CAMERA BASED ROBOTIC CONTROL

By

QIANG NIU

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2011

To my dear parents, thank you so much for your full support!

ACKNOWLEDGMENTS

I would like to acknowledge the inspirational guidance to Dr. Carl D. Crane. Without his patient support and kind advice, I could not have completed this thesis. I also would like to thank managers of Center for Intelligence Machines and Robotics, Dave Armstrong and Shannon Ridgeway for their valuable advice.

I would like to thank all my lab mates at CIMAR. Special thanks to Vishesh Vikas for his explanations of some terminologies and concepts. Junsu Shin and Sujin Jang have been very generous in supporting of programming assistance. I would also like to thank Anubi Olugbenga Moses and his fiancée, Serena Sealy for their patient support.

Finally I would like to thank all my friends for their support and encouragement, Joel Oliver, Andrew Sharkey, Ayman Abdellatief and Divik Schueller. They helped me a lot for revising my writing. Again, special thank my parents, Ying Hai and Jingyuan Niu. I could never have finished my thesis without their selfless love and support!

TABLE OF CONTENTS

page

LIST OF TABLES

Table page

LIST OF FIGURES

LIST OF ABBREVIATIONS

DOG	Difference of Gaussian
DOH	Difference of Hessian
LOG	Log of Gaussian
NN	Neural Network
PCA	Principal Components Analysis
RANSAC	Random Sample Consensus
SIFT	Scale Invariant Feature Transform
SSD	Sum of Squared Difference
SURF	Speed Up Robust Features
SUSAN	Small Univalue Segment Assimilating Nucleus

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

CAMERA BASED ROBOTIC CONTROL

By

Qiang Niu

August 2011

Chair: Carl D. Crane III
Major: Mechanical Engineering

Computer vision is a new field of study that can greatly be used in autonomous vehicle navigation or other industrial automation. In this thesis, the SIFT feature which is a scale invariant feature transform is applied in multiple computer vision tasks such as image stitching and object recognition.

The full derivation of the Lowe SIFT features is discussed in detail in this thesis. In the image stitching part, first the SIFT features are extracted from images. Then k-d tree method is used for feature matching between the sets of the SIFT features of two images. Once the matches are obtained, the homography transformation matrices are obtained with the matches using the RANSAC method. Image stitching is achieved after the image is transformed by the homography transform matrix.

Panorama stitching is a further development of image stitching. The goal is to produce a panoramic stitched image while the vehicle is moving through an unknown environment. There are some additional processes that need to be developed for the panorama stitching, such as color correction. Due to the illumination differences, there may be artificial edges in the stitched image, so color correction is used to minimize the artificial edges.

The SIFT feature can also be applied in object recognition which is called feature based object recognition. Since the SIFT features are distinctive and invariant, it is plausible to use them to achieve object recognition. Through the feature matching and verification processes, the SIFT feature is able to recognize the object from different scene images. The number of feature matches is also returned as a standard for object recognition.

CHAPTER 1
INTRODUCTION

Motivation

Computer Vision is a relatively new field of study and it can be applied to a variety of fields. It enables the machine to utilize images to extract information that is necessary to solve tasks. Many methods and applications about Computer Vision are still in the state of basic research, but more and more efforts have found their way into commercial products, such as automobile driver assistance, digital photography, people tracking, medical and biomedical images, and in the area of industrial automation and inspection. Camera sensors are becoming economical and thus problem solving is becoming more prominent with the use of camera.

Camera based robotic control is one of the applications of Computer Vision. It is widely used in the field of navigation for unmanned autonomous vehicles. It enables a robot to be controlled with the feedback information extracted from camera sensors. The images captured from the camera sensors provide a large amount of information. But image data analysis processes are computationally expensive so that making concise use of images becomes necessary.

Instead of processing all the image data, image features are more representative to be used for future tasks. There are varieties of methods to detect the features, such as corner detection, edge detection, or blob detection. The exact extraction of features depends on the problem or the type of application. To achieve the task of image stitching, features that are invariant to translations, rotations, scaling, and other transformations will be important. There are some methods that are able to detect critical features and they are discussed in the literature review sections.

Literature Review

There are different methods to detect features from images. Through comparison of these methods in this literature review, the most suitable feature is chosen for image stitching.

FAST Corner Detection

FAST corner detection is used as the first step of computer vision tasks such as tracking, localization, image matching, and recognition. Corner detectors have been widely used as feature point detectors because corners correspond to image locations with high information content and they can be matched between images reliably. FAST corner detection is high speed detection compared with Harris, SUSAN, or LOG, DOG, and DOH detection which yield high quality features but are computationally intensive (Luo Juan and Oubong Gwun in 2010). This is important because FAST can be used in real time applications. FAST corner detection is able to achieve high quality feature detection and it is repeatable between images.

But FAST corner detection also has its cons. The high speed doesn't generalize well so the choice of high speed is not optimal. It’s not robust to high levels of noise and it is dependent on threshold. But according to the high speed detection, FAST corner detection is often used in real time video tracking, such as the application in smart phones developed by Nokia Research Center.

SIFT Feature Detection

Lowe (2004) presented the SIFT for extracting distinguished invariant features from images that can be invariant to image scale and rotation. The SIFT features can also provide robust matching across the substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination.

Developed by Lowe in distinctive image features from scale invariant keypoints, the SIFT feature detection includes four stages: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor. DOG was used instead of Gaussian to identify the potential interested points in order to improve the computational speed. Hessian Matrix was used to compute the principal curvatures in order to eliminate low contrasts points and edge responses. Dominant orientations are assigned to localized keypoints. These steps ensure the invariant features. More details about the SIFT will be discussed in Chapter 3.

PCA-SIFT Feature Detector

The PCA-SIFT feature detector is a technique for dimension reduction. PCA-SIFT feature is able to represent the keypoint patches and enables one to linearly project high dimensional samples into a low dimensional feature space.

PCA-SIFT uses PCA instead of a histogram to normalize patches. The feature vector is significantly smaller than the standard SIFT feature vector and it can be used with the same matching method. The PCA based feature descriptors are also distinctive and robust to image deformations. However PCA still has its shortcomings, such as its implicit assumption of Gaussian distributions and its restriction to orthogonal linear combinations. The methods of extracting robust features are time intensive. As a result, the PCA-SIFT feature detector is often used in high dimension situations.

SURF Feature Detector

The SURF feature is partly inspired by the SIFT descriptor, but it’s several times faster and more robust against different image transformations than the SIFT. SURF employs a slightly different way to detect features from the SIFT. The SIFT builds an

image pyramid and then uses DOG. SURF creates a stack without down sampling for higher levels in the pyramid resulting in images of the same resolution.

The invariant features are always preferred but in some cases, rotation invariance can be left out, resulting in a scale-invariant only descriptor, which is ‘upright SURF’ (U-SURF). In the applications like mobile robot navigation, the camera only rotates about the vertical axis. The benefit of avoiding the rotation invariance is not only increased speed, but also increased discriminative power.

SURF relies on integral images to reduce the computation time so it is called a “Fast-Hessian” detector. The descriptor is based on sums of approximated 2D Haar wavelet responses within the interest point neighborhood, which can be computed extremely fast. However, SURF does have some loss of accuracy from the SIFT in certain situations, such as it’s not as invariant to illumination and viewpoint change as the SIFT.

CHAPTER 2
RESEARCH GOAL

Problem Statement

From the previous literature review, each type of image feature has advantages and drawbacks. For the image stitching task, it is desired to get the image features which do not vary with differences in translations, rotations, and changes in the 3D view. In order to produce correct matches for stitching, not only the features need to be distinguished, but also the number of features detected needs to be sufficient. The speed of finding the features is less critical than the accuracy. So the SIFT features are used to accomplish the image stitching task.

For image stitching, it is desired to seamlessly stitch together the images that have overlap area and make the stitched image result look natural. Once the features are indentified, the next step is to determine the common features that match between the two images. In order to stitch the image together, one of the images is transformed into the image plane of the other image using the matches between two images.

A further use of stitching is panorama stitching. The vehicle is expected to get multiple images while it is traversing through an unknown environment, then produce a single image of the surroundings for the unknown environment. However, there are some problems to solve when stitching several images together. The image may not be straightened after stitching and there may be some artificial edges between the images due to the differences in illumination. Straightening the image and gain compensation are necessary processes after the normal stitching.

The SIFT feature is very distinguished and large number of the SIFT features can be detected effectively from an image. It is plausible to achieve feature based object

recognition with the SIFT features. This feature based object recognition can be used when training images are stored as local data. Object recognition is then performed while the vehicle is capturing images. It recognizes an object defined in the training image is detected.

Development

As described in the problem statement section, three main tasks are discussed in this thesis: image stitching, panorama stitching, and object recognition using the SIFT features. The development of the tasks in this thesis is as follows:

- Lowe Approach to detect the SIFT features
- A method is developed to find the matches between the SIFT features. This method can be used in image stitching and in object recognition.
- Finding the homogenous transformation with the SIFT matches to achieve image stitching.
- Perform panorama stitching using the image stitching method. Color correction is applied additionally for elimination of the artificial edges caused by the differences in illumination.
- Feature based object recognition is achieved by matching the detected the SIFT features. A verification method is introduced to find better matches with fewer errors involved.

CHAPTER 3
IMAGE STITCHING

Background

Image stitching is a fundamental problem in computer vision. It is a process of combining multiple images with overlapping fields of view to produce a panoramic panel of a high resolution image. On an unmanned autonomous vehicle, image stitching can be used when processing images captured from a camera mounted on the vehicle, in order to extract more information from smaller image data files.

There are four steps to process image stitching: to find features of the overlapped area of two images, to get feature matches from each feature set of the images, to find the homogenous transformation of the two images, and to get warp image and blending. From the literature review section, research suggests use of the SIFT feature method, which is invariant to scaling, rotation and other changes. In feature matching, a special k-d tree method called Best Bin First search is used to get matching sets. Once the matching feature sets are obtained, the homogenous transformation matrix can be determined using the RANSAC method. Details are discussed in the following sections.

Features Finding

As the literature review explains, the SIFT features are used for image stitching. The reason of using the SIFT features is that the SIFT features are invariant to image scale and rotation. They can provide robust matching across the substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. Large numbers of distinctive features can be extracted from a typical image, which allows a single feature to be correctly matched with high probability from a database of many features.

There are four steps to get the SIFT features: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor.

Scale-Space Extrema Detection

The first step of detection is the Scale-Space Extrema detection. It searches over all scales of the image to get the potential points that are invariant to scale and rotation. The first stage of keypoint detection is to find the locations and scales that can be assigned under different views of the same object. Detecting the locations that are invariant to the scales can be achieved by searching for the stable features across all the possible scales using scale space.

The scale space representation of the image is usually parameterized by one parameter called the scale parameter such as the size of smoothing kernel, to represent the images as a one parameter family of smoothed images. The main reason for generating a scale-space representation is based on the observation that the real world objects are composed of different structures at different scales. When an unknown scene is analyzed, it is impossible to know a priori what scales are appropriate to describe the interesting structures from the image data. To combat this factor, descriptions at multiple scales will be able to capture the unknown scale variations. A scale space representation is considered to represent at all scales.

Koenderink (1984) and Lindeberg (1994) have shown that under a variety of reasonable assumptions, the only possible scale space kernel is the Gaussian function. The Gaussian filter is widely used because it is able to reduce noise and detail. It is a low pass filter but it doesn’t mean any low pass filter can be used to generate scale space. It is important that the smoothing filter does not introduce new spurious structures at coarse scales that don’t correspond to simplifications of corresponding

structures at finer scales. The Gaussian filter is presented to generate linear scale-space based on the foresaid requirement.

Therefore the scale-space of an image I(x,y) is defined as L(x,y,σ), which is produced from the convolution of the Gaussian function G(x,y,σ) and image I(x,y):

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3-1}$$

Where $*$ is the convolution operation in x and y, and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3-2}$$

In order to detect stable keypoint locations in scale-space, difference-of-Gaussian (DOG) function D(x,y,σ) is used to convolve with the image to construct the image pyramid.

$$\mathrm{D}(\mathrm{x}, \mathrm{y}, \sigma) = \big(\mathrm{G}(\mathrm{x}, \mathrm{y}, \mathrm{k}\sigma) - \mathrm{G}(\mathrm{x}, \mathrm{y}, \sigma)\big) * \mathrm{I}(\mathrm{x}, \mathrm{y}) = \mathrm{L}(\mathrm{x}, \mathrm{y}, \mathrm{k}\sigma) - \mathrm{L}(\mathrm{x}, \mathrm{y}, \sigma) \tag{3-3}$$

The reason to use DOG is that firstly it is an efficient function to compute to smooth the image, so is the image subtraction. Secondly it provides a close approximation to the scale normalized Laplacian of Gaussian (LOG) $\sigma^2\nabla^2\mathrm{G}$. From the heat diffusion equation 3-4, equation 3-5 can be derived as follows:

$$\sigma^2\nabla^2\mathrm{G} = \frac{\partial \mathrm{G}}{\partial \sigma} \approx \frac{\mathrm{G}(\mathrm{x},\mathrm{y},\mathrm{k}\sigma) - \mathrm{G}(\mathrm{x},\mathrm{y},\sigma)}{\mathrm{k}\sigma - \sigma} \tag{3-4}$$

$$\mathrm{G}(\mathrm{x}, \mathrm{y}, \mathrm{k}\sigma) - \mathrm{G}(\mathrm{x}, \mathrm{y}, \sigma) \approx (\mathrm{k} - 1)\sigma^2\nabla^2\mathrm{G} \tag{3-5}$$

In 2002, Mikolajczyk found that the maxima and minima of the LOG produce the most stable image features compared to other image functions such as gradient, Hessian, or the Harris corner function, through detailed experimental comparisons. The factor (k-1) is a constant over all scales so it doesn't affect the extrema location. To build the image pyramid D(x,y,σ), the initial image is convolved with Gaussians to produce one octave. The images in the same octave are separated by a constant factor

k in scale-space, shown on the left in Figure 3-1. Each octave is divided into an integral number s, of intervals, so $k=2^{\frac{1}{s}}$. It is necessary to produce s+3 images to cover the complete octave. Once a complete octave has been processed, the Gaussian image which has twice the initial value of σ is resampled and the process repeats. By this method, the accuracy of sampling relative to σ is the same as the starting octave but the computation is greatly reduced. Adjacent image scales are subtracted to produce the DOG image shown on the right in Figure 3-1.

Once the DOG is produced, the maxima and minima of $D(x,y,\sigma)$ can be detected by comparing each sample point with its eight neighbors in the current image and nine neighbors in the scale above and below (Figure 3-2). The point is selected when it is larger or smaller than all of its neighbors.

Keypoint Localization

After the keypoint candidates are found by comparison to its neighbors, it is necessary to find the accurate keypoint location, scale, and ratio of the principal curvatures. The ratio of the principal curvatures is for eliminating low contrast keypoints and edge responses.

In 2002, Brown and Lowe developed a method for fitting a 3D quadratic function to the local sample points to determine the interpolated location of the maximum. This approach uses the Taylor expansion of the scale-space function $D(x,y,\sigma)$. Let $X=(x,y,\sigma)$,

$$D(X) = D + \frac{\partial D^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial^2 D}{\partial X^2}X \tag{3-6}$$

Where D and its derivatives are evaluated at the sample point and X is the offset from the point. The location of the extremum $\hat{X}$ is determined by taking the derivative of this function with respect to X and setting it to zero:

$$\frac{\partial D}{\partial X} = \frac{\partial^2 D}{\partial X^2} X + \frac{\partial D}{\partial X} = 0 \tag{3-7}$$

$$\hat{X} = -\frac{\partial^2 D}{\partial X^2}^{-1} \frac{\partial D}{\partial X} \tag{3-8}$$

The Hessian and derivative of D are approximated by the differences of neighboring sample points. The function value at the extremum can be obtained by:

$$\mathrm{D}(\hat{\mathrm{X}}) = \mathrm{D} + \frac{1}{2} \frac{\partial \mathrm{D}^{\mathrm{T}}}{\partial \mathrm{X}} \hat{\mathrm{X}} \tag{3-9}$$

If the value of $|\mathrm{D}(\hat{\mathrm{X}})|$ is less than 0.03, the point will be discarded. Therefore the low contrast points can be eliminated by setting the contrast threshold. As the edge responses which were caused by the DOG function, the poorly defined peaks in DOG will have large principal curvatures across the edge but a small one in the perpendicular direction. The principal curvatures can be produced using the Hessian Matrix.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{3-10}$$

The eigenvalues of H are proportional to the principal curvatures of D. By calculating the ratio, computing the eigenvalues can be avoided. Let α be the eigenvalue with the larger magnitude and β be the smaller one, equations 3-11 and 3-12 are obtained as follows:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{3-11}$$

$$Det(H) = D_{xx} D_{yy} - {D_{xy}}^2 = \alpha\beta \tag{3-12}$$

Let γ be the ratio between the larger magnitude eigenvalue and the smaller one, so:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma} \quad (3\text{-}13)$$

which depends only on the ratio of the eigenvalues. The value goes to minimum when two eigenvalues are equal and it increases with γ. Therefore, in order to discard the edge responses, the threshold value for γ needs to be assigned, and it is implemented as:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(\gamma+1)^2}{\gamma} \quad . \quad (3\text{-}14)$$

Orientation Assignment

Once the keypoints that are invariant to scaling are found, the next step is to achieve invariance to image rotation. By assigning consistent orientation to each keypoint based on the local image properties, the keypoint descriptor can be represented relative to this orientation. Lowe (2004) showed that using the flowing approach to assign orientation gives the most stable results through experiments. The scale of the keypoint is used to select the Gaussian smoothed image L. For a certain scale, assign each image L(x,y) the gradient magnitude m(x,y) and orientation ϑ(x,y) as:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3\text{-}15)$$

$$\vartheta(x,y) = tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \quad . \quad (3\text{-}16)$$

By assigning the gradient orientations of sample points within a region around the keypoint, an orientation histogram is formed. It has 36 bins covering the 360 degree range of orientations, shown in Figure 3-3. Each sample to the histogram is added by weighting it with its gradient magnitude and a Gaussian weighted circular window with the scale parameter that is 1.5 times that of the scale of the keypoint.

Keypoint Descriptor

By doing the previous operations, the image location, scale, and orientation have been assigned to the keypoint. Therefore it is able to provide invariance to scale and orientation. The next step is to compute the keypoint descriptor for the highly distinctive image region so it can be invariant to the change in illumination or 3D viewpoint.

Edelman, Intrator, and Poggio (1997) proposed a representation based on a biological vision model, particularly in complex neurons in the visual cortex. They performed detailed experiments showing that matching gradients while allowing for position shift results in better classification. First the image gradient magnitudes and orientations are computed around the keypoint locations. Next the level of the Gaussian blur is chosen by the scale of the keypoint, shown on the left side of Figure 3-4. For orientation invariance, the coordinates of the keypoint and the gradient orientation are rotated relative to the keypoint orientation. A Gaussian weighting function is shown as the window on the left side of Figure 3-4. The reason of using the Gaussian window is to avoid sudden changes in the descriptor and to give less emphasis to the gradients on the edge as they are mostly affected by misregistration errors.

The keypoint descriptor is shown on the right side of Figure 3-4. It shows eight directions for each orientation histogram over 4×4 sample regions, with the length corresponding to the magnitude of the histogram entry. A gradient sample on the left can shift up to four sample positions while remaining the same histogram on the right, so that it achieves the objective of allowing large positional shifts.

The feature vector is modified to reduce the change of illumination. First, the vector is normalized to unit length. Image contrast change is the multiplication by a constant with each pixel value, the same as with the gradients, so it will be canceled by

vector normalization. A brightness change has no effect on the gradient values which are computed from the pixel differences. Therefore, the descriptor is invariant to affine changes in illumination. But there are also non-linear changes of illumination, which causes large change in magnitudes for some gradients but are less likely to affect the gradient orientations. By setting the threshold value in the unite feature vector, the influence of large gradient magnitudes can be reduced.

Implementation

There are four main steps of finding the SIFT features of the image. the feature structure is defined to have the following parameters: feature x coordinate, feature y coordinate, oxford type affine region parameter, scale of the Lowe style feature, orientation of the Lowe style feature, descriptor length, descriptor, feature type, and all purpose feature class. The result is shown in Figure 3-5. The length of the arrow shows the gradient magnitude of the feature point, and the orientation of the arrow is the orientation of the descriptor in the histogram. As shown in Figure 3-5, a 512×384 image is able to produce 305 SIFT features. These large numbers of keypoints extracted from the sample image are not only useful for the correct match for a keypoint from a large database of other image keypoints due to their distinctiveness, but also enables the robustness in extracting small objects among clusters. In the following section, the use of the SIFT features for matching will be discussed.

Features Matching

Through features finding, the set of the SIFT features is able to be extracted from a sample image. Given two images that are used for image stitching, two sets of the SIFT features can be detected. The next step is to find the matching keypoints from the first set of SIFT features to the other set. By identifying each keypoint’s nearest

neighbor in the database of keypoints from a training image, the best candidate match for the keypoint can be found. The nearest neighbor is defined as the keypoint with the minimum Euclidean distance for the invariant descriptor vector.

However there may be some incorrect matches in the database, so the features that are not well matched in the database should be discarded. Since some descriptors are more discriminative than others, it is not well performed to set the threshold on the distance to the closet neighbor. An effective way is to compare the distance of the closest neighbor to that of the second-closest neighbor. The reason is that the correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching. For false matches, there will possibly be a number of other matches within similar distances due to the high dimensionality of the feature space.

So the matching process is basically using the k-d tree to identify the nearest neighbors of keypoints. But the k-d tree provides no speedup over the search for high dimensional spaces as the keypoint descriptor has a high dimensional feature vector. So a modified k-d tree searching method called Best Bin First (BBF) search is used, which was first examined by Arya and Mount (1993). The details about BBF will be discussed in the next section. After indexing the set of the SIFT features for the second image using a BBF search, each SIFT feature from the first image is matched to the k-d tree index of the second image features.

Nearest Neighbor Indexing

Before introducing the BBF search algorithm, first the standard k-d tree is reviewed. A k-d tree is a data structure for storing a finite set of points from a k-dimensional space. It was examined in detail by J. Bentley (1980). It is a binary tree in

which every node is a k-dimensional point. The domain vector or the range vector component is the index for the node. Each non-leaf node is implicitly generating a splitting hyperplane that divides the space into two parts called subspaces. All the points in the left subspace are represented by the left subtree, and the points in the right subspace by the right subtree. The splitting hyperplane is a plan which passes through the domain vector and which is perpendicular to the left of the domain vector if and only if its *i*th component is less than the *i*th component of the domain vector. So for example, if a particular “x” axis is chosen, all the points in the subtree with a smaller “x” value than the node will be on the left subtree. An example is shown in Figure 3-6.

Beginning with a complete set of N points, the data space is split on the dimension *i* in which the data exhibits the greatest variance. A cut is made at the median value *m* of the data in that dimension, so that an equal number of points fall to one side or the other. An internal node is created to store *i* and *m*, and the process iterates with both halves of the data.

To look up the Neural Network (NN) to a query point *q*, a backtracking branch and bound search is used. First the tree is traversed to find the bin containing the query point. In the backtracking stage, whole branches of the tree can be pruned if the region of space they represent is further from the query point than the distance from *q* to the closest neighbor yet seen. However, the backtracking is still inefficient because the order of examining the leaf nodes is according to the tree structure, which depends only on the stored points and doesn’t take into account the position of the query point.

A better idea is to look in bins in order of increasing distance from the query point. The distance to a bin is defined to be the minimum distance between *q* and any point on

the bin boundary. It can be implemented using a priority queue. During the lookup, when a decision is made at an internal node to branch in one direction, an entry is added to the priority queue to hold information about the option not taken. It includes the current tree position and the distance of the query point from the node. After a leaf node has been examined, the top entry in the priority queue is removed and used to continue the search at the branch containing the next closest bin.

The Best Bin First (BBF) search provides a great improvement in the NN search for moderate dimensionality, making indexing in these regimes practical (Jeffrey S. Beis and David G. Lowe 2004).

Implementation

In order to build a k-d tree database from keypoints in an array, a k-d tree node is initialized with a set of features. Defined k-d node structures consist of the following parameters: partition key index, partition key value, leaf, features, number of features, k-d node left and k-d node right. The Partition key index is that along which descriptors have most variance. The Partition key value is the median of the descriptor values at the partition key index.

To expand the k-d tree, first the descriptor index at which the value with which to partition a k-d tree node's features is determined. Then partition the features at this specified k-d tree node to create its two children. When all records fall on same side of the partition, a leaf is found. When exploring a k-d tree from a given node to a leaf, branching decisions are made at each node based on the descriptor of a given feature. Each node examined but not explored is put into a priority queue to be explored later, keyed based on the distance from its partition key value to the given feature's descriptor.

Once the k-d tree is built, the image feature's approximate k nearest neighbors are found in a k-d tree using the BBF search. It returns k which is the number of neighbors found and stored in "nbrs". The "nbrs" is a pointer to an array in which to store pointers to neighbors in order to increase the descriptor distance. If k equals to 2, the squared Euclidian distance between the two feature descriptors is calculated. When the first distance is less than two times threshold on squared ratio, a match is found. A line is drawn between the matched keypoints in each image and then "nbrs" is released. An example of the implementation is shown in Figure 3-7.

Homography Transformation

Homography

Any two images of the same planar surface in space are related by a homography. A homography is an invertible transformation from the real projective plane to the projective plane that maps two straight lines together. It describes what happens to the perceived positions of observed objects when the point of view of the observer changes. From the previous section, the feature matches are extracted from adjacent images. These feature matches are used to get the homography of two images.

Assuming that the camera rotates about its optical center, the group of transformations which the images undergo is a special group of homographies. Each camera rotation is parameterized by the rotation vector $\theta = [\theta_1, \theta_2, \theta_3]$, and the focal length *f*. The homography can be expressed as $\widetilde{u_i} = H_{ij}\widetilde{u_j}$,

$$H_{ij} = K_i R_i R_j^T K_j^{-1} \tag{3-17}$$

where $\widetilde{u_i}$, $\widetilde{u_j}$ are the homogeneous image positions (3-dimentional) and i is the number of the captured images. The camera model is defined by

$$K_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3-18}$$

Using the exponential representation for rotations, the following equation is obtained:

$$R_i = e^{[\theta_i]\times}, \quad [\theta_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i1} & \theta_{i1} & 0 \end{bmatrix} \tag{3-19}$$

Ideally the image features would be invariant under the group of transformation. But for small changes in image position

$$u_i = u_{i0} + \frac{\partial u_i}{\partial u_j} |_{u_{i0}} \Delta u_j \tag{3-20}$$

or

$$\widetilde{u_\iota} = A_{ij} \widetilde{u_\jmath} \tag{3-21}$$

where

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{3-22}$$

is an affine transformation obtained by linearizing the homography about u_{i0} . It implies that each image patch undergoes an affine transformation, and justifies the use of the SIFT features which are only partially invariant to affine changes.

The homogeneous matrix can be calculated using two methods. The first method is when the camera calibration and rotation matrix are known. The second method is when one knows the correspondences between the two images. Since one can find the correspondences which are the SIFT feature matches between two images, the second method is used to find the homogeneous matrix.

Random Sample Consensus

As discussed in the previous section, the homography transformation matrix is a 3×3 homogeneous matrix with eight unknown variables, θ_i, θ_j(each of which has three components) and f_i and f_j. In order to get the transformation matrix, at least four pairs of correspondences should be given. Since many SIFT features are found, RANSAC (Fischler and Bolles in 1981) is used in this parameter estimation problem.

RANSAC is an iterative method to estimate parameters from a set of observed data, which contains outliers. Outliers are the data that do not fit in the model. Rather than using as much of the data as possible to obtain an initial solution and then attempting to eliminate the invalid data points, RANSAC uses as small of an initial data set as feasible and enlarges this set with consistent data when possible. It produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. RANSAC has two basic assumptions. Firstly, it assumes the data consists of inliers and outliers. The outliers can be subject to noise. It also assumes that, given a set of inliers, a procedure can estimate the parameters of a model that optimally explains or fits the data.

The input of RANSAC is a set of observed data which is the correspondences between the two images, a parameterized model which is the transformation matrix and some confidence parameters. RANSAC achieves its goal by iteratively selecting a random subset of the input observed data which are hypothetical inliers. Afterwards, the hypothesis is tested as shown in Table 3-1:

Table 3-1. Procedure of RANSAC Algorithm

Step	Action
1	A model is fitted to the hypothesis inliers
2	All other data are tested against the fitted model (if a point fits well to the estimated model, considered as a hypothetical inlier)
3	The estimated model is reasonably good if sufficiently many points have been classified as hypothetical inliers
4	The model is reestimated from all hypothetical inliers
5	The model is evaluated by estimating the error of the inliers relative to the model

Let parameter *k* be the number of iterations, *p* be the probability that the RANSAC algorithm in some iterations selects only inliers from the input data set from n points, *w* be the probability that an inlier is chosen each time a single point is selected. So that:

w = number of inliers/ number of points in the data

$1 - w^n$ is the probability that a bad model will be estimated from the point set.

$$1 - p = (1 - w^n)^k \tag{3-23}$$

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{3-24}$$

These parameters assume that the n data points are selected independently. However this does not happen when points are selected. Therefore, *k* should be the upper limit when the points are selected without replacement.

Implementation

From the previous step, all the correspondences between the two images are obtained. Using RANSAC, one can get the homography transformation matrix which can be implemented in OpenCV with the function cvFindHomography. Once the transformation matrix is known, the image can be transformed to the other image perspective. An example is shown in Figure 3-8. Here the transformation matrix is produced from the correspondences transformed from the second image to the first

one. Then by warping the second image using the cvWarpPerspective function, the second image is transformed to the perspective of the first image. Next, the first image is copied to the warped image of the second one. The final stitched image is produced and shown in Figure 3-10. Compared with Figure 3-9 which uses the SURF features, it shows that the SIFT features perform better than the SURF.

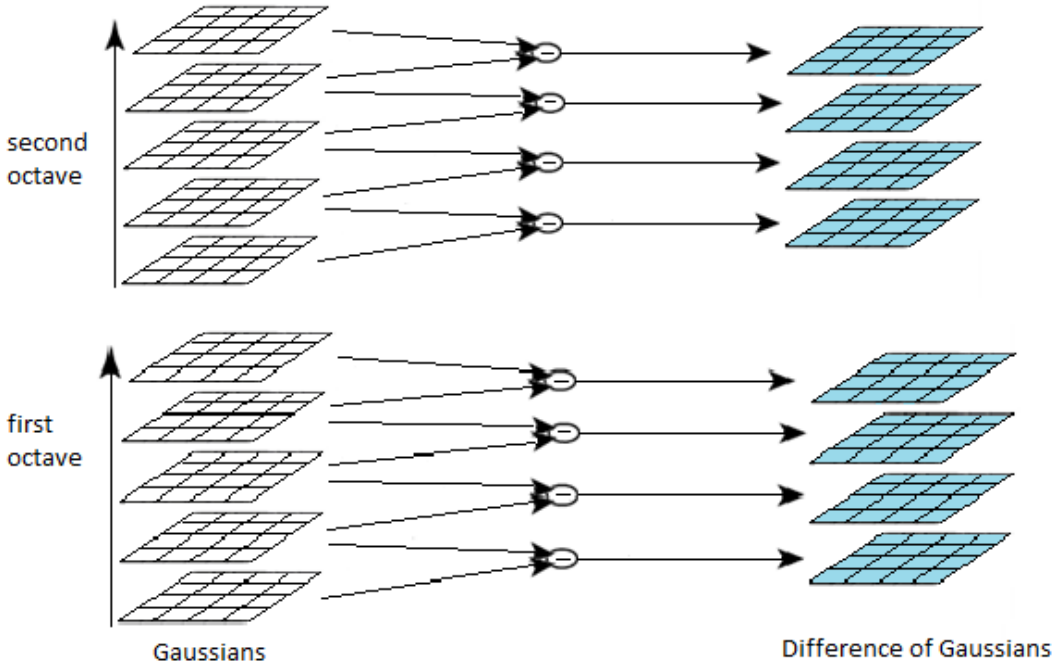


Figure 3-1. Building image pyramid and derivation of DOG

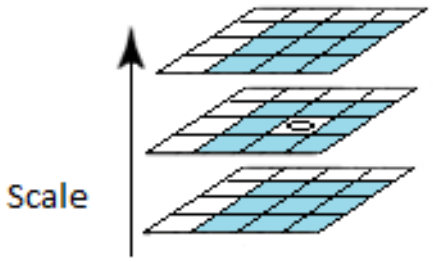


Figure 3-2. Detection of local minimum and maximum by comparing with the neighbors

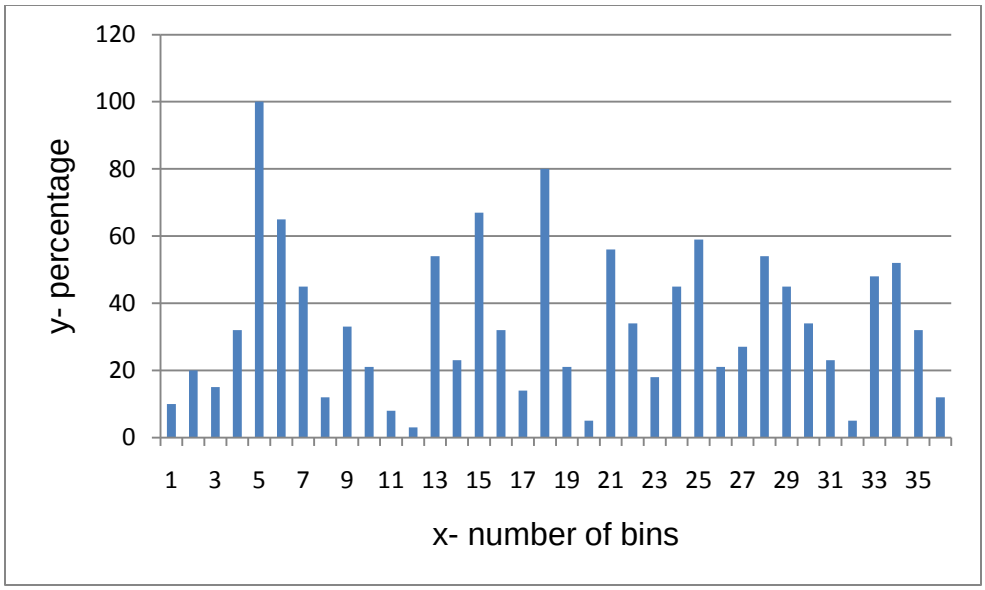


Figure 3-3. Gradient orientation histogram

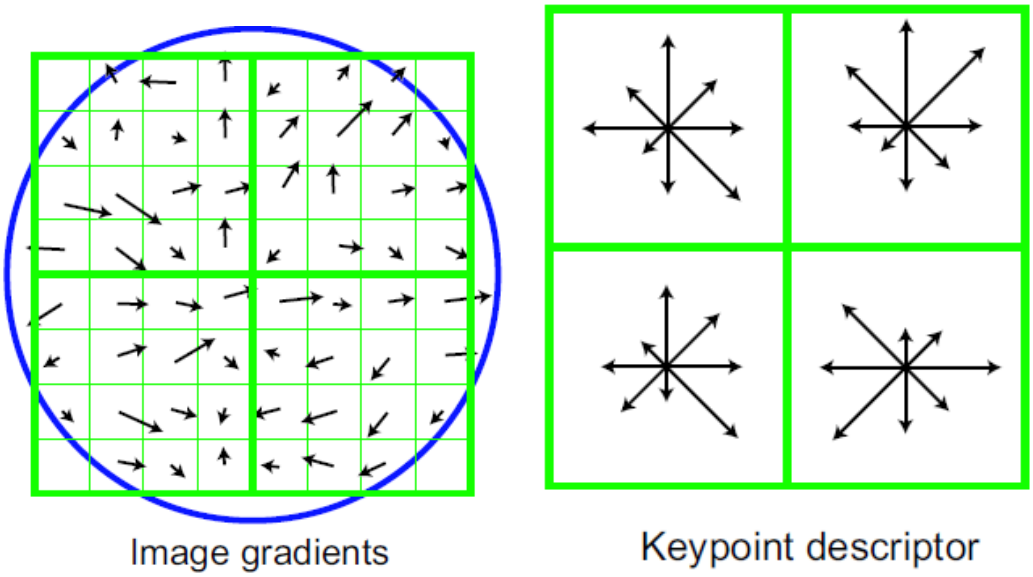


Figure 3-4. Producing the keypoint descriptor. Photo courtesy of David Lowe

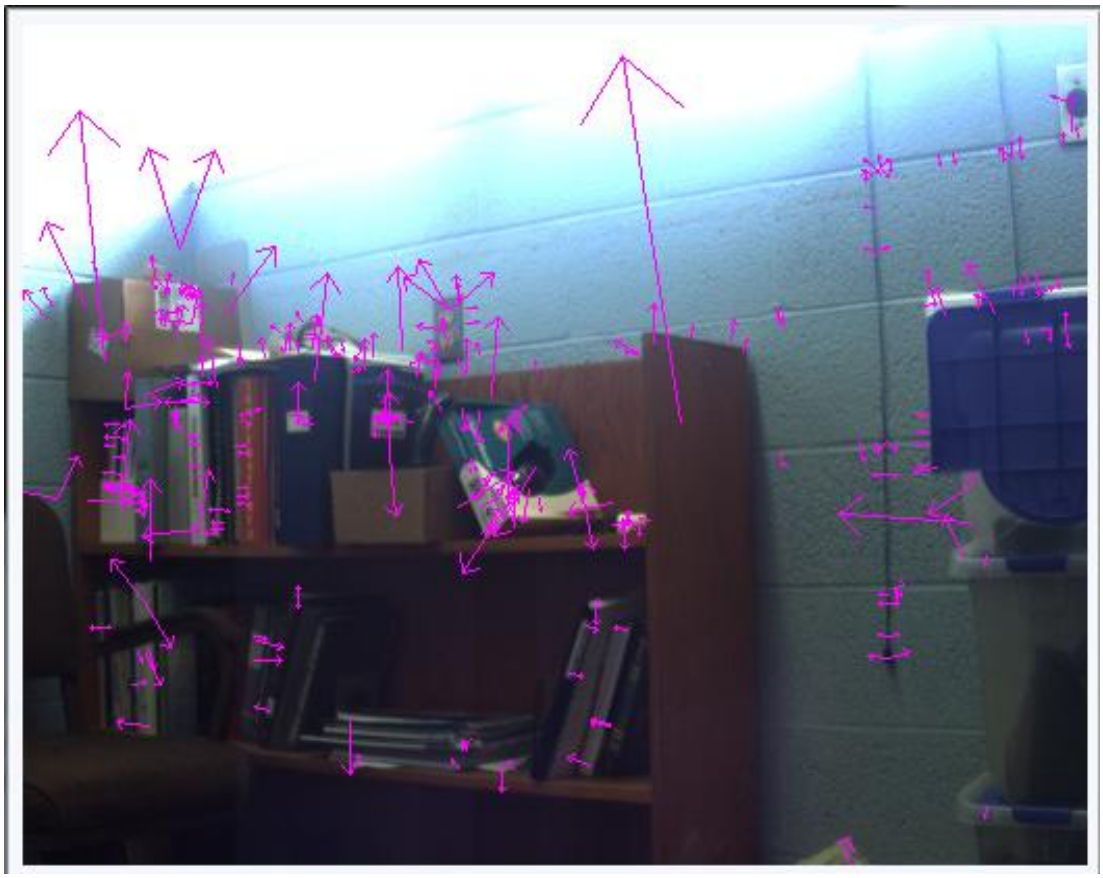


Figure 3-5. Implementation of finding the SIFT features. Photo taken by Qiang Niu

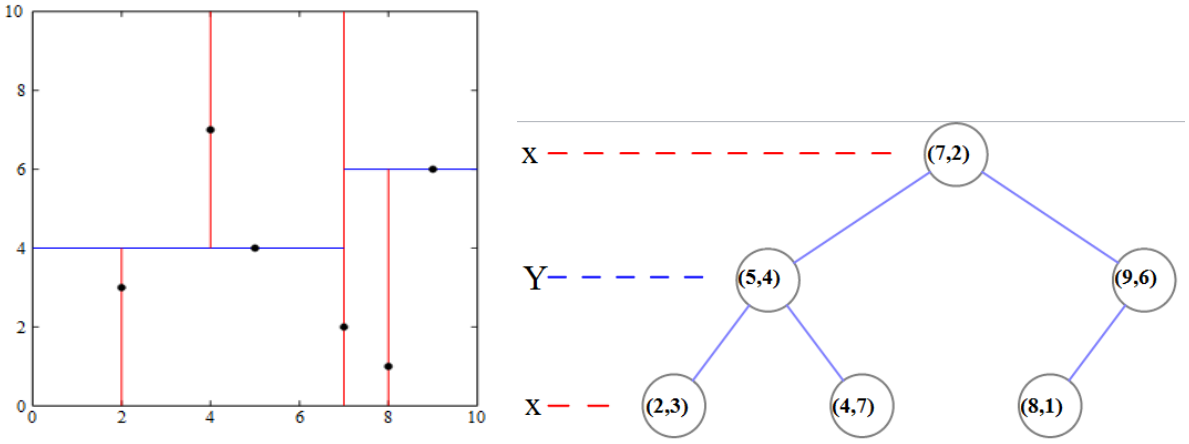
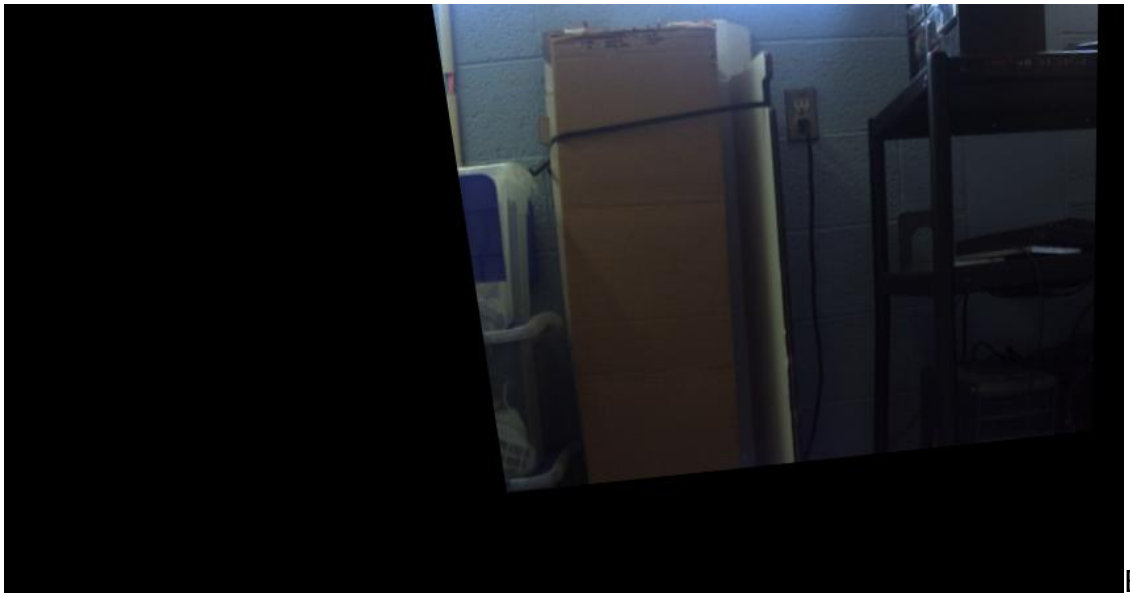


Figure 3-6. K-d tree decomposition

Figure 3-7. Features matching of two images using k-d tree. Photo taken by Qiang Niu

A



B

Figure 3-8 Homography transformation of image. A) Source image. B) Image after homography transform. Photo taken by Qiang Niu


A


B


C

Figure 3-9. Example of image stitching using SURF. A) First image to be stitched. B) Second image to be stitched. C) Stitched image using SURF. Photo taken by Qiang Niu


A


B


C

Figure 3-10. Implementation of image stitching using the SIFT. A) First image to be stitched. B) Second image to be stitched. C) Stitched image using SIFT. Photo taken by Qiang Niu

CHAPTER 4
PANORAMA STITCHING

Background

From the previous chapter, two images are able to be stitched together using the SIFT features. One use of that is to form a panorama which is a wide-angle view of a scene. It can be used on an unmanned vehicle when the vehicle is moving around an unknown scene and taking pictures through the camera mounted on it. Instead of storing all the images taken from the camera, a panorama image is stored after stitching. A panorama image is a high resolution image, so that more information can be extracted from less stored data.

To produce a panorama stitched image, the first step is to get a series of images from the camera mounted on the vehicle. The camera that was used for obtaining the images is an mvBlueFOX camera. A CameraDevice class is defined to hold information about the camera. The basic functions are defined in this class by the interface function called mvIMPACT _acquire, such as GetSerialID, GetImage, and so. After the camera configuration, images are obtained by setting the ‘key’ values equal to one, two, three and so until ‘key’ equals to ‘q’ which means quit getting images. The images are shown and saved for further processing and stitching.

However, once two images are stitched together, an artificial edge between two images appears. It is because of the colors and luminance levels of the two images are different sometimes. In order to construct a good panoramic image, one needs to reduce the color differences between the source images and smooth the color transition for the image sequence before stitching the image together. In order to reduce the color

differences, the overlap area between two images needs to be found first which is discussed in the next section.

Finding Overlap Area

It is a common task to find out the overlap area of coverage between two images, and there are several methods available to find it of varying time complexities. A method based on the Monte Carlo method in 1946 was used. This method takes less time than compared to the image matching methods through correlations when complete images are given. The method is discussed below.

If A and B are two 4-gons then three cases arise between A and B. First is that A and B intersect on two different edges with one point of B in A, second is that A and B intersect on the same edge with two points of B in A, third is that A and B intersect on the same edge with three points of B in A. The three cases are illustrated in Figure 4-1. When image B is entire within image A, the overlapping area is set to be image A. The first two cases are ruled out for finding the common points of intersection between them. In the last case, the number of intersection points may vary from one to eight depending on the orientation of both 4-gons.

The common points between two polygons are the intersection of edges of one polygon with edges of the other. In this method, the eight points of corner coordinates are known. Let (x_i, y_i), i=0, 1, 2, 3 be the corner coordinates of A, and (x_i', y_i'), i=0, 1, 2, 3 be the corner coordinates of B, ordered along a clock wise direction as shown in Figure 4-2. Let L1: $y = m_1 x + c_1$ be a line of (x_1, y_1) and (x_2, y_2), where

$m_1 = (y_2 - y_1)/(x_2 - x_1)$, $c_1 = (y_1 x_2 - y_2 x_1)/(x_2 - x_1)$, and L2: : $y = m_2 x + c_2$ be a line of (x_1', y_1') and (x_2', y_2'), where $m_2 = (y_2' - y_1')/(x_2' - x_1')$, $c_2 = (y_1' x_2' -$

$y_2'x_1')/(x_2' - x_1')$. The common point of intersection between L1 and L2 if they intersect (when $m_1 \neq m_2$) is given by (x_s, y_s) equals to $(\frac{c_2-c_1}{m_1-m_2}, \frac{m_1c_2-m_2c_1}{m_1-m_2})$. The parameters $\lambda_1 = (x_s - x_1)/(x_2 - x_1)$ and $\lambda_2 = (x_s - x_1')/(x_2' - x_1')$ are used to decide which solution is an intersecting point inside of both polygons.

Any point on the line segment joining (x_1, y_1) and (x_2, y_2) in parametric form is given by $(\lambda_1(x_2 - x_1) + x_1, \lambda_1(y_2 - y_1) + y_1)$, similarly any point on the line of (x_1', y_1') and (x_2', y_2') is $(\lambda_2(x_2' - x_1') + x_1', \lambda_2(y_2' - y_1') + y_1')$, where $0 \leq \lambda_1, \lambda_2 \leq 1$. At the point of intersection it follows $\lambda_1(x_2 - x_1) + x_1 = \lambda_2(x_2' - x_1') + x_1'$ and $\lambda_1(y_2 - y_1) + y_1 = \lambda_2(y_2' - y_1') + y_1'$. Solving for λ_1 and λ_2 gives:

$$\lambda_2 = \frac{(y_2-y_1)(x_1'-x_1)-(x_2-x_1)(y_1'-y_1)}{(y_2'-y_1')(x_2-x_1)-(x_2'-x_1')((y_2-y_1)} \quad (4\text{-}1)$$

$$\lambda_1 = \frac{(x_1'-x_1)+\lambda_2(x_2'-x_1')}{x_2-x_1} \quad (4\text{-}2)$$

When both λ_1 and λ_2 are between [0, 1], the common solution is $(\lambda_1(x_2 - x_1) + x_1, \lambda_1(y_2 - y_1) + y_1)$. The algorithm of finding the points of intersection is as follows, the overlap area of A and B are shown in Figure 4-2.

1. Read (x_i, y_i) and (x_i', y_i') for i = 0, 1, 2, 3.
2. For i = 0:3 (i-loop)
3. Calculate m_1 and c_1 of line segment A_iA_{i+1}
4. For j = 0:3 (j-loop)
5. Calculate m_2 and c_2 of line segment B_iB_{i+1}
6. If $m_1 = m_2$, go to step 4
7. Calculate $x_s, y_s, \lambda_1, \lambda_2$
8. If $0 \leq \lambda_1, \lambda_2 \leq 1$, then (x_s, y_s) is the point of intersection
9. End j-loop, end i-loop, and stop.

An example is implemented for finding the overlap area between image1 and image2, shown in Figure 4-3 A and B. Given two images, the second image needs to be transformed to the plane of the first image through the homography transformation

matrix discussed previously. Then the algorithm is used to find the overlap area. Figure 4-3, C is the overlap area of the first image, and D is the overlap area of the second one.

Color and Luminance Compensation

As stated previously, different view angles of the camera and changes in illumination in the scene will cause different exposure and colors in consecutive images. So it is necessary to perform color and luminance compensation for the source image to reduce the artifacts. The gamma correction for the luminance component of the source image is performed to avoid pixel saturation problems and obtain good transitions between source images. And the linear correction is performed for the color components of the source image, so that a good quality of color correction is obtained.

Gamma correction is a nonlinear operation used to code and decode luminance in images or videos. It is simply defined by the power-law expression as:

$$V_{out} = {V_{in}}^{\gamma}. \tag{4-3}$$

When γ is larger than 1, it is called encoding gamma, which makes the output image darker. When γ is smaller than 1, it is called decoding gamma, which makes the output image brighter.

The gamma correction is used to match colors of overlapping images areas in the RGB color space. From the previous section of finding the overlap area, the overlapping areas are obtained and the notations are illustrated in Figure 4-4.

Let $\mathrm{C_{i-1,i}(p)}$ be the color value of pixel p in image $\mathrm{S^{o}_{i-1,i}}$ and $\mathrm{C_{i,i-1}(p)}$ be the color value of corresponding pixel p in image $\mathrm{S^{o}_{i,i-1}}$, the following result is expected to be achieved through gamma correction:

$$\frac{1}{N_{i-1,i}}\sum_p C_{i-1,i}(p)^{\gamma_{i-1}} = \frac{1}{N_{i-1,i}}\sum_p C_{i,i-1}(p)^{\gamma_i} \quad (4\text{-}4)$$

where γ_{i-1} and γ_i are gamma correction coefficients for image *i*-1and *i*, $N_{i-1,i}$ is the number of pixels in the overlap area. To simplify the computation and make the algorithm robust to spatial alignment errors and speed up the algorithm, the means over the overlapping area is calculated then the gamma correction is applied, so the equation 4-4 becomes:

$$\left(\frac{1}{N_{i-1,i}}\sum_p C_{i-1,i}(p)\right)^{\gamma_{i-1}} = \left(\frac{1}{N_{i-1,i}}\sum_p C_{i,i-1}(p)\right)^{\gamma_i} . \quad (4\text{-}5)$$

Applying a logarithm to both sides of the equation 4-5 yields

$$B_{i,j} = \ln\left(\frac{1}{N_{i,j}}\sum_p C_{i,j}(p)\right) . \quad (4\text{-}6)$$

Equation 4-6 becomes

$$\gamma_{i-1}B_{i-1,i} = \gamma_i B_{i,i-1} . \quad (4\text{-}7)$$

In order to get the gamma correction coefficients γ_i(*i*= 1, 2,…, n), an error function is defined over all source images in the image sequence. The error function is the sum of the squared differences of gamma corrected colors between all corresponding pixels in the overlapping area. It is calculated as follows:

$$E = \frac{1}{2}\sum_{i=2}^{n}(\gamma_{i-1}\, B_{i-1,i} - \gamma_i B_{i,i-1})^2. \quad (4\text{-}8)$$

Since $\gamma_i = 0$ (i= 1, 2, …, n) is an optimal solution to this function, a prior term is added to keep the gamma correction coefficients close to unity since it makes no change when $\gamma = 1$. Also the normalization between color and the gamma correction coefficients errors should be considered. So the error function becomes:

$$E = \frac{1}{2}\left(\frac{\sum_{i=2}^{n}(\gamma_{i-1}B_{i-1,i}-\gamma_i B_{i,i-1})^2}{\sigma_N^2} + \sum_{i=1}^{n}\frac{(1-\gamma_i)^2}{\sigma_g^2}\right) . \quad (4\text{-}9)$$

To minimize this function to get γ_i (i=1, 2, …, n), the derivatives of the error function are set to zero, which gives:

$$\frac{\partial E_2}{\partial \gamma_1} = \left(\frac{B_{1,2}^2}{\sigma_N^2} + \frac{1}{\sigma_g^2}\right)\gamma_1 - \frac{B_{2,1}B_{1,2}}{\sigma_N^2}\gamma_2 - \frac{1}{\sigma_g^2} \tag{4-10}$$

$$\frac{\partial E_2}{\partial \gamma_n} = -\frac{B_{n-1,n}B_{n,n-1}}{\sigma_N^2}\gamma_{n-1} + \left(\frac{B_{n,n-1}^2}{\sigma_N^2} + \frac{1}{\sigma_g^2}\right)\gamma_n - \frac{1}{\sigma_g^2} \tag{4-11}$$

$$\frac{\partial E_2}{\partial \gamma_i} = -\frac{1}{\sigma_N^2}B_{i-1,i}B_{i,i-1}\gamma_{i-1}\left(\frac{B_{i,i-1}^2}{\sigma_N^2} + \frac{B_{i,i+1}^2}{\sigma_N^2} + \frac{1}{\sigma_g^2}\right)\gamma_i - \frac{B_{i+1,i}B_{i,i+1}\gamma_{i+1}}{\sigma_N^2} - \frac{1}{\sigma_g^2} \tag{4-12}$$

(i= 2, 3, …, n-1) .

By setting $\frac{\partial E_2}{\partial \gamma_i} = 0$, a tri-diagonal linear equations are obtained and can be solved by LU decomposition.

Once the optimal gamma correction coefficients have been obtained, color correction is performed for all source images in sequence. This method enables the compensation to have no color saturation and cumulative error problems. So before stitching the images together, the work flow of color correction is illustrated in Figure 4-5.

An example is implemented for the color compensation of the image shown in Figure 4-6. C is the stitched image without any color compensation; D is the result after processing color compensation.

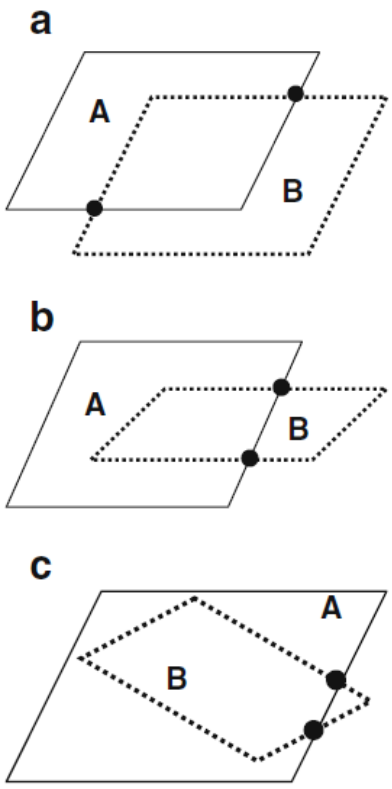


Figure 4-1. Three cases that image A and B are intersected

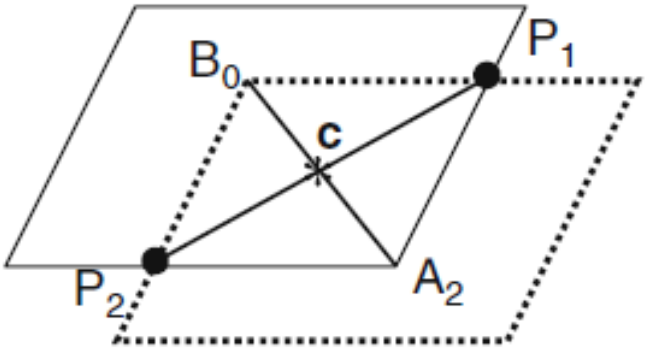
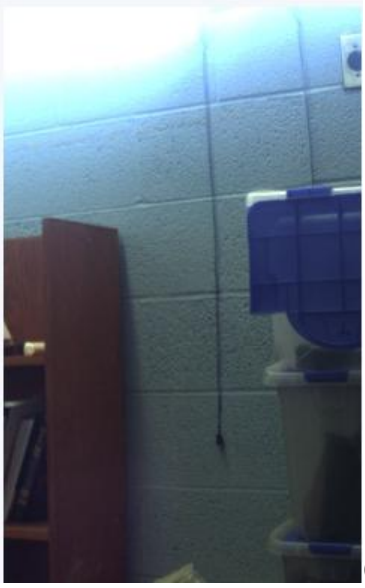


Figure 4-2. The overlap area with points of intersection


A


B


C

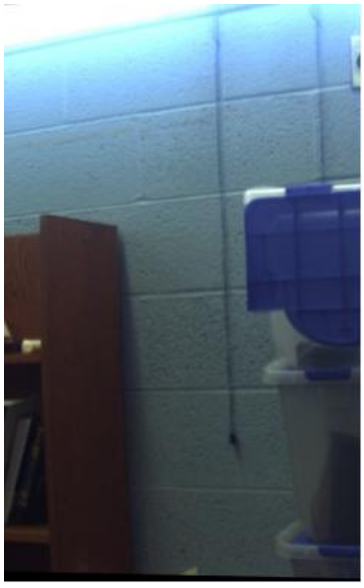

D

Figure 4-3. Finding overlap area of two images. A) Image1. B) Image2. C) Overlap area of image1. D) Overlap area of image2. Photo taken by Qiang Niu

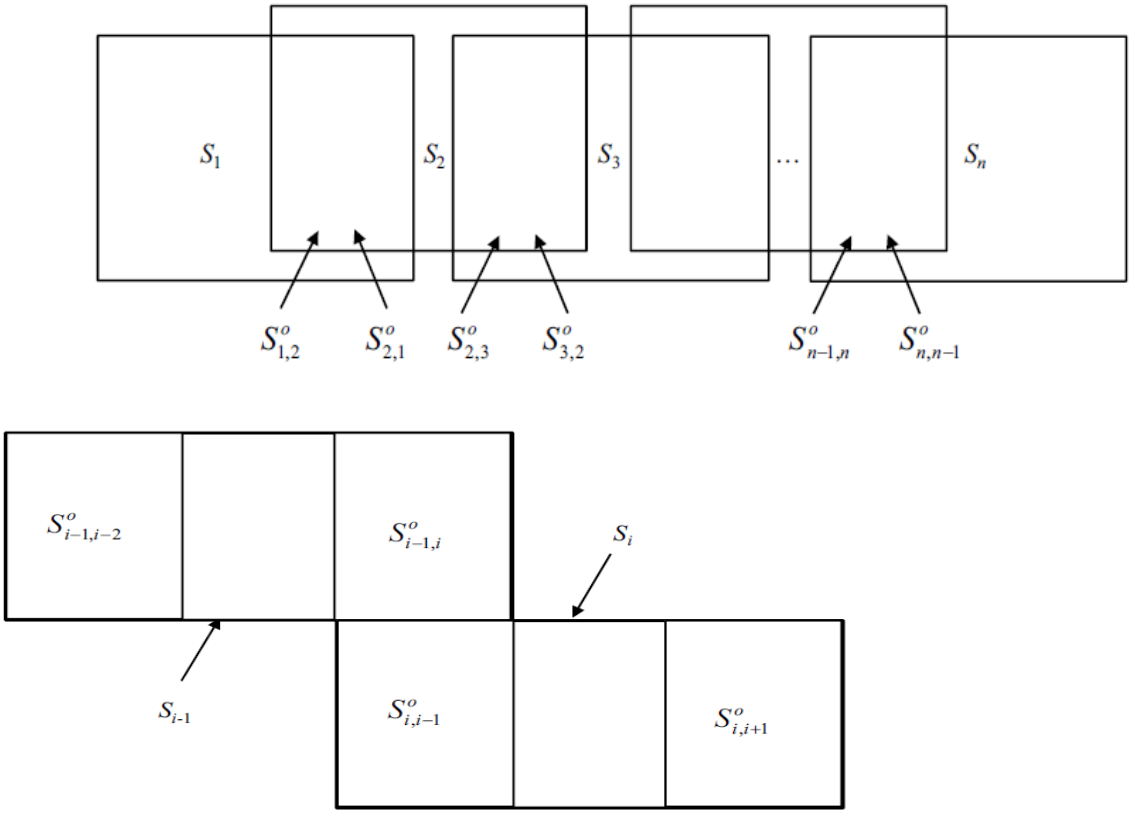


Figure 4-4. Overlapping source images notations. Photo courtesy of D. Sudheer Reddy

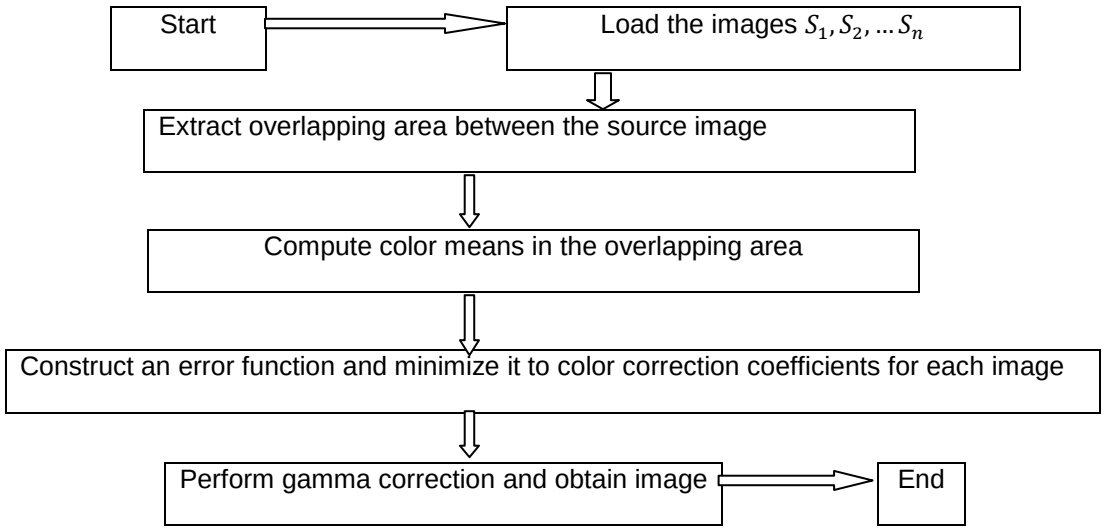
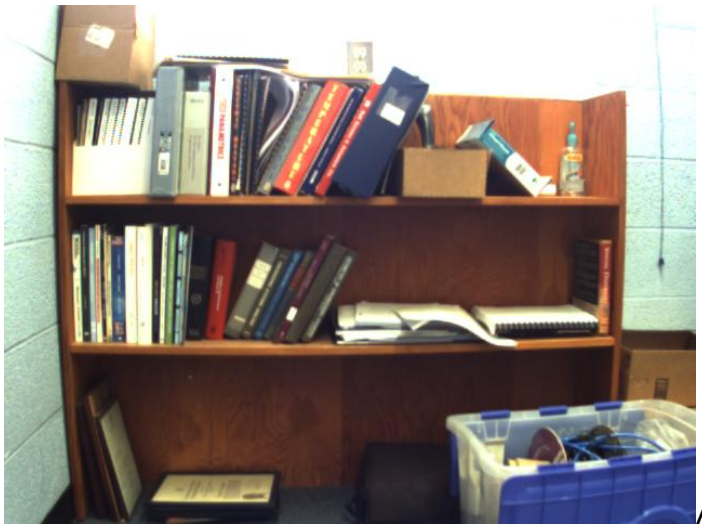
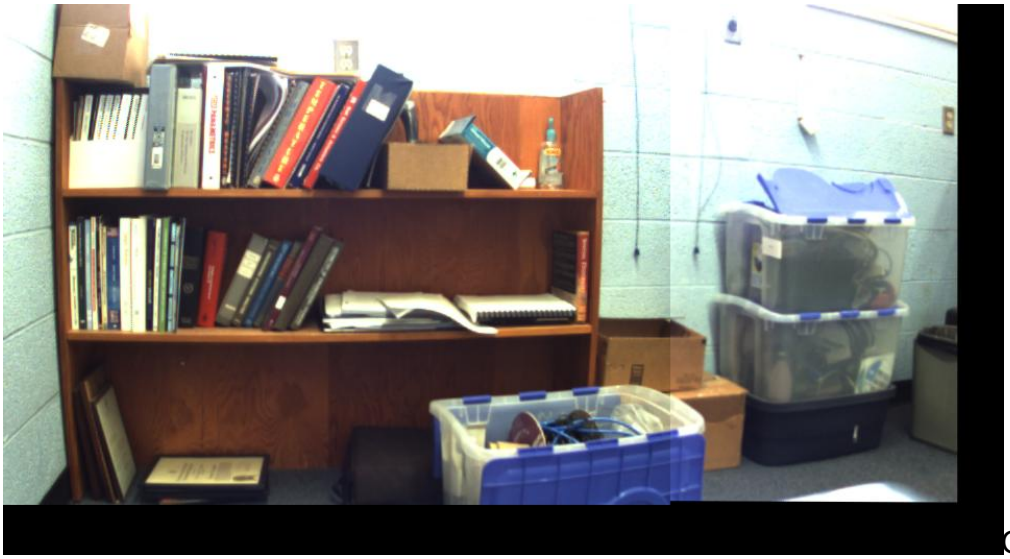


Figure 4-5. Work flow of the color correction



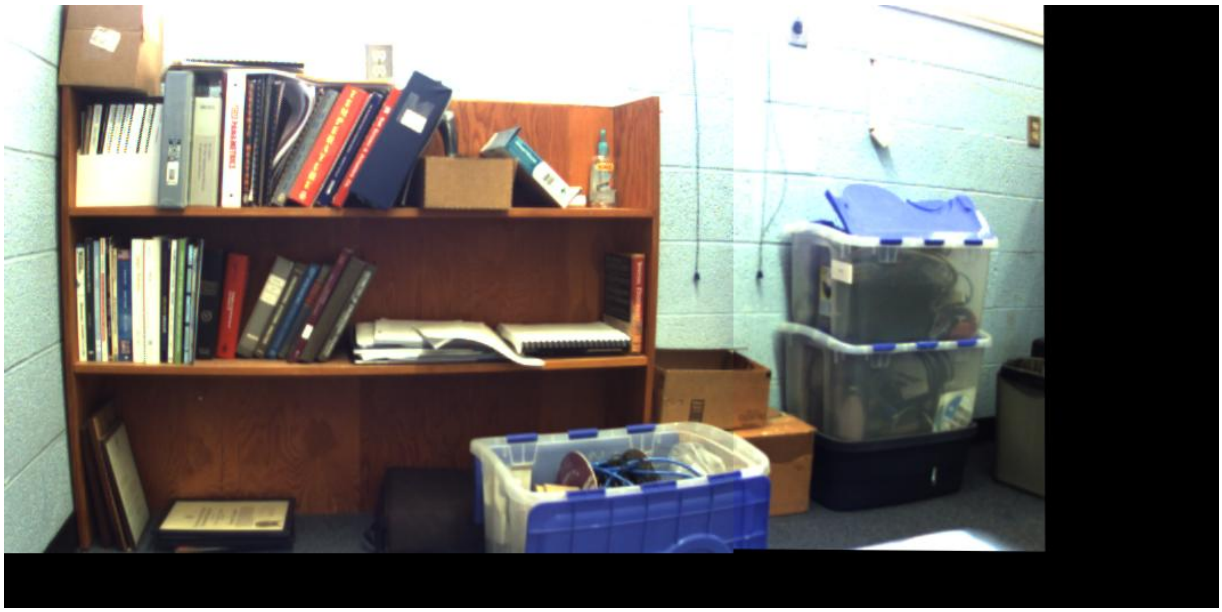
A

B



C

Figure 4-6. Implementation of Gamma correction. A) First image to be stitched. B) Second image to be stitched. C) The image stitching without gamma correction. D) The result after gamma correction. Photo taken by Qiang Niu



D

Figure 4-6. Continued

CHAPTER 5
OBJECT RECOGNITION

Background

Object recognition is widely used in the manufacturing industry such as for machine vision for registration, manipulation and so on. But current commercial object recognition systems are mainly based on template matching. Template matching is effective for some types of engineering environments, but when it comes to changes in illumination, rotation or scale, it becomes infeasible.

More and more progress has been made in developing real-world object recognition systems based on the use of invariant features. As the local features are complex, they are distinctive enough to determine the likely matches from a large database features. Also they are insensitive to clutter and occlusion. The difficulty of the object recognition problem is strongly due to the lack of success in finding such features. However from the previous discussion, the SIFT features are invariant to image scaling, translation, rotation, and partially invariant to illumination changes and affine transformations. So the SIFT features can be used efficiently as local image features for object recognition.

The process of object recognition using the SIFT is described as follows: First the SIFT features are detected from a local training image through filtering in scale space as discussed previously. Secondly, image keys are created which represent the blurred image gradients in different orientation planes and at different scales, as the input to a nearest neighbor indexing method to identify the matches. Thirdly, a verification of each match is achieved by finding a low residual least-squares solution for the unknown model parameters.

Object Recognition Using SIFT Features

From the previous discussion, the images are represented as a set of SIFT features. The approach that is used for object recognition is to match the features to individual training images. First the SIFT features extracted from the training image are stored as local data. Then images are captured from the camera and presented as sets of the SIFT features. The local data is matched to the sets of the SIFT features produced continuously. Once the matches are produced, the object is found. An example of finding a stop sign is shown in Figure 5-1.

There are some requirements on the training image. It is better taken on a uniform background, thus only few SIFT features will be produced from the uniform background. In this way, incorporation of spurious features will be minimized. From experiments (David Lowe 2004), the spurious features don't cause major problems other than some extra computation and memory usage. So it's still possible to use the training image without a uniform background as long as the majority of the image is the single object.

But there may be situations in which false matches are produced or a different number of matches is found from different images. So finding final matches should be achieved through verification.

The geometric verification is performed by using an affine transformation to verify the fit between local data and image features. Letting $[x,y]^T$ be a local data, $[u,v]^T$ be an image point, the affine transformation can be expressed as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (5\text{-}1)$$

Where $[\mathrm{t_x}, \mathrm{t_y}]^{\mathrm{T}}$ is the translation parameter and m_i are the affine parameters including rotation, scale, and stretch. In order to solve the parameters, the above equation 5-1 is rewritten as:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & & \dots & & \\ & & & \dots & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \tag{5-2}$$

There are six unknown parameters in equation 5-2, so at least three sets of matches are needed to solve the equation. The linear system can be expressed as:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{5-3}$$

Parameter x can be determined by:

$$\boldsymbol{x} = [\boldsymbol{A}^T \boldsymbol{A}]^{-1} \boldsymbol{A}^T \boldsymbol{b} \tag{5-4}$$

In this way, the sum of the squares of the distances from local data to image points will be minimized.

Outliers can now be removed by checking the verification with the affine parameter solution. In this test, each match is defined to agree within 15 degrees orientation, $\sqrt{2}$ changes in scale, and 0.2 times maximum in location. If fewer than 3 points remain after the test, the match will be rejected. If any outliers are discarded, the least-square solution is resolved within the remaining points. An example processed after verification is shown in Figure 5-2.

When there is a case of more than one image having the same object, the images should be able to be distinguished from the extent of recognizing the object. So another parameter, probability, is introduced. From the previous discussion, the number of the SIFT features is obtained from the training image. The number of the feature matches

between the object features from the training image and the image features from the captured image can also be obtained. The number of matches divided by the number of the SIFT features is used as the matching probability to measure the extent of object recognition, and the probability parameter is returned with the images.

As object recognition is an important and useful area of computer vision, there are varieties of approaches in feature based recognition, such as pose clustering using Hough Transform (Paul Hough in 1962). The SIFT features approach is only one of the feasible methods. So this chapter just briefly described another contribution of the SIFT features.

Figure 5-1. Attempt testing of object recognition. Training image is shown on the top; Image captured from camera is shown on the bottom. Photo taken by Qiang Niu

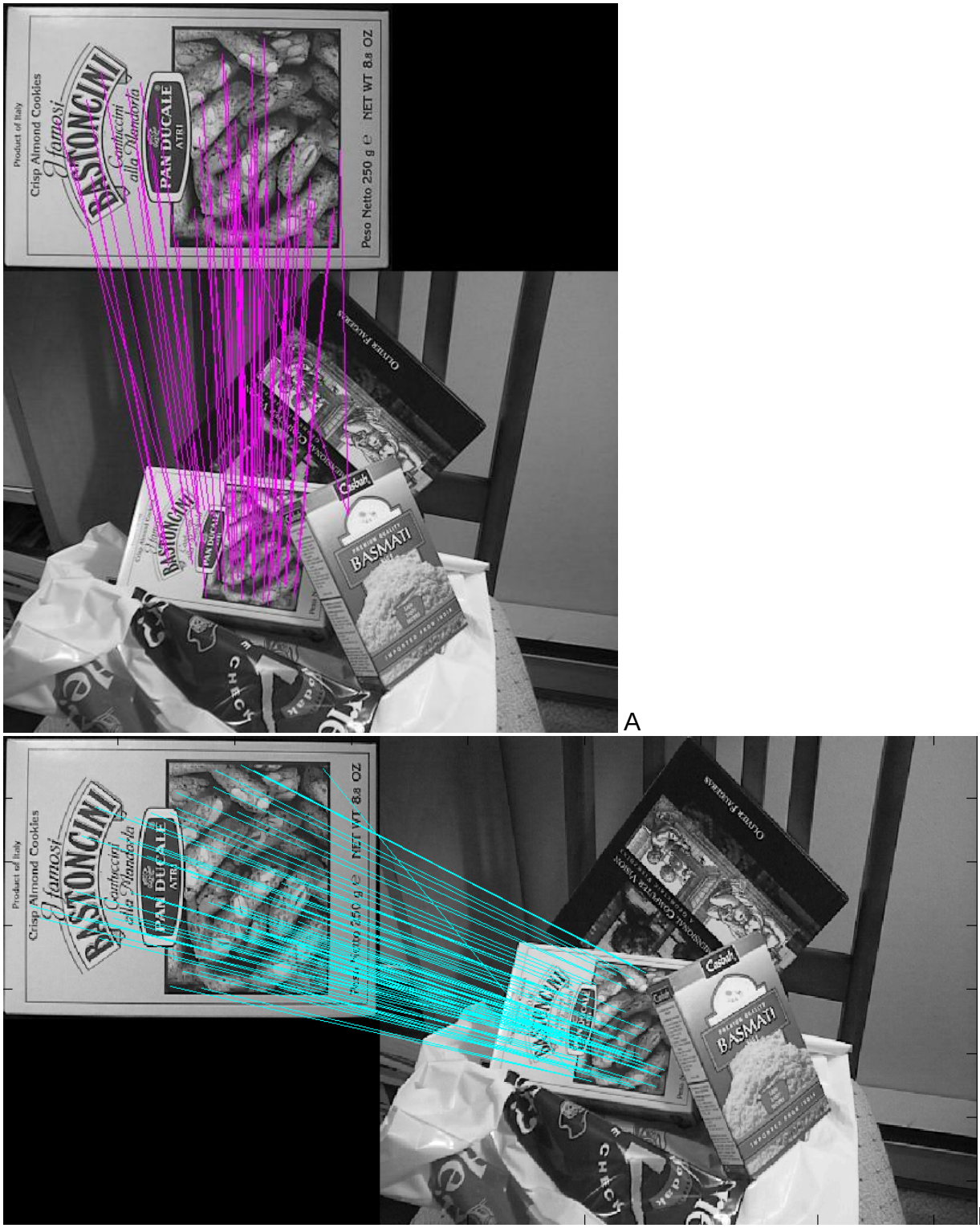


Figure 5-2. Comparison of object recognition using the SIFT with verification. A) The recognition before verification. B) The one after verification. Photo courtesy of David Lowe

CHAPTER 6
IMPLEMENTATIONS AND CONCLUSIONS

Results

The camera that was used for capturing images is the MvBlueFOX camera, which has the parameters are shown in Figure 6-1. Chapter 3 provides a technique to find the SIFT features. With the invariant features, further processing for image stitching can be achieved. Chapter 4 further discussed color correction for panorama stitching, which further improves the performance of image stitching in Chapter 3. The panorama stitching after color correction is shown in Figure 6-2. Chapter 5 briefly discussed another SIFT feature application which is feature based object recognition. It is faster and easier, it also requires less memory storage and computational time. Figure 6-3 shows how an object from a training image is recognized from sets of images captured from the camera.

Conclusions

Scale Invariant Feature Transform is the essential technique used in this thesis. It is an important invariant local feature for varieties of use. The idea of building an image pyramid by convolution with Gaussian function is essential for detection. In the image stitching area, large number of the SIFT features can be detected effectively. From the implementation, it only takes few seconds to extract the SIFT features. Feature descriptors are assigned for image matching, also for object recognition as it is able to achieve the invariance.

Stitching using the SIFT features does have better alignment than SURF as tested, but it is mostly only better aligned around the region of features that are detected. By just copying the untransformed image onto the transformed image, this

type of stitching cannot guarantee full alignment on the image edges as shown in Figure 6-2. For further improvement, an optimal seam method can be use to find the cutting curve in the overlapping area. This method requires one to minimize the difference between the pixels in overlapping area from both images. In addition, there are various algorithms to find the cutting curve such as Min-Cut (Kwatra et al in 2003). With this method, finding seamless edges may provides better alignment for image stitching.

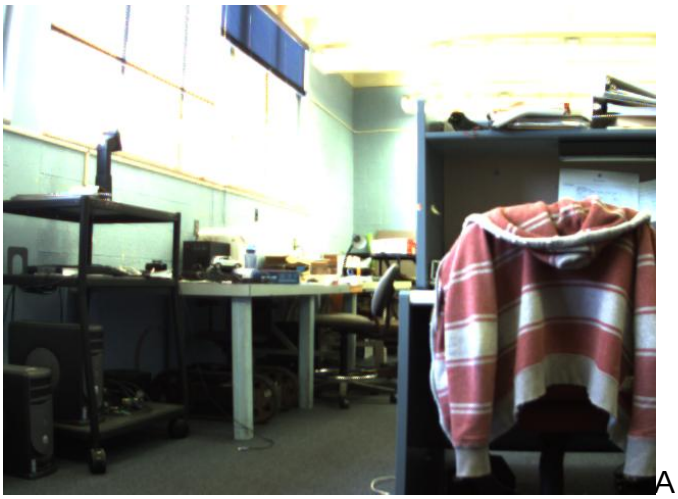
There is a constraint on the direction that the camera may move for capturing images. Since the order that images transform to the adjacent ones affects the order of stitching, there are different algorithms for the different directions that the camera moves. Clockwise and counter-clockwise directions were defined here to capture the images. For future development, there should be a way to combine them together.

Object recognition is another application of the SIFT features. The recognition from feature matches was briefly tested in this thesis. It works when the vehicle is driving around trying to find an object like a stop sign from the picture it captures. This method requires high feature based training image that contains mostly only the object. In this test, it was decided to use a black background image as the training image. There are also ways to find shapes that contain all the matched features from the training image within the captured image such as the Hough Transform, which would be developed in further object recognition problems. Also building a probability model helps the object recognition perform better.

In addition, there are also various applications of the SIFT features such as 3D scene modeling, video tracking, and so. It is a powerful invariant type of feature so it has the potential to make great contributions to many computer vision researches.

model name	-220
model variant	G,C
resolution of sensor's active area (width x height in [pixels])	640 x 480
maximum frame rate [Hz]	60
transfer type	full frame
sensor size	1/4"
pixel size (width x height in [µm])	5.6 x 5.6
readout type	progressive
integration time	44 µs - 10 s
ADC resolution	10 bit (up to 10 bit transmission)
overlap capabilities	yes
spectral sensitivity	G,C
USB type	USB 1.1 / USB 2.0
sensor manufacturer	Sony
sensor name	ICX098AL/BL

Figure 6-1. The parameters of the camera MvBlueFOX


A


B

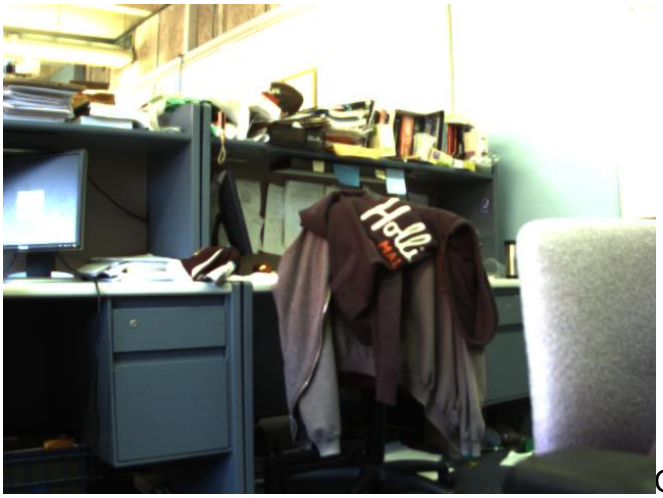

C

Figure 6-2. Panorama stitching. A) First image to be stitched. B) Second image to be stitched. C) Third image to be stitched. D) Window of the process. E) Stitched image. Photo taken by Qiang Niu

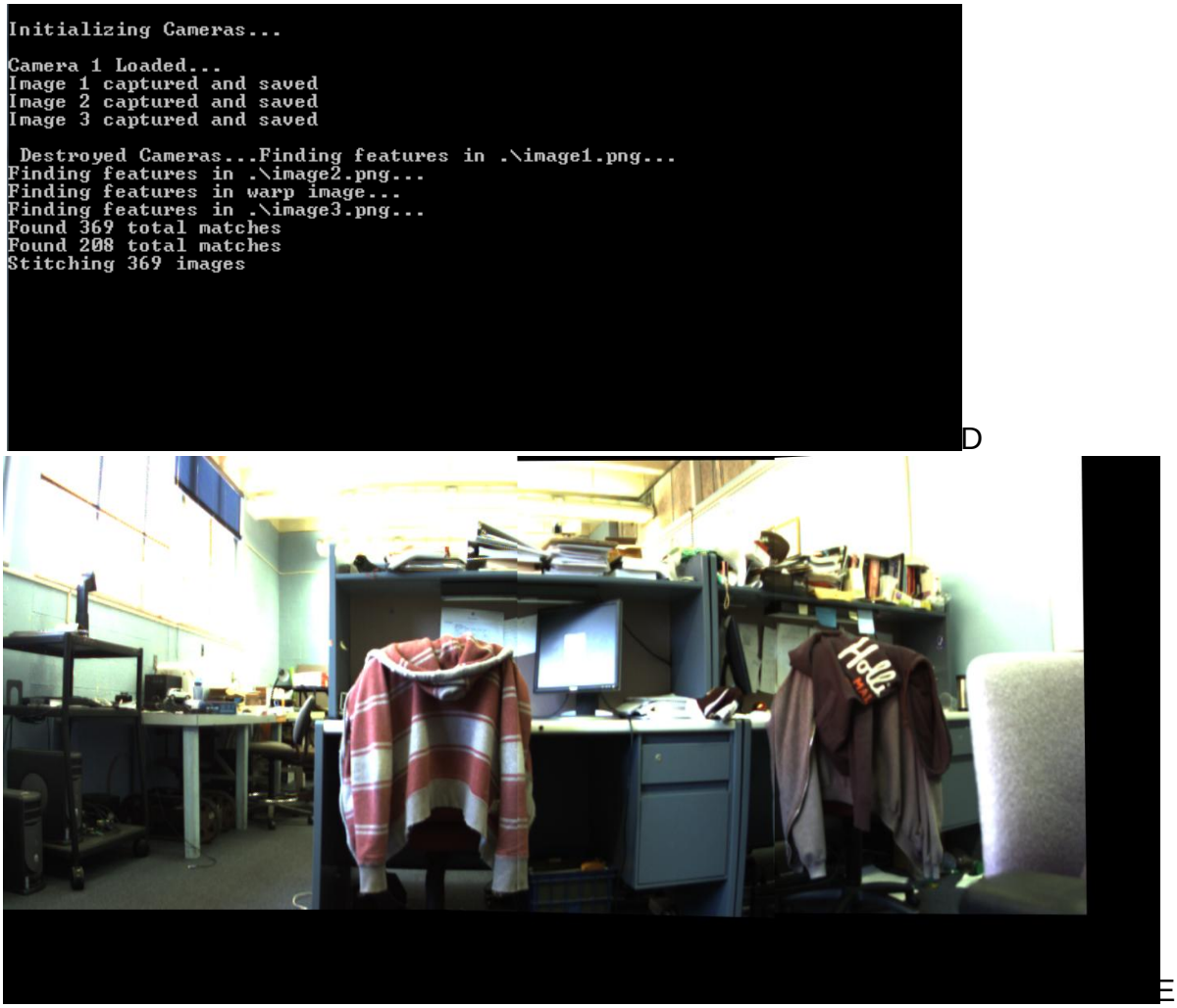
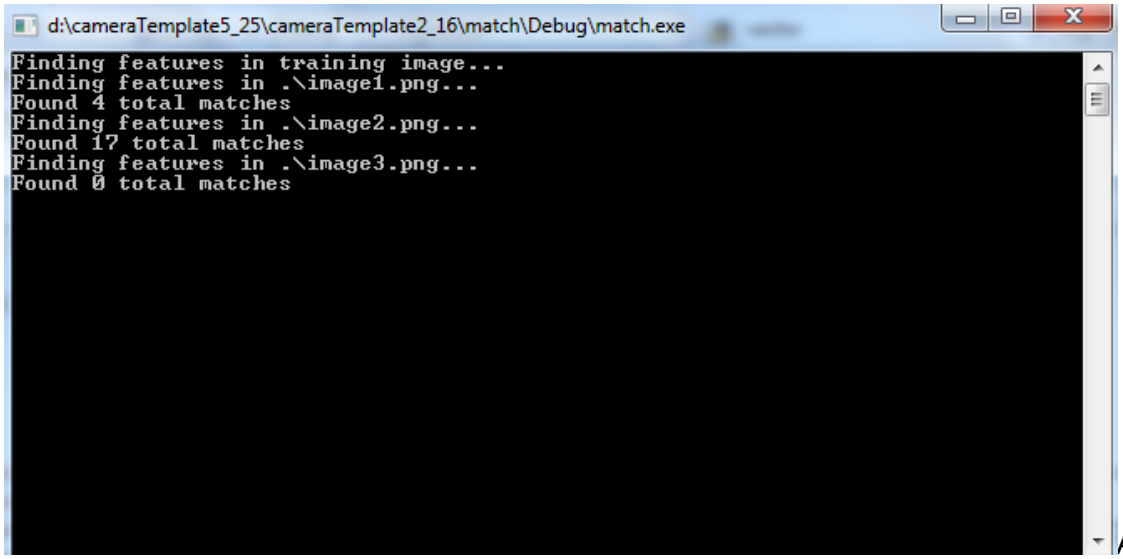


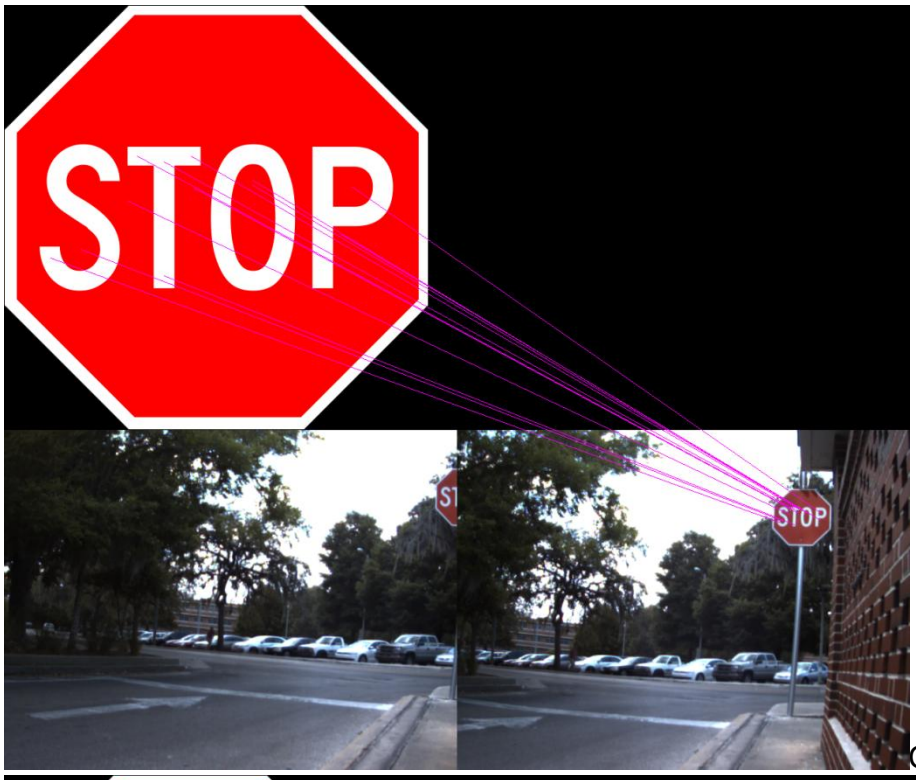
Figure 6-2. Continued



A

B

Figure 6-3. Object recognition using the SIFT features. A) Process of finding object in sets of images, returning the number of matches. B) Recognition between training image and the first captured image. C) Recognition between training image and the second captured image. D) Recognition between training image and the third captured image. Photo taken by Qiang Niu



C

D

Figure 6-3. Continued

LIST OF REFERENCES

Ayers, B. and Boutell, M. 2007. Home Interior classification using SIFT Keypoint Histograms. *Computer Vision and Pattern Recognition*, pp. 1-6.

Beis, J. S. and Lowe, D. G. 1997. Shape Indexing Using Approximate Nearest-Neighbor Search in High-Dimensional Spaces. *Computer Vision and Pattern Recognition, Vancouver*, B.C., Canada, pp. 1000-1006.

Brown, M. and Lowe, D. G. Automatic Panoramic Image Stitching Using Invariant Features. *International Journal of Computer Vision*, Vancouver, B.C., Canada, vol.74, pp. 59-73.

Fischler, M. A., and Bolles, R. C. 1981. Random Sample Consensus: A paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *SRI International, ACM*, vol. 24, pp. 381-395.

Li, Y., Wang Y., Huang, W. and Zhang, Z. 2008. Automatic Image Stitching Using SIFT. *International Conference on Language and Image Processing*, ICALIP, pp. 568-571.

Lowe, D. G. 2004. Distinctive Image Features from Scale-Invariant Keypoints. I*n the International Journal of Computer Vision*, Vancouver, B. C., Canada, vol. 60, pp. 91-110.

Lowe, D. G. 2001. Local Feature View Clustering for 3D Object Recognition. *Conference on Computer Vision and Pattern Recognition*, Vancouver, Canada, vol.1, pp. 682-688.

Lowe, D. G. 1999. Object Recognition from Local Scale-Invariant Features. *International Conference on Computer Vision*, vol.2, pp. 1150.

Reddy, D. S. 2010. Monte Carlo Method of Finding Overlap Area Between Two Image Scenes when Only Corner Coordinates are Known. *J Indian Society of Remote Sensing*, vol. 38, pp. 664-669.

Sadeghi, M. A., Hejrati, S. M. M. and Gheissari, N. 2008. Poisson Local Color Correction for Image Stitching. *Computer Vision Group*, pp. 275-282.

Stegmann, M. B. 2001. Image Warping. *Informatics and Mathematical Modeling.*

Sun, X., Pitsianis, N. P. and Bientinesi, P. Fast Computation of Local Correction Coefficients. *Advanced Signal Processing Algorithms*, vol.7074, pp. 1-8.

Szeliski, R. 2005. Image Alignment and Stitching. *Handbook of Mathematical Models in Computer Vision*, pp. 273- 292.

Wu, F. and Fang X.2007. An Improved RANSAC homography Algorithm for Feature Based Image Mosaic. *Conference on Signal Processing*, *Computer Geometry and Artificial Vision*, Athens, Greece, pp. 202-206.

Xiong, Y. and Pulli, K. 2010. Color and Luminace Compensation for Mobile Panorama Construction. *International Conference on ACM Multimedia*, Firenze, Italy, pp. 1547-1550.

Xiong, Y. and Pulli, K. 2010. Color Matching of Image Sequences with Combined Gamma and Linear Corrections. *International Conference on ACM Multimedia*, Firenze, Italy, pp. 261-270.

Xiong, Y. and Pulli, K. 2009. Color Correction for Mobile Panorama Imaging. *Proceedings of the First International Conference on the Internet Multimedia Computing and Service*, New York, 2009, pp. 219-226.

Xu, Y., Hu, K., Tian, Y. and Peng, F. 2008. Classification of Hyperspectral Imagery using SIFT for spectral matching. *Congress on Image and Signal Processing*, pp. 704-708.

Yianilos, P. N. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. *NEC Research Institute*, Princeton, New Jersey, pp. 311-321.

BIOGRAPHICAL SKETCH

Qiang Niu graduated from Nanjing University of Aeronautics and Astronautics in China with a Bachelor of Science degree in vehicle engineering. After graduating, Qiang Niu attended the University of Florida for her master's degree in mechanical engineering. At the same time, she pursued a minor in electrical and computer engineering at the University of Florida. Qiang Niu has worked on some projects involving vehicle dynamics and mechanical practice about automotives. She started doing research on computer vision, image processing and machine intelligence during her master's studies. Her research goal is develop computer vision system for autonomous vehicles and she will continue her career as an engineer in robotics field.