Vietnam National University, Hanoi
University of Engineering and Technology

Computer Vision - Mid-term

Extract Features using SIFT
Applications and Comparisons

Group 6	21020170	Tran Ngoc Bach
	21020692	Nguyen The Khang
	21021602	Le Trung Kien
	22029037	Ngo Quang Tang
Professor	Assoc. Prof. Dr. Le Thanh Ha	
Teacher Assistant	BA. Le Cong Thuong	
Class	INT3412E_21 - Computer Vision	
Report	Report Phase 2	**Date Report:** 26/10/2023

1 Abstract

Humans identify objects, people, and images through memory and understanding. The more times you see something, the easier it is for you to recollect it. Indeed, our human brains don't store the entire object's details, they define that object by several features called key points. Also, every time an image pops up in your mind, it relates that item or image to a bunch of related features. What if I told you we could teach a machine to do the same using a technique as human brain?
This paper presents a method for extracting those distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. It is based on an algorithm named: Scale Invariant Feature Transform **SIFT**, announced by D.Lowe (2004)[1], which proposed how a computer could extract key points, and compute its descriptors (could be referred to the "signature" of key points).
Thanks to the advance of keypoint descriptors, we can solve various problems in computer vision, image processing, ... This paper also describes how a panorama picture is created by multiple images using SIFT. After extracting features in those image, then we try to match them by KNN algorithm, by computing a homography matrix to project those images on the base plane then warp them into a panorama image.

2 Introduction

The key points are features in the source image that are invariant to image scaling and rotation, partially invariant to changes in illumination and 3D camera viewpoint, and could be compatible with reliable matching. They are well localized in both the spatial and frequency domains, and robust to disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. Moreover, the features are highly distinctive to perform a correct match with high probability against a large database of features, providing a basis for object and scene recognition.
SIFT's algorithm to extract keypoints follows four main cascade stages:

1. Scale-space extrema detection.
   The target is to identify locations and scales that can be repeatably assigned under differing views of the same object while ignoring any noise. So we consider the image in a space

using a continuous function of scale known as scale space. Locate keypoints by detecting the extrema in neighboring pixels.

2. Keypoint localization
   Remove low contrast keypoints and edge responses to remain the keypoints that are highly stable invariance to scale, orientation, and noise.

3. Orientation assignment
   To determine the keypoint orientation, a gradient orientation histogram is computed in the neighborhood of the keypoint, to achieve invariance in image rotation.

4. Keypoint descriptor
   Compute a descriptor for the local image region about each keypoint that is highly distinctive and invariant as possible to variations such as changes in viewpoint and illumination.

3 Extracting interested features by SIFT

3.1 Scale-space extrema detection

3.1.1 Scale-space construction

Early investigations into scale selection were performed by Lindeberg (1993; 1998b)[2], who first proposed using extrema in the Laplacian of Gaussian (LoG) function as interest point locations. Based on this work, D. Lowe proposed computing a set of sub-octave Differences of Gaussian (DoG) functions to give a rough approximation of the LoG solution. He also constructed a scale space extrema using the DoG function to detect stable keypoint locations.

$$\frac{\partial L}{\partial \sigma} = \sigma \nabla^2 L \approx \frac{L(x, y, k_i\sigma) - L(x, y, k_{i-1}\sigma)}{k_i - k_{i-1}} = \frac{D_f(x, y, \sigma_i)}{k_i - k_{i-1}} \tag{1}$$

Key points are extracted at different scales and blur levels and all subsequent computations $L(x, y, \sigma_i)$ are performed within the scale space framework. This will make the descriptors invariant to image scaling and small changes in perspective.

$$L(x, y, \sigma_i) = G(x, y, \sigma_i) * I(x, y) \tag{2}$$

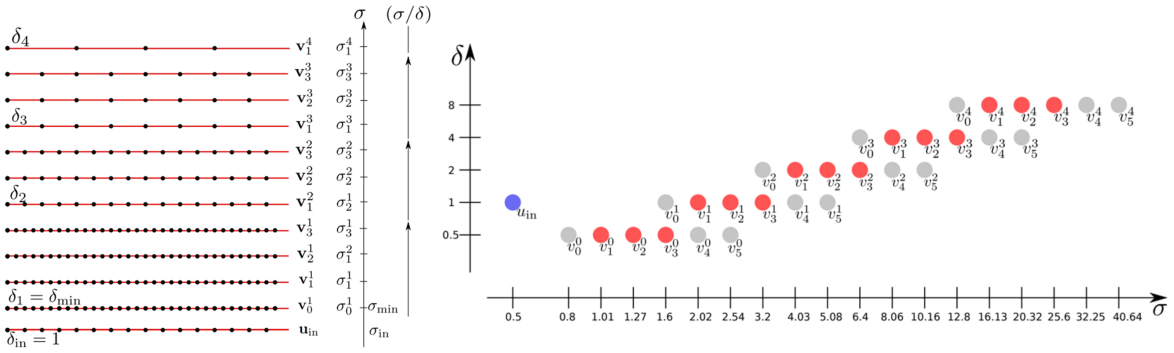


Figure 1: Pyramid of scale-space construction

Before constructing scale-space, if we pre-smooth the image before extrema detection, we are effectively discarding the highest spatial frequencies. Expanding input image by doubling the size of the input image using linear interpolation prior to building the first level of pyramid. Each image in the pyramid set of images should be specified a different blur variance σ_0, k, and $\sigma_i = \sigma_0 * k_i$ where $\sigma_0 = 0.8, k_i = (\sqrt[3]{2})^i$.

With the Gaussian blur functions $G(x, y, \sigma_i) = \frac{1}{2\pi\sigma_i^2}\exp(-\frac{x^2+y^2}{2\sigma_i^2})$, $\sigma_i = k_i\sigma_0$ are multiples of σ, and $D(x, y, \sigma_i)$ is our approximate DoG solutions.

$$\begin{aligned} D(x, y, \sigma_i) &= \big[G(x, y, \sigma_i) - G(x, y, \sigma_{i-1})\big] * I(x, y) \\ &= L(x, y, \sigma_i) - L(x, y, \sigma_{i-1}) \\ &= L(x, y, k_i\sigma) - L(x, y, k_{i-1}\sigma) \\ &\approx (k_i - k_{i-1})\sigma^2\nabla^2 L(x, y, k_i\sigma) \end{aligned} \tag{3}$$

Constructing scale-space by computing differnce of two adjacent images in the same scale level. This set is an discrete approximation of continous scale-space.

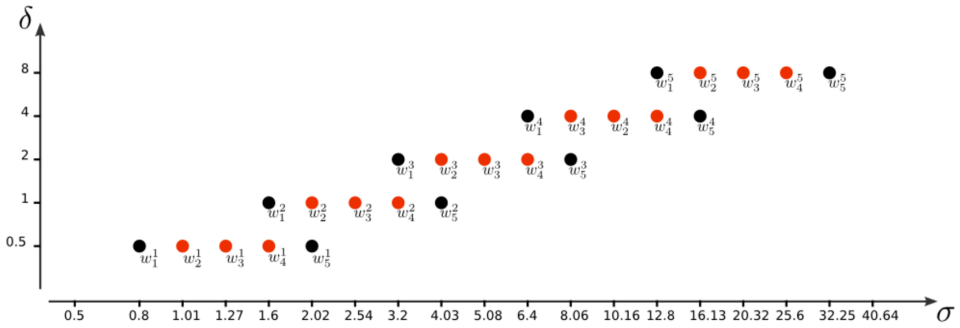


Figure 2: Set of DOG Images

3.1.2 Local extrema detection

In detailed experimental comparisons, Mikolajczyk (2002)[3] found that the maxima and minima of LoG (or DoG by an approximate solution) produce the most stable image features compared to a range of other possible image functions.
To detect the local maxima and minima of $D(\hat{x}, \hat{y}, \hat{\sigma_i})$, each sample point is compared to its eight neighbors $D(\mathbf{x}, \mathbf{y}, \sigma)$ in the current image and nine neighbors in the scale above $D(\mathbf{x}, \mathbf{y}, \sigma_{i+1})$ and below $D(\mathbf{x}, \mathbf{y}, \sigma_{i-1})$.
With $[\mathbf{x}, \mathbf{y}]$ is a 3x3 matrix which the coordinate of center point is $(\hat{x}, \hat{y})$, and $\boldsymbol{\sigma_i}$ is the level of scale and blur $\boldsymbol{\sigma_i} = [\sigma_{i-1}; \sigma_i; \sigma_{i+1}]$

$$D(\hat{x}, \hat{y}, \hat{\sigma_i}) = \mathop{\mathbf{min}}_{\mathbf{max}} \; D(\mathbf{x}, \mathbf{y}, \boldsymbol{\sigma_i}) \tag{4}$$

$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$

(a) $D(\mathbf{x}, \mathbf{y}, \sigma_{i-1})$

$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$

(b) $D(\mathbf{x}, \mathbf{y}, \sigma_{i+1})$

$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$

(c) $D(\mathbf{x}, \mathbf{y}, \sigma_i)$

3.2 Keypoints localization

Many potential key points are discarded if they are deemed unstable or hard to locate precisely. The remaining key points should thus be relatively immune to image noise. This step is to reject points that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

3.2.1 Removing low contrast keypoints

Firstly, we remove **low contrast** keypoints, since they are sensitive to noise. The function value at the extremum, $D(\hat{\mathbf{x}})$, is useful for rejecting unstable extrema with low contrast. All extrema with a value of $|D(\hat{\mathbf{x}})| < 0.03$ were discarded.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{5}$$

Consider D is Taylor expansion (up to the quadratic terms) of the scale-space function, $D(x, y, \sigma)$ and its derivatives are evaluated at the sample point. The location of the extremum $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to $\mathbf{x} = (x, y, \sigma)^T$ and setting it to zero.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad \hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}} \tag{6}$$

3.2.2 Eliminating edge responses

The difference-of-Gaussian function will have a strong response along edges, even if the edge keypoints location along the edge is poorly determined and therefore unstable to small amounts of noise. Due to that reason, we have to remove **edge locating** keypoints. Consider a 2x2 Hessian Matrix, $\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$, stable keypoints will be satisfied the following inequality:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} \leq \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r+1)^2}{r} \qquad \text{with } \alpha = r\beta \tag{7}$$

where α is the eigenvalue with the largest magnitude and β is the smaller.

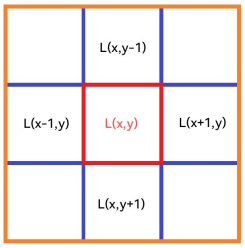
3.3 Orientation assignment

The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner.
For each image sample, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\Phi(x, y)$, is precomputed using pixel differences:

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

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

(a) The gradient magnitude $m(x, y)$ and orientation $\Phi(x, y)$



(b) Smoothed $L(x, y)$ & surrounding pixels

Figure 3: Calculate the gradient magnitude and orientation of an image pixel.

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360-degree range of orientations. Peaks in the orientation histogram correspond to the dominant directions of local gradients.[4]

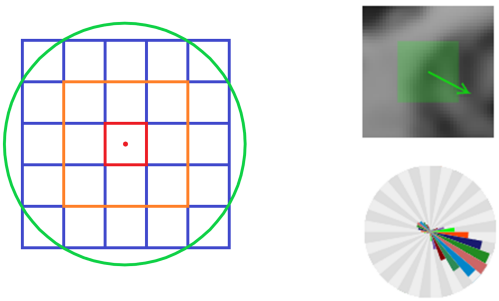


Figure 4: Calculate the dominant orientation of a keypoint with the closest scale.

The dominant orientation will be the highest peak of histogram, and if local peaks are higher than threshold value can be considered as a new keypoint for that orientation, at the same position, octave, scale region, and magnitude as dominant.
Finally, Keypoint location is defined by these following attributions:

1. Position by pixels in original image.
2. The octave and blur levels of image that keypoint derived from.
3. The scale region (neighboring) around keypoint.
4. Orientation of that keypoint.

3.4 Keypoints descriptor

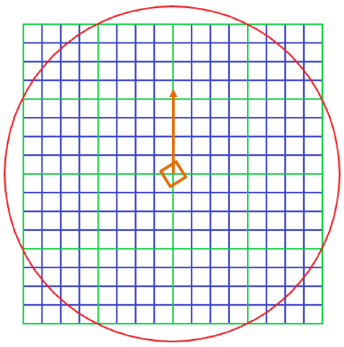
In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint reference (dominant) orientation. SIFT features are formed by computing the gradient at each pixel in a 16 × 16 window around the detected keypoint, using the appropriate level of the Gaussian pyramid at which the keypoint was detected, then applied by a Gaussian filter to reduce the contribution of more distant pixels.

$$\hat{x}_{m,n} = \frac{(m\delta_0 - x_{key})cos\theta_{key} + (n\delta_0 - y_{key})sin\theta_{key}}{\sigma_{key}}$$

$$\hat{y}_{m,n} = \frac{-(m\delta_0 - x_{key})sin\theta_{key} + (n\delta_0 - y_{key})cos\theta_{key}}{\sigma_{key}}$$

$$\Phi(\hat{x}, \hat{y}) = tan^{-1}\left(\frac{L(\hat{x}, \hat{y}+1) - L(\hat{x}, \hat{y}-1)}{L(\hat{x}+1, \hat{y}) - L(\hat{x}-1, \hat{y})}\right)$$

(a) The new rotated pixel formula

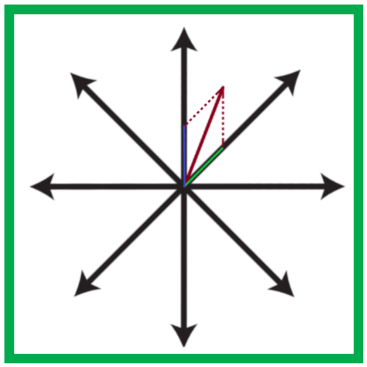


(b) A window of 16x16 pixel surrounded rotated keypoint

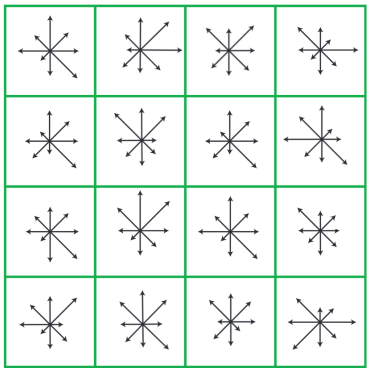
Figure 5: All detected keypoints are rotated for the same reference orientation

A Gaussian weighting function with σ equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point.

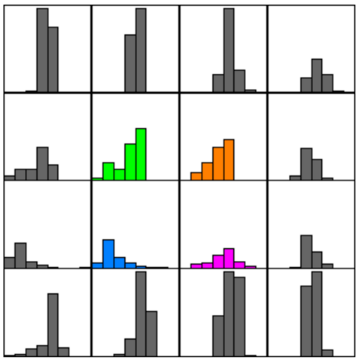
It allows for a significant shift in gradient positions by creating orientation histograms over 4x4 sample regions. The figure shows eight directions for each orientation histogram, with the length of each arrow corresponding to the magnitude of that histogram entry.



(a) Magnitude of each pixels are accumulated into the two nearest bins



(b) A 4x4 descriptors computed from a 16x16 sample array



(c) Flatten the 4x4 descriptor to form feature vector

Figure 6: 4x4 Keypoint descriptors form a 128 elements feature vector

These feature vectors are sensitive to illumination changes since they change the value of each pixel in image, so our vectors should be modified to reduce the effects of those changes. Non-linear illumination changes can also occur due to camera saturation or due to illumination changes

that affect 3D surfaces with differing orientations by different amounts. Therefore, we reduce the influence of large gradient magnitudes by thresholding the values in the unit feature vector to each be no larger than 0.2, and then renormalizing to unit length. The value of 0.2 was determined experimentally using images containing differing illuminations for the same 3D objects. [1]

$$\mathbf{f}_k \longleftarrow \min(\mathbf{f}_k, 0.2 * \|\mathbf{f}\|) \qquad \mathbf{f}_k \longleftarrow \min(\lfloor \mathbf{f}_k \rfloor, 255) \tag{8}$$

Finally, the feature vector is a 128 elements vector after normalized reference orientation. The descriptors will not store the magnitudes of the gradients, only their relations to each other. This should make the descriptors invariant against global, uniform **illumination changes**.

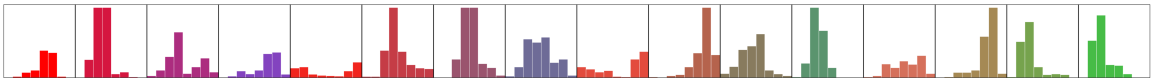


Figure 7: Keypoint Descriptor is a 128 elements vector

4 How to create panorama using SIFT

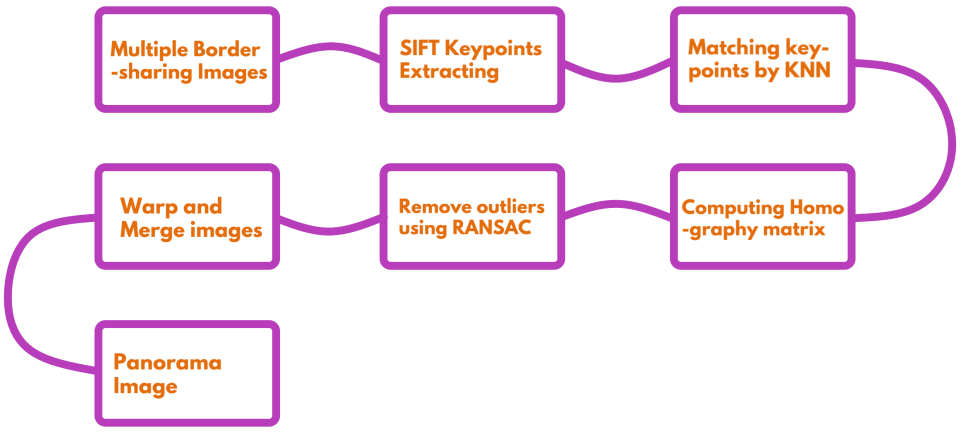


Figure 8: How to create panorama using SIFT

The major topic of this report is the derivation of distinctive invariant keypoints, as described above. To demonstrate their application, we will now give a brief description panorama stitching using those keypoints.

4.1 Matching Keypoints by KNN

Thanks to SIFT algorithm, we can describe a images by interested features. These features is stores by the sets of feature vectors, and these representations are "comprehensible" to machine. These feature vectors called descriptors, are computed by SIFT, and could be a reliable solution for matching similar images problems. Upon obtaining the descriptors for images, we will match them by some matching methods, they are Brute Force Matcher and KNN (K-Nearest Neighbors).

4.1.1 Brute Force Match

The idea of BFmatch is very simple that Brute Force calculates the distance between two points (vectors of 128 dimensions) using the Euclidean distance. With p is one descriptor in first image, q a random descriptor in second one, the Euclidean distance formula is given below:

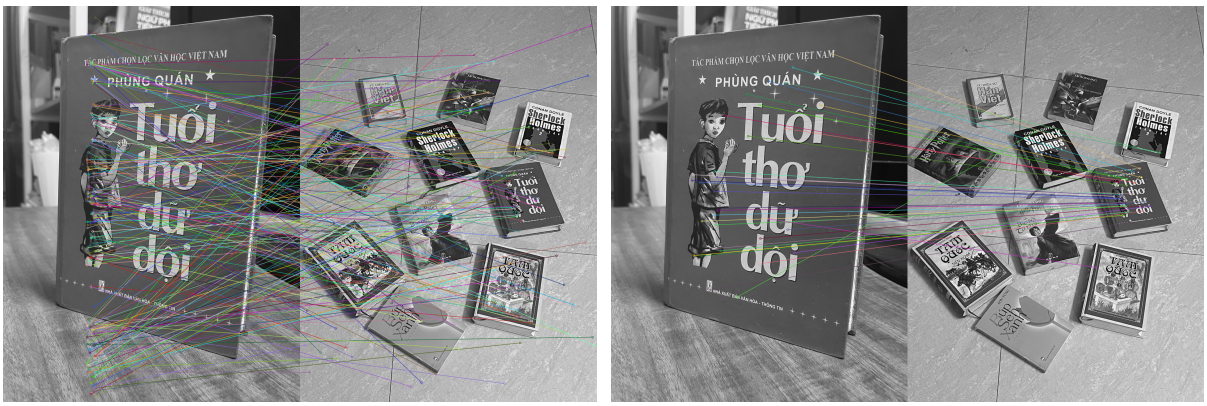
$$d(p,q) = \sqrt{\sum_{i=1}^{128}(p_i - q_i)^2} \tag{9}$$

Thus, for every descriptors in first image, it returns the closest descriptor in second image. The term "closest" here means they have the least Euclidean distance. However, BFmatch returns the single best match for a given descriptor, and has no validation for that return.

4.1.2 KNN (K – nearest neighbors)

Instead of returning the single best match for a given descriptor, KNN returns the k-best matches to validate the closest found match. How could we validate our "closest match"?. The closest match is considered a good match, only if it is superior to other k-best matches. This is why KNN result is more impressive than BFmatch result.

Following Lowe's paper which had chosen k = 2, we validate this good match by a ratio test.



(a) BFmatch, matches = 4270 (b) KNN, good matches = 37

Figure 9: Matching results by BF matches

With d_1 refers to Euclidean distance of closest matches, d_2 is the second one, the metrics R refered to ratio test is given below:

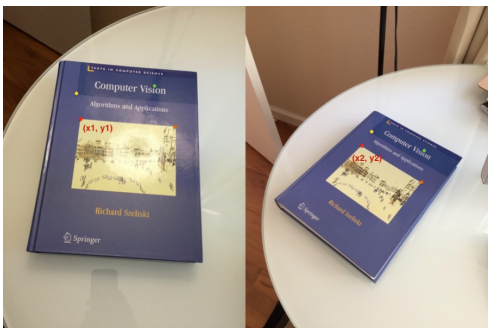
$$R = \frac{d_1}{d_2} < \text{threshold} \tag{10}$$

Threshold value has to be between 0 and 1, commonly around 0.5 or 0.6. If our metrics is satisfied the inequality, that means the closest match is is superior or considered a good match, then appended to good-match set. Otherwise, KNN will turn a blind eye to it, and continue with other descriptor.

4.2 Compute Homography Matrix

4.2.1 What is homography matrix?

A homography matrix, also known as a perspective transformation matrix, is a 3x3 matrix used in computer vision to map points between two images of the same scene taken from different viewpoints. It essentially captures the geometric relationship between the two images and allows you to transform points from one image onto the other.



(a) Two images of a 3D plane (top of the book) are related by a Homography

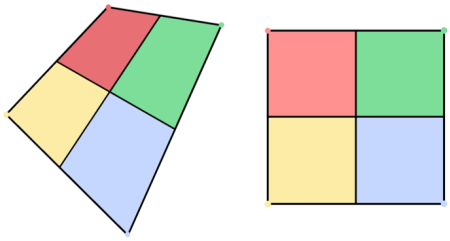
(b) One image of a 3D plane can be aligned with another image of the same plane using Homography

Figure 10: An illustration of using homography matrix

4.2.2 Why is homography matrix necessary?

Although the keypoints have been identified using the presented SIFT and matched by KNN algorithm, we still cannot create a complete panorama image. In particular, the two images we use to create the panorama are taken from different viewpoints, so they cannot be combined to create a panorama image right away. Therefore, the Homography matrix is used to solve this problem by processing the two images so that they look like they were taken from a single frame.

4.2.3 How to compute homography matrix?



(a) Four corners illustrated as four pairs of matched keypoints

$$\underbrace{\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}}_{\text{Destination}} \equiv \begin{bmatrix} \tilde{x_d} \\ \tilde{y_d} \\ \tilde{z_d} \end{bmatrix} = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{\text{Homography matrix}} \underbrace{\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}}_{\text{Source}}$$

(b) Homography maps source keypoints to desination keypoints

Figure 11: Four corners illustrated as four pairs of matched keypoints

Due to the fact that homography matrix is a 3x3 matrix which means we need to find its nine parameters to solve the matrix. Actually, we just need to find only eight parameters after

normalizing the matrix, which means a random parameter is considered as one (as an integer) and followed proportionally by the others. Eight parameters left require 4 equations to be solved, corresponding four pairs matched of keypoints (illustrated as 4 corners in the image above). Then we easily solve matrix after some mathematical transformation:

$$\begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ x_s^{(2)} & y_s^{(2)} & 1 & 0 & 0 & 0 & -x_d^{(2)}x_s^{(2)} & -x_d^{(2)}y_s^{(2)} & -x_d^{(2)} \\ 0 & 0 & 0 & x_s^{(2)} & y_s^{(2)} & 1 & -y_d^{(2)}x_s^{(2)} & -y_d^{(2)}y_s^{(2)} & -y_d^{(2)} \\ x_s^{(3)} & y_s^{(3)} & 1 & 0 & 0 & 0 & -x_d^{(3)}x_s^{(3)} & -x_d^{(3)}y_s^{(3)} & -x_d^{(3)} \\ 0 & 0 & 0 & x_s^{(3)} & y_s^{(3)} & 1 & -y_d^{(3)}x_s^{(3)} & -y_d^{(3)}y_s^{(3)} & -y_d^{(3)} \\ x_s^{(4)} & y_s^{(4)} & 1 & 0 & 0 & 0 & -x_d^{(4)}x_s^{(4)} & -x_d^{(4)}y_s^{(4)} & -x_d^{(4)} \\ 0 & 0 & 0 & x_s^{(4)} & y_s^{(4)} & 1 & -y_d^{(4)}x_s^{(4)} & -y_d^{(4)}y_s^{(4)} & -y_d^{(4)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

After solving this equation to find eight parameters of homography matrix $\mathbf{H}$, this matrix is reshaped to a 3x3 matrix.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \tag{12}$$

4.3 Dealing outliners: RANSAC

4.3.1 Problem with homography matrix

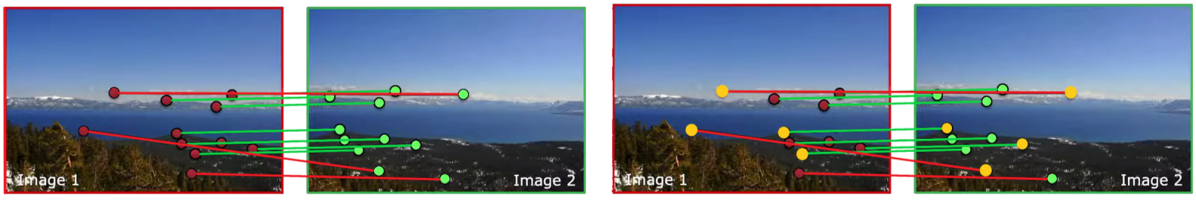
In fact, the homography matrix actually maps two images together if and only if the four pairs of points used to calculate the matrix are actually four pairs of keypoints that are well-matched. However, these two images have a lot of keypoints and the KNN algorithm does not always match them correctly. Therefore, we use a technique called RANSAC to find the best homography matrix, which means that it can map the points in the first image to the points in the second image in the best way, as will be presented in the following section.

4.3.2 RANSAC

RANSAC stands for RANdom SAmple Consensus :

1. Randomly choose s = 4 samples. Typically s is the minimum samples to compute the homography matrix.
2. Compute the homography matrix using chosen samples.
3. Count the number of M of data points(inliers) that fit the model (homography matrix) within a measure of threshold.
4. Repeat Steps 1-3 N times
5. Choose the best homography that has largest number M of inliers

These steps will be discussed in more detail:
Firstly, it is required four pairs of matched keypoint to compute homography matrix. Therefore, we choose four samples randomly (pairs of yellow points in the image below).



(a) Detected matches by KNN
good matches & **bad matches**

(b) Four random keypoints

Figure 12: An illustration of using homography matrix

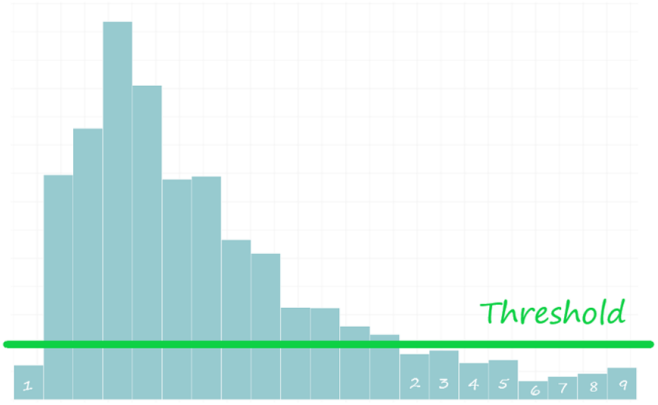
After that, we can compute the homography using four chosen pairs of matched keypoints. However, those keypoints could be well-matched or not which means the homography computed could be wrong. Therefore, we need to count the inliers that fit the model (homography matrix) within a measure of threshold.Specifically, we will consider if a points in the destination image (image 2) solved by multiplying homography and a keypoint found in the source image is actually a keypoint or not by computing the euclidean distance (more clearly in the inequality below)

The n matching pairs: $(x_s^{(i)}, x_d^{(i)})$
$[(x_s^{(1)}, x_d^{(1)}), (x_s^{(2)}, x_d^{(2)}), ..., (x_s^{(n)}, x_d^{(n)})]$

The metric Loss is computed by

$$\text{Loss} = \sum_{i=1}^{n} (H * x_s^{(i)}, x_d^{(i)}) < \text{threshold}$$

(a) Euclidean distance equation



(b) Illustrated chosen inliers by threshold

Figure 13: Choose inliers within a measure of threshold

Note: A keypoint is considered as an inliners when it satisfies the inequality above.
Then, we keep choosing four pairs of matched keypoints, compute the homography matrix, count the number of inliers to find the best homography matrix with the most inliers.

4.4 Warp and merge

We now have a homography matrix **H** allowing us to transform an image to another of the same objects but different viewpoint. This will be visually illustrated by the following images, showing how the homography matrix maps two images together.

(a) Original class image 1

(b) Original class image 2

(c) Warped class image 1

(d) Warped class image 2

Finally, it's time to create a complete panorama from the photos that we took.

Figure 15: Panorama

4.5 Summary

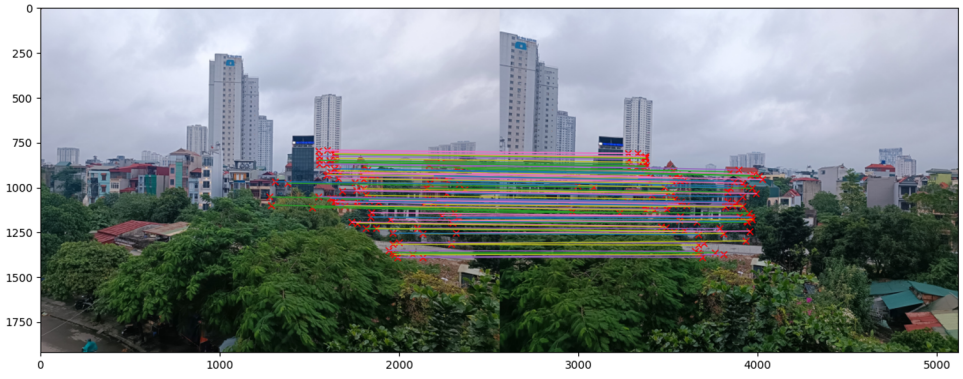
We will summarize the steps to create a panorama. Firstly, we have taken 2 photos of city scene as below:

(a) Origial Image 1

(b) Original Image 2

(c) Keypoints in Image 1

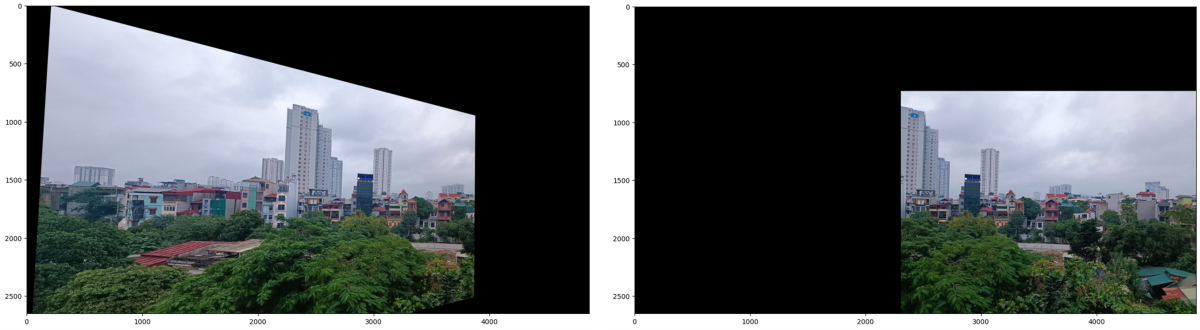
(d) Keypoints in Image 2



(e) Matched keypoints

Figure 16: Photos of City Scene, detected and matched keypoints

Using homography matrix to warp those 2 images.



(a) Warped Image 1 (b) Warped Image 2

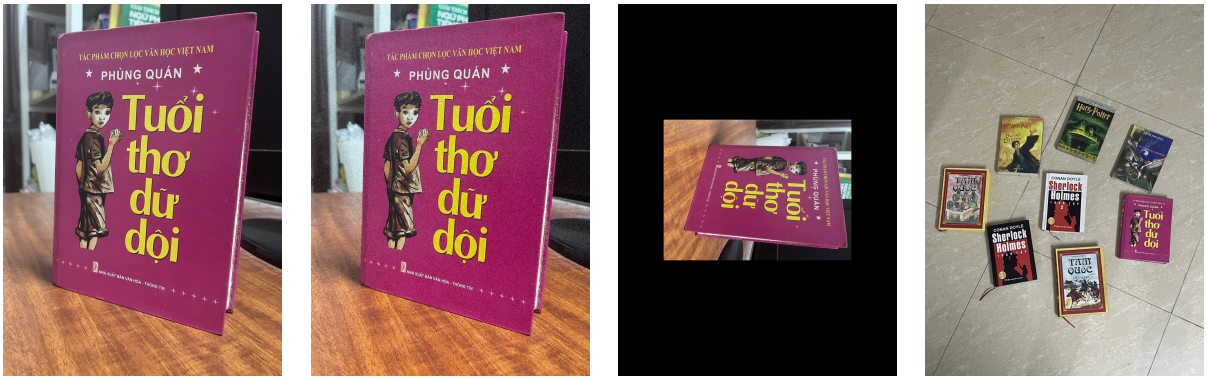
Figure 17: After warping

Now, just stitch them to complete the panorama.

Figure 18: Panorama made of those two image above

5 Harris Comparison

To evaluate the performance of two algorithms in the task of locating correspondences between two images, we implement and analyze through the following three problems.



(a) Original Image (b) Noisy Image (c) Transformed Image (d) Another Image

Figure 19: The following three problem's images

5.1 Problem 1: Original image and a noisy version

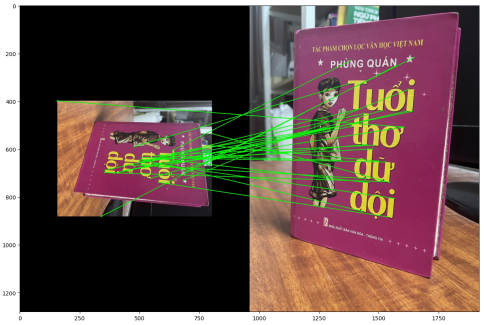
We conduct a locating correspondences experiment between an original image and a noisy version. Both of the results are perfect, but Harris's execution time is approximately four times as much as SIFT's execution time.

(a) Execution Time = 18.132s (b) Execution Time = 4.93s

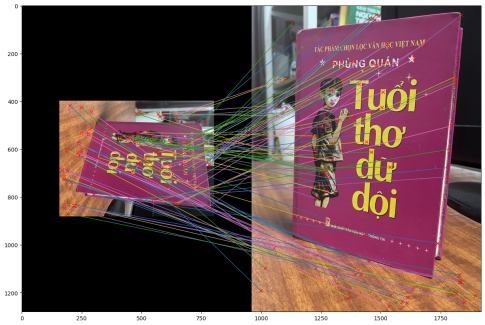
Figure 20: Comparison between Harris and SIFT about locating correspondences performance

5.2 Problem 2: Original image and rescaled & rotated version

We propose a rescaled and rotated image to compare with original image. In this case, the test became more challenging, the results of Harris are imprecise, and its execution time is slightly higher than SIFT execution time.



(a) Execution Time = 5.754s



(b) Execution Time = 4.93s

Figure 21: Comparison between Harris and SIFT about locating correspondences performance

5.3 Problem 3: Object detection

Finally, how can Harris corners detection deal with object detection, although it is not a strong point of Harris algorithm. See the results below, we easily point out that Harris matches are wrong at all, cause Harris approach is really sensitive to scale transforms, while SIFT is invariant with them. Moreover, Harris execution time is as 1.4 times as SIFT's execution time.

(a) Execution Time = 18.425s

(b) Execution Time = 13.49s

Figure 22: Comparison between Harris and SIFT about locating correspondences performance

6 Conclusion

6.1 Panorama Stitching

Feature Matching: SIFT plays a crucial role in panorama stitching by identifying common features in overlapping images. It can recognize and match key points across different images, even when there are variations in scale, rotation, or perspective.

Alignment: After feature matching, SIFT helps in aligning the images correctly. This ensures that corresponding features are in the same spatial locations, allowing for seamless blending of images to create a panoramic view.

Robustness: SIFT's scale invariance and robustness to transformations make it particularly effective in stitching images that may have been taken from different angles or distances.

6.2 Harris Comparison

Based on experiment results, we can briefly conclude that correspondence points from Harris features can be obtained with high execution time and it is also very difficult to get high correct points. Whereas from SIFT features, we can get high correctness and robustness correspondence points with fairly low execution time. For those reasons, Harris corner detection is simpler and computationally efficient, suitable for basic corner detection tasks. On the other hand, SIFT is more robust and versatile, making it suitable for applications requiring scale and rotation invariance, such as object recognition and image matching.

References

[1] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 1–28, 2004.

[2] Tony Lindeberg. Feature detection with automatic scale selection. *Computational Vision and Active Perception Laboratory*, pages 5–9, 1996, 1998.

[3] Krystian Mikolajczyk & Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, pages 64–67, 2004.

[4] Ives Rey-Otero & Mauricio Delbracio. Anatomy of the sift method. *Image Processing On Line*, pages 384–387, 2014.