Object Distance Estimation and Unsupervised Recognition using Non-calibrated Camera

A SURF & Spectral Residue Based Implementation

朱恬骅
09300240004

1 Introduction

1.1 Background

As robotics prospers, it becomes increasingly possible for household robots to prosper. Though many kinds of sensors, like ultrasonic sensors, photosensitive devices, or touch sensors may help them getting information about the environment, They are highly limited in resolution. We cannot expect ultrasonic sensors to provide a refined 3-D scene map, for they focus on point-to-point sceneries.

However, stereo vision technology using two or more cameras may fail in situations where careful calibration is almost impossible to carry on. Here we are trying to find out a way for automatic camera calibration and efficient object positioning without prior selection on which object to track with.

In this paper, we will introduce two major approaches to handle the following questions:

1. Which object(s) is of most likelihood in being interested?
2. What's the position of this object?
3. How far is it from the camera approximately, comparing with the farthest background in the current view?

1.2 Related Works

Literature [6] surveyed algorithms ranging from simple edge-based algorithms to composite high-level approaches utilizing advanced pattern recognition methods. The survey traced back the history of face detection. Literature [10] describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. Cascade classifiers combination is also proposed, and is contributing to speeding up processing by quickly discarding background regions of images. Such method, called AdaBoost, is proved to be a modern and useful approach nowadays.

For object recognition, using PCA (Principal Component Analysis) and FLD (Fisher Linear Discriminant), used to be very popular. Other popular methods include CAMShift, which stands for "Continuously Adaptive Mean Shift."[2] It is a histogram-based algorithm used for fast face tracking. Given the initial position of a face, CAMShift uses its colour statistic histogram as a pattern, search in the adjacent area around last face position to match for a face.

This method is fast and appears on initial testing to be moderately accurate. It may be possible to improve accuracy by using a different colour representation. It inspired us to find only objects in their last positions for fast matching. We can therefore make use of information provided by nearby frames.

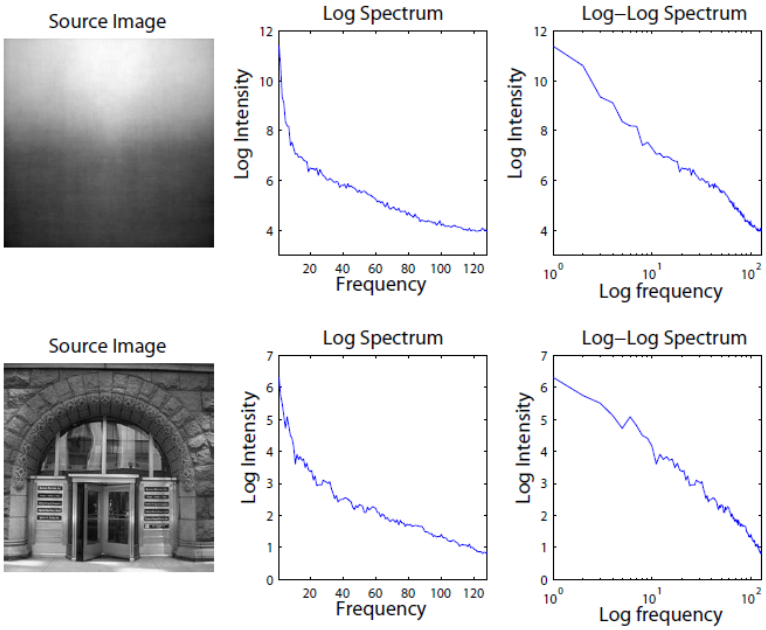

Figure 1: Spectrums as examplified in [7]

Scale Invariant Features Transformation (SIFT) was first proposed by Lowe in 1999 and accomplished in 2004 [8]. For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. [1] We will discuss this algorithm in detail later.

An evolved version of SIFT, Affine-SIFT (ASIFT), was proposed by Guoshen Yu and Jean-Michel Morel. [11] Their algorithm simulates a set of sample views of the initial images, obtainable by varying the two camera axis orientation parameters, namely the latitude and the longitude angles, which are not treated by the SIFT method. Then it applies the SIFT method itself to all images thus generated. Thus, ASIFT covers effectively all six parameters of the affine transform. And, against any prognosis, simulating a large enough set of sample views depending on the two camera orientation parameters is feasible with no dramatic computational load.

Region-Of-Interest (ROI) is also a hotspot of research in robotic artificial intelligence. Xiaodi Hou and Liping Zhang has proposed a quite easy way to find possible interesting objects.[7] They employed a model which is independent of features, categories, or other forms of prior knowledge of the objects. By analysing the log-spectrum of an input image, they extract the spectral residual of an image in spectral domain, and propose a fast method to construct the corresponding saliency map in spatial domain. We will also provide a perspective on this algorithm later.

2 Algorithms

2.1 ROI Determination Based on Spectral Residue

2.1.1 Spectral Residual Approach for Saliency Detection

Object detection aims at extracting an object from its background before recognition before performing recognitive feature analysis. Traditional models actually convert this problem to the detection of specific categories of objects. Most of them focus on summarizing the properties of target objects. However, general properties shared by various categories of objects are not likely to exist.

[1] http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Figure 2: An example of camera-based object extraction, with threshold set to 1e-2.

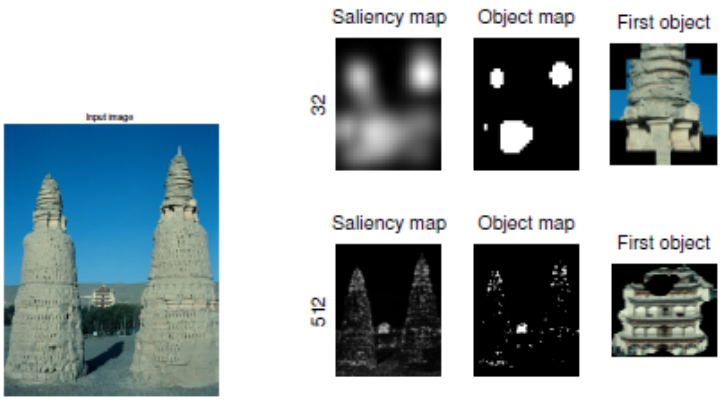

Figure 3: Results from inputs of different scales [7]

Xiaodi Hou and his group found that an image containing little information is prone to have a smooth spectrum, as is shown in figure 1, and another pictures, which may contain more details, are likely to have many rags in its spectrum. So we can simply subtract a smoothed value from the spectrum of an image to evaluate its information value. Hou's examples are shown in figure 3.

2.1.2 Multi-scale Object Detection

According to Hou, different scales of image may produce different object saliency, as is examplified in figure 3. We may conclude that object detection based on spectral residual saliency detection (SRSD for short) is scale variant. This may be a disadvantage in some cases, but in our scenario, this enables us to identify objects of different scales quickly and easily by applying SRSD for several times just adjusting its scale parameter. Generally speaking, we may find smaller objects in images of larger input size, and vice versa.

In our application, a 2-scale SRSD is performed to get objects of different scales. We may imply that objects in a further distance may became frequently detectable in larger sized inputs, and those objects found in smaller sized inputs can be identified as truly interesting. The larger scale is used to extract background information.

Figure 2 showed an implementation of 2-scale SRSD on camera. We here set an threshold to binarize the saliency map into mask image and apply it on the camera input for a more instinctive presentation. The largest two scales here are set to 256×192 and 64×48, respectively. We can see that it reserved the book title, the face, the shelf as distant object, and removed the ceiling and walls.

2.1.3 ROI Determination

After the SRSD phase, we have got a combined saliency map. For a given saliency map M, we choose at maxima k regions (denoted as k') of greatest saliency value as candidate objects, and get its rectangle bound $\{r_{i=1}^{k'}\}$. For each r_i, we simply set the ROI of the image to r_i. Obviously, r_i is much smaller than the whole grabbed image, and thus accelerates key-point finding in SIFT phase.

2.2 Key-point Matching and Recognition

2.2.1 SIFT Key-points

Scale Invariant Features Transformation (SIFT) was first proposed by Lowe in 1999 and accomplished in 2004 [8]. For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. [2]

Following are the major stages of computation used to generate the set of image features:

1. Scale-space extrema detection: it searches over all scales and image locations, implemented efficiently by using a difference-of-Gaussian function (as is shown in Figure 4 and 5) to identify potential interest points that are invariant to scale and orientation.
2. Key-point localization: At each candidate location, a detailed model is fit to determine location and scale.
3. Orientation assignment: One or more orientations are assigned to each key-point location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
4. Key-point descriptor: The local image gradients are measured at the selected scale in the region around each key-point. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

SIFT transforms image data into scale-invariant coordinates relative to local features. For image matching and recognition, SIFT features are first extracted from a set of reference images and stored

[2] http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

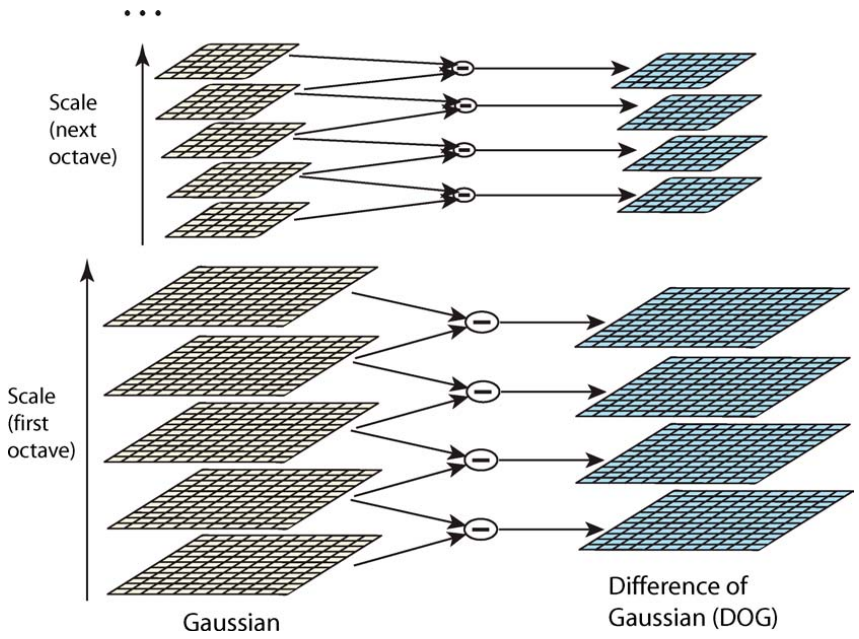

Figure 4: DoG operation

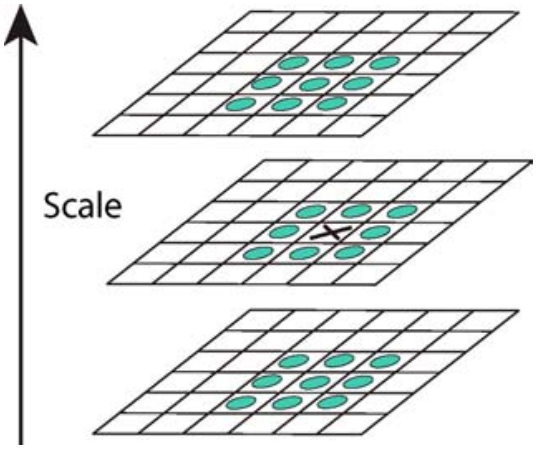

Figure 5: Scale-space extrema detection

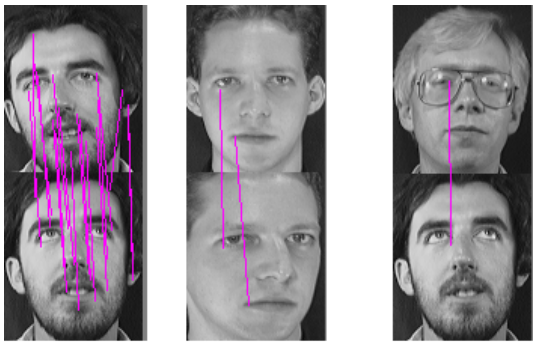

Figure 6: Tests on examples from the ORL dataset

in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors.

2.2.2 Key-point Matching Using FLANN

Fast Library for Approximate Nearest Neighbors (FLANN) is a library that contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. More information about FLANN can be found in [9].

In the implementation of SIFT algorithm by Rob Hess [3] [4], kd-tree is used to find k-NN of two SIFT features. The classical kd-tree algorithm is efficient in low dimensions, but in high dimensions the performance rapidly degrades. To obtain a speed-up over linear search, it becomes necessary to settle for an approximate nearest-neighbour. This improves the search speed at the cost of the algorithm not always returning the exact nearest neighbours.

When searching the trees, a single priority queue is maintained across all the randomized trees so that search can be ordered by increasing distance to each bin boundary. The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned.

2.2.3 Object Identifying Using SIFT

Due to these reasons, we can imagine using SIFT for object recognition. Here we take faces for example. Unfortunately, we find it insufficient to rely on SIFT. As is seen in figure 6, picture (a) showed a successful example, while (b) and (c) showed that SIFT may fail in distinguishing two faces. A test running on the whole ORL set indicated that pure SIFT method can reach 87.8% accuracy (49 errors in all 400 tests) when trained with 2 pictures per person and the least matches threshold set to zero, which is just a little better than face recognition using Eigenface.

2.2.4 Object Recognition: the Allocation Process

Having key-points matched, we have the object recognition phase. As is introduced in the previous sections, we have SIFT for extracting features from given images and can use them for recognition. Given an evaluation function $f(a, b)$ and a corresponding threshold θ, where a, b are two series of SIFs, we can perform face recognition using SIFT. The value of f increases as a and b differs more. The procedure, shown below, is on-line.

We just check similarities between the SIFs a of input and those of stored acquaintances, one by one. We may simply use the following equation to find the result:

$$\arg\min_{x \in L} f(a, x) \tag{1}$$

[3] http://web.engr.oregonstate.edu/ hess/

Figure 7: The two people are both labelled as yellow and same id

Such judgement will result in that an object occurs in more than one place in the image, which is impossible in reality (see figure 7). And also, it may cause chaos when multiple samples trained and recorded for one object.

So we may change our strategy to assigning each region with an object, rather than processing them independently. In the recognition phase, we have scored $f(a, x)$ for each $x \in L$, so that we can build a matrix $A = (f(a_i, l_j))$, where a_i is the i-th face and l_j is the j-th SURFs in database L. For those scores lower than threshold $theta$, we assign the id of l_j to face a_i if $l_j = \arg\min A_i$, otherwise we recognize it as unknown and leave it for acquainting phase.

In acquainting phase, we first find out the nearest face in the last frame, and compare it with that of current frame. If the score is lower than a looser bound $\hat{\theta} > \theta$, we regard the two regions as the same object, and then add the new sample SURFs to L. Otherwise, we will allocate a new id for it.

2.3 Speeding-up by Using SURF

Speeded-up Robust Features (SURF) was proposed by H. Bay et al. in 2008. [1] We here first introduce its basic idea and then have a look at its implementation in OpenCV.

2.3.1 A Concise Introduction

SURF provides also an approach towards scale- and rotation-invariant features like SIFT. SIFT costs much time due to its way of calculating the image pyramid. It repeats calculating Gaussian convolution on different scales. SURF improved this deficiency by relying on 'integral images' for image convolutions; and by reducing dimensions used. Typically, only 64 dimensions are used, and thus reduce the time cost in both feature computation and matching.

The entry of an integral image $I_\Sigma(x)$ at a location $x = (x, y)$ represents the sum of all pixels in the input image I of a rectangular region formed by the point x and the origin, $I_\Sigma(x) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i, j)$. [1] Actually, what they call 'integral image' is simply an easy application of local summation idea.

H. Bay et al. also adopted a novel detector, called Fast-Hessian Detector. They use box filters to simulate traditional Gaussian filters, called Laplacian of Gaussian Approximation (see figure 8).

Conventionally, we use Gaussian filters to re-sample a given image I repeatedly in order to get the image pyramid. The image size is varied and the Guassian filter is repeatedly applied to smooth subsequent layers. H. Bay et al. reversed this viewpoint by constructing a series of filters in different sizes, called the 'filter pyramid,' leaving the original image unchanged and varies only the filter size.

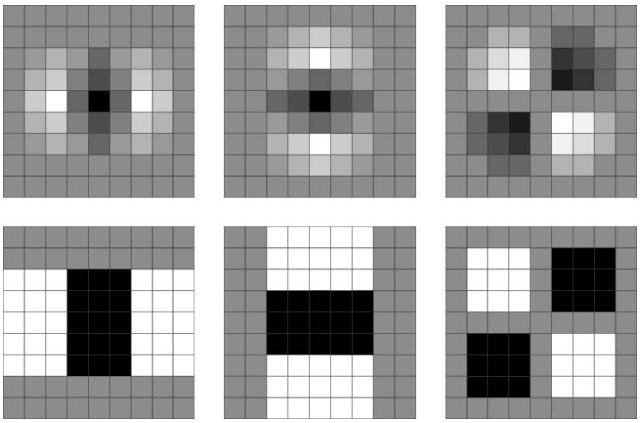

Figure 8: Laplacian of Gaussian Approximation. Top Row: The discretized and cropped second order Gaussian derivatives in the x, y and xy-directions. Bottom Row: Weighted Box filter approximations in the x, y and xy-directions. [1]

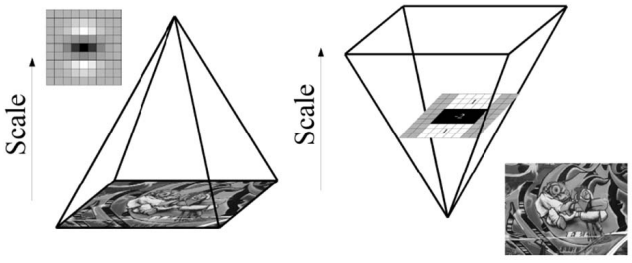

Figure 9: Filter Pyramid [1]

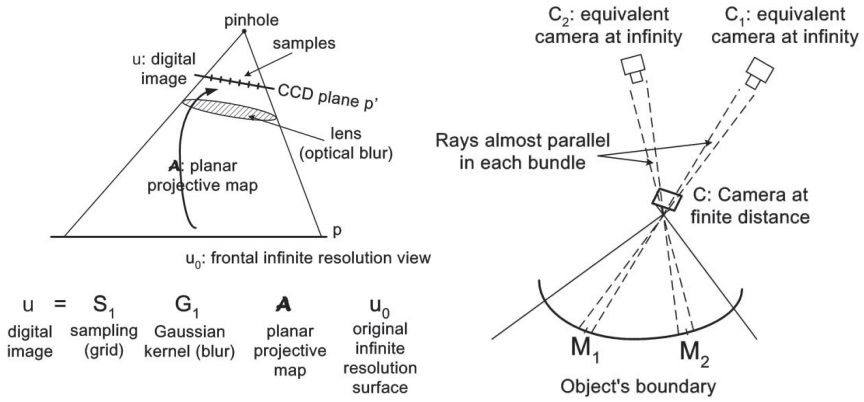

Figure 10: A camera at finite distance looking at a smooth object is equivalent to multiple local cameras at infinity. These cameras at infinity generate affine deformations. [11]

2.3.2 Using SURF instead of SIFT in Recognition Phase

OpenCV implemented SURF natively. It provides function **cvExtractSURF** to extract SURF from an image. By adopting SURF instead of SIFT, the same recognition process performed in the last subsection has accelerated to approximately 4 fps.

2.4 Camera Model and Distance Estimating

2.4.1 Camera Model

Here we adopt a model given in [11]. Figure 10 interprets the local behaviour of a camera as equivalence to multiple cameras at infinity. These cameras at infinity generate affine deformations.

We define a matrix $A = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ with a positive determinant (i.e., $a_1 a_4 - a_2 a_3 > 0$) as an affine map, and for a ground truth (ideal) image u_0, we have the following equation holds:

$$u = S_1 G_1 A T u_0 \tag{2}$$

where T is a plane translation due to the camera motion, G_1 is a Gaussian convolution modelling the optical blur, and S_1 is the standard sampling operator on a regular grid. Figure 10 illustrates the meaning of this model.

$$A = \lambda \begin{pmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{pmatrix} \begin{pmatrix} t & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix} \tag{3}$$

Figure 11 illustrates the equation 3. The image u is a flat physical object. The small parallelogram at the top right represents a camera looking at u. The angles ϕ and θ are, respectively, the camera optical axis longitude and latitude. A third angle ψ parametrizes the camera spin, and λ is a zoom parameter.

2.4.2 Distance Estimation without Prior Calibration

Popular camera calibration methods always involves manual operations. Chessboards or circle plates are required as an object to calibrate cameras. Though Zhengyou Zhang's method using chessboard images [12] is rather accurate, but this is at the cost of a person holding a chessboard, changing its position and rotation for many times. Once the relative position of the two cameras changed, the process has to be performed again.

We here will omit the introduction of Zhang's method and focus on our problem. According to the camera model we adopted in previous sections, we will ignore any kind of non-linear distortion of

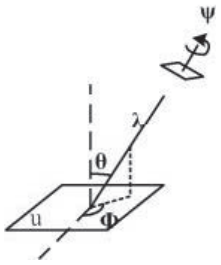

Figure 11: Parameter explanation [11]

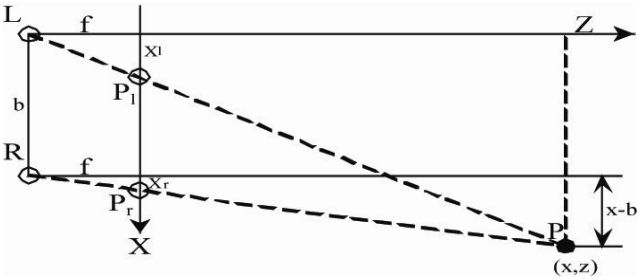

Figure 12: Distance estimation

the image, thus leaving only a parameter matrix A and a zoom parameter λ. In order to minimize non-linear distortion and difference between the optical structures of two cameras, we put two cameras in one plane and share a mutual foci f and zooming factor λ. Then, we can further simplify this model by removing λ from our consideration, for that we cannot actually acquire knowledge about actual size of any object, and what we are actually interested in is only relative positioning relationship between objects and their background. To further simplify our analysis, here we assume that two cameras are placed on one line (sharing a mutual X axis).

Consider the figure 12. L and R are two cameras, sharing their X axis, and their Y and Z axes paralleled respectively. Y axis is perpendicular to the X-Z plane, so we can omit it. The shadow centre for R has an offset of b. Target point P has a correspondent point P_l in the left image, and P_r in the right. We can induce that

$$z = \frac{fb}{x_l - x_r} = \frac{fb}{x_l - x_r} \tag{4}$$

and we have assumed that f is relevant to λ, which is ignored. And, if we take the left camera as a referential coordination, we will know that x_r in right camera coordination corresponds $x_r + b$ in the left. Thus leading to

$$z \propto b/(x_l - x_r - b) \tag{5}$$

For any given point shadowing on (x, y) in actual world (here we use left camera for reference), we have a correspond point in the right camera (maybe out of view range) $(x', y')^T = (x, y)^T A + (b, b')^T$, where b' denotes height difference from the left camera to the right one.

As is noted in previous sections, we use the most distant object as an estimation for background distance, because the vision difference in the left and right cameras is ideally $(b, b')^T$ when the background can be considered as infinitely distant, and closer objects for estimating A. The estimation process may repeat several times with an interval, for that camera situations may change in time.

Figure 13: Rotation rectification

2.4.3 Dealing with Rotation

The discussions above solved the problem when two cameras are in the same plane and have their X axices paralelled. In situations where the two cameras, though in the same plane, have rotations, we need to introduce a new parameter, called θ (actually $\Delta\psi$ as in figure 11), to represent the angle of the two X axices. This complicates our problem, because our cameras are not calibrated, so that the angle is hard to estimate. Using only SURF will not help us estimate θ.

We may assume that in some images, we have one or more lines in it, and we can estimate θ by comparing the lines in the left view with the right.

To detect lines, Hough tranformation is adopted.[3] Its idea is: represent a line with two parameters, obliquity α and the algebraic distance ρ from the origin. For each pair of parameters (α, ρ), we count the numbers of points on this line which are on the edge in this image (by using Canny operator). The bigger the number is, the more likely this image contains a line with this parameter (α, ρ). The line is determined by the following equation:

$$x\cos(\alpha - \pi/2) + y\sin(\alpha - \pi/2) = \rho. \tag{6}$$

Also, if we introduce a factor λ representing scaling, we can handle with situations where two cameras are in two paralelled planes. We can esitmate λ by calculating coordinate informatio of the two corresponding SURF key-points in the same distance. Detailed method is omitted here.

Ideally, we should pair the lines in two images, to get accurate θ. SURF key-points, again, will be involved in this process. However, it is not that easy to implement. In this project, I adopt the 'main direction' idea in SIFT. Use a histogram to count obliquities of the lines in one image, and choose the most frequent obliquity as the main direction of this image. Compare the two main directions, and we can define the difference as one estimation for θ.

Figure 14: Snapshots for the two video clips

Figure 15: Detection result for video clips

3 Experiments

3.1 SIFT and SRSD for People Detection in a Video Clip

In order to examine recognition on video, we use two short video clips recorded in class. One is a teacher's monologue, another is a side-frontal view of students. The snapshots is shown in figure 14.

The two video clips are all sampled at 29 fps, with height of 480 px and width of 720 px, lengthened 1' 42" and 20", respectively. They are converted from an Advanced Streaming Format (ASF) video of lecture to AVI format, and contain some codec errata, typically are scan lines and noises.

When facing many objects, it may recognize some but not all (due to face detection limitations), and tolerate rotation to some degree. If we just use the whole image for SURF, we may get better results when facing rotation problems, but we must sacrifice speed performance. The program is yet not so robust or stable, as illumination conditions differs and colour space is sensitive to it.

The processing speed is approximately 1.66 fps, and may get even slower as time grows, and so the number of features it acquainted keep increasing. To avoid this problem, we may set up some thresholds for the program, so that it may keep only features occurring frequently and/or recently and eliminate those seldom come around or out of sight for long. It still leaves large room for speed improvement.

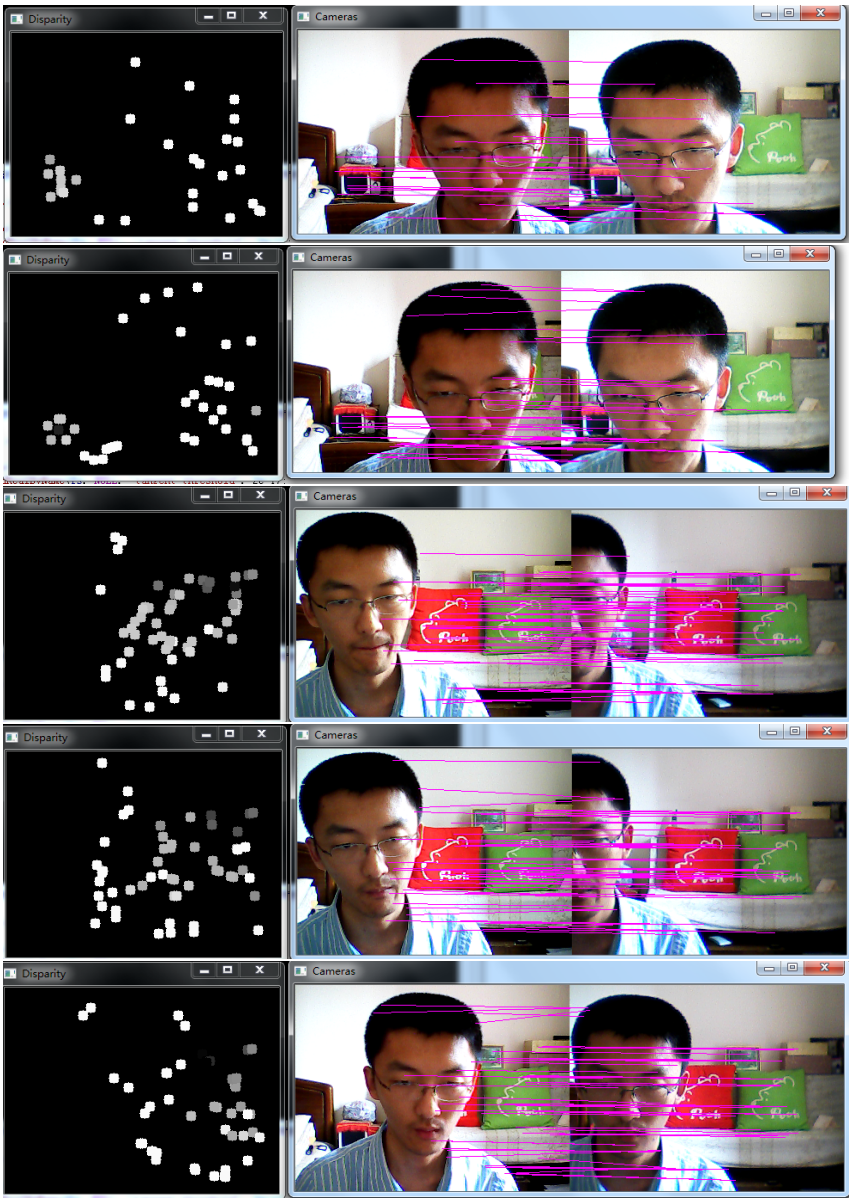

Figure 16: Key-point matching and distance estimation

3.2 Distance Estimation and its Comparison with SGBM

Figure 16 shows the result of my program. The magenta lines indicates key-point matching, and the grey-scale image showed in window 'Disparity' showed the distance estimation of the objects detected in the scene. We can see that these frames all showed that the person's head is nearest to the camera (white points in the 'disparity' window), and back cushions placing on the piano and the sound box are further (grey points in the 'disparity' window).

We also use distance estimation to refine image segmentation. We suppose that objects will not have an obvious change in z direction, so that we can divide key-point matches to several classes according to their distance estimation, and thus assigning them to different objects.

Comparison with SGBM Method in OpenCV OpenCV 2.3 provides a ready-to-wear implementation for SGBM (Semi Global Block Matching). [5] SGBM, proposed by H. Hirschmuller, finds

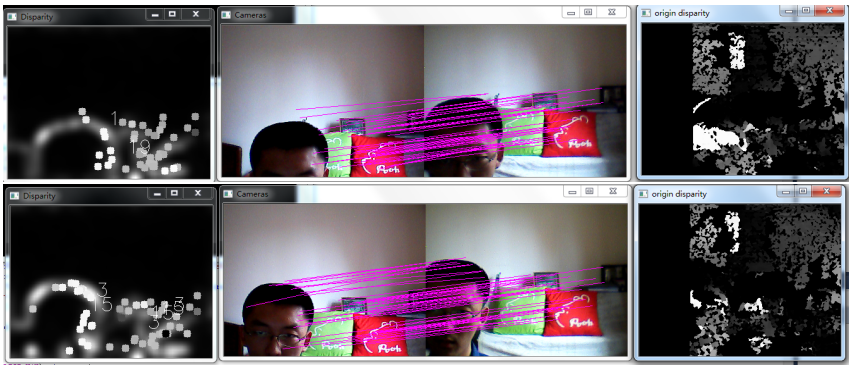

Figure 17: Comparison with SGBM Method in OpenCV

correspondent points in two views by pixel-wise matching based on Mutual Information and the approximation of a global smoothness constraint. The implementation typically performs at 5 fps.

This approach differs from SGBM in two respects. First, it considers only local information of interest in two views, precisely key-points only, while SGBM covers all information provided in the two images. Secondly, our approach combines object distance estimation (disparity calculation) with detection, recognition (as well as on-line learning) together. Though our approach may performs slower than SGBM, considering its coverage of object recognition relevant cost, our approach works better than SGBM + SURF combination.

Figure 17 showed a comparision of our approach with the SGBM method. For the reason that the wall (background) provides few details, our approach ignored it. But the shadow on the wall and the imaging difference of the two cameras makes SGBM to output a part of the wall as an object, almost in the same distance as the face.

4 Conclusion

In this project, I implemented a program using SURF for object distance estimation in stereo vision circumstances, and SRSD for object detection. The program satisfactorily produces distance distribution of key-points.

In this report, I have summarized the algorithms used in my project, and proposed a simple framework for distance estimation based on SRSD and SURF.

This approach has its limitations. Its recognition phase has much space for improvement.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:790–799, August 1995.

[3] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[4] Rob Hess. An open-source siftlibrary. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1493–1496, New York, NY, USA, 2010. ACM.

[5] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 807–814, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Erik Hjelmas and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[7] Xiaodi Hou and Liqing Zhang. Saliency detection: A spectral residual approach. In *CVPR*, 2007.

[8] D Lowe. Distinct image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91 – 110, 2004.

[9] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[10] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, (2):137–154, 2004.

[11] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line*, 2011.

[12] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, November 2000.