DSA5203
Visual Information processing and Interpretation

National University of Singapore
Data Science and Machine Learning Department
March 30, 2023

Assignment 2 Report

Wang Jiangyi, A0236307J
e0732857@u.nus.edu

1 Illustration and Discussion of Rectification Algorithm

Firstly, this assignment requires us to design an algorithm that rectifies one image to be horizontally placed.

Generally speaking, my algorithm can be decomposed into 8 parts:
1. change RGB image to GRAY image;
2. utilize Gaussian filter to remove noise (actually Gaussian filter will remove some complex pattern in image to make algorithm more robust);
3a. detect edge via Canny detector and change GRAY image to BINARY image;
3b. generate
4. detect all contours;
5. find the contour with the largest area for those 2 BINARY images;
6. find 4 corners of that contour and choose the BINARY image with exactly 4 corners;
7. fit perspective transformation from 4 corner points;
8. warp image with respect to the fitted perspective transformation via backward warping;

Afterwards, we will discuss more on the algorithm implementation and corresponding observation.

1.1 Gaussian Filter

Firstly, let's talk about Gaussian Filter. The main reasons for this low-pass filter are, flatten the noise in the image and kick out those complicated details in the image, like patterns in photo frame. This can be shown as the figure below:

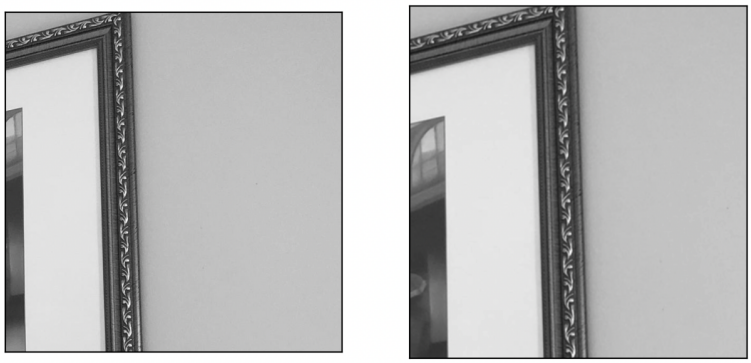

Figure 1: Raw Image Pattern (LHS) versus Gaussian Blur Pattern (RHS)

In Figure 1, we use 3×3 Gaussian Filter with standard devation 10 to smooth the image. It can be observed that, in the LHS, the image is more clear and in the RHS, the image is more blur. Moreover, in LHS, the edge is shaper due to the complicated pattern in photo frame.

Actually the complex pattern in image is not a good thing for image processing, since it will make the image gradient unstable. As a consequence, we will generate more useless edges which we are not interested in.

To give an example, in Figure 1, our interests can be the boundary of photo. However, if given complex patterns, then there are plenty of edges generated by complex patterns near the boundary, which will make the task more difficult. **This can be one motivation for us to choose Gaussian Filter**. Furthermore, the choice of filter size and corresponding std is critical, as we will show in the later discussion.

1.2 Canny Detector

One thing to discuss about Canny detector is, its choices of lower threshold and upper threshold. The mechanism of these 2 parameters is, if the magnitude of gradient is below T_{low}, then set the pixel value to 0. If the magnitude of gradient is above T_{upper}, then set it to 1 and marked as 'strong edge'. Otherwise, check whether this pixel is connected to one 'strong edge'. If so, set it to 1 and 0 instead. Noted that via Canny Detector, we can translate one gray image into bianry image, which can be used to find contours in the later stage.

Therefore, if we want to remove those complicated patterns, we should set those 2 thresholds to be sufficiently large, such that we are able to filter them out. Notice that, this process can cooperate with different sizes of Gaussian Filter to extract features we want.

Here, we use the previous example to illustrate the motivation. Observe that, there are plenty of thin edges between the inner and outer boundary of the painting. Based on this observation, it recommends us to use 'average' low-pass filter to smooth those thin edges, which is equivalent with Gaussian filter with large standard deviation. Here we choose 3×3 Gaussian Filter with $std = 10$. Afterwards, to filter out those smoothed thin edges, we should choose a relatively big threshold. Here, we choose $T_{upper} = 300, T_{lower} = 100$. The final result can be shown in Figure 2:

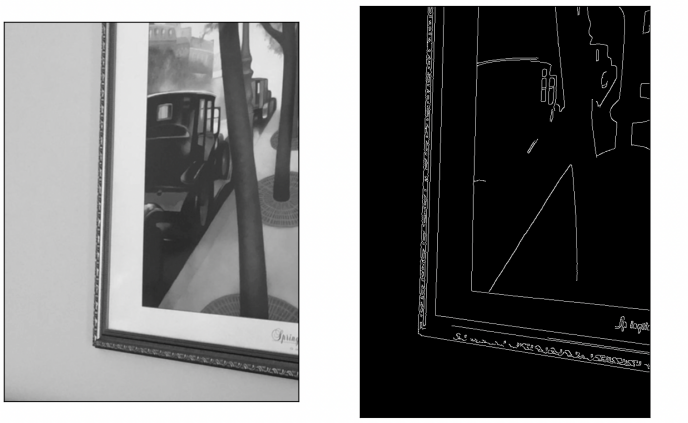

Figure 2: Gassuain Blur and Canny Detector

As you can see, the patterns between boundaries have already been smoothened. This makes the outer boundary smoother, which is approximately a line.

To make comparison, if we use Gaussian Filter with small *std* and Canny Detector with small threshold, then we will attain a more messy image. This can been shown as follows:

Figure 3: Gassuain Blur and Canny Detector with small *std*

In Figure 3, you can easily observe that, there are more edges in the binary image generated by Canny Detector. That is, we preserve more detailed pattern of the raw image. This will bring great difficutly to detect contour since these detailed patterns can be viewed as noise with respect to the interested outer boundaries.

1.3 Contours

After deriving those edges described by one binary image, we are abled to find contours via function pre-defined in cv2 package. This function actually returns all the contours.

In my application, I select the interested outer boundary via the largest-area contour. After that, find its 4 corner points, which will be used in perspective transformation fitting. Given utilize appropriate Gaussian Filter and Canny Detector (as shown in Figure 2), the result of contours is shown as follows:

Figure 4: Contours

To make the algorithm more robust, we consider using the BINARY image via threshold for contour generation.

1.4 Perspective Transfom

After we attain the 4 corner points of outer boundary, we can fit for perspective tranformation matrix via Least Square Fitting.

It is worthwhile to notice that, here we choose 'Perpsective Transformation' to approximate the rectified transformation. We do not choose affine transformation because in unrectified image, parallel lines will not be paralleled. We only want to guarantee the preservation of straight lines. This is why I use 'Perspective Transformation' instead of 'Affine Transformation' here.

Lastly, achieving the estimated perspective transformation, I warp the image via backward warping and the result is shown as follows:

Figure 5: Rectified image

The result is pretty good!