Undergraduate Thesis
Xiaoxin Zhou
Computer Science
Ryerson University
Toronto, Canada
xiaoxin.zhou@ryerson.ca

Abstract:

Feature extraction is one of the most important tasks in computer vision. It is critical in many applications such as structure from motion (SFM), closed-loop control for simultaneous localization and mapping (SLAM), camera calibration, and feature matching. In previous studies, there has been a lot of research and use of feature extraction algorithms such as SIFT, SURF, LBP, ORB, and more. This paper introduces and compares ORB, SIFT and Affine-SIFT by feature matching processing time. This paper also considers the feature point's quality after using Affine-SIFT, reduces the number of Affine-SIFT feature points to meet the exact quantity requirement, and checks the matching accuracy.

Introduction:

Feature extraction is a fundamental task in computer vision. Feature extraction aims to detect the feature point from 2D images with keypoints and descriptors. Many widespread feature extraction algorithms have been developed in the past few years. The most representative extraction methods are Oriented FAST and Rotated BRIEF (ORB) [1], Scale-Invariant Feature Transform (SIFT) [2], and feature extraction using convolution neural network (CNN) [3].

In recent years, the widespread application of feature extraction includes reducing redundant data from the data set as 2D images, and applying the feature point to machine learning, feature matching for structure from motion (SFM) [4] and Simultaneous localization and mapping (SLAM) landmark extraction.

In this work we will compare the number of feature points, quality of the feature extraction (feature matching), and extraction timing for ORB, SIFT, Affine SIFT, and CNN feature extraction. The goal is to determine the best feature matching performance from different feature extraction methods.

Oriented FAST and Rotated BRIEF:

Features from Accelerated and Segments Test (Fast) Algorithm
[https://www.edwardrosten.com/work/rosten_2006_machine.pdf]
The Fast algorithm is for identifying the interest points in an image.

1. The FAST algorithm will select a pixel P in the image (In figure 1),
2. Set a threshold intensity value T,
3. Using Bresenham circle to create a 16-pixel circle surrounding the pixel P.
4. Set N contiguous pixels out of the 16 and they need to be either darker or brighter than the pixel P by the threshold value.
5. Check pixels 1 5 9 13. If three of them are not brighter or darker, then the pixel P is not in a corner and move to the next candidate corner, otherwise, move to the next steps.
6. Check all 16 pixels and verify a contiguous pixel fall in the criterion or not.

7. Repeat steps one to six for all the pixels in the image.

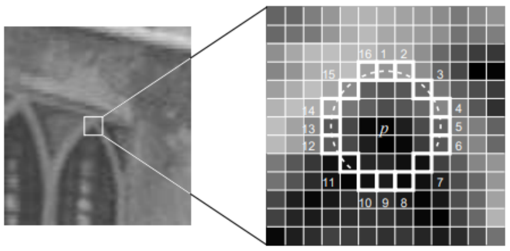

Figure 1: The pixel at P is the at the center of a candidate corner, and there are 16 numbers around the pixel at P. Set a threshold and detect 16 pixels around the candidate corner; in the figure, there are 11 pixels that are brighter than the pixel at P (threshold), FAST detects the pixel at P as a corner on this image.

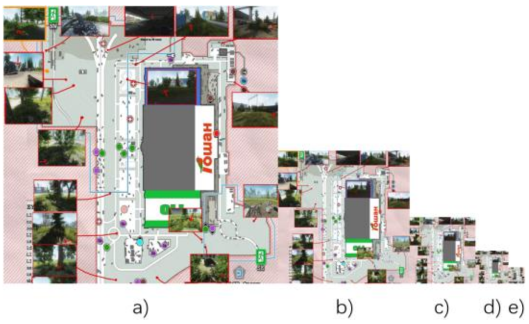

Figure 2: Multiscale image pyramid. a) Level 0 Original image, b) Level 1 1/2 resolution, c) Level 2 1/4 resolution, d) Level 3 1/8 resolution, e) Level 4 1/16 resolution.

However, the FAST algorithm does not include an orientation component and multiscale features. Hence, ORB introduces a multiscale image pyramid (Figure 2). The image pyramid is a multiscale representation of a single image, each level in the pyramid contains 1/2 down sample version of the previous level's image. Detecting keypoints from each level of the image effectively locates keypoints at a different scale. Therefore, ORB will achieve partial scale invariant.

Since ORB found the keypoint with scale invariance, ORB also needs to have the rotation invariance.

The moments of a patch are defined as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y)$$

With the above defined, we can find the centroid, which is the center of mass:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

Construct the vector from the corner's center to the centroid. The orientation of the patch is:

$$\theta = \text{atan2}(m_{01}, m_{10})$$

After the orientation of the patch is calculated, the ORB obtains the rotation invariant.

rBRIEF converts all the keypoints from the FAST algorithm into a binary feature vector (only contains 1 and 0) to represent an object. In BRIEF, each keypoint is described by a feature vector which contain 128–512 bit strings. The benefit of using binary features is that it can speed up data processing and reduce computing resource requirements. In the process, rBRIEF is smoothing the target image using a gaussian kernel to prevent the descriptor from being sensitive to high-frequency noise. Then it defines a neighborhood around the pixel as a patch and selects a random pair of pixels in the patch. The first pixel in the random pair is drawn from the gaussian distribution centred around the keypoint with the standard deviation. The second pixel in the random pair is drawn from a gaussian distribution centred around the first pixel with a standard deviation of two. If the first pixel is brighter than the second, rBRIEF assigns the value of 1 to the corresponding bit else 0. In rBRIEF, it requires the above process 128 times to make a 128-bit vector.

P(x) is the intensity of p at a point x, the binary test τ defined as:

$$\text{Where } \tau(p;x,y) \text{ is defined as}: \quad \tau(p;x,y)=\begin{cases}1 & : p(x)<p(y)\\ 0 & : p(x)\ge p(y)\end{cases}$$
$$p(x) \text{ is the intensity value at pixel } x.$$

The binary feature is defined as a vector:

$$f(n)=\sum_{1<i<n} 2^{i-1}\tau(p;x_i,y_i)$$

To increase the performance of BRIEF, we need a two by n matrix to steer BRIEF according to the orientation of the keypoints.

$$S=\begin{pmatrix} x1,\dots x_n \\ y1,\dots y_n \end{pmatrix}$$

Using the patch orientation θ and the corresponding rotation matrix. The construction as a steered version is:

$$S_\theta = R_\theta S$$

The rBRIEF operator is:

$$g_n(p,\theta)=f_n(p)\,|\,(\mathbf{x}_i,\mathbf{y}_i)\in S_\theta$$

Then discretizes the angle to increments of 12 degrees and constructs a lookup table of precomputed BRIEF patterns. From the above steps, ORB will have an invariant orientation descriptor.

Scale Invariant Feature Transform (SIFT)

SIFT is a common method used for feature detection on an image regardless of the scale of the image. The approach goes through four different stages to achieve the above:

Scale-space

The multi-scale nature of objects is quite common in nature, SIFT also uses a similar way to solve the scale invariant as ORB. SIFT convolved with Gaussian filters at different scales to have a difference of successive gaussian blurred.

$$D(x,y,\sigma)=L(x,y,k_i\sigma)-L(x,y,k_j\sigma)$$

L (x, y, kσ) is the convolution of the original image, *I (x, y)* with the gaussian blur *G (x, y, k)* at scale kσ, and the Gaussian blur operator is:

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

Convolution images and gaussian blur result in the formula:

$$L(x,y,k\sigma)=G(x,y,k\sigma)*I(x,y)$$

Definition of Difference of Gaussians (DoG):

First, we use blurred images to generate another set of images, and use DoG to find interesting keypoints in the image. The DoG process is done from different octaves of the images in the Gaussian pyramid.

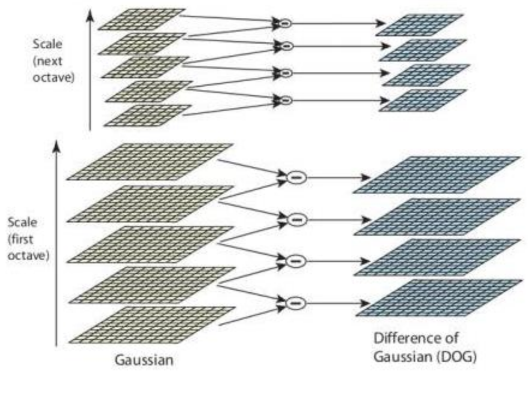

Figure 3: DoG is an approximation of LoG, Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let them be σ and kσ [5].

Once the DoG is found, images are searched for local extrema over scale and space.

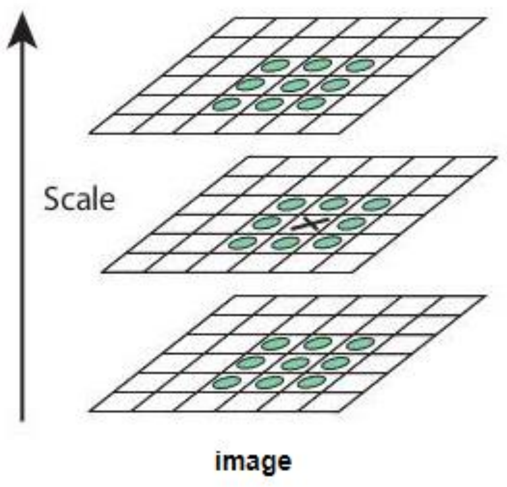

Figure 4: One pixel in an image is compared with its eight neighbors, and nine pixels in its next and previous scales. This process means that this keypoint is the best represented in that scale [5].

Keypoint Localization

To find keypoint localization we can use Taylor series expansion to scale space to get more accurate location of extrema.

Taylor series expansion:

$$f\left(\begin{bmatrix} x \\ y \\ \sigma \end{bmatrix}\right) \approx f\left(\begin{bmatrix} x_0 \\ y_0 \\ \sigma_0 \end{bmatrix}\right) + \left[\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\frac{\partial f}{\partial \sigma}\right]\left(\begin{bmatrix} x \\ y \\ \sigma \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ \sigma_0 \end{bmatrix}\right) +$$
$$\frac{1}{2}\left(\begin{bmatrix} x & y & \sigma \end{bmatrix} - \begin{bmatrix} x_0 & y_0 & \sigma_0 \end{bmatrix}\right)\begin{bmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial \sigma} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y \partial y} & \frac{\partial^2 f}{\partial y \partial \sigma} \\ \frac{\partial^2 f}{\partial x \partial \sigma} & \frac{\partial^2 f}{\partial y \partial \sigma} & \frac{\partial^2 f}{\partial \sigma \partial \sigma} \end{bmatrix}\left(\begin{bmatrix} x \\ y \\ \sigma \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ \sigma_0 \end{bmatrix}\right)$$

In vector form:

$$f(X) = f(X_0) + \frac{\partial f^T}{\partial X}(X - X_0) + \frac{1}{2}(X - X_0)^T \frac{\partial^2 f}{\partial X^2}(X - X_0)$$

$X - X_0$ is the offset from this point and is represented by $\hat{\boldsymbol{X}}$. $\hat{\boldsymbol{X}}$ subbed into vector from is:

$$\hat{X} = -\frac{\partial^2 f^{-1}}{\partial X^2}\frac{\partial f}{\partial X}$$

In the $\hat{\boldsymbol{X}}$, if the offset is over 0.5, then that is an indication that the extremum lies closer to another candidate's keypoint. In this case, the candidate keypoint will be changed. Interplation is performed around the orginial keypoint to find the new canidate. Otherwise, the offset is added to the candidate keypoint list and gets the interpolated estimate for the location of the extremum.

$$f(\hat{X}) = f(X_0) + \frac{1}{2}\frac{\partial f^T}{\partial X}\hat{X}$$

In f($\hat{\boldsymbol{X}}$), if the result is smaller than 0.03 threshold, this keypoint needs to be removed.

EdgeThreshold:

To increase SIFT keypoint stability, we need to eliminate the keypoints that have poorly determined locations but have high edge responses from DoG.

Since the principal curvature across the edge would be much larger than the principal curvature along it. Solve eigenvalues of the second-order Hessian matrix to find the principal curvatures.

Hessian matrix:

$$H = \begin{bmatrix} D_{xx}(x,y) & D_{xy}(x,y) \\ D_{xy}(x,y) & D_{yy}(x,y) \end{bmatrix}$$

There are two eigenvalues, α and β. Note $\alpha=\gamma\beta$

$$Tr(H) = \alpha + \beta$$

$$Det(H) = \alpha\beta$$

The Tr(H) represents the matrix trace, and Det(H) represents the matrix determinant. Firstly, remove all the negative points from the matrix determinant, and remove the high value of principal curvature.

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma}$$

If γ is bigger, it indicates the point might be on the edge, so to identify the principal curvature is over the threshold γ0, we do

$$\frac{Tr(H)^2}{Det(H)} < \frac{(\gamma_0+1)^2}{\gamma_0}$$

In Lowe's paper the $\gamma_0 = 10$.

Orientation Assignment:
Each keypoint is assigned one or more orientations based on local image gradient directions to achieve invariance to image rotation for SIFT keypoints. Use the Gaussian-smoothed image L(x,y,σ).

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

$$\theta(x,y) = arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

For image L(x, y) at σ level, the gradient magnitude, m(x, y), and orientation θ(x, y) need to be precomputed using pixel difference. Then compute every pixel in a neighboring region around the keypoint in the Gaussian-blurred image L to get the magnitude and direction. Next, generate a 36 bin histogram, with each bin containing 10 degrees. Then, a gaussian-weighted circular window is set with 1.5 σ times of the keypoint scale and weighted by the neighboring window gradient magnitude to add to a histogram bin. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. It will create keypoints with different directions with the same location and scale. This action will increase the stability of matching.

Keypoint Descriptor:

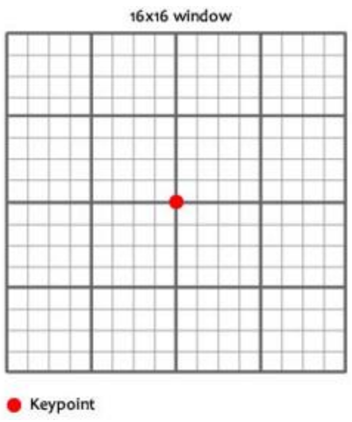

Figure 5: Create a 16x16 window around the keypoint, it is divided into 16 sub-blocks of 4x4 size.

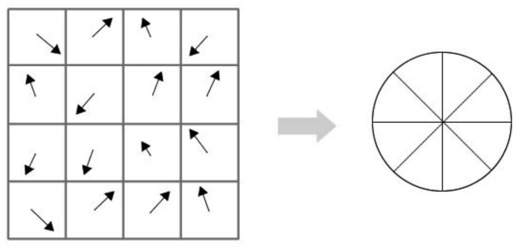

Figure 6: For each sub-block, an 8 bin orientation histogram is created. It's represented as a vector to form keypoint descriptor, one keypoint corresponding to 128 bin value in total.

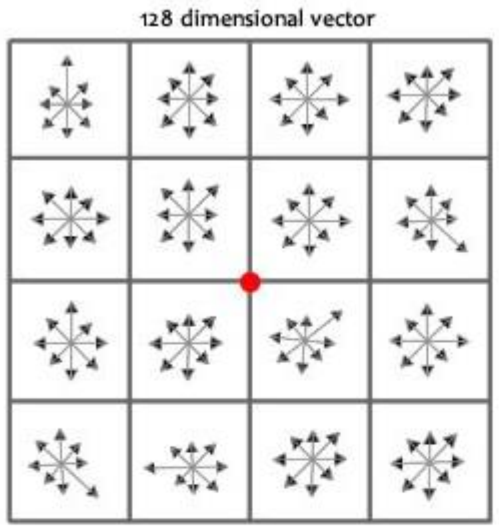

Figure 7: This vector is then normalized to unit length in order to enhance invariance to affine change in illumination. Apply a threshold of 0.2, and the vector is again normalized to reduce the effects of non-linear illumination.

Affine-SIFT:
Affine-SIFT simulates a set of sample view of the initial images obtainable by varying the two camera axis orientation parameters, namely the latitude and the longitude angle, it effectively covers all six parameters (invariance to translation, rotation, and zoom) of the affine transform. Affine-SIFT solves the images high degree tilt invalid feature matching problem.

Image Acquisition Model:

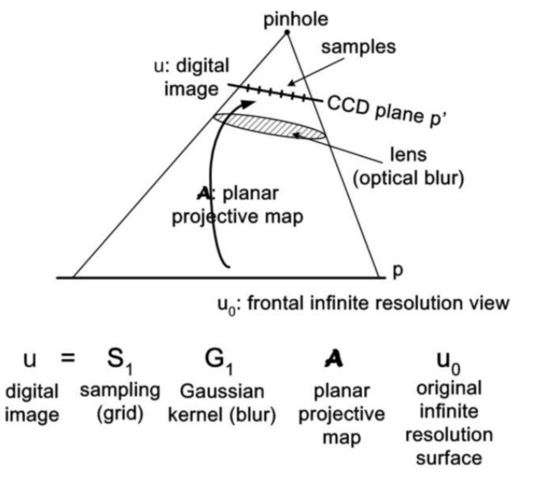

Figure 8: Illustration of the image acquisition model.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = H_\lambda R_1(\psi) T_t R_2(\phi) = \lambda \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

Where $\lambda > 0$, λt is the determinant of A. R is the rotations. $\theta \in [0, \Pi)$, and T is tilt, namely a diagonal matrix with first eigenvalue $t > 1$ and the second one equal to 1.

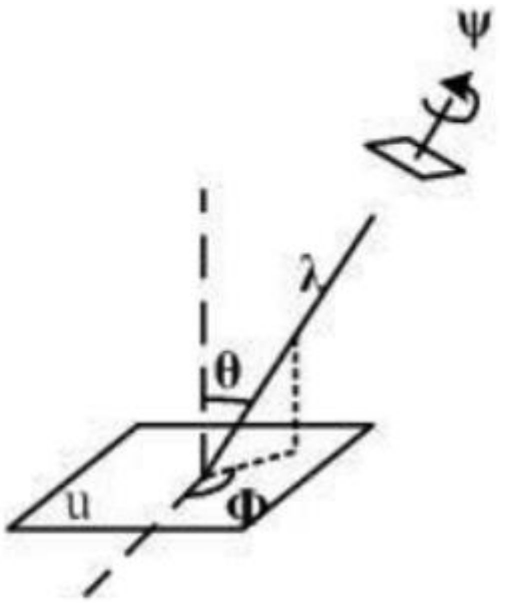

Figure 9: Geometric interpretation of affine decomposition

Assume the camera is far away from the images u, $\lambda = 1$, $t = 1$, $\theta = \psi = 0$

The camera can plane movement by changing the φ (transformation), self-rotation by ψ, changing the angle of longitude by θ, and zoom camera by λ.

Measure Image Between the Frontal View and Slanted View

The t value is used to measure the image tilt between frontal and slanted view. Assume v (x, y) = u(A(x, y)) and w (x, y) = u(B(x, y)) are two slanted views of a flat scene whose image is u (x, y), where A and B are two affine maps. Then v (x, y) = w (AB^{-1}(x, y)). The transition tile between v and w is defined as the absolute tilt associated with the affine map AB^{-1}. Let t and t' the absolute tilts of two images u and u', and let φ and φ' be their longitude angles. The transition tilt t(u, u') between the two images depends on the absolute tiles and the longitude angle, and whether it satisfies $t/t' \leqslant t(u, u') = t(u', u) \leqslant t\,t'$, where assume $t \geqslant t'$. The transition tile can therefore be much higher than an absolute tilt. Hence, it's important for an image matching algorithm to be invariant to high transition tilts.

Algorithm Description:

1. Images are transformed by simulating all possible affine distortions caused from the change of camera pose, which is camera optical axis orientation from a frontal position. Those distortions depend on longitude φ and latitude θ. The images rotate with angle φ followed by tilts with $t = 1 / |\cos\theta|$. For all digital images, the tilt is performed by a directional t-subsampling. Use Gaussian with standard deviation $c\sqrt{(t^2-1)}$. The value c = 0.8 is used to ensure a very small aliasing error. The rotations and tilts are performed at a finite and small number of latitude and

longitude angles. The sampling step is to ensure that the simulated images keep close to any other possible view generated by other values of φ and θ.

2. All the simulated images are compared by SIFT or use other feature extraction as ORB.
3. After feature extraction from the SIFT, we still have many wrong matchings. The criterion used is that the retained matches must be compatible with an epipolar geometry. In ASIFT, use ORSA method to filter out the wrong matching from the SIFT.

Parameter Sampling:

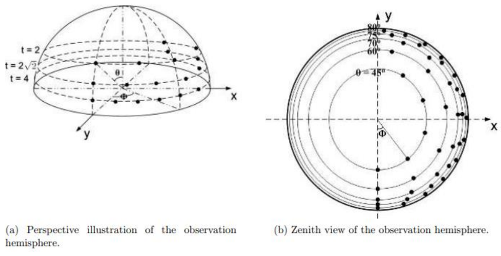
t = 2
t = 2√3
t = 4
x
y
θ
Φ
(a) Perspective illustration of the observation hemisphere.
80°
75°
70°
60°
θ = 45°
(b) Zenith view of the observation hemisphere.

Figure 10: Sampling of the parameters. The samples are the black dots.

The latitudes θ are sampled also such that the asocial tilts follow a geometric series 1, a, a^2, ⋯. , a^n, with a > 1. In the ASIFT paper, the best value of a was √2. The value of n can also increase up to 5, in the consequence transition tilts going up to 32 or more degrees can be explored.

The longitude φ are for each tilt an arithmetic series 0, b/t, .., kb/t, where b ≃ 72 degrees seems again a good compromise, and k is the last integer such that kb/t < 180 degrees.

Computational Complexity:

With the proposed simulated tilt range [tmin, tmax] = [1, 4√2], which allows to cover a transition tilt as high as 32, the total simulated area is about 13.5 times the area of the original image. Since ASIFT simulates 13.5 times the area of the original images, it generates about 13.5 times more features on both the query and the search images. The complexity of ASIFT feature comparison is 13.5^2 = 182.25 times as much as that of SIFT.

CNN Feature Extraction

D2-Net [6] is a single convolutional neural network that plays a dual role: It is simultaneously a dense feature descriptor and a feature detector. D2-Net computes a set of feature maps via a Deep Convolution Neural Network (CNN). These feature maps are then used to compute the descriptors and detect keypoints. Hence, the feature detector corresponds with the feature descriptor tightly.

The first step of the method is to apply a CNN F on the input image I to obtain a 3D tensor F = F(I).

Feature Description:

The 3D tensor F is as a dense set of descriptor vectors d:

$$\mathbf{d}_{ij} = F_{ij:}, \mathbf{d} \in \mathbb{R}^n$$

With i = 1, ..., h and j = 1, ..., w. Using Euclidean distance, descriptors can readily be compared between images to establish correspondences. During the training stage, these descriptors will be adjusted such that the same point in the scene produce similar descriptors, even when the images contain a strong appearance change. Use an L2 normalization on the descriptors prior to comparing them:

$$\hat{\mathbf{d}}_{ij} = \mathbf{d}_{ij} / \|\mathbf{d}_{ij}\|_2$$

Feature Detection:

A different interpretation of the 3D tensor F is as a collection of 2D responses D:

$$D^k = F_{::k}, \ D^k \in R^{h \times w}$$

When k = 1, ..., n. The feature extraction function F can be thought of as n different feature detector functions D^k, each producing a 2D response map D^k. These detection response maps are similar to the Difference of Gaussians (DoG). Like DoG, the detection map would be sparsified by performing a spatial non-local- maximum suppression. In D2-Net, it is contrary to traditional feature detectors, there exist multiple detections maps D^k (k = 1, ⋯, n), and detection can take place on any of them. Based on the detection maps to process detection, If the point (i, j) needs to be detected, it requires:

$$(i, j) \text{ is a detection} \iff D_{ij}^k \text{ is a local max. in } D^k \text{ with } k = \arg\max_t D_{ij}^t \ .$$

For each pixel (i, j), this corresponds to selecting the post preeminent detector Dk, and then verifying whether there is a local maximum at position (i, j) on that particular detector's response map D^k.
In the training stage, the hard detection procedure described above is softened to be amenable for back-propagation. Define a soft local-max:

$$\alpha_{ij}^k = \frac{\exp\left(D_{ij}^k\right)}{\sum_{(i',j')\in\mathcal{N}(i,j)} \exp\left(D_{i'j'}^k\right)}$$

Where N (I, j) is the set of 9 neighbors (3x3 box) of pixel (I, j), define the soft channel selection, which computes a ratio-to-max. Below is the pre descriptor formula that emulates channel-wise non-maximum suppression:

$$\beta_{ij}^k = D_{ij}^k / \max_t D_{ij}^t$$

In order to take α and β into account, we maximize the product of both scores across all feature maps k to obtain a single score map:

$$\gamma_{ij} = \max_k \left(\alpha_{ij}^k \beta_{ij}^k\right)$$

Finally, the soft detection scours s at a point (I, j) is obtained by performing an image level normalization:

$$s_{ij} = \gamma_{ij} \Big/ \sum_{(i',j')} \gamma_{i'j'}$$

For the robustness of scale invariance, D2-Net uses image pyramid and this is only performed during test time. Given the input image I, an image pyramid Ip containing three different resolutions p = 0.5, 1, 2 is constructed and used to extract feature maps *Fρ* at each resolution. Then the larger images structures are propagated from the lower resolution feature maps to the higher resolution ones, in the following way:

$$\tilde{F}^\rho = F^\rho + \sum_{\gamma<\rho} F^\gamma$$

The F^γ are resized to the resolution of *Fp* using bilinear interpolation.

Training Loss:

Given a pair of images (I1, I2) and a correspondence c : A ↔ B between them where A ∈ I1, B ∈I2), the triplet margin ranking loss seeks to minimize the distance of the corresponding descriptors $\hat{d}_A^{(1)}$ and $\hat{d}_B^{(2)}$, while maximizing the distance to other confounding descriptors $\hat{d}_A^{(1)}$ or $\hat{d}_B^{(2)}$ in either images, which might exist due to similar looking image structures. D2-Net defines the positive descriptor distance p(c) between the corresponding descriptor as:

$$p(c) = \|\hat{\mathbf{d}}_A^{(1)} - \hat{\mathbf{d}}_B^{(2)}\|_2$$

The negative distance n(c):

$$n(c) = \min\left(\|\hat{\mathbf{d}}_A^{(1)} - \hat{\mathbf{d}}_{N_2}^{(2)}\|_2, \|\hat{\mathbf{d}}_{N_1}^{(1)} - \hat{\mathbf{d}}_B^{(2)}\|_2\right)$$

Where the negative samples $d_{N1}^{(1)}$ and $d_{N2}^{(2)}$ are the hardest negative that lie outside of a square local neighbourhood of the correct correspondence:

$$N_1 = \underset{P \in I_1}{\arg\min} \|\hat{\mathbf{d}}_P^{(1)} - \hat{\mathbf{d}}_B^{(2)}\|_2 \text{ s.t. } \|P - A\|_\infty > K$$

Same as N₂, the triplet margin ranking loss for a margin M can be then defined as:

$$m(c) = \max\left(0, M + p(c)^2 - n(c)^2\right)$$

The final loss function can be written as:

$$\mathcal{L}(I_1, I_2) = \sum_{c \in \mathcal{C}} \frac{s_c^{(1)} s_c^{(2)}}{\sum_{q \in \mathcal{C}} s_q^{(1)} s_q^{(2)}} m(p(c), n(c))$$

Where $S_c^{(1)}$ and $S_c^{(2)}$ are the soft detection scores at point A and B in I_1 and I_2, respectively, and C is the set of all correspondences between I_1 and I_2.

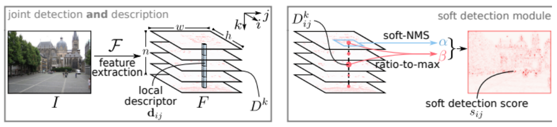

Figure 11: Local descriptors dij are simply obtained by traversing all the n feature maps Dk at a spatial position (i, j), 2. Detections are obtained by performing a non-local-maximum suppression on a feature map followed by a non-maximum suppression across each descriptor - during training. Keypoint detection scores sij are computed from a soft local-maximum score α and a ratio-to-maximum score per descriptor β.

Feature Matching:

Comparing ORB, SIFT, ASIFT and CNN feature extraction, we use feature extraction time, feature matching time, number of features, and matched number. All feature matching has many wrong matchings. In this paper, we use RANSAC to filter out the incorrect matching.

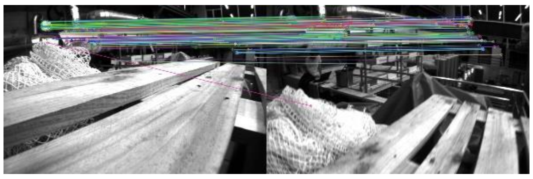

Figure 12: ORB feature matching result.

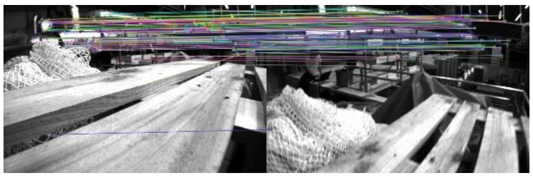

Figure 13: SIFT feature matching result.

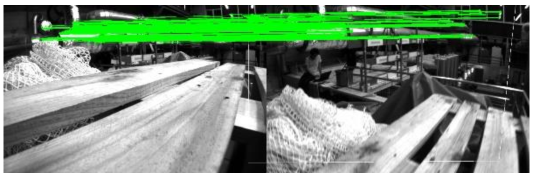

Figure 14: ASIFT feature matching result.

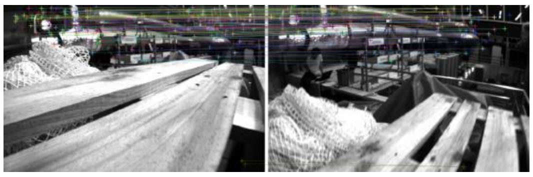

Figure 15: CNN feature matching result.

Feature Extraction	Feature extraction time (second)	First image number of keypoints	Second image number of keypoints	Feature extraction matching time (second)	Number of matched points	Matched rate (%)
ORB	0.024	7823	8306	0.167	343	4.253
SIFT	0.096	2293	2341	0.120	200	8.632
ASIFT	0.263	18094	18609	0.486	707	3.853
CNN	0.3595	896	1359	0.434	273	24.213

Table 1: The matching rate is calculated as the number of matching points divided by the average of the number of keypoints of the two images. The total processing time for

each algorithm is ORB (0.191s), SIFT (0.216s), ASIFT (0.749), and CNN (0.7935s).

Conclusion:

The paper introduces four feature extraction algorithms: ORB, SIFT, ASIFT, and CNN feature extraction. We have also experimented with them to find out the different performances. Based on the matched rate, we find that the CNN feature extraction shows the highest rate, which means the feature points have high precision. However, the feature extraction time and feature matching of CNN take up a lot of computation time, and its number of keypoints is the lowest, which is not efficient. The highest number of matched points is ASIFT (707), but the matched rate of ASIFT is 3.853%, and it also takes up a lot of time for extraction and matching keypoints. This means there are many ASIFT keypoints that are not precise. The ORB took the lowest time for feature extraction from the experiment and matched 343 points in 0.167 seconds. The most effective algorithm we found is SIFT, where the matched rate is 8.632%, and the total processing time is 0.216s.

Reference

[1] Rublee, Ethan; Rabaud, Vincent; Konolige, Kurt; Bradski, Gary (2011). "ORB: an efficient alternative to SIFT or SURF" (PDF). IEEE International Conference on Computer Vision (ICCV).

[2] Lowe, David G. (1999). "Object recognition from local scale-invariant features" (PDF). Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

[3] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. (2019). "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features". CVPR.

[4] S. Ullman (1979). "The interpretation of structure from motion". Proceedings of the Royal Society of London. 203 (1153): 405–426. Bibcode:1979RSPSB.203..405U. doi:10.1098/rspb.1979.0006. hdl:1721.1/6298

[5] OpenCV. "Introduction to SIFT (Scale-Invariant Feature Transform)". https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[6] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, Torsten Sattler. (2019). "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features". CVPR.