrow Q$ by direct closest point finding from P to Q as follows:

$$m_d(i) = \arg \min_{a \in Q, (i,a) \in ml} (d(p_i, q_a)). \quad (6.3)$$

Since we have defined the mapping, data instance and variables, we can turn to a definition of planar graph matching.

6.2.2 Problem statement

Given the definition and notation of the previous section, we can define a planar graph matching problem by translating current score functions of IQP model into a minimization function. Here, we use the preservation of distance to model rigid transformation, as usual. Then, the goal of graph matching between G_P and G_Q, given planar locations of features in the plane and a matching list of candidate correspondence pairs, consists in finding new vertex positions p_i of graph G_P in order to minimize the following objective function:

$$f(p) = \sum_{i,j \in E^P} \left[\left| d(p_i^0, p_j^0) - d\left(q_{m(i)}, q_{m(j)}\right) \right| \right] , \quad (6.4)$$

where p_i^0, p_j^0, are the initial locations of feature points i, j respectively, as given by the input, and $m : P \rightarrow Q \cup \{-1\}$ is the double-checked closest point one-to-one mapping defined by equation 6.1. It is the locations of points p_i that determine the mapping result. Note, that we could change the one-to-one mapping by a many-to-one mapping as defined in equation 6.3, relaxing one of the two mapping constraints.

6.3 Solution method for GPU planar graph matching

6.3.1 Outline of proposed solution method

The solution method indirectly addresses the main objective by customizing and combining, two types of tools from previous work from [WZC+15, WZC17, WMC17, QC19]. They were applied to clustering, traveling salesman problem and stereo-matching. Here, we customize these tools to the context of graph matching. We present a GPU implementation of the well known self-organizing map [Koh12], called SOM, and SOMGM here, and a customized version of the idea of distributed neighborhood local search, called DLS, as presented initially by [VAS95]. Both algorithms are fully executed on GPU, by a succession of kernel calls (*i.e.* parallel GPU function calls), with only global control tasks, such as termination tests, being executed on CPU. An illustration of the solution process to planar graph matching is given in Figure 6.2.

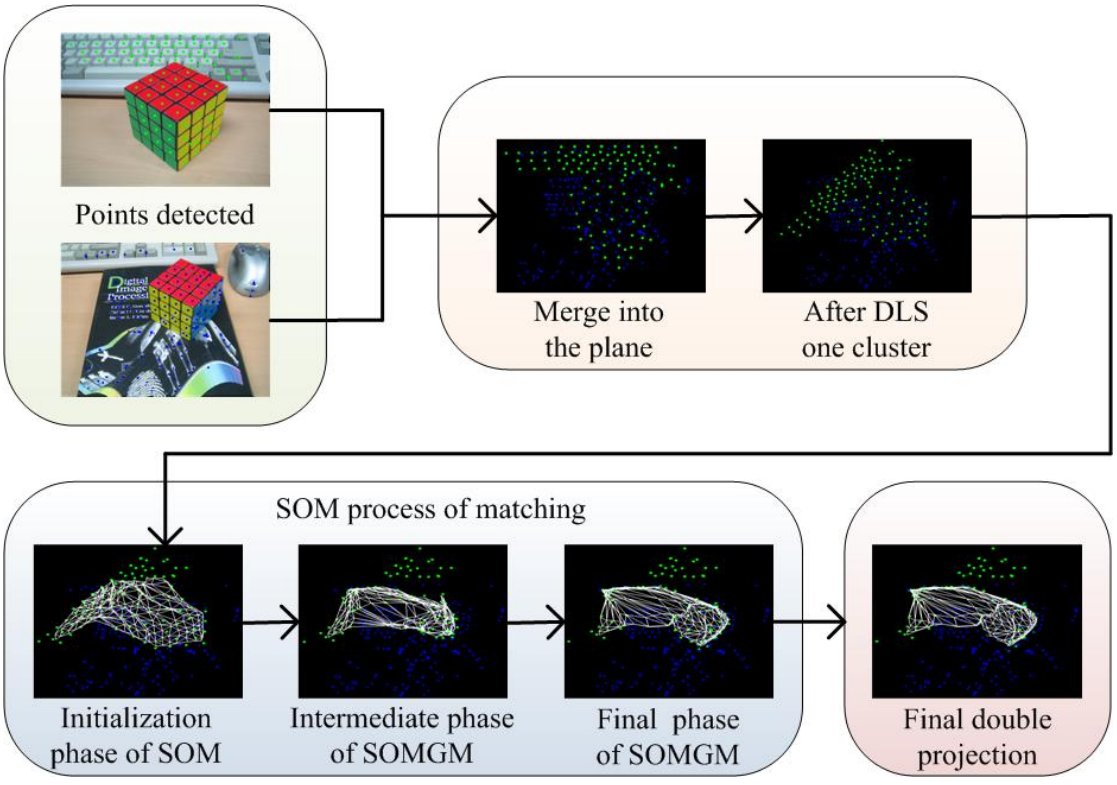


FIGURE 6.2: Flowchart of proposed DLS-SOM combination.

Many combinations of the tools are possible, whereas we only report the current best configuration. As a first step, the two instances are merged into the plane, then a round of parallel local search is performed. DLS works on a cluster map. It consists of finding a best transformation of the first point set through translations and rotations of clusters. In parallel, each feature point computes a best neighbor position through a transformation for the cluster to which it is assigned to. Then, the root of a cluster

select a best move among its cluster points, and execute the move. Here, DLS is applied to a single cluster representing the single object in the image.

Then, a self organizing map process can start. Its main property is to both preserve density and topology. A SOM process in the plane first creates an adapted mesh on the first point set. It generates clusters as well. Then, a second self-organizing map process, specifically customized to using the matching list data, and further called SOMGM, projects the first mesh onto the second point set. It is expected that the smooth transformation of the mesh, from a data set to the other, should preserve the topology of the network and its clustering. This should favor preservation of geometric structures. Finally, a double projection is performed to generate the final assignment one-to-one map. Here, a ground truth evaluation will allow to evaluate and compare the GPU process to IQP methods. Next, we present the DLS algorithm and its objective function. Then, we present the SOM and its adaptation to graph matching SOMGM.

6.3.2 Distributed local search for graph matching

We present the parallel local search method. We first present the objective function, then present a pseudo-code of the algorithm.

6.3.2.1 Objective function

The role of DLS is to match a partition of the first data point set, between clusters, to the second point set through rigid independent transformations of the individual clusters. Here, transformations are combined translation and rotation in the plane. Given a set of clusters C_k that constitute a partition of the first point set P, with K the number of clusters, the purpose is to find a set of rotation/translation transformations tr_k, one for each cluster, in order to minimize the following objective function:

$$g(tr) = \sum_{k\in[K]} \sum_{i\in C_k} d(tr_k(p_i^0), p_{c_i^{ml}}), \tag{6.1}$$

where c_i^{ml} is the closest point in Q obtained by matching list ml to point p_i, $p_{c_i^{ml}}$ its coordinates in the plane, and p_i^0 is the initial location of feature point i as given by the input. Once translated and rotated, the objective function evaluates a point-to-point distance to the second point cloud, which is to minimize. It is the rigidity of the cluster that should allow alignment to the second point set and may be good matching.

6.3.2.2 Algorithm

The algorithm behaves just like a standard local search, except that the process is executed in parallel by all the feature points of first graph G_P at the same time. Decision is taken based on a cluster partition. Each node computes the best move for its whole cluster, and once completed, the cluster root selects and applies the best move to the cluster. We report in Algorithm 6.1 the pseudo-code of the parallel kernel executed by the point set to get a best transformation related to the cluster of that point. To each node $i \in P$ corresponds a processor with the same identifier. A root map indicates for each node i, the identifier $root_i$ of its cluster root. A node can then access its root distributed list data structures to visit the cluster nodes and evaluate the cost.

Algorithm 6.1 Distributed local search for graph matching (DLSGM)

Input: $P, Q, root, tr_i, i \in P, tr_i = Id$//transform set to identity
Output: best transformations $bestTr_i, i \in P$//best transformation found by each node

1: **for** each node $i \in P$ in parallel do **do**
2: $bestTr_i \leftarrow tr_i$;
3: $improvement \leftarrow true$;
4: **while** *improvement* **do**
5: $count \leftarrow 0$;
6: $improvement \leftarrow false$;
7: **while** $count < neighborSize$ and $\neg$ improvement **do**
8: $count \leftarrow count + 1$;
9: $tr'_i \leftarrow generateTranform(tr_i)$;//generate small random move
10: **if** $isBest(tr'_i, bestTr_i, root_i)$ **then**//evaluate cluster transformation
11: $bestTr_i \leftarrow tr'_i$;
12: $improvement \leftarrow true$
13: **end if**
14: **end while**
15: $tr_i \leftarrow bestTr_i$
16: **end while**
17: **end for**
18: **return** $bestTr_i, i \in P$

In practical experiments, the intensity of a small random move was of 1 pixel translation, 0.1 degree of angle rotation. We found that a single cluster implementation behaves well because of the presence of a single object in image, as seen in Figure 6.3. In the case of a single cluster, time complexity becomes to $O(N_P)$ by processor for each evaluation. Once the parallel local search has resumed, a final one-to-one matching by double-checked projection can be performed.

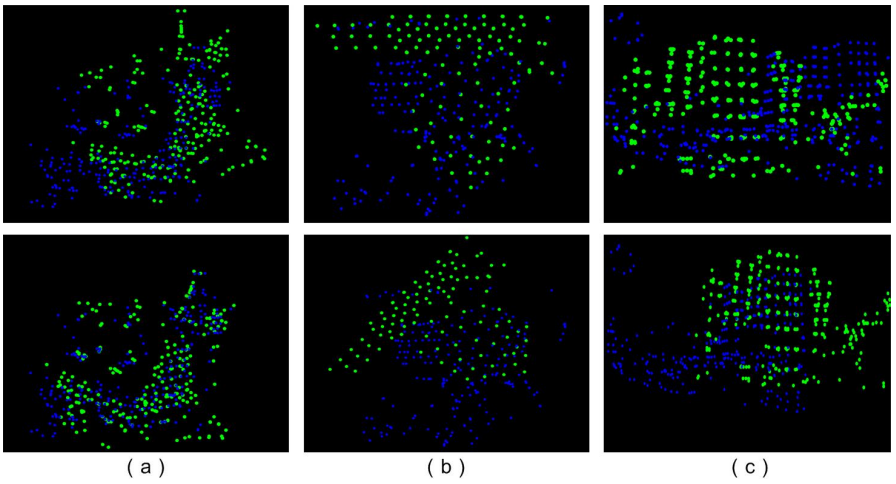


FIGURE 6.3: First line is the merge points of two images, the second line are results dealed by DLS for (a) CMU, (b) Cube, and (c) Building, respectively.

6.3.3 Self-organizing map

6.3.3.1 Self-organizing map in plane for clustering and meshing

A tool for implementing a smooth deformation from one graph to another is the self-organizing mapping (SOM) algorithm [Koh12], a neural network method that adopts a graph to some underlying data distribution, through density and topology preservation. Therefore, SOM naturally handles incomplete or noisy data as well as random requirements. We also use SOM as a center-based clustering algorithm to generate a partition of clusters by Voronoï assignation.

Here, the SOM algorithm runs on a moving graph $G = (N, E)$, with regular hexagonal topology, each vertex having 6 neighbors into the grid. Each vertex n has planar location $w_n \in \mathbb{R}^2$. The variables are the vertex locations. A distance in the graph is defined by the canonical expression $d_G(n, n') = 1$ if and only if $(n, n') \in E$, whereas the usual Euclidean distance $d(n, n')$ operates on coordinates in the plane.

The algorithm consists of triggering a moving graph G to a given data point set P in the plane. It applies a given number of parallel iterations to the moving graph as summarized in Algorithm 6.2. At given parallel iteration *niter*, points are randomly extracted from P according to probability π, with $\pi = 0.1$ in practice. Given an extracted point $p_i(t)$, a spiral search is performed in the plane to find its closest vertex n_i^* based on Euclidean distance.

Then, using learning rate $\alpha(t)$ and function profile h_t, a third triggering step is applied to all neurons in a finite neighborhood $\mathcal{N}_{n_i^*}$ of n_i^*. The neighborhood is defined by radius σ_t according to the distance in graph d_G. For each node $n \in \mathcal{N}_{n_i^*}$ a moving operation

Algorithm 6.2 Parallel self-organizing map algorithm (SOM)

Input: Moving graph G, initialized as regular grid, and data point set P
Output: Vertex locations of G and Voronoi cluster map *root*

```
t ← 1;
for t ⩽ niter do
    for each node i ∈ P in parallel with some probability π do// π = 0.1 in practice
        Find closest point n_i^* according to p_i;//Voronoi assignation
        Move neighborhood of n_i^* according to learning law;//atomic operation
    end for
    Slightly decrease intensity rate α and radius σ of neighborhood;
    t ← t + 1;//next parallel iteration
end for
return G and cluster map root
```

defined by the following Equation 6.2, called learning law, is applied:

$$w_n(t+1) = w_n(t) + \alpha(t) h_t(n_i^*, n)(p_i(t) - w_n(t)), \tag{6.2}$$

where function profile h_t is given by a Gaussian in following equation:

$$h_t(n^*, n) = exp(-d_G(n_i^*, n)^2/\sigma_t^2). \tag{6.3}$$

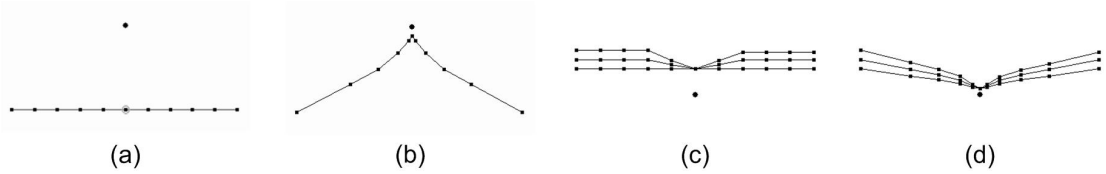


FIGURE 6.4: A single SOM iteration with learning rate α and radius σ. (a)(c) Initial configuration. (b) α =0.9, σ =4. (d) α =0.75, σ =4.

Finally, intensity $\alpha(t)$ and radius σ_t slightly decrease as geometric function of time after each parallel execution. A single SOM iteration, applied to a single point, with learning rate α and radius σ is shown in the Figure 6.4. In the GPU implementation, each parallel iteration involves $O(|N|)$ concurrent moves. Closest point findings are implemented by spiral search into a grid decomposition of the plane between cells for the fastest search. In experiments, parameters are fixed in a standard way as $(niter, \alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}) = (100, 0.5, 0.05, 12, 0.9)$.

Examples of results for cluster generation and graph generation are presented in Figure 6.5. The figures represent different sizes for graph G. We can see that the mesh shapes globally follow the point density with smooth deformations. Another way to generate clustering is the use of spanning tree, spanning forest, algorithm as shown in the last image, that we employ to get a root map with a single cluster. Here, SOM in the plane

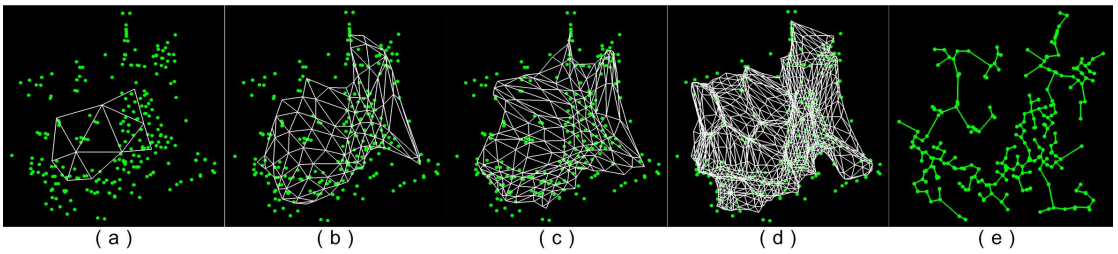


FIGURE 6.5: SOM meshing results with different cluster map sizes: (a) 2×5, (b) 7×13, (c) 9×18, (d) 21×42, and (e) single cluster by spanning tree.

will be used to initialize a minimum graph, as a first step, before applying a specific SOMGM that we describe now.

6.3.3.2 Self-organizing map for graph matching

In order to apply SOM to graph matching, we consider the two input graphs G_P and G_Q and we introduce an intermediate moving graph G. As presented in the general view, we first optimize G among the first graph G_P by Euclidean SOM to model its underlying distribution by an adapted mesh, then we apply a second round of SOM process to adapt G and its attached underlying distribution P to the second graph point set Q. The process is presented in Figure 6.6. To address good potential correspondence points, the SOM is modified as follows. After initialization by SOM in Euclidean space, the second SOM process, called SOMGM, only applies relative to the matching list of candidate correspondence pair ml, for closest point findings. As well, the triggering step also trigger point set P together with graph G point set, since P have been attached to the data distribution by the first round of SOM clustering. Finally, a double projection generates a one-to-one mapping that can be evaluated according to ground truth as the examples of sample results in Figure 6.7. The figure also presents IQP results. Note that SOM results are random and may have variations of few units. In experiments, parameters are fixed for smoothing as follows $(niter, \alpha_{init}, \alpha_{final}, \sigma_{init}, \sigma_{final}) = (500, 0.1, 0.01, 6, 0.9)$.

6.3.3.3 Objective function

There is no definitive objective function for the SOM process, but we present a function here, that should be closed to the behavior of the SOM, as we expected. We use it for its property of clustering and topology preservation. It is known that it is an extension of the very standard k-means algorithms by adding neighborhood relationships. Hence, it allows also to address problems, such as the traveling salesman problem in parallel. Given the notation as above, we should use the SOM as a heuristic for the following combination of a data cost and a smoothing cost by length minimization. Given a data

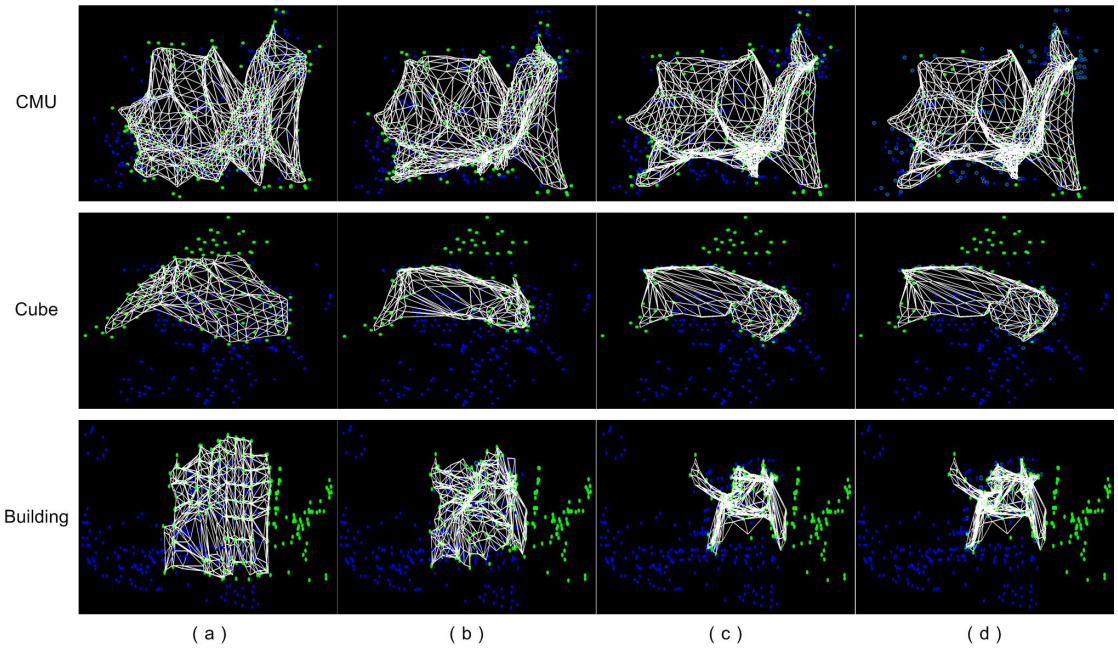


FIGURE 6.6: SOM meshing process under different steps: (a) Initialization phase of SOM, (b) Intermediate phase of SOMGM, (c) Final phase of SOMGM, and (d) Final double projection.

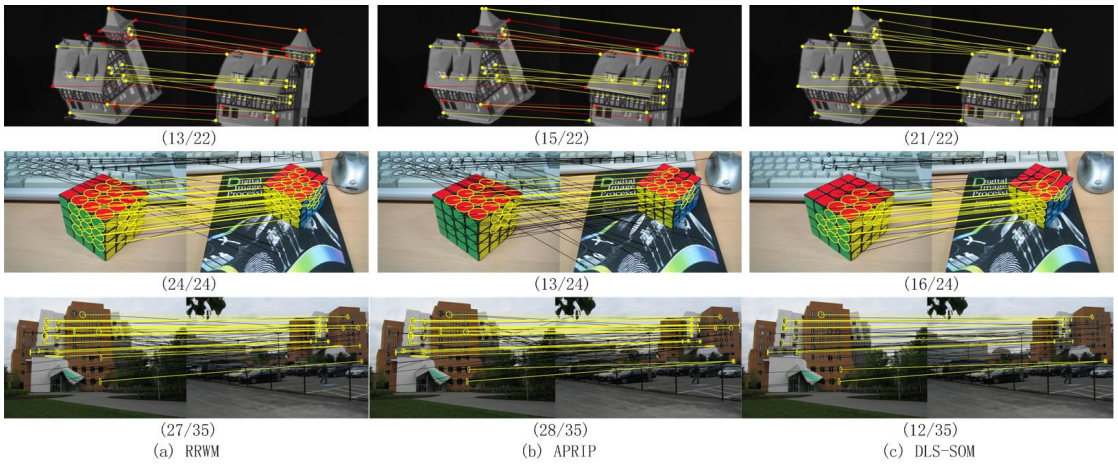


FIGURE 6.7: Matching results of (a) RRWM, (b) APRIP, and (c) DLS-SOM algorithms.

point set P and a moving graph $G = (N, E)$ with the hexagonal neighborhood, the objective function can be given by :

$$h(w) = \sum_{a \in P} d(a, c_a^N) + \sum_{(i,j) \in E} d(i, j), \tag{6.4}$$

where w represents coordinates of moving graph, c_a^N denotes the closest point to a in the moving graph G, and $d(.,.)$ is the Euclidean distance. The proposed objective function combines proximity to data points, as well as smoothing, as often encountered for image processing problems. The first term represents the k-means algorithm, whereas the

second term represents the minimization of the graph length.

6.4 Experimental evaluation

The main SOM and DLS experiments were conducted on a GPU card GeForce RTX 2070, with C++ CUDA program, Toolkit v9.0. The host computer is a CPU Intel Core(TM) i7-7700K 4.5 GHz. The experimental explanation will be divided into two parts: on CMU house sequence dataset[1] and on CALTECH database[2], as we did in chapter 5. Experiments are executed on the basis of 10 runs by instance. We report the mean value accuracy over 10 runs, and also the standard error $s_{err} = std.deviation/\sqrt{10}$. Multiplying s_{err} by factor 2.262 of Student's distribution would provide a 95% confidence interval on the mean value reported. The size of the SOM moving graph was set to four times the number of features.

6.4.1 Tests on CMU database

In this section, we do experiment on CMU data set. We first provide 10 experimental results with CMU house test set, between sequence 0 and sequence gaps from 10 to 90 with an interval of 10, as shown in Figure 6.8. Correct and incorrect matches are marked with yellow and black lines, respectively. As to quantitative evaluation, a set of experiments are conducted. Comparative experiments are divided into Projection only, SOM only, DLS one cluster, and DLS+SOM, as shown in Table 6.1. In this section, we focus on accuracy, stander error, and time as the main judging criteria. The detail data information are shown in Table B.1 of Appendix B. We also made an overall comparison chart among RRWM, SM, IPFP, APRIP, DLS one cluster, and DLS+SOM algorithms following the same experimental protocol, as shown in Table 6.2. Its detail comparison data information can be founded in Table B.2 of Appendix B.

TABLE 6.1: The summary table of comparative evaluation on CMU database for Projection only, SOM only, DLS one cluster, and DLS+SOM.

	Methods	Accuracy	Std error	Time(s)
1	Projection only	14.4/23.2	0.00	0.09
2	SOM only	19.6/23.2	0.38	0.19
3	DLS one cluster	22.08/23.2	0.20	0.43
4	DLS+SOM	21.22/23.2	0.33	0.43

In this typical and classical test, the algorithms DLS one cluster, and DLS+SOM that we proposed perform competitively to the IQP approaches. Experiment results show that

[1] *http://vasc.ri.cmu.edu/idb/html/motion/*
[2] *https://cv.snu.ac.kr/research/ RRWM/*

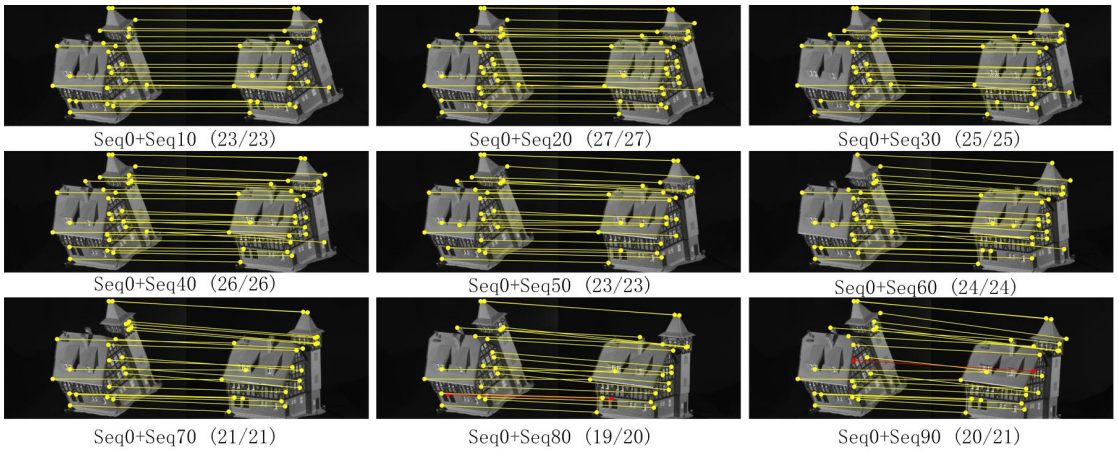


FIGURE 6.8: Matching results of DLS algorithm to CMU dataset.

TABLE 6.2: The summary table of comparative evaluation on CMU database for RRWM, SM, IPFP, APRIP, DLS one cluster, and DLS+SOM.

	Methods	Accuracy	Std error	Time(s)
1	RRWM	21.6/23.2		0.544
2	SM	20.9/23.2		0.194
3	IPFP	22/23.2		3.493
4	APRIP	21.4/23.2		0.49
5	DLS one cluster	22.08/23.2	0.20	0.43
6	DLS+SOM	21.22/23.2	0.33	0.43

the proposed algorithm has competitive accuracy on average for similar computation time. On such CMU instances, it is shown that finding a simple rotation/translation transformation, by parallel local search with a single cluster, is sufficient to achieve competitive performance, with the proposed GPU tools, against IQP methods.

6.4.2 Tests on real images

In the experiment of real image matching, we still use the CALTECH database as used in chapter 5. Accuracy, standard error and time are the main judging criteria. In this subsection, the proposed DLS-SOM algorithm is tested in two correspondence matching methods: one to one and many to one. Table 6.3 shows the final average results for 30 pairs of pictures through RRWM, SM, IPFP, APRIP, DLS-SOM (one to one), and DLS-SOM (many to one) algorithms. We give some corresponding visual maps, as shown in Figure B.1 and Figure B.2 in Appendix B for DLS-SOM (one to one) and DLS-SOM (many to one) algorithms, respectively. See the Table B.3 in Appendix B for specific data information for overall comparison chart among RRWM, SM, IPFP, APRIP, DLS-SOM (one to one), and DLS-SOM (many to one) algorithms.

From these results, we can find that the proposed DLS-SOM combination do not satisfactory addresses the real image test cases of CALTECH database in comparison to IQP

TABLE 6.3: The summary table of comparative evaluation on CALTECH database for RRWM, SM, IPFP, APRIP, DLS-SOM (one to one), and DLS-SOM (many to one).

	Methods	Accuracy	Std error	Time(s)
1	RRWM	12.5/21.5		0.15
2	SM	10.2/21.5		0.02
3	IPFP	11.9/21.5		0.64
4	APRIP	12.0/21.5		0.07
5	DLS-SOM (one to one)	7.54/21.5	0.51	0.42
6	DLS-SOM (many to one)	11.28/21.5	0.57	0.40

methods. This is not surprising, since the GPU tools provided, only indirectly adress the problem, by referring to their own objective functions, rather than to the global objective function of the planar graph matching that is not yet implemented. Moreover, this primary GPU framework is now available for further developments and improvements on basis of soon comparisons with IQP methods.

6.5 Conclusion

The main result of this chapter is that we have provided a C++ GPU framework available, that allows to address graph matching or derived sub-problems with a closed relationship with experiments on IQP models in the Matlab platform. We can adopt the same matching list data as input, and compare results based on the same context. Within this framework, we propose a combination and adaptation of the SOM algorithm and the DLS algorithm, to gauge their potential in the domain of GPU graph marching. We found that the DLS-SOM approach performs competitively to IQP models on CMU images on accuracy. The performance looks less satisfactory for real case images of the CALTECH database. Remember that the proposed GPU algorithms do not reuse the high dimensional affinity matrix, but only use lower memory complex data structures in the plane. However, relaxing the one-to-one mapping constraint to a many-to-one mapping constraint should lead to a significant improvement of accuracy. A next step should be to study many-to-one implementation in comparison to IQP. Some results on particularly complex cases, such as the Cube test case, look encouraging since they reflect the principle of a smooth deformation from one graph to the other, as expected. Further work will consist of embedding the proposed DLS-SOM tools as operators into a metaheuristic framework driven by the main objective function of the problem.

Chapter 7

Conclusion and future work

7.1 Conclusions

The main contribution of this thesis was to try to present some insightful analysis on the latest findings on local feature extraction methods, object detection for tracking, general problematic of feature correspondence, graph matching by integer quadratic program, and finally on planar graph matching by parallel GPU programs. In general, we tried to provide an overall view of the graph-based approaches to detection and matching. New combinations of methods based on a powerful framework (Matlab) were presented and a state-of-the-art experimental framework for algorithm comparisons was provided. We studied different core problems of the chain tool of graph-based detection and matching with the aim to determine which could be good candidates for parallel implementation on GPU. More precisely, about the goals to achieve, we can mention the followings elements.

We proposed new combinations of methods based on the powerful Matlab framework. The proposed system for object detection is implemented by embedding a set of MATLAB functions into a real-time video rate-driven loop. Combining the frame difference method and the background subtraction method with Laplace filters and edge detectors provides fast sparse detection. A simple and intuitive graphical user interface for target silhouette extraction in real-time monitoring has been realized.

We proposed a Marr-wavelets automatic feature detection algorithm applied to first-order correspondence based on cross-correlation that matches local feature descriptors to local feature descriptors. This graph matching framework can realize real-time execution. The proposed feature extractor tool allows improving the accuracy of the feature matching step, providing a set of highly representative object features in two images. In

terms of improving reliability, we propose to use local entropy in the meshing strategy in conjunction with the sensitive feature selection stage.

When assumption of background stability and brightness hypothesis are relaxed, with cluttered scenes, high-order potentials become necessary. The geometric relationship between pairs or k-tuple of features are the key elements for finding correspondences. We proposed APRIP algorithm for two-order graph matching. The algorithm is a combination version of elements from different IQP algorithms. It combines affinity-preserving process within a customized integer projected fixed-point algorithm improved by spectral technique. The algorithm competes in accuracy with existing state-of-the-art algorithms with the presence of clutters, deformations, and outliers, with often being faster.

Then, avoiding the requirement of a large affinity matrix, as required in the IQP model, we implemented a GPU platform based on parallel algorithms stated in the Euclidean plane. We provided a parallel algorithm for GM problem in GPU, which exploits geometry in the plane in order to save time and space complexity, running with $O(N)$ memory complexity. We combine the self-organizing map and parallel local search as the main operators for experimentation on graph matching. The proposed operators appear to be competitive on artificial (CMU) cases, even with large gap transformations, but not on real image cases actually, for similar computation times. Note that the affinity matrix has not to be computed, which is usually a time-consuming task applied before the matching process takes place in IQP. The problem was only indirectly addressed on GPU and implementing a more elaborated metaheuristic framework by embedding parallel operators should be a further task.

7.2 Future works

Because GPU parallel processors become more and more available and cheaper, we think that it should be of interest to improve our proposed parallel approach. Our main extensions of the GPU framework will focus on the following directions:

- Improve basic GPU operations of SOM and DLS in parameter setting and moving operations. Experiment different graph connection between features, such as triangulation and spanning-tree, rather than only the actual SOM meshing.
- Implement an adequate main objective function and embed the basic parallel operations into a metaheuristic framework. An implementation we envisage is a population-based memetic algorithm by using a multi-core and GPU system in conjunction.

- Extend to multiple objects matching with a map of multiple clusters instead of a single cluster. Recall that GM generally operates with a single moving object in the image.

- Implement more elaborated data structures, in order to advance one step more into the direction of GPU simulation of IQP models. For example, a good candidate for GPU implementation should be the max-pooling approach, which uses the adjacency list of neighbor nodes for graph representation with selection of only the maximum affinity links. This could be done by slight modifications of our GPU implemented data-structures.

- Another direction should be the simulation of a random walk in the adjacency graph in a distributed way rather than by using matrix product as power-iteration. A distributed version of the famous Pagerank was recently proposed by [SMPU13]. Perhaps, it could serve as a basis for adaptation to graph matching.

- Use other feature extraction methods, as our proposal, for application to graph matching in order to generate the matching list of candidate correspondence pairs by the first round of closest feature search.

- Generate large size artificial or real data instances for a better evaluation of computation time and limits of algorithms, since instances for testing GM were actually of very small sizes.

List of Figures

List of Tables

List of Algorithms

Appendix A

Experimental results of chapter 5

TABLE A.1: Detail experiment results for RRWM, SM, IPFP, and APRIP on 30 tests of CALTECH database.

	RRWM			SM			IPFP			APRIP		
Dataset	accuracy	score	time(s)	accuracy	score	time(s)	accuracy	score	time(s)	accuracy	score	time(s)
1 Cube	24/24	100.00	0.24	15/24	87.27	0.01	10/24	91.55	0.98	13/24	89.78	0.20
2 Motorcycle	9/37	100.00	0.28	9/37	91.44	0.02	13/37	99.77	9.06	14/37	94.95	0.64
3 Car	23/31	98.47	0.22	13/31	45.24	0.02	23/31	100.00	0.75	17/31	64.45	0.05
4 Butterfly	28/35	100.00	0.09	12/35	43.21	0.02	23/35	88.26	0.29	23/35	88.24	0.06
5 Dragonfly	14/29	95.98	0.08	12/29	81.05	0.02	17/29	100.00	0.10	20/29	99.06	0.02
6 Building	27/35	95.31	0.22	26/35	87.91	0.02	27/35	100.00	4.29	28/35	98.78	0.45
7 TaiChi1	4/8	100.00	0.04	4/8	95.61	0.01	4/8	95.61	0.02	4/8	99.05	0.01
8 TaiChi2	3/6	92.53	0.04	1/6	61.90	0.02	2/6	90.63	0.04	4/6	100.00	0.02
9 Face 1	8/9	92.09	0.10	8/9	83.18	0.02	7/9	100.00	0.08	7/9	95.06	0.02
10 Face 2	15/27	100.00	0.14	11/27	52.19	0.02	14/27	97.99	0.36	14/27	98.39	0.07
11 Ant	19/29	94.80	0.08	9/29	56.29	0.02	18/29	100.00	0.06	18/29	100.00	0.03
12 Cat 1	26/49	100.00	0.23	28/49	94.59	0.02	24/49	90.26	0.34	24/49	90.24	0.10
13 Cat 2	17/29	97.72	0.13	18/29	85.08	0.02	15/29	100.00	0.10	15/29	100.00	0.04
14 Crab	12/26	98.26	0.12	12/26	91.84	0.01	12/26	99.93	0.38	12/26	100.00	0.07
15 Ostrich	15/28	95.00	0.18	11/28	89.61	0.02	14/28	100.00	0.27	14/28	100.00	0.05
16 Fish 1	17/42	93.21	0.23	15/42	75.19	0.02	17/42	100.00	0.17	14/42	99.33	0.04
17 Fish 2	14/18	89.77	0.34	15/18	79.04	0.02	14/18	100.00	0.13	11/18	92.82	0.03
18 Dinosaur	5/22	92.54	0.11	10/22	67.58	0.02	7/22	100.00	0.08	14/22	89.70	0.03
19 Wrench	12/29	89.91	0.30	10/29	91.60	0.02	11/29	100.00	0.16	10/29	99.64	0.03
20 Football	3/3	92.16	0.07	3/3	68.56	0.01	3/3	100.00	0.03	3/3	99.82	0.02
21 Bicycle	5/13	95.79	0.02	3/13	80.04	0.02	5/13	99.92	0.22	5/13	100.00	0.04
22 Cattle	6/8	98.54	0.04	3/8	71.98	0.01	6/8	100.00	0.03	6/8	98.65	0.01
23 Car(side)	6/9	93.96	0.22	7/9	87.56	0.02	7/9	100.00	0.36	7/9	98.96	0.06
24 Car(back)	13/16	100.00	0.07	7/16	67.78	0.02	14/16	92.10	0.11	14/16	92.84	0.03
25 Car(front)	5/19	93.62	0.36	5/19	60.95	0.02	6/19	100.00	0.35	6/19	97.31	0.06
26 Boat	13/15	100.00	0.06	13/15	90.68	0.02	14/15	96.94	0.08	14/15	96.75	0.03
27 Face 3	8/11	97.97	0.06	8/11	86.78	0.01	7/11	100.00	0.03	7/11	99.08	0.01
28 Sheep	11/17	100.00	0.05	11/17	93.84	0.02	11/17	93.63	0.05	11/17	93.60	0.02
29 Cat 3	6/11	100.00	0.06	6/11	90.20	0.01	6/11	97.60	0.03	6/11	98.32	0.01
30 Shepherd	6/10	93.05	0.16	1/10	58.98	0.02	6/10	100.00	0.11	6/10	98.63	0.02
average	12.5/21.5	96.36	0.15	10.2/21.5	77.24	0.02	11.9/21.5	97.81	0.64	12.0/21.5	95.78	0.07

Appendix B

Experimental results of chapter 6

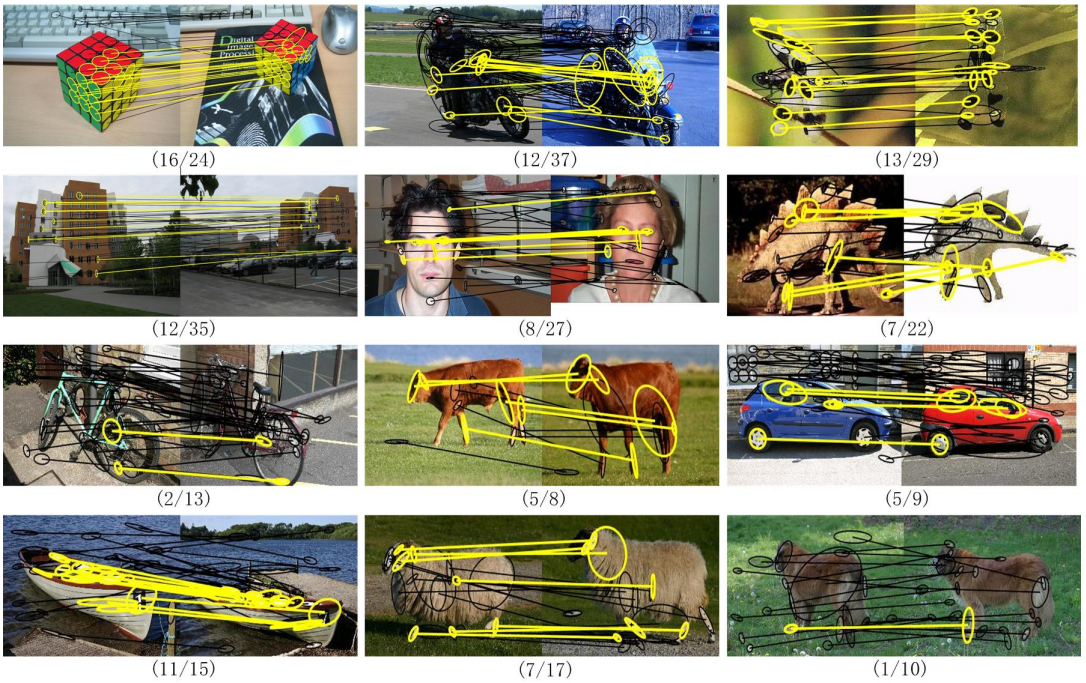


FIGURE B.1: Matching results of proposed DLS-SOM (one to one) algorithm in CALTECH dataset.

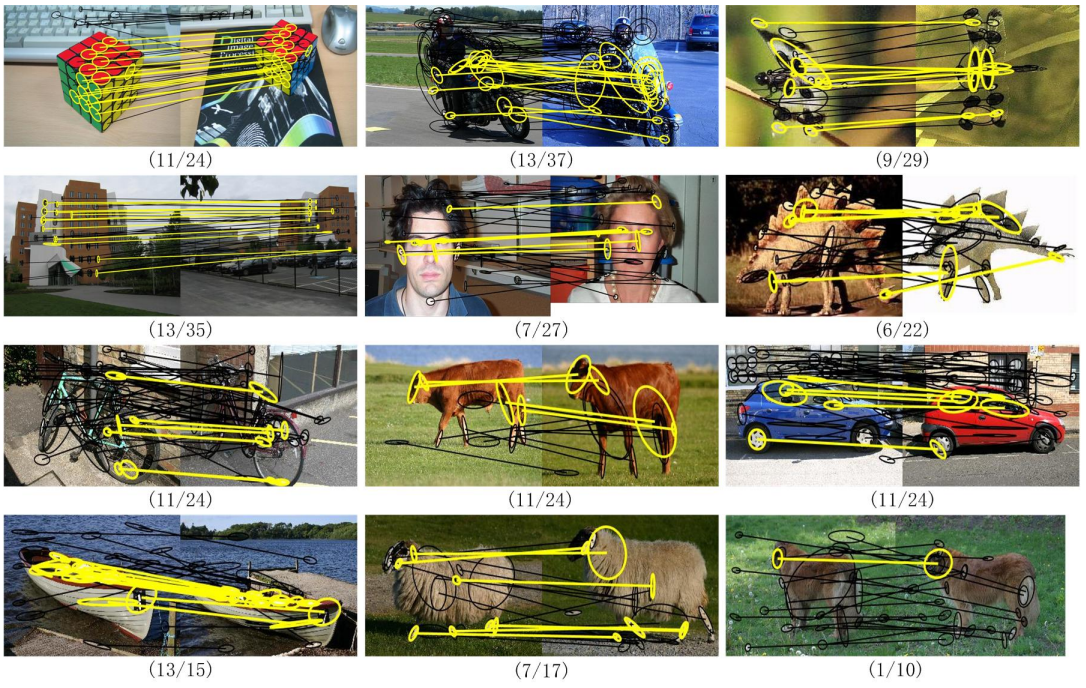


FIGURE B.2: Matching results of proposed DLS-SOM (many to one) algorithm in CALTECH dataset.

TABLE B.1: Detail experiment results for Projection only, SOM only, DLS one cluster, and DLS+SOM on 10 tests of CMU database.

	Projection only			SOM only			DLS one cluster			DLS+SOM		
Dataset	accuracy	std error	time(s)	accuracy	std error	time(s)	accuracy	std error	time(s)	accuracy	std error	time(s)
seq10	23/23	0.00	0.09	22.7/23	0.15	0.28	23/23	0.00	0.40	22.5/23	0.17	0.40
seq20	25/27	0.00	0.09	26.7/27	0.21	0.29	27/27	0.00	0.43	26.7/27	0.15	0.42
seq30	20/25	0.00	0.10	24.9/25	0.10	0.27	25/25	0.00	0.48	23.9/25	0.31	0.46
seq40	22/26	0.00	0.10	25.5/26	0.17	0.28	25.1/26	0.10	0.52	24.5/26	0.31	0.51
seq50	17/23	0.00	0.09	21.7/23	0.37	0.18	22.2/23	0.29	0.41	20.3/23	0.33	0.43
seq60	11/24	0.00	0.08	21.9/24	0.62	0.11	23/24	0.33	0.37	22.3/24	0.37	0.38
seq70	11/21	0.00	0.08	17.4/21	0.81	0.12	19.5/21	0.40	0.39	20.1/21	0.48	0.41
seq80	7/20	0.00	0.08	11.1/20	0.46	0.11	17.9/20	0.38	0.41	16.7/20	0.21	0.41
seq90	3/21	0.00	0.08	13.1/21	0.50	0.11	17.3/21	0.33	0.43	16.8/21	0.57	0.41
seq100	5/22	0.00	0.08	11/22	0.42	0.11	20.8/22	0.20	0.44	18.4/22	0.40	0.44
average	14.4/23.2	0.00	0.09	19.6/23.2	0.38	0.19	22.08/23.2	0.20	0.43	21.22/23.2	0.33	0.43

TABLE B.2: Comparative experimental results for RRWM, SM, IPFP, APRIP, DLS, and DLS+SOM on 10 tests of CMU database.

	RRWM		SM		IPFP		APRIP		DLS one cluster			DLS+SOM		
Dataset	accuracy	time(s)	accuracy	time(s)	accuracy	time(s)	accuracy	time(s)	accuracy	std error	time(s)	accuracy	std error	time(s)
seq10	23/23	0.40	23/23	0.18	21/23	1.90	21/23	0.07	23/23	0.00	0.40	22.5/23	0.17	0.40
seq20	27/27	0.49	27/27	0.19	27/27	1.56	27/27	0.19	27/27	0.00	0.43	26.7/27	0.15	0.42
seq30	25/25	0.43	25/25	0.18	25/25	2.43	25/25	0.25	25/25	0.00	0.48	23.9/25	0.31	0.46
seq40	26/26	0.64	26/26	0.18	26/26	1.67	26/26	0.27	25.1/26	0.10	0.52	24.5/26	0.31	0.51
seq50	21/23	0.44	21/23	0.20	23/23	4.25	22/23	0.76	22.2/23	0.29	0.41	20.3/23	0.33	0.43
seq60	24/24	0.45	24/24	0.20	24/24	2.08	24/24	0.30	23/24	0.33	0.37	22.3/24	0.37	0.38
seq70	21/21	0.61	21/21	0.20	21/21	5.18	21/21	0.98	19.5/21	0.40	0.39	20.1/21	0.48	0.41
seq80	18/20	0.77	15/20	0.20	19/20	5.21	18/20	0.21	17.9/20	0.38	0.41	16.7/20	0.21	0.41
seq90	18/21	0.63	14/21	0.21	17/21	5.36	15/21	0.95	17.3/21	0.33	0.43	16.8/21	0.57	0.41
seq100	13/22	0.58	13/22	0.20	17/22	5.29	15/22	0.92	20.8/22	0.20	0.44	18.4/22	0.40	0.44
average	21.6/23.2	0.544	20.9/23.2	0.194	22/23.2	3.493	21.4/23.2	0.49	22.08/23.2	0.20	0.43	21.22/23.2	0.33	0.43

TABLE B.3: Comparative experimental results for RRWM, SM, IPFP, APRIP, DLS-SOM (one to one), and DLS-SOM (many to one) on 30 tests of CALTECH database.

	RRWM		SM		IPFP		APRIP		DLS-SOM (one to one)			DLS-SOM (many to one)		
Dataset	accuracy	time(s)	accuracy	time(s)	accuracy	time(s)	accuracy	time(s)	accuracy	std error	time(s)	accuracy	std error	time(s)
1 Cube	24/24	0.24	15/24	0.01	10/24	0.98	13/24	0.20	12.5/24	1.04	0.77	14.3/24	0.54	0.67
2 Motorcycle	9/37	0.28	9/37	0.02	13/37	9.06	14/37	0.64	10.8/37	0.51	0.85	18.2/37	0.44	0.74
3 Car	23/31	0.22	13/31	0.02	23/31	0.75	17/31	0.05	12.7/31	0.54	0.57	14.3/31	0.52	0.48
4 Butterfly	28/35	0.09	12/35	0.02	23/35	0.29	23/35	0.06	18.9/35	0.81	0.42	24.5/35	0.67	0.37
5 Dragonfly	14/29	0.08	12/29	0.02	17/29	0.10	20/29	0.02	13.6/29	0.73	0.30	20.4/29	0.65	0.28
6 Building	27/35	0.22	26/35	0.02	27/35	4.29	28/35	0.45	10/35	0.87	0.75	19.6/35	1.09	0.75
7 TaiChi1	4/8	0.04	4/8	0.01	4/8	0.02	4/8	0.01	1.7/8	0.15	0.22	3.3/8	0.26	0.21
8 TaiChi2	3/6	0.04	1/6	0.02	2/6	0.04	4/6	0.02	1.6/6	0.22	0.23	1.8/6	0.33	0.22
9 Face 1	8/9	0.10	8/9	0.02	7/9	0.08	7/9	0.02	3.5/9	0.79	0.32	3.1/9	0.74	0.31
10 Face 2	15/27	0.14	11/27	0.02	14/27	0.36	14/27	0.07	4.8/27	0.53	0.34	12.5/27	0.91	0.33
11 Ant	19/29	0.08	9/29	0.02	18/29	0.06	18/29	0.03	15.8/29	0.29	0.38	21.8/29	0.79	0.40
12 Cat 1	26/49	0.23	28/49	0.02	24/49	0.34	24/49	0.10	10.8/49	0.66	0.66	20.7/49	0.83	0.64
13 Cat 2	17/29	0.13	18/29	0.02	15/29	0.10	15/29	0.04	14/29	0.68	0.45	16.7/29	0.98	0.42
14 Crab	12/26	0.12	12/26	0.01	12/26	0.38	12/26	0.07	6.4/26	0.65	0.48	12.9/26	1.08	0.46
15 Ostrich	15/28	0.18	11/28	0.02	14/28	0.27	14/28	0.05	11.5/28	0.99	0.46	10.6/28	0.91	0.44
16 Fish 1	17/42	0.23	15/42	0.02	17/42	0.17	14/42	0.04	12.6/42	0.60	0.40	20.3/42	0.97	0.39
17 Fish 2	14/18	0.34	15/18	0.02	14/18	0.13	11/18	0.03	4.6/18	0.48	0.40	7.6/18	0.48	0.38
18 Dinosaur	5/22	0.11	10/22	0.02	7/22	0.08	14/22	0.03	7.2/22	0.29	0.35	14.6/22	0.37	0.33
19 Wrench	12/29	0.30	10/29	0.02	11/29	0.16	10/29	0.03	5.1/29	0.57	0.34	8.7/29	0.47	0.33
20 Football	3/3	0.07	3/3	0.01	3/3	0.03	3/3	0.02	0.9/3	0.18	0.25	1.4/3	0.27	0.53
21 Bicycle	5/13	0.02	3/13	0.02	5/13	0.22	5/13	0.04	4.2/13	0.36	0.52	8.8/13	0.25	0.53
22 Cattle	6/8	0.04	3/8	0.01	6/8	0.03	6/8	0.01	4/8	0.33	0.27	6.1/8	0.18	0.26
23 Car(side)	6/9	0.22	7/9	0.02	7/9	0.36	7/9	0.06	3.4/9	0.31	0.38	3.5/9	0.37	0.37
24 Car(back)	13/16	0.07	7/16	0.02	14/16	0.11	14/16	0.03	4.6/16	0.31	0.47	6.6/16	0.31	0.44
25 Car(front)	5/19	0.36	5/19	0.02	6/19	0.35	6/19	0.06	5.2/19	1.15	0.37	9.4/19	0.56	0.37
26 Boat	13/15	0.06	13/15	0.02	14/15	0.08	14/15	0.03	11.5/15	0.34	0.32	13.1/15	0.38	0.31
27 Face 3	8/11	0.06	8/11	0.01	7/11	0.03	7/11	0.01	0.8/11	0.29	0.28	1.7/11	0.50	0.28
28 Sheep	11/17	0.05	11/17	0.02	11/17	0.05	11/17	0.02	7.5/17	0.31	0.31	12.2/17	0.57	0.30
29 Cat 3	6/11	0.06	6/11	0.01	6/11	0.03	6/11	0.01	4.7/11	0.21	0.27	6.8/11	0.25	0.26
30 Shepherd	6/10	0.16	1/10	0.02	6/10	0.11	6/10	0.02	1.4/10	0.16	0.39	2.8/10	0.47	0.41
average	12.5/21.5	0.15	10.2/21.5	0.02	11.9/21.5	0.64	12.0/21.5	0.07	7.54/21.5	0.51	0.42	11.28/21.5	0.57	0.40

Appendix C

Publications

Journal

1. *A systematic algorithm for moving object detection with application in real-time surveillance*
   Beibei CUI, Jean-Charles CREPUT
   Accepted for publication in SN Computer Science, Springer, 2020. DOI: 10.1007/s42979-020-0118-5

Conferences

1. *Affinity-Preserving Integer Projected Fixed Point Under Spectral Technique for Graph Matching*
   Beibei CUI, Jean-Charles CREPUT
   In Proc. of The 4th International Conference on Computer, Communication and Computational Sciences, Thailand, in Advances in Computer Communication and Computational Sciences, AISC volume 924, Springer, 2019.

2. *Using Entropy and Marr Wavelets to Automatic Feature Detection for Image Matching*
   Beibei CUI, Jean-Charles CREPUT
   In Proc. of The 15th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2019, IEEE sponsorship, Italy, 2019.

3. *Moving Object Detection and Tracking Based on Three-Frame Difference and Background Subtraction with Laplace Filter*
   Beibei CUI, Jean-Charles CREPUT

International Conference on Artificial Intelligence and Soft Computing. Springer, Cham, 2018: 3-13.

4. *Matlab GUI Application for Moving Object Detection and Tracking*
   Beibei CUI, Jean-Charles CREPUT
   International Symposium on Distributed Computing and Artificial Intelligence. Springer, Cham, 2018: 353-356.

5. *An Object Tracking Parallel Algorithm Based on Multi-Frame Difference and Background Subtraction.*
   Beibei CUI, Abdelkhalek MANSOURI, Jean-Charles CREPUT.
   In Proc. of FUTURMOB : préparer la transition vers la mobilité autonome, Montbéliard, France, 2017.

6. *Parallelisation of Optical Flow.*
   Abdelkhalek MANSOURI, Beibei CUI, Jean-Charles CREPUT, Fabrice LAURI
   In Proc. of FUTURMOB : préparer la transition vers la mobilité autonome, Montbéliard, France, 2017.

Bibliography

[AB12] Bas O Fagginger Auer and Rob H Bisseling. A gpu algorithm for greedy graph matching. In *Facing the Multicore-Challenge II*, pages 108–119. Springer, 2012.

[AM96] J-P Antoine and Romain Murenzi. Two-dimensional directional wavelets and the scale-angle representation. *Signal processing*, 52(3):259–281, 1996.

[AZ09] Antonios S Andreatos and A Zagorianos. Matlab gui application for teaching control systems. In *Proceedings of the 6th WSEAS International Conference on ENGINEERING EDUCATION*, volume 208, 2009.

[BDYR18] Iljoo Baek, Albert Davies, Geng Yan, and Ragunathan Raj Rajkumar. Real-time detection, tracking, and classification of moving and stationary objects using multiple fisheye images. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 447–452. IEEE, 2018.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[BHH11] Sebastian Brutzer, Benjamin Höferlin, and Gunther Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1937–1944. IEEE, 2011.

[BK98] Sushil K Bhattacharjee and Martin Kutter. Compression tolerant image authentication. In *ICIP (1)*, pages 435–439, 1998.

[Bou96] Paul Bourke. Cross correlation. *Cross Correlation", Auto Correlation—2D Pattern Identification*, 1996.

[CC18] Beibei Cui and Jean-Charles Créput. Matlab gui application for moving object detection and tracking. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 353–356. Springer, 2018.

[CLL09] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1280–1287. IEEE, 2009.

[CLL10] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *European conference on Computer vision*, pages 492–505. Springer, 2010.

[CSDP14] Minsu Cho, Jian Sun, Olivier Duchenne, and Jean Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2014.

[CSS07] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. In *Advances in Neural Information Processing Systems*, pages 313–320, 2007.

[Dau85] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

[DBKP11] Olivier Duchenne, Francis Bach, In-So Kweon, and Jean Ponce. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2383–2395, 2011.

[Der04] Konstantinos G Derpanis. The harris corner detector. *York University*, 2004.

[Der10] Konstantinos G Derpanis. Overview of the ransac algorithm. *Image Rochester NY*, 4(1):2–3, 2010.

[DFI+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[DLJZ10] Nannan Ding, Yanying Liu, Yongliang Jin, and Ming Zhu. Image registration based on log-polar transform and sift features. In *Computational and Information Sciences (ICCIS), 2010 International Conference on*, pages 749–752. IEEE, 2010.

[DZC89] Allan Dobbins, Steven W Zucker, and Max S Cynader. Endstopping and curvature. *Vision research*, 29(10):1371–1387, 1989.

[FSRDR14] Enrique J Fernandez-Sanchez, Leonardo Rubio, Javier Diaz, and Eduardo Ros. Background subtraction model based on color and depth cues. *Machine vision and applications*, 25(5):1211–1225, 2014.

[GDNB10] Vincent Garcia, Eric Debreuve, Frank Nielsen, and Michel Barlaud. K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching. In *2010 IEEE International Conference on Image Processing*, pages 3757–3760. IEEE, 2010.

[GHOI14] Fabian Gieseke, Justin Heinermann, Cosmin Oancea, and Christian Igel. Buffer kd trees: processing massive nearest neighbor queries on gpus. In *International Conference on Machine Learning*, pages 172–180, 2014.

[Gil98] Sébastian Gilles. Robust description and matching of images. *Ph. D. thesis, Dept. Eng. Sci., Univ. Oxford*, 1998.

[GSY+13] Liu Gang, Ning Shangkun, You Yugan, Wen Guanglei, and Zheng Siguo. An improved moving objects detection algorithm. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2013 International Conference on*, pages 96–102. IEEE, 2013.

[HCC+15] Wu-Chih Hu, Chao-Ho Chen, Tsong-Yi Chen, Deng-Yuan Huang, and Zong-Che Wu. Moving object detection and tracking from video captured by moving camera. *Journal of Visual Communication and Image Representation*, 30:164–180, 2015.

[HCW+19] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5390–5399, 2019.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[Jag13] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th international conference on machine learning*, number CONF, pages 427–435, 2013.

[JDJ19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Koh12] Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

[KPC+09] Junchul Kim, Eunsoo Park, Xuenan Cui, Hakil Kim, and William A Gruver. A fast feature extraction in object recognition using parallel processing on cpu and gpu. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3842–3847. IEEE, 2009.

[KSK08] Marek Kraft, Adam Schmidt, and Andrzej J Kasinski. High-speed image feature detection using fpga implementation of fast algorithm. *VISAPP (1)*, 8:174–9, 2008.

[KSW85] Jagat Narain Kapur, Prasanna K Sahoo, and Andrew KC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[Lav14] MP Lavanya. Real time motion detection using background subtraction method and frame difference. *Int. J. Sci. Res.(IJSR)*, 3(6):1857–1861, 2014.

[LBB+98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCL10] Jungmin Lee, Minsu Cho, and Kyoung Mu Lee. A graph matching algorithm using data-driven markov chain monte carlo sampling. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2816–2819. IEEE, 2010.

[LCL11] Jungmin Lee, Minsu Cho, and Kyoung Mu Lee. Hyper-graph matching via reweighted random walks. In *CVPR 2011*, pages 1633–1640. IEEE, 2011.

[LDW+16] Hongkun Liu, Jialun Dai, Ruchen Wang, Haiyong Zheng, and Bing Zheng. Combining background subtraction and three-frame difference to detect moving object from underwater video. In *OCEANS 2016-Shanghai*, pages 1–5. IEEE, 2016.

[LH05] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1482–1489. IEEE, 2005.

[LHS09] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and map inference. In *Advances in neural information processing systems*, pages 1114–1122, 2009.

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

[LM18] Chanho Lee and Ji-Hyun Moon. Robust lane detection and tracking for real-time applications. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):4043–4048, 2018.

[LNZ+19] Xue Lin, Dongmei Niu, Xiuyang Zhao, Bo Yang, and Caiming Zhang. A novel method for graph matching based on belief propagation. *Neurocomputing*, 325:131–141, 2019.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[LTRC11] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104. IEEE, 2011.

[LXM+14] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

[LZL+18] Chengcai Leng, Hai Zhang, Bo Li, Guorong Cai, Zhao Pei, and Li He. Local feature descriptor for image matching: A survey. *IEEE Access*, 7:6424–6434, 2018.

[MCvdM92] BS Manjunath, Rama Chellappa, and Christoph von der Malsburg. A feature based approach to face recognition. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 373–378. IEEE, 1992.

[MLSL15] Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE international conference on computer vision*, pages 2623–2631, 2015.

[MM12] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2681–2684. IEEE, 2012.

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

[MSC+96] BS Manjunath, Chandra Shekhar, Rama Chellappa, et al. A new approach to image feature detection with applications. *Pattern Recognition*, 29(4):627–640, 1996.

[Nak12] Naohito Nakasato. Implementation of a parallel tree method on a gpu. *Journal of Computational Science*, 3(3):132–141, 2012.

[NJ08] Lianqiang Niu and Nan Jiang. A moving objects detection algorithm based on improved background subtraction. In *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, volume 3, pages 604–607. IEEE, 2008.

[OLG+07] John D Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E Lefohn, and Timothy J Purcell. A survey of general-purpose computation on graphics hardware. In *Computer graphics forum*, volume 26, pages 80–113. Wiley Online Library, 2007.

[QC19] Wen-Bao Qiao and Jean-Charles Créput. Gpu implementation of boruvka's algorithm to euclidean minimum spanning tree based on elias method. *Applied Soft Computing*, 76:105–120, 2019.

[Qia18] Wenbao Qiao. *GPU component-based neighborhood search for Euclidean graph minimization problems.* PhD thesis, Bourgogne Franche-Comté, 2018.

[RAS17] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6148–6157, 2017.

[RCP+13] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. Usac: a universal framework for random sample consensus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):2022–2038, 2013.

[RK00] Phillip A Regalia and Eleftherios Kofidis. The higher-order power method revisited: convergence proofs and effective initialization. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 5, pages 2709–2712. IEEE, 2000.

[RPD10] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119, 2010.

[SB97] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

[SFPG06] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Gpu-based video feature tracking and matching. In *EDGE, workshop on edge computing using new commodity architectures*, volume 278, page 4321, 2006.

[SMPU13] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. Fast distributed pagerank computation. In *International Conference on Distributed Computing and Networking*, pages 11–26. Springer, 2013.

[SPS12] Nikos Sismanis, Nikos Pitsianis, and Xiaobai Sun. Parallel search of k-nearest neighbors with synchronous operations. In *2012 IEEE Conference on High Performance Extreme Computing*, pages 1–6. IEEE, 2012.

[SS13] C Saravanan and M Surender. Algorithm for face matching using normalized cross-correlation. *International Journal of Engineering and Advanced Technology (IJEAT) ISSN*, pages 2249–8958, 2013.

[Ste99] Charles V Stewart. Robust parameter estimation in computer vision. *SIAM review*, 41(3):513–537, 1999.

[TFP06] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast random walk with restart and its applications. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 613–622. IEEE, 2006.

[TGY+10] Chen Tang, Tao Gao, Si Yan, Linlin Wang, and Jian Wu. The oriented spatial filter masks for electronic speckle pattern interferometry phase patterns. *Optics express*, 18(9):8942–8947, 2010.

[TKBM99] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 255–261. IEEE, 1999.

[TXZ+19] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco C Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and cnn-based optical flow techniques. *Signal Processing: Image Communication*, 72:9–24, 2019.

[UO17] Nikolai Ufer and Bjorn Ommer. Deep semantic feature matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6914–6923, 2017.

[VAS95] MGA Verhoeven, Emile HL Aarts, and PCJ Swinkels. A parallel 2-opt algorithm for the traveling salesman problem. *Future Generation Computer Systems*, 11(2):175–182, 1995.

[VR09] Cristina Nader Vasconcelos and Bodo Rosenhahn. Bipartite graph matching computation on gpu. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 42–55. Springer, 2009.

[vVYB89] Lucas J van Vliet, Ian T Young, and Guus L Beckers. A nonlinear laplace operator as edge detector in noisy images. *Computer vision, graphics, and image processing*, 45(2):167–195, 1989.

[Wan15] Hongjian Wang. *Cellular matrix for parallel k-means and local search to Euclidean grid matching*. PhD thesis, Belfort-Montbéliard, 2015.

[WGY17] Qi Wang, Junyu Gao, and Yuan Yuan. Embedding structured contour and location prior in siamesed fully convolutional networks for road detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):230–241, 2017.

[WHD10] Muyun Weng, Guoce Huang, and Xinyu Da. A new interframe difference algorithm for moving target detection. In *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 1, pages 285–289. IEEE, 2010.

[WMC17] Hongjian Wang, Abdelkhalek Mansouri, and Jean-Charles Créput. Cellular matrix model for parallel combinatorial optimization algorithms in euclidean plane. *Applied Soft Computing*, 61:642–660, 2017.

[WZC+15] Hongjian Wang, Naiyu Zhang, Jean-Charles Créput, Julien Moreau, and Yassine Ruichek. Parallel structured mesh generation with disparity maps by gpu implementation. *IEEE transactions on visualization and computer graphics*, 21(9):1045–1057, 2015.

[WZC+16] Hongjian Wang, Naiyu Zhang, Jean-Charles Créput, Yassine Ruichek, and Julien Moreau. Massively parallel gpu computing for fast stereo correspondence algorithms. *Journal of Systems Architecture*, 65:46–58, 2016.

[WZC17] Hongjian Wang, Naiyu Zhang, and Jean-Charles Créput. A massively parallel neural network approach to large-scale euclidean traveling salesman problems. *Neurocomputing*, 240:137–151, 2017.

[Yan10] Zhuo Yang. Fast template matching based on normalized cross correlation with centroid bounding. In *2010 International Conference on Measuring Technology and Mechatronics Automation*, volume 2, pages 224–227. IEEE, 2010.

[YB18] Mehran Yazdi and Thierry Bouwmans. New trends on moving object detection in video images captured by a moving camera: A survey. *Computer Science Review*, 28:157–177, 2018.

[YH09] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *Circuits, systems and signal processing*, 28(6):819, 2009.

[YZZ+15] Junchi Yan, Chao Zhang, Hongyuan Zha, Wei Liu, Xiaokang Yang, and Stephen M Chu. Discrete hyper-graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1520–1528, 2015.

[ZDX+07] Chaohui Zhan, Xiaohui Duan, Shuoyu Xu, Zheng Song, and Min Luo. An improved moving object detection algorithm based on frame difference and edge detection. In *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on*, pages 519–523. IEEE, 2007.

[Zha13] Naiyu Zhang. *Cellular GPU Models to Euclidean Optimization Problems: Applications from Stereo Matching to Structured Adaptive Meshing and Traveling Salesman Problem.* PhD thesis, 2013.

[ZL10] Lijing Zhang and Yingli Liang. Motion human detection based on background subtraction. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 1, pages 284–287. IEEE, 2010.

[ZS08] Ron Zass and Amnon Shashua. Probabilistic graph and hypergraph matching. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[ZWQ12] Yanzhu Zhang, Xiaoyan Wang, and Biao Qu. Three-frame difference algorithm research based on mathematical morphology. *Procedia Engineering*, 29:2705–2709, 2012.

[ZZL+17] Zuofeng Zhong, Bob Zhang, Guangming Lu, Yong Zhao, and Yong Xu. An adaptive background modeling method for foreground segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1109–1121, 2017.