2024 COMP6528

Computer Vision

Assignment 3 Report

Name: Xing Chen

UID: u7725171

# Task1 Model Fitting with the Hough Circle Transform

To improve code readability and encapsulation, the implementation for Task 1 was separated into two parts: the core Hough transform and non-maximum suppression functions were placed in HoughCircleTransform.py, while the main execution flow was in Task1.py.

## Read images and convert to grayscale.

As shown in Figure 1-1, the RGB images were read in as NumPy arrays using the cv2.imread() function and then were converted Grayscale。

文本

描述已自动生成

Figure 1‑1 images read and conversion.

## Apply Canny edge detection

It’s very convenient to use canny edge detection here.

The Canny edge detection treats those pixels with gradient higher than high threshold as edges. Pixels with gradient lower than low threshold will be ignored. Only pixels those are neighboring with edge pixels and have gradient between two thresholds will be recon as edges.

To find the best parameters, I plot the gradient histogram of the images as shown in Figure 1-1 and Figure 1-2 to find the proper low and high threshold.

图片包含 形状

描述已自动生成

Figure 1‑2 gradient histogram of coins.png

图表

描述已自动生成

Figure 1‑3 gradient histogram of smarties.png

According to the histograms, a proper threshold for coins.png is around between (100, 200), while for smarties the best choice is about (25,60). After fine-tuning, I choose to use (108, 211) and (30, 65) for coins.png and smarties.png respectively.

Below in Figure 1-4 is the code used to apply canny edge detection.

文本

描述已自动生成

Figure 1‑4 canny edge detection.

## Hough Transform Function

To implement this function, use the algorithm below:

for each pixel in edge image:

for each possible r:

iterate in 360 degrees:

calculate a, b

accumulate the voting if in image range

end

end

end

The code of this function is shown below in Figure 1-5.

文本

描述已自动生成

Figure 1‑5 code of hough circle transform.

## NMS

For the non-maximum suppression, the algorithm below was implemented:

for each circle in the circles:

compare the circles with the remaining circles:

check the difference between them

if difference threshold:

suppress the latter

end

end

Code screenshot is listed below in Figure 1-7.

文本

描述已自动生成

Figure 1‑6 NMS code.

## Result from 1.1-1.4

After implementing all the code in previous tasks, parameters below in Table 1-1 were used and then plot the raw result here.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image | Radius range | Radius step | Theta step | Threshold ratio | NMS threshold |
| coins | (10,100) | 1 | 2 | 0.75 | 10 |
| smarties | (10,100) | 1 | 2 | 0.75 | 10 |

Table 1‑1 parameters before adjustment.

背景图案

描述已自动生成图表, 气泡图

描述已自动生成

Figure 1‑7 circle detected with Hough circle transform.

From Figure 1-8 above, circle transform had detected almost all of the circles in images, but with multiple circles within or around the target.

背景图案

描述已自动生成图表, 气泡图

描述已自动生成

Figure 1‑8 circle detected with NMS

Using NMS, multiple circles are eliminated, but there are still some circles we don’t want. （e.g. Figure 1-9）

## Assessment on result

### Raw result assessment

For coins.png, we can see that there are 10 coins in it. All 10 coins are detected, but 3 false circles are detected too.

* **False negative rate:** 0%.
* **False positive rate:** 23.1%.
* **Accuracy:** 76.9%.

For smarties.png, we can see that there are 14 smarties in it. 11 smarties are detected, missing 3 smarties. No false circles.

* **False negative rate:** 21.4%.
* **False positive rate:** 0%.
* **Accuracy:** 100%.

Effectiveness and shortcomings of the result.

* **Effectiveness:** The algorithm performs well on recognizing desired items.
* **Shortcomings:**
  + Image with complex texture will cause false positive. This need to be improved.
  + Some items cannot be detected.
  + Single set of parameters is not universal.

### Adjustment on parameters

To adjust the result, we need to modify the parameters.

Finaly, fetching the accuracy and preserving the time efficiency at the same time, I choose the parameters as in Table 1-2 below to better fit the task:

* Higher the min\_radius to 25 to ignore circles that are smaller than our item.
* Lower the max\_radius to 50 to increase time efficiency.
* Increase the radius step to 2 and angle step to 10 to increase time efficiency.
* Lower thresholde ratio to 0.7 and 0.6 respectively to include items that are not detected but should be.
* The NMS threshold distance was set to 10 pixels, as this value eliminated duplicate detections while preserving adjacent distinct circles.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image | Radius range | Radius step | Theta step | Threshold ratio | NMS threshold |
| coins | (25,50) | 2 | 10 | 0.7 | 10 |
| smarties | (25,50) | 2 | 10 | 0.6 | 10 |

Table 1‑2 parameters after adjustment.

### Final result

Using this parameters, we can see that the result is much better in Figure 1-9 and 1-10, with 100% accuracy upon coins and very high accuracy upon smarties.

背景图案

描述已自动生成图表, 气泡图

描述已自动生成

Figure 1‑9 Hough circle detection after parameter optimization.

背景图案

描述已自动生成图表, 气泡图

描述已自动生成

Figure 1‑10 NMS on circle detection after parameter optimization.

Almost all circle are detected and no false positive are produced. The time efficiency has also increased dramatically as we can see in Figure 1-11.

文本

描述已自动生成

Figure 1‑11 only 3 seconds are needed.

### Circles not in image

I also noticed that some circle that are not totally in the smarties image are ignored. If we want to keep them, we need to adjust our algorithm like shown in Figure, Figure and Figure.

文本

描述已自动生成

Figure 1‑12 Expending the accumulator by an offset.

文本

描述已自动生成

Figure 1‑13 set a, b back to original coordinates.

## Impact of different parameters

This part we will discuss the impact of different parameters of the Hough circle transform. We will supplement them with pictures. All parameters in this section are the same as Table 1-2 in **1.6.2**.

* **Radius range:** Table 1-3
  + **Time Efficiency:** Expanding the range will increase time cost of the algorithm. **Impact level: high.**
  + **Result Accuracy:** If the range is not proper (e.g. min range higher than the items) , will loose information. **Impact level: high.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image | Radius range | (10, 100) | (25, 50) | (30, 40) |
| coins | Accuracy |  |  |  |
| Run time | 11s | 4s | 1s |

Table 1‑3

* **Radius step:** Table 1-4
  + **Time Efficiency:** Higher step will dramatically reduce time cost. **Impact level: high.**
  + **Result Accuracy:** Expanding the radius step has significantly impact on item detecting (e.g. losing circles or circle with offset). Better keep it 1. **Impact level: high.1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image | Radius step | 1 | 2 | 5 |
| coins | Accuracy |  | 背景图案  描述已自动生成 |  |
| Run time | 8s | 4s | <1s |

Table 1‑4

* **Theta step:** Table 1-5
  + **Time Efficiency:** Higher step will dramatically reduce time cost. **Impact level: high.**
  + **Result Accuracy:** Expanding the theta step in a proper range has less impact on result. **Impact level: medium.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image | Theta step | 5 | 10 | 20 |
| coins | Accuracy |  | 背景图案  描述已自动生成 |  |
| Run time | 8s | 4s | 2s |

Table 1‑5

* **Threshold ratio:** Table 1-6
  + **Time Efficiency: Impact Level: low**.
  + **Result Accuracy:** Low threshold will include noise in to result and high threshold will lose vital information. **Impact level: high.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image | Threshold ratio | 0.5 | 0.7 | 0.9 |
| coins | Accuracy |  | 背景图案  描述已自动生成 |  |
| Run time | 4s | 4s | 3s |

Table 1‑6

* **NMS threshold:** Table 1-7
  + **Time Efficiency: Impact Level: ignorable**.
  + **Result Accuracy**: Huge numbers will cause circles that are close to or overlapping with each other to be suppressed. On the other hand, small values of threshold can’t detect some of the circles that should be suppressed. **Impact level: high.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| image | NMS threshold | 5 | 10 | 20 |
| coins | Accuracy |  | 背景图案  描述已自动生成 |  |

Table 1‑7

## Time and memory efficiency

### hough\_circle\_transform

**Time complexity - O(E \* R \* (360 / angle\_step))**

* initialization:
  + initializing accumulator - O(H\*W\*R)
  + calculating cos and sin - O(360 / angle\_step)
* iterate edge pixels: O(E)
* iterate radius: O(R)
* iterate angle: O(360 / angle\_step)

where E represents the number of edge pixels, R is the number of radius.

**Space complexity - O(H \* W \* R)**

* accumulator: O(H \* W \* R)
* rad array: O(360 / angle\_step)

### non\_maximum\_suppression

**Time complexity** - **O(C^2)**

* iterate circles: O(C)
* compare with other circles: O(C)

where C is the numbers of circles.

**Space complexity - O(C)**

* marker ‘supressed’ array: O(C)
* output ‘suppressed\_circles’ array: O(C)

# Task2 Two-view Homography Estimation

## Read images

As shown in the code in Figure 2-1, two images were read using cv2.

文本

描述已自动生成

Figure 2‑1 images reading using cv2.

## Homography function

My function implements the algorithm below:

For each correspondence coordinates in image 1 and 2:

Compute matrix A

Assemble A

Compute SVD of A

Take the last column of V and reshape it as H

Normalize H

The code is listed below in Figure 2-2.

文本

描述已自动生成

Figure 2‑2homography function.

## Transform unnormalized points.

The algorithms used in these two functions are:

def compute\_normalisation\_matrix:

Compute the centroid of these points

Shift the points to center the centroid at the origin

Compute average distance and then the scale factor

Construct the normalization matrix T

def homography\_w\_normalisation:

compute the normalization matrixes of two points set

normalize the coordinates using the matrixes

compute homography H

denormalized H using H = T2^(-1) \* H\_norm \* T1

Code used in this section is listed below in Figure 2-3 and 2-4.

文本

描述已自动生成

Figure 2‑3 compute normalization matrix

文本

描述已自动生成

Figure 2‑4 homography\_w\_normalisation

## SIFT

To extract features, cv2.SIFT was used to detect and compute the SIFT features.

To find matching feature points across the images, BFMatcher was found useful.

To get the sample points, seed 42 was used to randomly sample half of the matched points. The code is listed below in Figure 2-5.

文本

描述已自动生成

Figure 2‑5 get sample points.

## The RANSAC algorithm

The RANSAC algorithm is like this:

Randomly select 4 point pairs from the matched feature points between image 1 and image 2.

Compute the homography matrix H using the selected 4 point pairs and the normalized DLT algorithm.

Apply the computed homography matrix H to transform all the points from image 1.

Compute the Euclidean distance (error) between the transformed points from image 1 and their corresponding points in image 2.

Classify the point pairs as inliers if their distance (error) is below a specified threshold.

Update the best homography matrix if the current iteration yields a higher number of inliers than the previous best.

Repeat steps 1-6 for a maximum number of iterations or until the maximum number of iterations is reached.

Return the best homography matrix found across all iterations.

The code is listed in Figure 2-6.

文本

描述已自动生成

Figure 2‑6 homography\_w\_normalisation\_ransac.

## Warp and stitch

### algorithm

To implement the warp and stitch function, I followed these steps:

Use the homography matrix to transform the corner points of the first image.

Calculate the minimum and maximum coordinates of the transformed corners of the first image and the corners of the second image.

Use these coordinates to determine the size of the resulting stitched image.

Use the homography matrix to warp the first image to align with the second image.

Overlay the warped first image onto the second image to create the stitched image.

The code for the warp and stitch function is shown below in Figure 2-7:

文本

描述已自动生成

Figure 2‑7 function for warp and stitch

### Result

After processing and stitching the images mountain1.jpg and mountain2.jpg, the result is shown in Figure 2-8. The homography estimation successfully aligned and blended the overlapping areas of the two images to produce a seamless stitched image. The visual inspection confirms that the stitching process effectively merged the images, preserving the continuity of the scene.



Figure 2‑8 Result

## Discuss the factors that affect the rectified results theoretically and empirically.

### Theoretical Analysis

Quality of Feature Extraction and Matching

* Number and Distribution of Feature Points: The quantity and distribution of feature points impact the quality of matching. If there are too few feature points or if they are unevenly distributed, it may be difficult to find enough matching points, leading to registration failures or inaccuracies.
* Discriminability and Robustness of Feature Descriptors: The distinctiveness and robustness of feature descriptors are also crucial. Good feature descriptors should remain stable under image transformations (such as rotation and scaling) and noise interference to ensure correct matching.

Parameter Settings of the RANSAC Algorithm

* Distance Threshold: The RANSAC distance threshold determines which points are considered inliers. A threshold that is too small may exclude many correct matches, while a threshold that is too large may include more incorrect matches, reducing the accuracy of registration.
* Maximum Iterations: The maximum number of iterations in RANSAC affects the probability of finding the optimal model. Insufficient iterations may miss the best model, while too many iterations will increase computational costs.

Changes in Viewing Angles and Overlapping Area Size Between Images

* Viewing Angle Changes: Larger changes in viewing angles lead to greater differences in the appearance of corresponding points, making matching more challenging. When the viewing angle changes beyond a certain range, the homography model may not accurately describe the transformation between images.
* Size of the Overlapping Area: The size of the overlapping area also affects the registration results. A small overlapping area means a limited number of feature points available for matching, and the coplanar assumption may not hold, leading to inaccurate homography estimation.

Other Factors

* Lighting Changes: Changes in lighting can alter the appearance of images, affecting feature extraction and matching.
* Image Distortion: If image distortions (such as lens distortion) are not well corrected, they can cause registration errors.
* Low-Texture or Repetitive-Texture Areas: In areas with low texture or repetitive texture, feature points may be difficult to accurately locate and match.

### Experimental Analysis

Ground Truth:

Ginput was use here to get the points manually selected and then used to calculate the ground truth of H.

Below in Figure 2-9 is the code used in this section. We manually choose 7 points from the image and use them to get the H.

文本

描述已自动生成

Figure 2‑9 code for task 2-7

Using these points, we get our “ground truth” H in Figure 2-10 and the result stitched image using this H in Figure 2-11 below. Compared with the result in Figure 2-8, we can see that out algorithm get a very familiar H with the “ground truth”.

文本

描述已自动生成

Figure 2‑10 H from manually choose points

山上的风景

描述已自动生成

Figure 2‑11 result using "ground truth"

This picture in Figure 2-12 is the points selected using ginput on image1.



Figure 2‑12 points selected manually in image1

Threshold:

A threshold too low or too high will cause inaccuracy because low thresholds will exclude many correct matches, while high thresholds will include many errors.

The impacts on result were listed below in Table 2-1.

|  |  |  |  |
| --- | --- | --- | --- |
| Threshold | 0.05 | 1 | 20 |
| Result |  |  |  |
| Max inliers | 7 | 95 | 134 |

Table 2‑1 when iterations = 1000

Iterations

Number of iterations decide the possibility of getting best model. Insufficient iterations could miss the best model. A big number of iterations may increase time cost.

The impacts on result were listed below in Table 2-2.

|  |  |  |  |
| --- | --- | --- | --- |
| Iterations | 500 | 1000 | 2000 |
| Result |  |  |  |
| Max inliers | 87 | 95 | 109 |

Table 2‑2 when threshold = 1000

# END