Moving Object Reconstruction on Background Mosaics of Dynamic Video Sequences

Shen Hui

October 10, 2004

Acknowledgement

It goes without saying that the process of doing the research and writing a thesis is really a collaborative effort. Therefore I would like to thank everyone who has made this thesis possible, bearable, or both.

I have been lucky to join the CHIME\DIVA lab, which allowed me to work with so many brilliant researchers. First and most importantly, I would like to show my gratitude to my supervisor Dr.Mohan Kankanhalli. He continuously gives me the most valuable guidance, helpful comments, and insightful criticism, which are absolutely necessary in my academic life. Besides, he also encourages me a lot when I met difficulties or feel depression during my research. I believe what he has given me is the most that an supervisor can give his students. I am also very grateful to Dr.Sengamedu Hanumantharao Srinivasan for his remarkable directions and advices. He has helped me continuously, even after he left the DIVA group during my research period. He has given me a lot of helpful advice and contributed tremendously to my work, especially in the beginning part. Moreover, I want to show my thankfulness to Dr.Yan Weiqi, for his work in mosaic and shaky removal, which enlightened my idea, also for his valuable discussion and suggestions. My examiners, Leow Wee Kheng and Ng Teck Khim, gave me quite valuable suggestions and comments to help me amend my thesis. I want to show my thankfulness to them too. The rest of the DIVA students past and present were no less critical to my research. In alphabetical order, I want to acknowledge Abhinav Singh, Achanta Shri Venkata Radhakrishna, Chen Lei, Chitra Lalita Madhwacharyula, Ji Yi, Meera Gajanan Nayak, Mohammad Awrangjeb, Pradeep Kumar Atrey, Stephen Bissol, Wang Jun, and Zhang Sheng. Their selfless efforts have significantly aided my work. I also would like to thank School of Computing at the National University of Singapore

for providing the research facilities that permitted me to complete this thesis.

Finally, I want to express my deepest love and gratitude that I feel for my parents. They instilled in me the love of learning from my childhood and always demanded that I live up to my ability. During this research period, they have been supporting me and encouraging me from the very beginning, which makes me feel confident to overcome any difficulty, and feel free to do exactly what I like to do, not only at present time, but also in the future. I owe everything that I have done and I will do to them.

Contents

1 Introduction **9**

1.1 Background 9

1.2 Motivation 11

1.3 Problem statement 12

1.4 Overview 13

2 Related Works **14**

2.1 Related works about static mosaic 14

2.2 Related works about image registration 25

2.3 Related works about moving object detection 26

3 Algorithm **33**

3.1 Static Mosaic Construction 34

3.1.1 Video Frame Separation 35

3.1.2 Corresponding Points Establishment 35

3.1.3 Forward Homographies Computation 38

3.1.4 Bounding Box Computation 40

3.1.5 Backward Homographies Computation 41
3.1.6 Mosaic Image Integration 42
3.2 Coarse Frame Registration 48
3.2.1 Reference Matrix Computation 48
3.2.2 Frame Warping and Registration 50
3.3 Moving Objects Detection 51
3.3.1 Difference Picture Computation 52
3.3.2 Optical Flow Computation 57
3.3.3 Moving Blocks Detection 63
3.3.4 Moving Blocks Refinement 64
3.4 Moving Objects Reconstruction 67
3.4.1 Moving Objects Separation 67
3.4.2 Static Background Construction 69
3.4.3 Pixel-Based Reconstruction 69

4 Experimental Results **74**

5 Conclusion **84**

List of Tables

2.1 A Taxonomy of plenoptic functions. 15

3.1 Summarization of Two Color Blending Algorithms. 45

4.1 Quantitative Evaluation of Examples 81

List of Figures

3.1 Corresponding Points for Static Scene 37

3.2 Corresponding Points for Dynamic Scene 38

3.3 Two Input Images for Static Mosaic Construction. 46

3.4 Mosaic for Building without Edge Blending Technique. 47

3.5 Mosaic for the Entire Scene without Edge Blending Technique. 47

3.6 Mosaic for Building with Edge Blending Technique. 47

3.7 The Relationship among Matrices in Mosaic Construction 49

3.8 Two Input Images in Static Scene 53

3.9 The Difference Picture of Static Scene 53

3.10 Two Input Images in Dynamic Scene 54

3.11 The Difference Picture of Dynamic Scene 54

3.12 Two Input Images with both Camera Motion and Object Motion 55

3.13 The Difference Picture before Image Registration 55

3.14 Two Input Images after Image Registration 56

3.15 The Difference Picture after Image Registration 56

3.16 The change of image intensity equation constrains the optical flow velocity 60

3.17 The Aperture Problem. 60

3.18 The Motion Constancy Property in Successive Frames. 66

4.1 Selected Frames from Walking Video Sequence. 75

4.2 Mosaic Image after Static Mosaic Construction. 75

4.3 Coarse Frame Registration. 76

4.4 Moving Blocks Detection. 77

4.5 Pixel Compensation in Moving Objects. 78

4.6 Moving Objects Reconstruction on the Static Background Mosaic. 79

4.7 Shake Removal. 80

4.8 Two Input Images. 82

4.9 Static Mosaic Image. 83

Abstract

Traditional video consists of frames along the time axis, so we need many frames to represent a complete scene. If we change frame-based video to be scene-based, i.e., make a mosaic from the frame sequence, we can get a more efficient representation of the scene without any redundant information.

However, a mosaic deals only with the static scene information and has difficulties in displaying moving objects if the scene is dynamic. It would be more useful if we can retain motion information, that is the dynamic information, which is one of the main advantages of video. Therefore, here, we have developed a novel technique to reconstruct moving objects on the static background mosaic.

Our approach is based on the separation of static and dynamic information in a video sequence. Then we build the mosaic from the background information and reconstruct the moving objects using dynamic information. The layer separation is again based on mosaic so that our algorithm is simple, integrated, as well as efficient. Moreover, this technique has been tested for real videos and achieved pleasing result.

Actually, in our work, we try to convert the video from its traditional format, which is inefficient and hard to manipulate, to a novel representation, which is more efficient, easy to access and control, without any information loss. Hence, our work can be viewed as a starting point to a technique which can completely decompose video into some descriptors and reassemble them in a new format according to users' need or application requirements.

Chapter 1

Introduction

This thesis attempts to improve the traditional mosaicing technique so that it can be applied on digital video with moving objects instead of only on static image sequence. The two main improvements that we want to explore are: separate the moving layer from the static layer in the video input and combine the static as well as the dynamic data together.

1.1 Background

Digital video is a rich source of information. It provides spatial information as well as temporal information so that viewers can observe the dynamics of a scene. However, video sequence also has large amount of information redundancy because there is a large overlap of information across consecutive frames. Therefore, researchers have developed many different methods to represent a digital video more efficiently and make the access and control of digital video more convenient. For example, we can encode the video in the MPEG format using motion compensation so that it requires less storage space. We can also find "key frame" with techniques such as "user attention

model" or "action retrieval" to make video indexing so that users can search and access the most interesting periods quickly and efficiently.

Among so many different techniques, mosaicing has been an active area of research field for a long time. One of the most important usage of mosaic is for panorama. Traditional video is based on frame, so we need many frames to represent a complete scene, even if actually those frames have large overlapped part. However, if we change frame-based video to be scene-based, i.e., we "stitch" the frames of the same scene together and remove the overlapped part, we can get a large panoramic image which can represent the original scene without any redundant information. Thus, mosaic can be used to gain more efficient storage, representation, and transmission of digital video data.

For example, we can add some "frame-to-scene" conversion in video encoding process as a novel compression method. In this situation, the unit of storage is scene rather than frame, which has no redundant information. When we want to reconstruct the original video, we can do the reverse conversion.

Another usage of mosaic is to enhance resolution. Some lens can help to get panoramic image, without the help of mosaic techniques, but the mosaic can do more. Take for an example, in a video we want to show a large museum to viewers. Of course we can take a complete view of the whole building, but due to the resolution restriction of the video camera, the scene will be very small and difficult to observe. Alternatively, we can capture the building from doors to windows, or from bottom to top, to make every frame larger and clearer, but in this way, there will be large overlapped part between every two adjacent frames because of the camera moving, and it's not

so easy for the viewers to know how this building exactly looks like. In this situation, utilizing mosaics is a good method to perform image-based rendering, which can show each part of the scene clearly as well as it allows for the gain of high resolution.

1.2 Motivation

Till now, traditional mosaic techniques are all for static image sequences. The scene is static and the final mosaic is also static. However, one of the most important properties of video is that there are various kinds of motion in it, not only the camera motion, but also the object motion. Therefore, we argue that a complete mosaic should also include the moving parts based on the original video.

Concerning the static mosaic we talked before, we can get rid of redundant information and view the complete scene with high resolution. Then, what else advantages can we gain if we succeed in building dynamic mosaic? Besides those mentioned before, another important advantage only of dynamic mosaic is that we can get clear knowledge of how the objects move in the scene.

For example, suppose we have a video clip of football game. Traditionally, the cameramen will trace the football and capture the players who are close to the ball. In this situation, if there is little information about background, it's difficult to tell in which direction the football is moving, from left to right, or from middle court to gate. If we can make a static mosaic of the whole court, and place the "moving" football on the mosaic, then viewers can see clearly how the football is moving, because the background is complete and static.

Actually, there do exist some techniques which can deal with video input with moving objects,

but what they have done is only to extract moving objects from background or to display the moving trajectory, not to reconstruct the entire moving objects. Therefore, the dynamic video mosaic we want to build here, is the one which not only can represent the background completely, but also can reconstruct the moving object completely to show its trajectory as well as the object itself.

If we are able to do this, there are many kinds of application which can use this techniques. For example, we can make panoramic video, which is the video with complete and static background. Also, we can deal with camera shaking and remove it totally. The idea behind this application is that we build the mosaic for the original video first, in which the camera shaking between frames disappears naturally, then we reconstruct the frames from the sprite in a stable manner. After these two steps, we can get perfect video under a real "stable" virtual camera. Conversely, we can also add shakes to static video to add excitement.

1.3 Problem statement

Strictly speaking, the problem can be stated as following. Given a video sequence input, we should be able to detect the foreground moving objects and separate them from the static background. After we get these two different "layers" of the scene data, we apply traditional mosaic technique only on static part to build a large panoramic background, which represents the scene completely and concisely. The most important improvement is done with moving part. We want to place the detected moving part on their corresponding location in panoramic background so that viewers can have a clear view of how the objects is really moving with respect to the background as well as other moving objects.

The ideal system should be fully automatic. The input is video with moving camera and moving object, of lower resolution. After a series of automatic processing, the output is still a video (that's why it is called "dynamic mosaic"), with a static "virtual camera" and the same moving objects, having a higher resolution.

1.4 Overview

The rest of the thesis is organized as follows.

In Chapter 2, we review the past work of the mosaicing technique. It includes those which depend more on hardware and those which depends more on software. The static image mosaic falls into the category of software, which is the foundation of our work.

Chapter 3 describes the algorithm we used in the system. Actually, the procedure can be divided into three steps: preprocessing, layer separation, and object registration. The technique used in preprocessing is almost the same as what is used in static image mosaic, so we will focus on the next two steps.

Chapter 4 presents the result of dynamic mosaic. The experiment demonstrates how the algorithm can be applied to some practical video and achieve pleasing result. The sample video we used in our experiment is recorded by ourselves, because our work is done on video with special properties, i.e., moving camera and moving object. Therefore, we have to pick the video of this kind to demonstrate the algorithm clearer and better.

Chapter 5 gives the conclusion of our work. What we did is just the first step towards dynamic mosaic. We summarize our contributions and discuss some problems for future research.

Chapter 2

Related Works

We will explore three areas of previous research in this section: static mosaics, image registration and moving object detection, which form the basis of our work. There are many works in all of these areas, but we will only include latest works which are most relevant.

2.1 Related works about static mosaic

A mosaicing technique renders a complete image from a set of small images or a series of video frames, which is tightly related to image-based rendering. According to [9], image-based rending techniques can be classified into several categories according to the data dimension and viewing space. The one with almost complete information is the 7D plenoptic function, which was defined as the intensity of light rays passing through the camera center at every location, at every possible viewing angle, for every wavelength and at any time. By ignoring time and wavelength, we generate the 5D plenoptic function. The Lumigraph and Lightfield systems presented a clever 4D parameterization of the plenoptic function if the scene can be constrained to a bounding box.

Finally, if the viewpoint is fixed and only the viewing directions and camera zoom can be altered, the plenoptic function simply becomes a 2D panorama.

Table 2.1 describes the above classification clearly.

Dimension	Viewing Space	Name	Reference
7	free	plenoptic function	[39]
5	free	plenoptic function	[21]
4	inside a 3D box	Lightfield/Lumigraph	[36], [29]
3	inside a 2D circle	concentric mosaics	[9]
2	at a fixed point	panorama	[35], [38]

Table 2.1. A Taxonomy of plenoptic functions.

The main area of our work is the 2D panorama, but we still explore some techniques on higher dimension.

Panorama mosaic as a field of research came into existence at the beginning of 1990s. A number of techniques have been developed in this area, which can be roughly divided into two categories.

The first category mostly depends on the hardware. One way is to record an image onto a long firm strip using a panoramic camera to directly capture a cylindrical panoramic image directly [16]. Another way is to use a lens with a very large field of view such as a fisheye lens. In [7], such kind of camera is used to capture a distortion-free 360 view of the scene so that we can capture it all and view what we want.

The hardware method of mosaic is easy to use, and it requires little additional work. However, it is less flexible and not so convenient to apply on all kinds of applications, especially when people

want some unusual effects of mosaic, such as shake artifact removal or moving object detection.

The second category depends more on software, and this category is more tightly related to our work, so we will focus our attention on this kind of method. In order to describe the techniques more clearly, we again divide the "software" method into two parts.

The first part is about the techniques which totally deal with static scenes.

Before we build the mosaic, first we have to get the relative position of two or more images. In [18], Anandan et al give the traditional way of motion estimation, the beginning step of which is actually also the traditional way to get the corresponding points in two images. The method is the minimization of sum-of-squared-differences (SSD) of the Laplacian filtered intensity images. This algorithm is done separately at each level, starting from a user-specified coarse level and refining the result down to the finest level (usually it is the resolution of the original image). This method can get the result as accurate as possible because it iterates several times within each level, but it requires very large amount of calculation and needs long time.

The hierarchical techniques used in [18] is actually the pyramid based image encoding and decoding scheme, and the details can be found in [31]. We outline the coding scheme as follows. For the original image, we get the predicted image using a unimodal Gaussian-like (or related trimodal) weighting function centered on each pixel. The predicted value for each pixel is first obtained by convolving this weighting function with the original image. The result is a low-pass filtered image which is subtracted from the original. The subtraction result is the prediction error. Rather than encode the original image, we encode the prediction error and the filtered image, which results in a net data compression, because (a) the prediction error is largely decorrelated,

so may be represented pixel by pixel with many fewer bits than the original, and (b) the filtered image is low-pass filtered, so it may be encoded at a reduced sample rate. This process can be iterated and results in a tapering pyramid data structure. This image representation is referred as "Laplacian-pyramid code". This is a classical image coding scheme with many attractive features. First, we can choose the parameters of the encoding and quantizing scheme by ourselves so that we can substantially reduce the entropy in the representation and simultaneously stay within the distortion limits imposed by the sensitivity of the human visual system. Second, this encoding scheme requires relatively simple computations, which are local and maybe performed in parallel, so that we can perform them in real time applications. Third, in this representation, image features of various size are enhanced and are directly available for various image processing. Therefore, we can use this scheme to downsample and upsample images to perform our task if necessary.

In [40], Bhosle et al describe a new method for automatic image mosaic. The novel part of their method is to use geometric hashing in the image alignment step to reduce time complexity. The algorithm adopts geometric properties such as angle formed by vectors and length between points as parameters to build hash table between two images, and compare these values in the hash table to get possible match. This method takes much less time in finding transformation between two images, which is better than [18], but the method used in mosaic building step is not so satisfactory. They deal with overlapping area by taking part from only one of the images. Although they said there is no effect of blurring in the mosaic, the edges of every image is actually visible.

In [23], Gonzalez et al provide a method of mosaic construction which can deal with looping path problem. Traditional mosaic construction only aligns consecutive frames, so the accumulation of

small alignment errors will occur. Especially, when the image path returns to a previous position (looping path), a significant mismatch between non-consecutive frames will result, which is called "looping path problem". The proposed solution in this paper is to distribute the accumulated error of positions of all images in the mosaic, and the premise of this solution is that the relative position of a neighbor pair of images can be modified slightly without introducing a visible loss in quality. However, the number of the equations this algorithm has to solve is the same as the number of images. Therefore, if there are large number of images in the mosaic, this algorithm will become very complex and time-consuming.

In [32], Jones et al propose a mosaic method which can work in compressed domain. Usually, methods for building mosaic work in pixel domain, but this one can create mosaic directly from an MPEG video sequence. The most important work they do is to compute the camera motion directly from the motion vector encoded in MPEG stream, which enables mosaic building more simply and quickly. In the frame integration step, they use a combined method of replacement and average, but it still doesn't work well in dealing with moving objects. The moving objects are removed in replacement method or its position becomes transparent in average method.

In [38], Szeliski et al give some detailed description of panoramas building, including cylindrical and spherical panoramas, perspective (8-parameter) panoramas and rotational (3-parameter) panoramas. The novel part of their work is to close the gap in a panorama. One method is to estimate the focal length, because once an initial set of focal length is available, we can improve all the motion parameters estimation so as to improve the image registration process. The other method is to register the same image at both the beginning and the end of the sequence. Also, they use almost the same idea as [23], which distributes the misregistration error evenly across

the whole sequence. The more important in "gap closing" technique here is that they update the focal length and estimated parameters after a complete panorama is constructed, then the process is repeated. Using this iterative process, more accurate and robust results can be obtained.

[11] is an improved version of [38], and the most important improvement is in the method to reduce accumulated registration errors. In this paper, besides the original methods such as estimating the focal length and iterative update, some new algorithms are proposed, for example, global alignment and local alignment. The global alignment is actually the block adjustment applied to the whole sequence of images, which reduces accumulated error by simultaneously minimizing the misregistration between all overlapping pairs of images. The local alignment is actually the deghosting techniques which warps each image based on the results of pairwise local image registrations. By combining these two alignment, the quality of image mosaics is significantly improved.

Besides planar 2D panorama techniques, we also explore some algorithms dealing with mosaics of higher dimensions.

In [9], Shum et al describe the concept of *concentric mosaics*. Concentric mosaics are a set of manifold mosaics constructed from slit images taken by cameras rotating on concentric circles. In this algorithm, they constrain camera motion to planar concentric circles and create concentric mosaics using slit images taken at different locations along the circle. The input image rays are indexed naturally in 3 parameters: radius, rotation angle and vertical elevation. The advantage here is that, since this mosaicing technique is in 3D space, it provides a much richer user experience by allowing the user to move freely in a circular region and observe significant parallax and lighting

changes. However, there are two disadvantages here. The first one is that the camera motion is constrained in a circle. Although this makes capture and construction easy and convenient, it is not so flexible and free. The second one is that rendering with concentric mosaics needs depth correction otherwise vertical distortions will be caused.

In [30], Lhuillier et al describe another concept of *relief mosaics*. Relief mosaics are collections of registered images that extend traditional mosaics by supporting motion parallax. Traditional mosaicing algorithms always assume that the input images are free from motion parallax, i.e., the camera translation is small or the scene is shallow or near planar. If there is a deep scene, special equipment and calibration may be required to fix the viewing position to sufficient accuracy. However, this algorithm assembles images into a composite image using view morphing to cancel their relative motion parallax on the registered overlapping sections, and a heuristic default mapping on the non-overlapping sections to provide visual continuity with the registered ones. This can be viewed as a '2.5D' plenoptic function, intermediate between 2D panorama and 3D concentric mosaics in [9]. The main advantage here is that this algorithm allows motion parallax with less amount of data and no camera geometry information is necessary.

The second part in first area is about the techniques which can deal with scene with moving objects although the result mosaic is still almost static.

In [28], the concept of "dynamic mosaic" is first proposed. The cause to propose this concept is that the issue of how to develop a *complete* representation of scenes based on mosaics, so that the sequence can be fully recovered from the mosaic image, has not been adequately treated. The information not captured by the traditional static mosaic techniques, which is described in

the first part, and needing additional representation are the changes in the scene with respect to the background. That's why we need to create *dynamic* mosaic, which is a sequence of evolving mosaic images, where the content of each new mosaic image is updated with the most current information from the most recent frame. According to the authors, the complete dynamic mosaic representation of the video sequence consists of the first dynamic mosaic and the incremental alignment parameters and the incremental residuals, which represents the changes. Besides this novel idea of "dynamic mosaic", they also give several kinds of mosaic applications, including mosaic based video compression, mosaic based visualization and mosaic based video enhancement. Therefore, this is a classic paper in mosaicing techniquesis. However, the "dynamic mosaic" here is still different from our idea "mosaic with moving object". Their "dynamic mosaic" is to update the mosaic image with the most current frame, so that they can keep the information of each new frame, but they do nothing on layer separation or object detection and the system does not *understand* the scene completely.

In [41], Bhosle et al present a "static" method to deal with moving objects. The key element here is to do background extraction before doing feature extraction so that the mosaic is built only from background region. Using this algorithm, the moving object will be removed in the final mosaic, so the result is static background only. The criterion they use to extract the background is as follows:

1. It is situated behind the rest of the scene.

2. Appearance of the scene remains constant over the time; the only changes in the grey levels are due to global motion.

3. Background pixels occupy the main part of the image.

The alignment method here is almost the same as [40], which seems to be a quick and efficient method in alignment. However, although this paper can deal with video input with moving objects, it simply removes them and only the static information is retained. Therefore, except the first process step to extract moving part, the technique itself is still static mosaic.

In [14], Davis provides another algorithm to deal with moving objects in the scene. There are three primary contributions here. The first is a registration method that remains unbiased by movement–the Mellin transform is extended to register images related by a projective transform. The second is an efficient method for finding a globally consistent registration of all images. Lastly, a new method of compositing images is presented. Blurred areas due to moving objects are avoided by segmenting the mosaic into disjoint regions and sampling pixels in each region from a single source image. The key element here related to moving objects is to divide the final mosaic into a set of regions and the pixels in each region are sampled from a single "correct" source image. Therefore, it's important to find a best path to divide regions so that any moving object falls in a single region, which can avoid object truncation. Using this algorithm, the moving objects will display once in the final mosaic, and the result is almost static background with static foreground.

In [26], Irani et al give the concept of *frame-based* video data and *scene-based* video data. They presents an approach for efficient access, use, and manipulation of video data based on mosaic representation. The video data are first transformed from their sequential and redundant frame-based representation, in which the information about the scene is distributed over many frames, to an explicit and compact scene-based representations, to which each frame can be directly related.

According to the authors, the scene representation is composed of three components:

1. Extended spatial information, which is represented in the form of panoramic mosaic image.

2. Extended temporal information, which is represented in the form of trajectories of independently moving objects.

3. Geometric information, which captures the 3D scene structure as well as the geometric transformation which maps the location of each scene point back and forth between the mosaic images and the individual frames.

However, to recover the geometric transformations and the 3D scene structure, the regions of the video frames corresponding to the static and dynamic portions of the scene should be determined. Using this algorithm, we can show the trajectory of moving object on the final mosaic. The result is static background with moving foreground, but the foreground is only the trajectory and we still don't know what the object really is.

Mosaicing techniques have various kinds of applications. For example, in [20], Leung and Chen present a mosaic based compression scheme for image-based rending application. A sequence of images is first captured by a camera located at different positions along a circle, then the mosaic image is constructed and used to predictively encode the original images. The corresponding part of the mosaic image is taken as the prediction image. Furthermore, motion compensation is applied to provide a closer match at the block level between the prediction image and the original image. The main advantage here is that mosaic based compression with motion compensation uses less storage space but still allows random access of individual images, which is better than

only inter-code scheme or intra-code scheme. However, the camera motion is constrained to a preset circle, so that the mosaic construction is also limited in this constrained situation.

Besides, Song et al also present a possible application in [37]. They concern the problem of collaborative frame selection, which arises when one robotic pan, tilt, zoom camera is shared by many users. They describe an algorithm which compute optimal camera parameters based on simultaneous frame requests from all users. Although they did not mention mosaic techniques obviously, mosaic can be definitely useful in generating the whole scene ready for frame selection.

From the above literature survey, we can find that mosaicing technique has developed a lot since it appeared as a field of investigation. If we classify it by dimension, we get 2D panorama [41] [16] [28] [11] [40] [32] [28] [14], 2.5D relief mosaic [30], 3D concentric mosaic [9] and mosaics of even higher dimension. In this thesis, our focus is on 2D panorama. In this 2D panorama subarea, we can still classify the techniques into static mosaic [38] [11], dynamic mosaic [26] [28].

Till now, we can construct static mosaics from a set of images in a static scene, and the algorithms work well whether the input images have motion parallax or not. Moreover, we can also do something on scenes with moving object, such as extracting the background only to still make a static mosaic, retaining one position of the moving object, or displaying the trajectory of moving object in the final mosaic. However, people haven't tried to keep the whole moving object in the mosaic scene and retain its motion information at the same time. If we can do this, viewers can know how the object is moving as well as what the object looks like, so that they can get more information about the scene and the activities inside it. We always keep in our mind that the mosaic is a basis for an *efficient* and *complete* representation of video sequences, so we should try to include information as much as possible when we build the mosaic.

2.2 Related works about image registration

Image registration is a fundamental task in image processing used to match two or more images. This is also an important step in solving our problem because we need to match images and mosaics to find the difference.

[4] is a survey paper of image registration techniques. Brown organizes the research field by establishing the relationship between the variations in the images and the type of registration techniques which can most appropriately be applied. Three major types of variations are described here. The first type are the variations due to the differences in acquisition which cause the images to be misaligned. The second type of variations are those which are also due to differences in acquisition, but cannot be modelled easily such as lighting and atmospheric conditions. The third type of variations are differences in the images that are of interest such as object movements, growth, or other scene changes. The author also gives a framework of image registration techniques: feature space, search space, search strategy and similarity metric. The feature space extracts the information in the images that will be used for matching. The search space is the class of transformation that is capable of align the images. The search strategy decides how to choose the next transformation from the search space. The similarity metric determines the relative merit for each test. Search continues according to the search strategy until a transformation is found whose similarity metric is satisfactory. Of course, the type of variations present in images determines the selection for each of these frame components. After these basic introduction, Brown presents the theory of image registration as well as some applicable methods. Therefore, we can gain a complete overview of image registration techniques from this paper.

2.3 Related works about moving object detection

Object detection is an important problem which interests many researchers in image and video analysis. Specific objects can be detected by means of specialized detectors, motion, sounds, and appearance in the textual modality. Among these, grouping object based on motion is the best in absence of other knowledge. In other words, motion is one of the most valuable features in detection since the appearance of objects might vary widely. This technique is necessary for surveillance applications, for guidance of autonomous vehicles, for efficient video compression, for smart tracking of moving objects and many other applications.

Since we want to improve the static mosaic techniques so that they can deal with scenes with moving objects, moving object detection forms an important part in our work.

In [24], Irani et al present a method for detecting and tracking occluding and transparent moving objects, which uses temporal integration without assuming motion constancy. Here, motion constancy is an assumption which assumes that motion remains uniform in the analyzed sequence. According to the authors, the analysis of multiple motions can be divided into two categories: motion analysis without segmentation and motion analysis with segmentation. In the former case, the dominant motion approach is used, which finds the parameters of a single translation in a scene with multiple motions. In the latter case, a region-based tracking method is used, where we should initially separate the moving objects. To detect the moving objects, a single motion is first computed, which is called the "dominant motion", and the corresponding object is called the "dominant object". Once a dominant object has been detected, it's excluded from the region of analysis and the process is repeated on the remaining region to find other objects and their

motion. This algorithm yields a *continuous* function, an taking a threshold on this function yields partitioning of the image to moving and stationary regions. Also, the problem of noise can be overcome once the algorithm is extended to handle longer sequences using temporal integration. The temporal integration is to construct a dynamic internal representation image for each tracked moving object, by taking a weighted average of recent frames, registered with respect to the tracked motion (to cancel the motion). This image contains, after a few frames, a sharp image of the tracked object and a blurred image of all the other objects. However, this detecting and tracking methods are not able to deal with several moving objects at the same time due to the concept of *dominant* motion and object.

While [24] presents a method to moving object detection only in 2D scene, [25] describes a unified approach to handling moving object detection in both 2D and 3D scenes. The key step in moving object detection is accounting for (or compensating for) the camera-induced image motion. After compensation for camera-induced image motion, the remaining residual motion must be due to the moving objects. The approach used here is based on a stratification of the moving object detection problem into scenarios which gradually increase in their complexity.

1. Scenarios in which the camera-induced motion can be modelled by a single 2D parametric transformation.

2. Those in which the camera-induced motion can be modelled in terms of a small number of layers of parametric transformation.

3. General 3D scenes, in which a more complete parallax motion analysis is required.

Of course, the techniques matching the above stratification also progressively increase in their

complexity. The computations at one complexity level become the initial processing step at the next complexity level. The main contribution of this paper is that it provides a unified approach for handling moving object detection in both 2D and 3D scenes, with a strategy to gracefully bridge the gap between those two extremes, while past techniques can only deal with one case and fail on the other case. However, the core elements of the unified approach has been given, but the integration into a single algorithm still remains an unsolved task in this paper.

In [33], Fablet et al describe a region-based approach with a view to directly detecting moving objects in the scene from a color image sequence acquired by a mobile camera. The detection includes three steps. First, they compute the 2D affine motion model accounting for the dominant image motion. Second, a spatial graph, whose nodes correspond to spatial regions, is derived from the color-based segmentation. Third, a Markovian framework is introduced to assign to each node of the graph a binary label stating if a region is conform or not to the dominant motion. If the dominant motion is due to camera motion, the set of regions labelled as non-conform includes moving objects. The advantage of this algorithm is that it does not require to attach a parametric motion model to each extracted region, and only the estimation of the dominant image motion is computed. Also, it benefits from the integration of local motion-related measures to determine the relevance of the estimated dominant motion in each spatial region. However, the disadvantage is that the computation procedure here is very complicated, and it is necessary to decide many parameters to make the algorithm work well. The authors do not tell how to set these parameters.

In [5], Nguyen et al present a method that segments a single video frame into independently moving visual objects. This method follows a bottom-up approach, starting with a color-based decomposition of the frame. Regions are then merged based on their motion parameters via a

statistical test. The main contribution of this paper is a new well founded measure for motion similarity leading to a robust method for merging regions.

In [15], Courtney uses the typical procedure to manage video indexing: detecting moving objects using motion segmentation method, tracking individual objects and generating symbolic representation. The steps we are most interested is object detection and tracking stages. In these stages, the author uses feature-based method. The feature set used is named "V-object", containing the label, centroid, bounding box and shape mask of its corresponding region, as well as object velocity and trajectory information generated by tracking process. The tracking process "links" V-object V_{np} and V_{n+1q} if their position and estimated velocity indicate that they correspond to the same real-world object appearing in frame F_n and F_{n+1}. According to the size of feature set, there will be a trade-off between computation complexity and tracking efficiency. To make tracking more accurate, the V-objects are tracked both forward and backward. The main contribution is a novel directed graph to describe the objects and their movement, and annotates it using a rule-based scheme to identify events of interest. Besides, the utilization of motion continuity to track objects both forward and backward is also very helpful in motion detection. However, the motion segmentation here is only based on the absolute difference of images, which is difficult to gain accurate results if there are both camera motion and object motion in the scene. Therefore, it is only suitable for video sequence within static scene, such as in surveillance and scene monitoring applications.

In [2], Badenas et al present a motion-based segmentation method and a region-based tracking method to deal with moving objects, which is a part of a traffic monitoring system. First, it carries out a frame-to-frame motion segmentation. Three features: x-y coordinates and intensity of pixels

are used to divide every image into unsupervised regions, which is the initial segmentation. Then motion estimation involves finding the translation parameters for every region that minimize the sum of displaced frame differences (DFD), which actually follows the idea from [18]. Next, it matches regions based on similarity, which is formulated as a distance. Five features are used to form a feature vector, which are region centroid(x-y), intensity mean and velocity(x-y). A weighted squared Euclidean distance is measured. Third is to estimate motion parameters. Here they use a recursive estimator, the Kalman filter. Once the motion parameters have been estimated, it compares motion parameters of neighboring regions and merges them when their motions are similar enough. It is like a feedback procedure to make segmentation more accurate. The advantage of this paper is that it uses some new techniques to manage the region segmentation better. One is accumulating evidence, which utilizes the motion continuity property. The other is recovering lost region, which can help to detect moving object even if it stops moving temporarily in the scene. However, the same problem as in [15] also exists here. This algorithm is only suitable for static scene. If the there are both camera motion and object motion in the scene, this algorithm won't get satisfactory result.

In [13], Jang et al present an improved version of Kalman filter, to be called Structural Kalman filter, which can successfully work under some deteriorating condition such as occlusion. The idea is to partition a target into several meaningful sub-regions instead of treating a target as one entity. Each sub-region is evaluated independently together with their relationship and the overall evaluation is then used to estimate the motion information on possibly occlude sub-regions. Under this idea, the new Structural Kalman filter is a composite of two types of the Kalman filters: Cell Kalman filters and Relation Kalman filters. The Cell Kalman filter is allocated to each sub-region

and the Relation Kalman filter is allocated to the connection between two adjacent sub-regions. If the sub-region is not occluded, the Cell Kalman filter is enough. If it is occluded, the related Relation Kalman filters are used to estimate its motion. However, it is not clear at present how many sub-regions are suitable in an object and the compulsory partition will cause problem in the process of object matching.

From the above literature survey and other papers which have not been mentioned here, we can find that motion detection techniques usually rely on region-level classification schemes and exploit local motion-related information, which can be the DFD (Displaced Frame Difference) [2], or the normal flow. Concerning the classification step to get moving or static part, most techniques either use threshold [24] [25] or Bayesian framework [17]. Besides, as far as motion-based segmentation is concerned, pixel-level and region-level labelling are often used, and it seems that region-level labelling is more popular [33] [2], because region-level labelling is closer to the "object" concept and easier to track than pixel-level labelling. Of course, there are also some techniques called "subregion" [13], which is a combination of region-level and pixel-level, and can deal with occlusion. Moreover, the computation of a primary separation of spatial regions is processed either relying on motion-based criterion [15] [12], or on intensity, texture or color information [5] [42]. Using intensity, texture or color contours usually supplies a better localization partition. In fact, most techniques start from this initial spatial partition, then a 2D parametric motion model, generally an affine one [33], is attached to each spatial region. After that, the original spatial regions are merged according to the motion properties. To this end, the merge techniques can be classified into clustering schemes in motion parameter space [19], MDL criterion [12], and Markovian graph labelling approach [22].

Till now, we can detect motion in both 2D and 3D scenes, in either pixel-level or region-level. We can detect object motion even when it is occluded by others sometimes. Moreover, we can deduce which activities happen in the scene based on objects motion and their interaction using some predefined rules. However, there are few techniques concerning motion detection in dynamic scene, i.e., there are both camera motion and object motion in the scene. Although there do exist a few algorithms which can work in dynamic scenes, they are either too complicated or with tremendous computational load. If we can get a relatively simple algorithm which can also work well in dynamic scenes, it would be a very meaningful work.

Chapter 3

Algorithm

In this chapter, we discuss our approaches for building complete mosaic of dynamic scene and reconstructing moving objects. This can be called *dynamic video mosaic*. We divide the whole algorithm into four steps as given in follows. Of course our approaches is based on others' works, but the integration of these all is novel. Besides, we give our novel methods and make quite a few improvements, which we will state clearly in the following parts.

Step 1: **Static Mosaic Construction**. The input video is preprocessed and a static mosaic is built without separating objects from background.

Step 2: **Coarse Frame Registration.** The whole frame is registered according to the mosaic, which actually is a preparation step of object detection.

Step 3: **Moving Objects Detection.** The moving objects are detected using difference pictures and optical flow. In this step, we get two different layers of the input video.

Step 4: **Moving Objects Reconstruction.** The moving objects are reconstructed in the mosaic scene, so that the final mosaic has both static scene information and dynamic

motion information.

One of the difficulties in this chapter is that we want to detect the moving objects based on mosaic techniques. In this way, the mosaic step is not only a step to process the scene, but also a step to prepare for the motion detection. Another difficulty is that to reconstruct the detected objects in the scene, we have to get both the position and the warp information, which requires a complete list of motion estimation matrix as well as some matrix transformation such as inverse and productive.

Additionally, we want to clarify the preconditions of our work. Due to the motion model we used and the limitations of our algorithms, there are constrains on both camera and objects. The camera should move in a line and the scene should be nearly a flat plane. Concerning the object, currently, our algorithm works well only when there is one moving object and its size is proportional to the scene.

3.1 Static Mosaic Construction

The technique for static mosaic construction has been explored by many researchers, and various kinds of methods have been presented. However, to make this thesis self-contained, we will still describe our algorithm in this section. It is a classical one, which has been used by many researchers as a basis and even appears in textbooks.

The input in this step is many frames with overlap and the output is a large mosaic picture representing the complete scene.

3.1.1 Video Frame Separation

Before we do anything about mosaic, we should first slice frames from video clips. Here, we use MPEG2Decoder [6] to separate individual frames from MPEG encoded video, and the number of frames we get is equal to the frame rate, usually 30 frames per second. Therefore, if the time length is a bit large, we would get a large number of frames and there is only very little change between two adjacent ones. To avoid processing too many similar frames, we use one out of five or ten frames instead of using frame one by one.

3.1.2 Corresponding Points Establishment

The first step to construct a mosaic based representation is image alignment, which depends on both the chosen world model and the motion model. The alignment can be limited to 2D parametric motion models, or can utilize more complex 3D motion models and layered representations. Till now, most techniques utilize 2D alignment models.

In our work, the most important part is to include motion information into the mosaic representation, so we simply use 2D parametric models to simplify the mosaic construction procedure and pay more attention to the moving part.

In our current implementation, the 2D parametric motion models we use is an 8-parameter projective model, which requires at least 4 corresponding points for one pair of images. Moreover, the mosaic we want to build here is not the traditional static mosaic, but the *dynamic* mosaic, the concept of which is presented in [28]. This is a *dynamic* mosaic which increases every time when a new frame is added, so the corresponding points we establish are between the latest frame and the current mosaic.

To establish corresponding points of two adjacent frames in a static scene, we can use a relatively simple algorithm presented at [27]. To make the description clear, we refer the first frame as *the source frame*, and the second as *the destination frame*.

1: Detect the corners in two frames and mark every corner point (x,y) in the source frame.

2: Extract an $r \times r$ square window of pixels with the marked point in the center of the square, in the source frame.

3: Overlay the source square window onto the destination frame at the point (u,v), so that (x,y) coincides with (u,v).

4: Calculate the sum-of-square difference (SSD) of the pixel values to measure the match quality. The SSD measure is:

$$E(\{\mathbf{r}\}) = \sum_{r} (I(\mathbf{x}) - I(\mathbf{u}))^2 \tag{3.1}$$

where $\mathbf{x} = (x,y)$ denotes the position of the point in the source frame, $\mathbf{u} = (u,v)$ the position of the coincident point in the destination frame, and I denotes the image intensity. The sum is computed over all the points within the region and $\{\mathbf{r}\}$ is used to denote the entire field within that region.

5: If the source window perfectly matches the destination window, then the value of SSD measure would be 0. If not, shift the square window in destination frame so that (x,y) in the source frame coincides with $(u+1,v)$ in the destination frame, and compute the SSD again.

6: Continue shifting the window within an $r \times r$ region in the destination frame and

Figure 3.1. Corresponding Points for Static Scene

computing the SSD each time until the smallest SSD value is got. The center of the destination window that gives the smallest SSD is the correct (u,v).

For simplicity, we do this corresponding points establishment using grey-scale images, and the sample result is shown in Figure 3.1.

However, at this step, the scene still appears as a whole, i.e., we have no information about which part is the background and which part is the moving objects. Therefore, if we apply this algorithm directly in dynamic scenes, it would mark some incorrect corresponding points in the moving part and cause image misalignment in the later construction procedure. One example is shown in Figure 3.2.

One possible solution for this problem is that we enable user to select the suitable points after we get a complete set of corresponding points. However, this would make the system slow to process. Therefore, for simplicity, at the current stage, we establish the corresponding points in the frame and mosaic manually, and import it as input datafiles to the program. To get sub-pixel level accuracy, we utilize a build-in function *cpselect* in MATLAB. Furthermore, although the

Figure 3.2. Corresponding Points for Dynamic Scene

required number of corresponding pairs is 4 according to the motion model, we select 8-9 pairs to minimize the misalignment of the mosaic.

3.1.3 Forward Homographies Computation

Forward homography is an estimation matrix which maps the points in the source image to their corresponding points in the destination image. Since we build dynamic increasing mosaic here, the source image refers to the current built mosaic, and the destination image refers to the latest to-be-added frame. This definition remains effective in the rest part of this thesis.

As we mentioned before, we use 2D 8-parameter perspective motion model here [10]. Such a model works well under the scenarios where there is little translation of the camera, or the entire scene can be approximated by a single parametric surface (typically a plane). This motion model can be described in matrix format as follows:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.2}$$

where (x_1, y_1) refers to the point in source image and (x_1, y_1) refers to its corresponding point in the destination image, and the equality is up to scale. Although there are 9 unknowns a,. . . ,i in the homography matrix, only 8 of them need to be found because we are working in homogeneous coordinates. It is customary to let i = 1, and then seek to determine the other unknowns. We can rewrite the above matrix equation for each pair of corresponding points in terms of the unknowns a,. . . ,h and get an 8 × 8 system as follows:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -u_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -u_4x_4 & -u_4y_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -v_1y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -v_2x_2 & -v_2y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -v_3x_3 & -v_3y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -v_4x_4 & -v_4y_4 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}}_{p} = \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}}_{b} \tag{3.3}$$

If we note the structure of above matrix **A**, we can easily understand why the minimum requirement is 4 pairs of corresponding points. Of course, this matrix equation can be extended to handle $n > 4$ pairs of corresponding points. What we want to get is the vector **p**, which can be solved using the pseudoinverse:

$$\mathbf{p} = \mathbf{A}^{\dagger}\mathbf{b} \tag{3.4}$$

where $\mathbf{A}^{\dagger} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$.

Rewrite vector $\mathbf{p}$ in matrix format as follows:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

then we get forward homography $\mathbf{H_{SD}}$[1], which maps each point in the source image to its corresponding point in the destination image.

3.1.4 Bounding Box Computation

In this subsection, we will describe the method on how to compute the bounding box. Bounding box is the smallest rectangle that contain the final mosaic completely, so it is the same as how to compute the size of the output mosaic image. To compute the size of the mosaic, we have to compute the maximum extent of both two images after they are warped.

However, we have to specify a reference image first. This is the image to whose viewpoint the other image will be warped. Since we are working on a *dynamic increasing mosaic*, we update the mosaic every time when we get a new frame. Therefore, we designate the current frame as the reference image to produce the new mosaic. Since we already have $\mathbf{H_{SD}}$ as the forward homography which maps points from mosaic to current frame, the two reference matrices are:

[1]The subscript **SD** denotes that this homography is from source image to destination image, i.e., the computation direction is forward.

$$\begin{cases} \mathbf{H_{SR}} = \mathbf{H_{SD}} \\ \mathbf{H_{DR}} = \mathbf{I} \end{cases} \tag{3.5}$$

where $\mathbf{I}$ denotes the identity matrix[2].

Normally, homographies transform rectangles into quadrilaterals, so what we need to do is to keep track of the 4 corners of both images to be warped. The four corners in one image are (1,1), (w,1), (1,h), (w,h), where w is the width of the image and h is its height. We have two images as the input, so we will get 8 corners. After warping, we find the minimum and maximum **X**-**Y** coordinates from all 8 corners, which will become the upper left corner and the lower right corner of the result bounding box, respectively. If we let (x_{min}, y_{min}) and (x_{max}, y_{max}) denote these two corners, the width and height of the bounding box, i.e., the output mosaic are:

$$\begin{cases} \mathbf{bw} = x_{max} - x_{min} \\ \mathbf{bh} = y_{max} - y_{min} \end{cases} \tag{3.6}$$

where **bw** and **bh** refer to the width and height respectively.

3.1.5 Backward Homographies Computation

In this subsection, we will describe how to compute backward homography. The backward homography is used to find the corresponding points in two input images given their position in the reference image. However, it is not just the inverse of the reference matrix mentioned in last section, because the origin of the coordinate has already been shifted due to the bounding box

[2]The subscript **SR** and **DR** denote that the reference matrices are from the source frame and the destination frame to the reference frame, respectively.

computation. With respect to the reference image, the upper left corner of the bounding box is at (x_{min}, y_{min}), but we have to make this the new origin for convenience when producing the output image.

Therefore, considering the inverse and the translation of the coordinate origin together, the backward homographies of the two inputs can be achieved as follows:

$$\begin{cases} \mathbf{H_{BS}} = \mathbf{H_{SR}}^{-1}\mathbf{T} \\ \mathbf{H_{BD}} = \mathbf{H_{DR}}^{-1}\mathbf{T} \end{cases} \tag{3.7}$$

where the translation matrix is given by: $\mathbf{T} = \begin{bmatrix} 1 & 0 & x_{min} \\ 0 & 1 & y_{min} \\ 0 & 0 & 1 \end{bmatrix}$

3.1.6 Mosaic Image Integration

In the above five subsections, we have prepared all the necessary information for the mosaic construction. In this subsection, we finally reach the mosaic construction stage. The method we use here is the *destination scan* method, which can be described as follows.

1. for x = 1 to **bw**
2. for y = 1 to **bh**
3. Find (u,v), the point in the source image where (x,y), the point in the mosaic image warps to. Use the backward homography $\mathbf{H_{BS}}$.
4. Get c, the color at (u,v) in the source image. If (u,v) falls outside the source image, no color is assigned to c.

5. Repeat steps 3 and 4 for the destination image, using backward homography $\mathbf{H_{BD}}$.

6. Compute the final color of the mosaic image based on the colors we get in two input images. Let the final color be $\mathbf{C_{Final}}$.

7. Set the color in the mosaic image at (x,y) to be $\mathbf{C_{Final}}$.

NOTE: The homographies here specify only the geometric information, not the color. To know the color of an output pixel, we trace where that pixel comes from (via the backward homographies) to get the color from the input images. These colors together contribute something to the output mosaic.

We should pay attention that there are several kinds of methods to get color of input images in step 3 and to compute final color of mosaic image in step 6. We will describe some of them below and choose the one which is suitable in our work.

Color Selection

In most cases, the (u,v) coordinates we get are not integer values, i.e., the point falls between the pixels of the input images. Therefore, we need some methods to get the approximate color. To get the color representation in the input images, we can directly use the color in the nearest position or use interpolation method to get better result.

If we get the color directly, we should find the nearest point from the result point traced using backward homographies. The color of this point is used to represent the color of the warped point.

If we want to get better result, we can compute the color using interpolation method. One of

the widely used interpolation methods is *bilinear interpolation*. We describe this method in the following.

Without loss of generality, let the 4 pixels immediately surrounding (u,v) have coordinates (0,0), (0,1), (1,0), (1,1), and let (u,v) have values between 0 and 1. Thus (u,v) falls within the square whose corners are the coordinates given above. Let the colors at these corners be c_{00}, c_{01}, c_{10}, c_{11} respectively. Then:

$$c_{uv} = (1-u)(1-v)c_{00} + u(1-v)c_{10} + (1-u)vc_{01} + uvc_{11} \tag{3.8}$$

This equation can also be applied to RGB color system by simply interpolating for each channel, i.e., doing it for R,G,B separately. To optimize the above formula, we can transform it to the following equations, which use 3 multiplies instead of 8:

$$\begin{cases} c_{u0} = c_{00} + u(c_{10} - c_{00}) \\ c_{u1} = c_{01} + u(c_{11} - c_{01}) \\ c_{uv} = c_{u0} + v(c_{u1} - c_{u0}) \end{cases} \tag{3.9}$$

In our work, we use the direct method, not only because it is easier and faster, but also because the color retrieval function we use from the public library [8] can only deal with integer coordinate values.

Color Blending

Each point in the mosaic image comes from all the corresponding points in the input images. The question is that how to decide the final color based on several color inputs. Color blending

algorithm is used to solve this question. Generally speaking, there are two widely used ways to determine the overlapped region: **selection method** and **combination method**.

In selection method, we only select one point from all the available points and use its color to represent the final color. Although it looks not so accurate because only one point is selected to represent several points, this method works better in cases where camera motion, scene geometry and image conditions are challenging[1]. However, it requires accurate alignment along the seams.

In combination method, we compute all the available points using a suitable function such as median, average, etc, to get the final color. This method requires an accurate alignment over the entire overlapped region, otherwise the result will be blurred due to the color combination. This method works better in relative static scene.

These two color blending algorithms can be summarized as Table 3.1.

	Definition	**Requirement**	**Suitable Case**
Selection	Select one point from several ones	alignment along seams	dynamic scenes
Combination	Combine several points together	alignment over entire region	static scenes

Table 3.1. Summarization of Two Color Blending Algorithms.

In this thesis, we work in scenes where there are both camera motion and object motion, so we can't use combination method only because it will cause serious blur in moving regions. However, if we use selection method, it will still cause visible seams along the edges of every input image. Therefore, we make our improvement to try to use both of them to deal with different regions of input images.

Figure 3.3. Two Input Images for Static Mosaic Construction.

In the regions where the points only come from one input image, the only method we can choose is the selection method. In the regions where the points come from two input images, both methods can be used. To get better result, we choose to use combination method along the edge to make the seam invisible, and use selection method in other regions to maintain the clear shape moving objects. Since our dynamic mosaic is updated every time when a new frame is coming, we always select the point from latest input image instead of from old mosaic.

In the combination method to deal with the edge, we make some improvement to do better. The original combination method uses the combination function such as median or average, but we change to use distance-based parameter function instead. Concerning the two corresponding points from two input images, the one which is closer to the edge contributes more to the final color, so that there is stable transition across the edge, which makes it invisible.

The following example can be used to show the intermediate result after this step.

Figure 3.3 shows two input images.

Figure 3.4 shows mosaic result for two images and Figure 3.5 shows mosaic result for the entire

Figure 3.4. Mosaic for Building without Edge Blending Technique.

Figure 3.5. Mosaic for the Entire Scene without Edge Blending Technique.

Figure 3.6. Mosaic for Building with Edge Blending Technique.

scene. Both are without edge blending technique. Figure 3.6 shows the mosaic result for two images, using the same color combination method, but with edge blending technique. We can see clearly that with edge blending technique, the edges are almost invisible in the result mosaic.

3.2 Coarse Frame Registration

After finishing the above section, we now get a complete mosaic consisting of all input frames, and several lists of motion matrix computed in the construction process. This part is mostly based on others' work. In this section, we will describe our own development about how to register each input frame on the mosaic coarsely (without any detection or separation). Why we need to do this step? Remember that we always update the mosaic on every input, so that on the last output the only complete frame on the mosaic is the last frame. However, we need to present every input frame completely on the mosaic to detect the moving objects. Therefore, this step is necessary to cancel the camera motion, as a preparation for the moving object detection.

3.2.1 Reference Matrix Computation

To register each frame on the final mosaic, we have to compute the reference matrix from the final mosaic to each input images. Although we have already got $\mathbf{H_{BS}}$, $\mathbf{H_{SR}}$, $\mathbf{H_{BD}}$ and $\mathbf{H_{DR}}$ in the above sections, these matrix are only between the input images and the intermediate constructed mosaic instead of the final mosaic. Therefore, we have to compute *final* reference matrix. The relationship among these matrices can be shown in Figure 3.7.

In Figure 3.7, the computation method to get the final reference matrix is very clear.

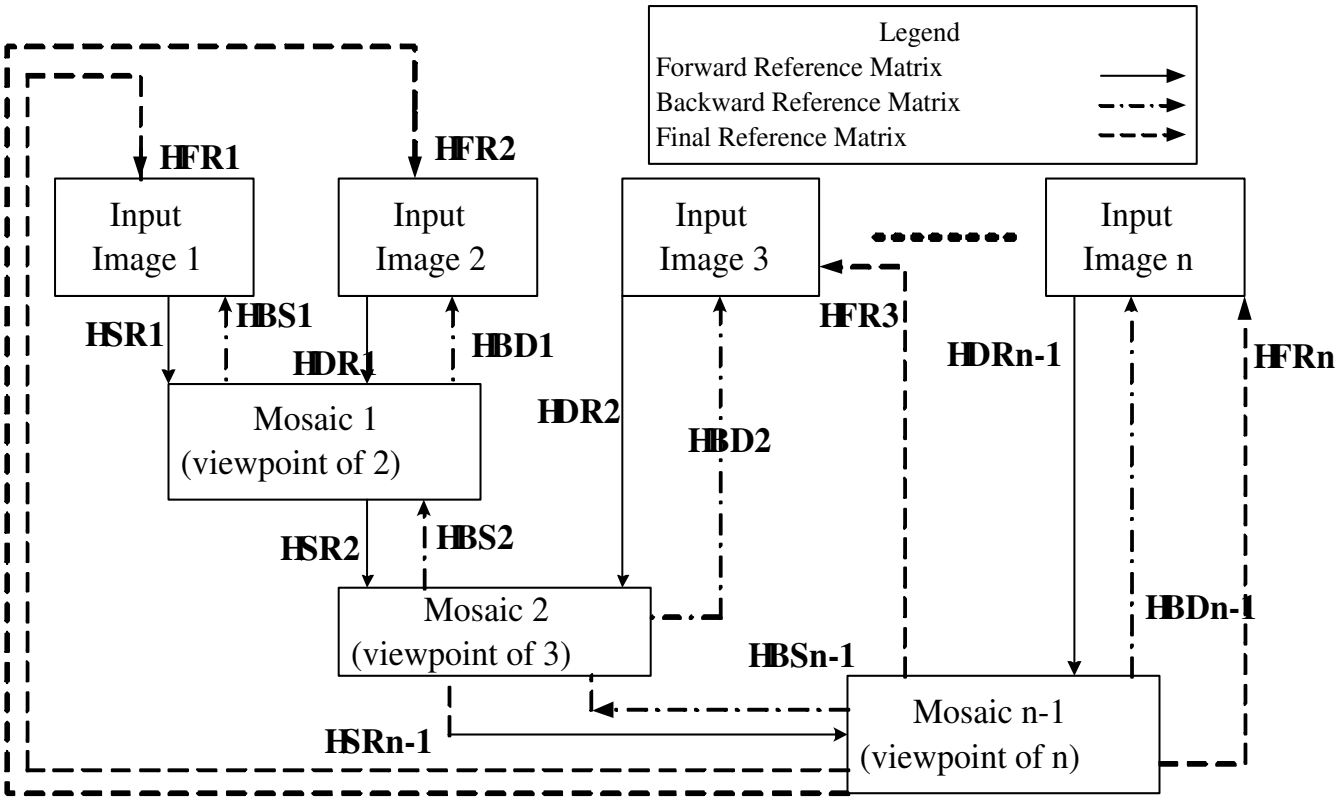

Figure 3.7. The Relationship among Matrices in Mosaic Construction

$$\begin{cases} \mathbf{H_{FR1}} = \mathbf{H_{BSn\text{-}1}} \times \mathbf{H_{BSn\text{-}2}} \times \cdots \times \mathbf{H_{BS1}} \\ \mathbf{H_{FR2}} = \mathbf{H_{BSn\text{-}1}} \times \mathbf{H_{BSn\text{-}2}} \times \cdots \times \mathbf{H_{BS2}} \times \mathbf{H_{BD1}} \\ \mathbf{H_{FR3}} = \mathbf{H_{BSn\text{-}1}} \times \mathbf{H_{BSn\text{-}2}} \times \cdots \times \mathbf{H_{BS3}} \times \mathbf{H_{BD2}} \\ \vdots \\ \mathbf{H_{FRn\text{-}1}} = \mathbf{H_{BSn\text{-}1}} \times \mathbf{H_{BDn\text{-}2}} \\ \mathbf{H_{FRn}} = \mathbf{H_{BDn\text{-}1}} \end{cases} \tag{3.10}$$

If we summarize the equations above, we can have the following formulas[3]:

$$\begin{cases} \mathbf{H_{FR1}} = \prod_{i=1}^{n-1} \mathbf{H_{BSi}} \\ \mathbf{H_{FRk}} = \prod_{i=k}^{n-1} \mathbf{H_{BSi}} \times \mathbf{H_{BDk\text{-}1}} & (2 \leq k \leq n-1) \\ \mathbf{H_{FRn}} = \mathbf{H_{BDn\text{-}1}} \end{cases} \tag{3.11}$$

3.2.2 Frame Warping and Registration

In the above subsection, we got the final reference matrix for each input images from the final mosaic. In this subsection, we describe how to register the whole input images on the final mosaic. For each input image F_k, do the followings:

1. for x = 1 to **image width**
2. for y = 1 to **image height**
3. Find (u,v), the point in the input image F_k where (x,y), the point in the mosaic image warps to. Use the final reference matrix $\mathbf{H_{FRk}}$.

[3]The subscript **FRk** denotes the final reference from the final mosaic to the k^{th} input frame.

4. If (u,v) falls inside the input image, the color at (x,y) in the mosaic is set as the same color at (u,v) in the input image.

5. If (u,v) falls outside the input image, the color at (x,y) in the mosaic remains unchanged.

After this coarse registration, we have a series of mosaic images, on each of which one input frames has been registered. These images represent a complete background, on which dynamic frames move following the trajectory of camera motion.

3.3 Moving Objects Detection

Moving objects detection can be generalized to change detection. Any perceptible motion in a scene results in some change in the sequence of frames of the scene. Motion characteristics can be analyzed if such changes are detected. In our situation, the motion is restricted to a plane that is parallel to the image plane, so a good quantitative estimate of the motion components of objects can be obtained.

Starting with frame-to-frame change detection, a global analysis of the sequence may be performed. Changes can be detected at different levels: pixel, edge, or region. Changes detected at the pixel level can be aggregated to obtain useful information with which the computational requirements of later phases can be constrained.

In this section, we will start with one of the simplest, yet one of the most useful change detection techniques, *difference pictures*, and then discuss one of the motion analysis techniques, *optical flow*. Although there is much work done in both fields, here we make our improvement to combine these

two and reach a rather simple motion detection algorithm. Moreover, we develop our own method to discard the noise according to *motion constancy* concept and *camera attention* property, which will be explained later, and get the regions of moving objects more accurately.

3.3.1 Difference Picture Computation

The most obvious method of detecting change between two frames is to directly compare the corresponding pixels of the two frames to determine whether they are the same.

In the simplest form, if we use $\mathbf{F_{(x,\ y,\ j)}}$ and $\mathbf{F_{(x,\ y,\ k)}}$ to denote two input frames with the same size, the difference picture $\mathbf{DP_{jk(x,\ y)}}$ is obtained by:

$$\mathbf{DP_{jk(x,\ y)}} = \begin{cases} 1 & if \quad |\mathbf{F_{(x,\ y,\ j)}} - \mathbf{F_{(x,\ y,\ k)}}| > \tau \\ 0 & otherwise \end{cases} \tag{3.12}$$

where τ is a threshold [34].

The result difference picture $\mathbf{DP_{jk}}$ is a binary picture. In this picture, pixels which have value **1** may be considered to be the result of object motion or illumination changes. One example is shown in Figure 3.8 and Figure 3.9. The Figure 3.8 shows two input images from a static scene, in which only one object is moving. The Figure 3.9 shows the difference picture from the above input.

However, to get correct result, this simple method assumes that the frames are properly registered or the scene is static. Only in these situation, the changed parts detected are caused by object motion. If there are both camera motion and object motion in the scene, most pixels in the difference picture will have value **1**, and we can not tell which ones are caused by object motion or

Figure 3.8. Two Input Images in Static Scene

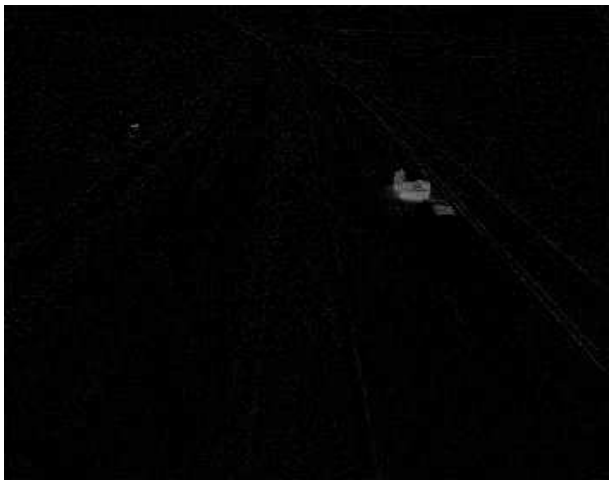

Figure 3.9. The Difference Picture of Static Scene

which ones are caused by camera motion. One example in dynamic scene is shown in Figure 3.10 and Figure 3.11. The Figure 3.10 shows two input images from a dynamic scene, in which the whole scene is moving. The Figure 3.11 shows the difference picture got from the above input.

Figure 3.10. Two Input Images in Dynamic Scene

Figure 3.11. The Difference Picture of Dynamic Scene

In this example, there is only camera motion in the scene, all the objects remaining still. However, we can not tell this from the difference picture, because it detects camera motion as

well as object motion. To make the difference picture detect object motion correctly, we have to compensate the camera motion, or register the input frames properly. That's why we need mosaic construction and image registration before this detection step.

The original input with both camera motion and object motion is shown in Figure 3.12.

Figure 3.12. Two Input Images with both Camera Motion and Object Motion

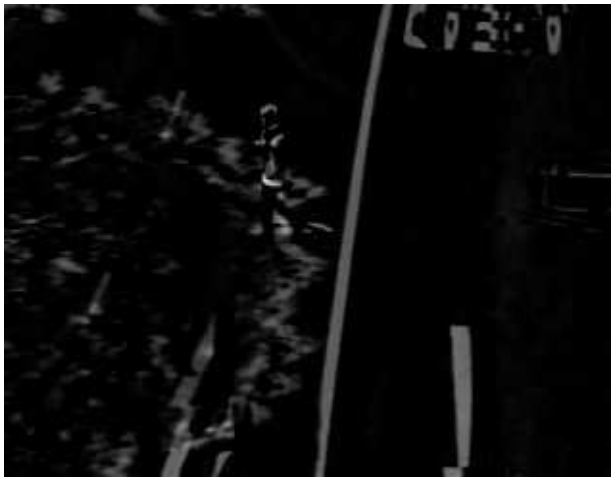

Figure 3.13. The Difference Picture before Image Registration

The corresponding difference picture of Figure 3.12 is shown in Figure 3.13.

If we build mosaic and register frames properly on it before computing difference picture, we can get registered frames as shown in Figure 3.14.

Figure 3.14. Two Input Images with both Camera Motion and Object Motion after Image Registration

Figure 3.15. The Difference Picture after Image Registration

The corresponding difference picture of Figure 3.14 is shown in Figure 3.15. We can see that in the final difference picture after image registration, the correct shape and position of the changed parts are detected correctly.

3.3.2 Optical Flow Computation

The most attractive aspect of the difference picture for motion detection is its simplicity. However, this simplest method is noise-prone. Changes in illumination and misregistration, in addition to electronic noise of the camera, can result in many false alarms. Changes in illumination will create problems for all intensity-based approaches. Misregistration of frames results in the assignment of false motion components, which can be eliminated using accumulative difference pictures or other additional information.

What we actually want to detect is motion, but difference picture only shows the changes, which can be caused by other factors in addition to motion. To detect the motion more accurately, we choose to compute the *optical flow* as another measurement for motion detection.

According to [34], we have the following definition:

Definition 3.3.1 *Optical flow is the velocity field in the image plane due to the motion of the observer, the motion of objects in the scene, or apparent motion which is a change in the image intensity between frames that mimics object or observer motion.*

Generally speaking, optical flow is the distribution of velocity of movement of brightness patterns over the points in an image. It can arise from relative motion of objects and the viewer. It can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects. Therefore, optical flow carries information which is very valuable for our analysis of dynamic scenes.

The computing of optical flow can be classified into two general categories: feature-based and gradient-based. If a stationary camera is used, most of the points in an image will have zero velocity. This is assuming that a very small subset of the scene is in motion, which is usually true. Thus, most application for optical flow involve a moving camera.

Feature-Based Methods

Feature-based methods for computing optical flow first select some features in the image frames and then match these features and calculate the disparities between frames. The correspondence may be solved on a stereo image pair using relaxation. The same approach may also be used to solve the correspondence problem in dynamic scenes. However, the problem of selecting features and establishing correspondence is not so easy. Furthermore, this method only produces velocity vectors at sparse points.

Gradient-Based Methods

Gradient-based methods exploit the relationship between the spatial and temporal gradients of intensity. This relationship can be used to segment images based on the velocity of points. One of the classic papers, which presented this method first, is [3], and the method we use here also follows this paper. In the follows, we will describe how to compute optical flow using gradient-based method.

Suppose the image intensity at a point in the image plane is given as $\boldsymbol{E}(x, y, t)$. Assuming small motion in successive frames in a video sequence, the intensity at this point will remain constant, so that:

$$\frac{d\boldsymbol{E}}{dt} = 0 \tag{3.13}$$

Using the chain rule for differentiation, we can get:

$$\frac{\partial \boldsymbol{E}}{\partial x}\frac{dx}{dt} + \frac{\partial \boldsymbol{E}}{\partial y}\frac{dy}{dt} + \frac{\partial \boldsymbol{E}}{\partial t} = 0 \tag{3.14}$$

If we let the velocities

$$u = \frac{dx}{dt} \tag{3.15}$$

and

$$v = \frac{dy}{dt} \tag{3.16}$$

we can easily have a single linear equation in the two unknowns u and v

$$\boldsymbol{E}_x u + \boldsymbol{E}_y v + \boldsymbol{E}_t = 0 \tag{3.17}$$

where we have introduced the additional abbreviations $\boldsymbol{E}_x$, $\boldsymbol{E}_y$, and $\boldsymbol{E}_t$ for the partial derivatives of image intensity with respect to x, y and t, respectively. The constraint on the local optical flow expressed by this equation can be illustrated in Figure 3.16.

$\boldsymbol{E}_x$, $\boldsymbol{E}_y$, and $\boldsymbol{E}_t$ can be computed directly from the image. Thus, at every point in an image, there are two unknowns, u and v, and only one equation. Therefore, using information only at one point, the optical flow can not be computed. This can be explained using Figure 3.17.

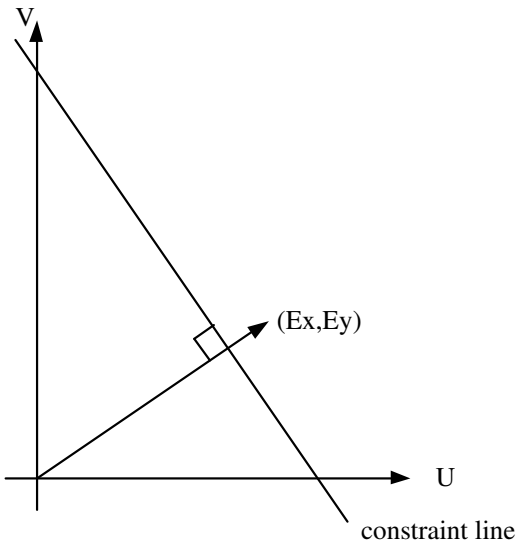

Figure 3.16. The change of image intensity equation constrains the optical flow velocity. The velocity (u, v) has to lie along a line perpendicular to the intensity gradient vector (E_x, E_y). The distance of this line from the origin equals E_t divided by the magnitude of (E_x, E_y).

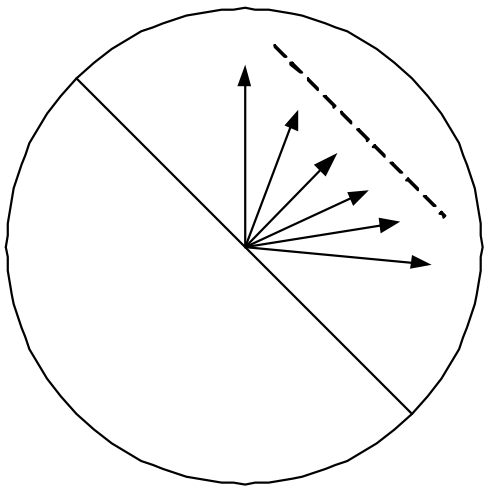

Figure 3.17. The Aperture Problem.

If one sees a point using a tube such that only one point is visible, then motion of the point can not be determined. One can only get the sense of the motion, not the components of the motion vector. This problem is commonly known as the *aperture problem*. The velocity components at one point can not be determined using the information at only one point in the image without making further assumption.

The new assumption we make is *smoothness constraint*, which assumes that the velocity field varies smoothly over an image. This assumption is reasonable because if every point of the intensity pattern can move independently, there is little hope of recovering the velocities. Therefore, we can combine the optical flow constraint equation with the above assumption about the smoothness of the optical flow velocity field to compute the optical flow using variational methods.

We recall that the optical flow constraint equation is

$$\boldsymbol{E}_x u + \boldsymbol{E}_y v + \boldsymbol{E}_t = 0 \tag{3.18}$$

where u and v are the x and y components of the optical flow, respectively, and $\boldsymbol{E}_x$, $\boldsymbol{E}_y$, and $\boldsymbol{E}_t$ are the spatial and temporal derivatives of the image intensity. For a smoothness measure, we use the sum of the squared magnitudes of each optical flow component as the integrand in the regularization term:

$$\int\int \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right] dx\, dy \tag{3.19}$$

Then we combine the smoothness measure with a measure of deviations from the optical flow constraint weighted by a parameter which controls the balance between deviations from the optical

flow constraint and deviations from smoothness constraint. What we get is:

$$\int\int \left\{ (\boldsymbol{E}_x u + \boldsymbol{E}_y v + \boldsymbol{E}_t)^2 + \nu^2 \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right] \right\} dx\, dy \tag{3.20}$$

Use the calculus of variations to transform this norm into a pair of partial differential equations:

$$\begin{cases} \nu^2 \nabla^2 u = \boldsymbol{E}_x^2 u + \boldsymbol{E}_x \boldsymbol{E}_y v + \boldsymbol{E}_x \boldsymbol{E}_t \\ \nu^2 \nabla^2 v = \boldsymbol{E}_x \boldsymbol{E}_y u + \boldsymbol{E}_y^2 v + \boldsymbol{E}_y \boldsymbol{E}_t \end{cases} \tag{3.21}$$

Then, we use finite difference methods to replace the Laplacian in each equation with a weighted sum of the flow vectors in a local neighborhood, and use iterative methods to solve the difference equations. The iterative equations are as follows:

$$\begin{cases} u = u_{average} - \boldsymbol{E}_x \frac{\boldsymbol{P}}{\boldsymbol{D}} \\ v = v_{average} - \boldsymbol{E}_y \frac{\boldsymbol{P}}{\boldsymbol{D}} \end{cases} \tag{3.22}$$

where

$$\boldsymbol{P} = \boldsymbol{E}_x u_{average} + \boldsymbol{E}_y v_{average} + \boldsymbol{E}_t \tag{3.23}$$

and

$$\boldsymbol{D} = \lambda^2 + \boldsymbol{E}_x^2 + \boldsymbol{E}_y^2 \tag{3.24}$$

In the above equations, $\boldsymbol{E}_x$, $\boldsymbol{E}_y$, $\boldsymbol{E}_t$, and λ represent the spatial gradients in the **X** and **Y** directions, the temporal gradient, and a constant multiplier, respectively. When only two frames

are used, the computation is iterated over the same frames many times. For more than two frames, each iteration uses a new frame.

This classic method to compute optical flow has been used for more than twenty years since it first appeared, but it still has some problems. One important fact to remember about this method is that it assumes a linear variation of intensities and computes the point-wise velocities under this assumption. It is typically expected that this assumption is satisfied at edge points in images and, hence, the velocity can be computed at these points. The smoothness constraint is not satisfied at the boundaries of objects because the surfaces of objects may be at different depths. When overlapping objects are moving in different directions, the constraint will also be violated. These abrupt changes in the velocity field at the boundaries cause problems. If we want to remove these problems, we should use some other information to refine the above optical flow determining method. However, since our algorithm is not totally dependent on optical flow computation, this method is enough for us.

3.3.3 Moving Blocks Detection

In the above two subsection, we detect the changed parts using difference picture and compute the optical flow for each point using gradient-based method. In this subsection, we will describe how to really detect the moving parts using these two kinds of information. The method we use is based on block.

For two input frames $\mathbf{F_{(x,\ y,\ j)}}$ and $\mathbf{F_{(x,\ y,\ k)}}$, we now can get the difference picture $\mathbf{DP_{jk(x,\ y)}}$, and compute the optical flow $(u,\ v)$ for each pixel $(x,\ y)$, using the above-mentioned algorithms. Then we divide the images into blocks, the size of which is denoted by $\mathbf{S}_{block}$.

The following detection is based on block. For each block, we assign a value $\mathbf{M}_{block}$ to indicate the motion property of this block, which is calculated using the following equation:

$$\mathbf{M}_{block} = \sum_{\substack{\mathbf{S}_{block} \\ \mathbf{DP}(x,y)=1}} \sqrt{u^2_{(x,y)} + v^2_{(x,y)}} \tag{3.25}$$

where $\mathbf{S}_{block}$ means this computation is done within the size of one block.

This equation summarizes the motion value from the optical flow. To get more accurate result, we only calculate the pixels with great difference, which can eliminate some noise from optical flow. Furthermore, we can use smaller block size to make the block-based detection more closed to the object boundary.

The judgement of the moving block is based on the $\mathbf{M}_{block}$ using the following criterion:

$$The\ block\ is\ moving: \begin{cases} true & if \quad \mathbf{M}_{block} > \tau \\ false & otherwise \end{cases} \tag{3.26}$$

The input images in this detection are coarse registered images, in which the background is already made static. Therefore, the moving blocks we detect here are *really* caused by object moving instead of camera moving.

However, we need a pair of frames at one time. Therefore, if we have **N** input frames in total, we can detect moving blocks in only **N-1** frames.

3.3.4 Moving Blocks Refinement

In the last step, we detect the moving blocks using the combination of difference picture and optical flow. However, due to the slight alignment error in the coarse registration step, the

computation limitation of optical flow, and the image noise, there will be false alarms in the detection results out of question. In order to refine the detection, we utilize some properties in object motion and camera shoot. We will describe these useful properties in the following respectively.

Motion Constancy

One of most obvious properties we can use is *motion constancy*. In [24], motion constancy is an assumption which assumes that motion remains uniform in the analyzed sequence. This assumption is reasonable under the similar reason of smoothness constraint in optical flow computation, because If the motion is not stable and pixels can move independently, the computation of velocity and the detecting as well as tracking are almost impossible.

According to this assumption, we can eliminate some noise blocks which are wrongly detected in the last step. We use $\mathbf{B}_(a, b, n)$ to denote one detected moving block in frame **n**, then its neighbor blocks in the next frame **n+1** are denoted as $\mathbf{B}_(a, b-1, n+1)$, $\mathbf{B}_(a, b+1, n+1)$, $\mathbf{B}_(a-1, b, n+1)$ and $\mathbf{B}_(a+1, b, n+1)$. The refinement is done as followings:

1. for every moving block in frame **n**
2. Search its neighbor blocks in frame **n+1**.
3. If at least one of the neighbor blocks is also detected as *moving* in frame **n+1**, the original block in frame **n** is confirmed as *moving*.
4. Otherwise, if none of neighbor blocks are moving in the next frame, this block is discarded as noise block.

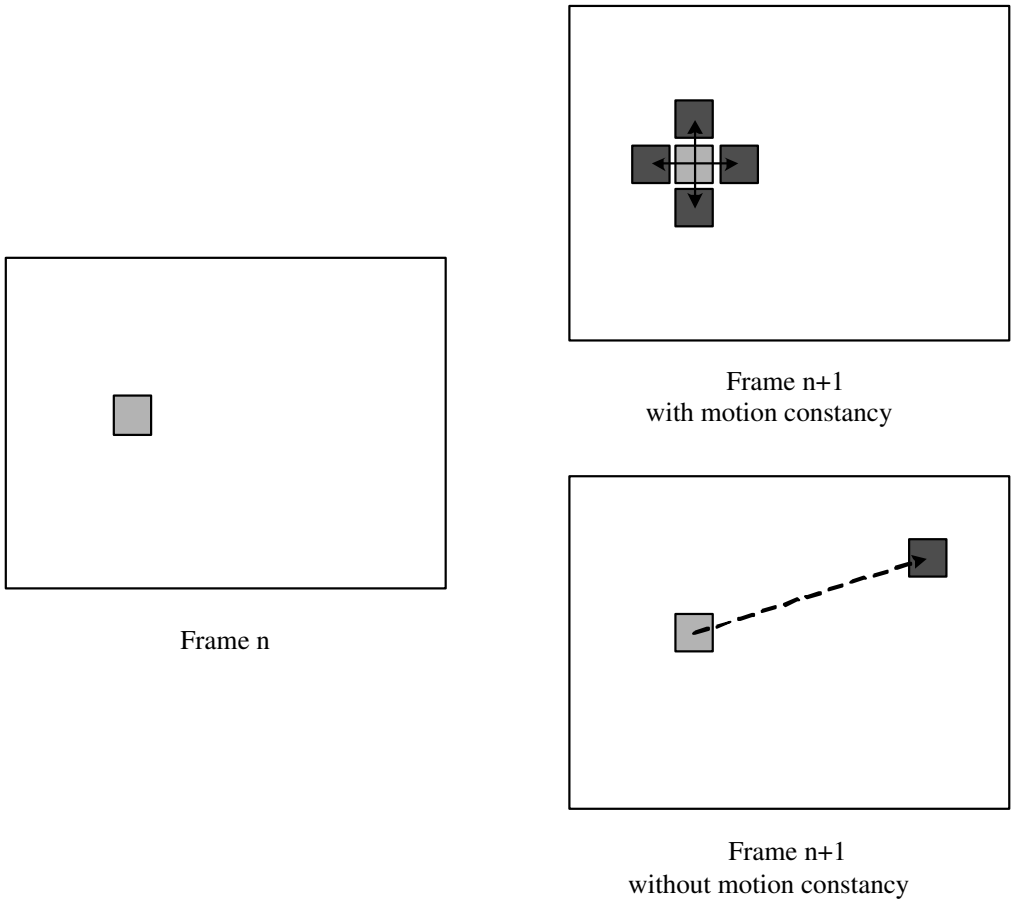

Figure 3.18. The Motion Constancy Property in Successive Frames.

This idea can be illustrated clearly in Figure 3.18. The block filled with thin grey color denotes the moving block in the first frame, and the blocks filled with thick grey color denote the moving blocks in the next frame.

Camera Focus Property

In this part, we describe another property coming from cinematography practice, which can also be used to improve the detection results. Generally speaking, if there are moving objects in the view scope, cameramen will follow these objects and place them in the central part to attract more attention from viewers. Therefore, if the detected moving blocks are very close to the edge

of the frames, we can assume that they are noise blocks and discard them safely.

In a complete refinement process, a detected moving block should meet these two criteria simultaneously, then it can be confirmed to be *really* moving.

3.4 Moving Objects Reconstruction

Till now, we have got the whole mosaic, the estimated reference matrix, and the moving blocks in every input frame. Using these information, we can build static background without moving objects and reconstruct these objects on the final mosaic.

To fulfil this task, we have to do the following things: separate moving objects from input frames, build static mosaic using processed inputs, reconstruct moving objects. The concept of *reconstruction* is first proposed here, therefore, our methods to do the relative work are newly developed based on this concept. We will describe them respectively.

3.4.1 Moving Objects Separation

In this subsection, we will discuss how to separate the moving objects from the input images.

One important fact we should remember is that the moving block detection is done in coarse registered images, but the moving objects separation should be done in original input images. Therefore, it is necessary to map the blocks detected in coarse registered images to input images, using final reference matrix computed before. This map is done pixel by pixel within the blocks, and the method is almost the same as that in section **Frame Warping and Registration**. We use $\mathbf{RI}_1$, $\mathbf{RI}_2$, $\cdots$, $\mathbf{RI}_n$ to denote the coarse registered images, and $\mathbf{OI}_1$, $\mathbf{OI}_2$, $\cdots$, $\mathbf{OI}_n$ to denote the corresponding original input images.

Now we get the corresponding position and scope of moving blocks in original input image. However, to separate them, we can not simply set the intensity value of the pixels within these blocks to zero, because if we do like that, there will be *black holes* in these images, and these holes will remain in the background mosaic, which impacts the result badly.

In order to solve this problem, we do *pixel compensation* instead of simply resetting the value. The algorithm is presented as follows:

1. for each pixel $\mathbf{p}$ within each moving block in one registered image $\mathbf{RI}_k$
2. Using the final reference matrix $\mathbf{FR}$ to get its corresponding pixel in each original input image, denoted by $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$. Of course, these pixels should within the image scope, otherwise, it is set to null.
3. Compute the intensity value of pixel $\mathbf{p}_k$ in corresponding input image $\mathbf{OI}_k$ using the following formula:

$$\mathbf{p}_k = \frac{1}{c} \sum_{\substack{i=1 \\ i \neq k}}^{n} \mathbf{p}_i \tag{3.27}$$

   where $\mathbf{p}_i$ should within the background part and constant $\mathbf{c}$ denotes the number of this kind of pixels.

4. Repeat the step 1 to 3 for each registered image to process each original input image.

The idea of the above algorithm is to replace the pixel value in the moving parts in one input image with the average value of all the corresponding pixels in background parts in other input images. In the *processed* images, moving parts are removed from the background, and the *hole*

after removal is compensated using the information from other images. However, sometimes *ghost* will appear in the processed images, which is caused by illumination change or misalignment error. In order to remove the *ghost*, we use selected input frames instead of all, and calculate the pixel value using alpha-blending technique instead of simple average algorithm.

3.4.2 Static Background Construction

In the last step, we get a series of images which are very similar to the original images, and the only difference between them is that, in the former, the moving parts have already been removed and compensated.

Now we can build a background mosaic which is completely static. The construction process is the same as that in first step, and the homographies are also the same, because we only change some pixel values in the images and the alignment remains unaltered.

3.4.3 Pixel-Based Reconstruction

Till now, we get the static background mosaic, input images with moving blocks detected, and reference matrix. These information is enough for us to do the last step: moving objects reconstruction.

Actually, the reconstruction here is a refined version of the coarse registration. In the **coarse frame registration** step, we register the whole frame using the reference matrix. Here, we register the moving blocks using the same matrix. Therefore, we won't repeat the registration algorithm here, but still need to mention one important fact.

Unlike the coarse registration, which place the whole frame on the mosaic pixel by pixel, here

we do not place the whole moving blocks. This is because our motion detection is based on block, so that the detection result doesn't follow the boundaries of objects accurately. If we place the whole blocks, there will be visible *box* around the moving objects.

To solve this problem, we only select suitable pixels in the moving blocks to replace the background pixels in the final static mosaic. The selection of pixel is based on the difference of the intensity value from pixel in moving blocks to pixels in mosaic background. If there is great difference, we can assume that this pixel is really from moving object so that we can replace the background with it, otherwise, we know that this pixel is quite similar to the background so that the original background remains unaltered.

Adding this comparison step before refined block registration, we can almost register the objects in their really shapes. Strictly speaking, what we use is a combination of block-based method and pixel-based method. The block-based method is used to find the moving parts roughly, while the pixel-based method is used to pick the moving pixels accurately in the blocks. Combining these two, we can detect and register moving objects efficiently and accurately.

Overall, the whole algorithm can be summarized as the following sequences.

The first subsection, static mosaic construction, is shown in Algorithm 1.

The second subsection, coarse frame registration, is shown in Algorithm 2.

The third subsection, moving objects detection, is shown in Algorithm 3. This part should be done for every two coarse registered mosaics.

The fourth subsection, moving objects reconstruction, is shown in Algorithm 4.

Input : One input video: V
Output: One large mosaic image: M

Procedure:

1. Separate the input video into a series of n images

$V = (I_1, I_2 \ldots, I_n)$.

2. Calculate a group of corresponding points between I_1 and I_2

$p_1 = \text{Correspond}(I_1,\ I_2)$.

3. Compute forward and backward homographies from corresponding points

$(H_F, H_B)_1 = \text{Homography}(p_1)$.

4. Integrate I_1 and I_2 into I_{12}

$I_{12} = I_1 + I_2$

5. Repeat 2-4, but replace the input of I_1 and I_2 with I_{12} and I_3, and so on.

6. $M = I_{12\ldots n}$.

Algorithm 1: Static Mosaic Construction

Input : One large mosaic image: M, Separated n images: $I_1, I_2 \ldots, I_n$
Output: Coarse registered n Mosaics: $M_1, M_2, \ldots, M_n$

Procedure:

1. Compute final reference matrix for every image from (H_F, H_B) in last subsection

$H_{FR} = \text{Final}(H_F, H_B)$.

2. Register every image on the mosaic

$M_1 = \text{Register}(M, I_1, H_{FR1})$.

3. Repeat step 2 for every image.

Algorithm 2: Coarse Frame Registration

Input : Coarse registered mosaics:M_k, M_{k+1}
Output: Image with moving Blocks: Mk_{MB1}, Mk_{MB2}, ..., Mk_{MBn}

Procedure:

1. Calculate the difference picture between M_1 and M_2

$DP_{k,k+1}$ = Difference(M_k, M_{k+1}).

2. Calculate the optical flow between M_k and M_{k+1}

(u, v) = OpticalFlow(M_k, M_{k+1}).

3. Create a moving image and divide the image into blocks

$Mk = (Mk_{B1}, Mk_{B2}, \ldots, Mk_{Bn})$.

4. Calculate the motion value for each block. If this value exceeds a threshold, we primarily detect this block as moving

$M_{block} = M(u, v, DP)$

5. Discard a moving block if its motion is not constant.

6. Discard a moving block if it is close to the edge.

Algorithm 3: Moving Objects Detection

Input : n images with detected moving blocks: (Mk_{MB1}, Mk_{MB2}, ..., Mk_{MBn}) for every image

Output: The result video: V_{output}

Procedure:

1. Extract moving parts from the original images and do compensation

I_k = Moving Objects I_{mk} + Background I_{bk} .

2. Build new background mosaic using Algorithm 1, but this time the input images are

$I_{b1}, \ldots, I_{bn}$

M_b = Mosaic($I_{b1}, I_{b2}, \ldots, I_{bn}$).

3. Register moving objects on the background mosaic

M_{bk} = Register($M_b, I_k, Mk_{MB1}, Mk_{B2}, \ldots, Mk_{Bn}$).

4. Make output video from n mosaic images after reconstruction

$V_{output} = (M_{b1}, M_{b2}, \ldots, M_{bn})$

Algorithm 4: Moving Objects Reconstruction

Chapter 4

Experimental Results

In this chapter, we will use some video examples to evaluate our approach. Due to our simple algorithms used in the processing steps, there are some constraints in the scene situation, such as depth variation and motion parallax. Therefore, the video examples we use here are captured by ourselves.

Example No.1 is used to show the result of the whole process.

Figure 4.1 is the selected frames from a video sequence in which a man is walking across the road.

Figure 4.2 is the static mosaic from all available frames without layer separation. Since we update this mosaic every time when we get a new frame, this mosaic only displays the final position of the moving object.

Figure 4.3 is the selected result of coarse frame registration. We can see clearly the edges of the registered frames due to little warping and alignment error. However, these kinds of error won't

Figure 4.1. Selected Frames from Walking Video Sequence.

Figure 4.2. Mosaic Image after Static Mosaic Construction.

Figure 4.3. Coarse Frame Registration.

Figure 4.4. Moving Blocks Detection.

impact our final result, because we only use the difference pictures of these frames in the later steps, not the pixel position and their intensity value.

Figure 4.4 shows the result of moving object detection. The black parts show the corresponding position of moving objects. Please notice that the size of the block has great influence on the detection result. At the present, the size is determined empirically. We find that when there is a suitable scale, such as 1:8 $\sim$ 1:10 between the block size and the object size, we can get the better detection result.

Figure 4.5 shows the result of pixel compensation after we detect the moving blocks. We can see that in the second image, the pixels of moving object is replaced by average the background pixels from other images.

However, we still can notice shadow in the second image. The shadow in this frame is caused by pixel compensation because we have to remove the object from the scene. Actually, the shadow exists in all frames, but it is covered in other frames by the mosaic constructing process (we

Figure 4.5. Pixel Compensation in Moving Objects.

update the mosaic using the most current frame every time when we get a new input). The final frame appears as a whole in the final mosaic, so the shadow is visible. That is to say, the shadow in other frames is covered by an entire frame, not pixel, so it is "seamless". If we use pixel to cover the shadow (the original position of the object), as we did in the last frame, it is still visible, even if we already tried our best to choose the most appropriate pixels. Remember, we still have illumination change in the sequence, which cause pixel value to change even if the pixel is in the same position throughout all frames.

Figure 4.6 shows the final static mosaic with moving objects reconstruction. Now the background is static and the moving object is still moving.

Example No.6 shows one kind of application of our work, which can be called *shake removal*. Since we can build a complete static background mosaic of the scene, it does not matter whether the input video is shaky or not. Then we can retrieve the frames from the mosaic by imagining that there is a *virtual stable camera* and build a new video clip using the new frames. After this frame-scene-frame conversion, we can change shaky video to stable video easily and quickly. The

Figure 4.6. Moving Objects Reconstruction on the Static Background Mosaic.

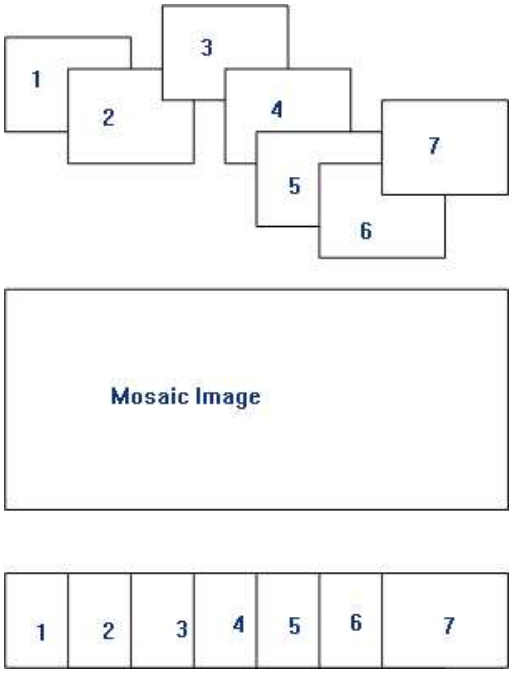

Figure 4.7. Shake Removal.

idea of this application can be shown as Figure 4.7.

However, we can only view the result of example No.6 in video format, so we will give the link for it later. Moreover, example No.6 is also better viewed in video format, and we have some other test examples on our website too. All these experiment results can be accessed in our website given by URL:

http://www.comp.nus.edu.sg/∼shenhui/project.htm

All the experiments can be summarized using Table 4.1 (The test platform is Pentium IV 1.6G with Windows 2000. All examples have 10 frames.).

From Table 4.1, we find that there are quite a few false alarms in coarse detection, because it is easily influenced by the mosaic alignment. However, our two improvement methods can eliminate

Property	Ex 1	Ex 2	Ex 3	Ex 4	Ex 5a	Ex 5b
Frame Size	320×240	320×240	320×240	320×240	320×240	320×240
Block Size	15×15	20×20	20×20	15×15	20×20	25×25
Time (ms)	16163	14671	18456	15152	17164	17796
Coarse Blocks	275	152	528	384	482	405
Refined Blocks	109	75	83	82	119	109

Table 4.1. Quantitative Evaluation of Examples

Figure 4.8. Two Input Images.

most of the noise and smoothen the results.

Furthermore, we should point out that the scenes in our examples are similar so that our algorithm seems not to be applicable to various situations. Actually, this constraint is caused by our quite simple mosaic construction method. It requires the scene to be almost planar and without motion parallax. If we take a look at the following example, where Figure 4.8 is two selected input images and Figure 4.9 is the static mosaic image, we can find that there are quite a few misalignments in the mosaic image and the scene is a bit distorted. Therefore, our algorithm is applicable only to some certain kinds of scenes. One problem in this example is that the scene is not planar, and object moves not only in X-Y plane, but also in Z direction. One possible solution is to add zoom parameters in the algorithm to deal with motion parallax situation. Of course, we can also change to a more complex motion model which we use in mosaic phase to release the scene limitations in a more general way.

Meanwhile, the objects in the scene should have great contrast with the background. This assumption is reasonable because otherwise, it is even difficult for human beings to detect the objects, let alone computer vision techniques.

Figure 4.9. Static Mosaic Image.

Therefore, a better mosaicing technique for more complex scenes can help to generalize our algorithm. Also, we can utilize more parameters to detect the moving objects because we find the coarse detection still has a long way to go. For example, there are still some noise blocks in several experiments even after detection refinement, and these noise blocks cause problems again in later steps.

Chapter 5

Conclusion

In this thesis, we designed a system to separate the moving parts from static parts in input video sequence, then reconstruct the moving parts on the background mosaic which is built from the static parts. In Chapter 1, we introduced the research background, state our motivation and describe the problem which we want to solve. Chapter 2 is the literature survey, which provided the theoretical background of our work. Since our work roughly consists of three parts: static mosaic, image registration, and motion detection, this chapter was also divided into three sections. Chapter 3 was the main body of this thesis. We presented the complete methods on how to build the mosaic, to detect the moving objects, and to reconstruct them on the mosaic background. We have proposed a novel method to register images on their mosaic which can lead to more interesting applications. We have then developed a new algorithm for motion detection based on our mosaic techniques, which is both simple and efficient. Of course, we have implemented our techniques and tried them on several real videos. The experimental results shown in Chapter 4 are promising despite the fact that the input videos are with some constraints due to the limitation

of our simple algorithms.

The most important contribution of our work is that our new mosaic technique is able to retain all the information in dynamic scenes, especially the motion, which is usually omitted in past work. However, our work is only a preliminary step towards the development of the complete system and the accurate theory. All the results should be evaluated with more accurate mathematical parameters instead of dependent on human feelings. Moreover, we can apply more advanced basic mosaic techniques in our first step to enable the complete algorithm to deal with more complicated scenes. Actually, the main objective of our work is simple and effective, so there are limitations in almost every step due to the simple algorithms we used.

In our mind, the perfect system should be able to process various kinds of input videos automatically, which may be with different camera motion and much object motion, separate the moving objects along their accurate boundaries, and reconstruct them on the final background mosaic at their accurate positions. If this kind of system is the eventual goal, there are quite a few interesting and challenge problems for future study:

1. Automatic correspondence establishment in a dynamic scene

    - To construct mosaic, the first step is to find correspondence between images as the information for alignment. If the scene is static, many techniques can be applied. However, important problem rises here if the scene is dynamic with object motion. Even if we still can find correspondence within the region of moving objects, this information cannot be used to build mosaic, otherwise it will cause alignment errors. Therefore, we should have a method to tell the difference between *true*

correspondence and *false* correspondence and how to select the correct one.

2. Illumination change in coarse registration

   - As we described before, we do coarse frame registration as a preparation step before motion detection. However, if the illumination in the scene changes during camera shooting, it will cause additional "alignment error" in the registration. The cause is like the following: The final mosaic is always updated until the last frame, so that if we register successive frames in the middle part of the stream, the illumination change will be great enough between them and the mosaic to cause detectable different part in difference picture, which may be wrongly detected as moving objects later. To solve this problem, maybe we can compensate for the illumination before frame registration according to its change.

3. Moving object detection along its accurate shape

   - Our present method for motion detection is based on block, so that we cannot follow the real shape of the objects, which also causes problems in later steps. Of course, we cannot change the block-based method to pixel-based, expecting that this can solve the problem. Pure pixel-based method will be prone to noise and difficult to group in object regions. Actually, some techniques are already available for accurate edge detection for moving objects, such as [33], but it is too complicated to be incorporated. We want to have a more simple one on the same basis of ours but presenting accurate shape.

4. Sequence-to-sequence alignment

- Presently, our method is applied to video sequence within one camera shot, because it is easy to locate and track the moving objects within the same shot. However, it would be better if we can automatically align two sequences in the same scene from different shot, and still be able to detect and reconstruct the moving objects at their correct positions.

5. Extension of availability

   - Apply our algorithm to more general videos. If some of the above problems can be solved so that there are less limitations, we can extend the availability of our algorithm to videos from more complicated scenes or with more complicated camera motion.

The long-term goal of our research is to build novel video descriptors, not only the mosaic, but also in other formats. Following this explanation, in our present work, we only separate two layers and make a few descriptors such as background mosaic and moving object regions. From this starting point, it is possible to extract more information from videos and make more kinds of descriptors. The more interesting thing we can imagine is that: we can make an intelligent video analysis and processing system. We tell it what we want to know from a video, then it can analyze our requirements, extract the necessary descriptors automatically, and finally combine these descriptors together to make a new representation of video, other than the traditional time-based frame stream, according to the requirements of users and applications.

Bibliography

[1] A.Zisserman and D.Capel. Automated mosaicing with super-resolution zoom. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 120–128, 1998.

[2] Jorge Badenas, Jose Miguel Sanchiz, and Filiberto Pla. Motion-based segmentation and region tracking in image sequences. *Journal of Pattern Recognition*, 34(3):661–670, 2001.

[3] Berthold.K.P.Horn and Brian.G.Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[4] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[5] H.T.Nguyen, M.Worring, and A.Dev. Detection of moving objects in video using a robust motion similarity measure. *IEEE Transactions on Image Processing*, 9(1):137–141, 2000.

[6] http://tns www.lcs.mit.edu/manuals/mpeg2/.

[7] http://www.behere.com.

[8] http://www.intel.com/research/mrl/research/opencv/.

[9] H.Y.Shum and L.W.He. Rendering with concentric mosaic. In *ACM SIGGRAPH Computer Graphics Annual Conference*, pages 299–306, CA, USA, August 1999.

[10] H.Y.Shum and R.Szeliski. Panoramic image mosaics. Technical Report MSR-TR-97-23, Microsoft Research, 1997.

[11] H.Y.Shum and R.Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, February 2000.

[12] H.Zheng and D.Blostein. Motion-based object segmentation and estimation using the mdl principle. *IEEE Transactions on Image Processing*, 4(9):1223–1235, September 1995.

[13] Dae-Sik Jang, Seok-Woo Jang, and Hyung-Il Choi. 2d human body tracking with structural kalman filter. *Journal of Pattern Recognition*, 35(10):2041–2049, October 2002.

[14] J.Davis. Mosaics of scenes with moving objects. In *Proc. of IEEE International Conference on CVPR*, pages 354–360, Santa Barbara, USA, June 1998.

[15] J.D.Courtney. Automatic video indexing via object motion analysis. *Journal of Pattern Recognition*, 30(4):607–625, April 1997.

[16] J.Meehan. *Panoramic Photography*. Watson-Guptill, 1990.

[17] J.M.Odobez and P.Bouthemy. *Video Data Compression for Multimedia Computing*, chapter Chapter 8, Separation of Moving Regions from Background in an Image Sequence Acquired with a Mobile Camera, pages 295–311. Kluwer Academic Publisher edition, 1997.

[18] J.R.Bergen, P.Anandan, K.J.Hanna, and R.Hingorani. Hierarchical model based motion estimation. In *Proc. of Second European Conference on Computer Vision*, pages 237–252, 1992.

[19] J.Y.A.Wang and E.H.Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, September 1994.

[20] Wing Ho Leung and Tsuhan Chen. Compression with mosaic prediction for image-based rendering applications. In *IEEE Internation Conference on Multimedia and Expo*, pages 1649–1652, New York, USA, July 2000.

[21] L.McMillan and G.Bishop. Plenoptic modeling: An image-based rendering system. In *ACM SIGGRAPH Computer Graphics Annual Conference*, pages 39–46, August 1995.

[22] M.Gelgon and P.Bouthemy. A region-level motion-based graph representation and labeling for tracking a spatial image region. *Journal of Pattern Recognition*, 33(4):725–745, April 2000.

[23] M.G.Gonzalez, P.Holifield, and M.Varley. Improved video mosaic construction by accumulated alignment error distribution. In *Proc. of 9th British Machine Vision Conference*, http://www.bmva.ac.uk/bmvc/1998/papers/d024/h024.htm, 1997.

[24] M.Irani, B.Rousso, and S.Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16, January 1994.

[25] M.Irani and P.Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):577–589, June 1998.

[26] M.Irani and P.Anandan. Video indexing based on mosaic representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 86(5):905–921, May 1998.

[27] M.Irani, P.Anandan, J.Bergen, R.Kumar, and S.Hsu. Efficient representations of video sequences and their applications. *Signal Processing:Image Communication*, 8:327–351, 1996.

[28] M.Irani, P.Anandan, and S.Hsu. Mosaic based representation of video sequences and their applications. In *Proc. of International Conference on Computer Vision*, pages 605–611, Boston, USA, June 1995.

[29] M.Levoy and P.Hanrahan. Light field rendering. In *ACM SIGGRAPH Computer Graphics Annual Conference*, pages 31–42, August 1996.

[30] M.Lhuillier, L.Quan, H.Shum, and H.T.Tsui. Relief mosaics by joint view triangulation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, USA, December 2001.

[31] P.J.Burt and E.H.Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[32] R.C.Jones, D.DeMenthon, and D.S.Doermann. Building mosaics from video using mpeg motion vectors. In *Proc. of 7th ACM Multimedia*, pages 29–32, Florida, USA, 1999.

[33] R.Fablet, P.Bouthemy, and M.Gelgon. Moving object detection in color image sequences using region-level graph labeling. In *Proc. 6th IEEE International Conference on Image Processing*, pages 939–943, Kobe, Japan, October 1999.

[34] R.Jain, R.Kasturi, and B.G.Schunck. *Machine Vision*. McGraw-Hill, 1995.

[35] S.E.Chen. Quicktime vr - an image-based approach to virtual environment navigation. In *ACM SIGGRAPH Computer Graphics Annual Conference*, number 29, pages 29–38, August 1995.

[36] S.J.Gortler, R.Grzeszczuk, R.Szeliski, and M.F.Cohen. The lumigraph. In *ACM SIGGRAPH Computer Graphics Annual Conference*, pages 43–54, August 1996.

[37] Dezhen Song, A.Frank van der Stappen, and Ken Goldberg. Collaborative frame selection: Exact and distributed algorithms for a networked robotic camera with discrete zoom levels. Submitted to the International Journal of Robotics Research, May 2003.

[38] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *ACM SIGGRAPH Computer Graphics Annual Conference*, number 31, pages 251–258, 1997.

[39] T.Wong, P.Heng, S.Or, and W.Ng. Image-based rendering with controllable illumination. In *Proceedings of the 8th Eurographics Workshop on Rendering*, pages 13–22, France, June 1997.

[40] U.Bhosle, S.Chaudhuri, and S.D.Roy. The use of geometric hashing for automatic image mosaicing. In *Proc. of National Conference on Communications*, pages 533–537, India, 2002.

[41] U.Bhosle, S.Chaudhuri, and S.D.Roy. Background mosaicing for scenes with moving objects. In *Proc. of National Conference on Communications*, pages 85–89, India, 2003.

[42] P.Ehran Y.Altunbasak and A.Murat Tekalp. Region-based parametric motion segmentation using color information. *Graphical Models and Image Processing*, 60(1):13–23, January 1998.