Dongheng Lin
Feb 25th, 2024

CS 549: Computer Vision
Assignment 1

1 Contour Detection

1.1 Method Description

- **Warm-up**: In the warm-up section, the convolution part is changed to handle the boundary artifacts by treating the boundary symmetrically, avoiding creating edges where there are none at the boundaries.
- **Smooth**: In the smoothing part, we used a Gaussian kernel of size 5×5 and $\sigma = 1$ to smooth the input image while keeping the original gradient computation kernel to be the same, thereby realizing an equivalent effect.
- **NMS**: The Non-Maximum Suppression (NMS) applied to the computed gradients is implemented by first calculating the orientation of the gradient at each pixel. The gradient magnitude is then compared with the magnitudes of the pixels in the direction of the gradient (perpendicular to the edge direction). Only the pixel with the maximum value in its neighborhood along the gradient direction is retained, ensuring that edges are thin and well-defined. This process is crucial for identifying prominent edges while minimizing the effect of noise and less significant variations in the image.

1.2 Precision Recall Plot

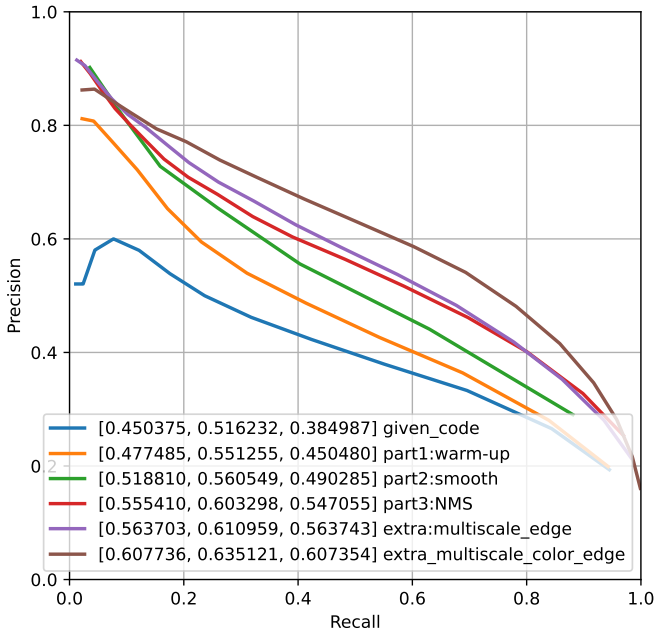

Figure 1: Contour Benchmarking

1.3 Results Table

1.4 Visualizations

As visualized in figure below, compared with baseline, the warm-up part resolved the corner artifacts, the smooth part gives softer contours, while the NMS makes it more precise and sharp. Finally, with Multi-scale gradients and

Table 1:

Method	overall max F-score	average max F-score	AP	Runtime (seconds)
Initial implementation	0.450	0.516	0.385	0.004091
Warm-up [remove boundary artifacts]	0.477485	0.477485	0.450480	0.004173
Smoothing	0.518810	0.560549	0.547055	0.006852
Non-maximum Suppression	0.555410	0.603298	0.547055	0.080247
Val set numbers of best model [From gradescope]	0.604549	0.635462	0.604829	1.099172

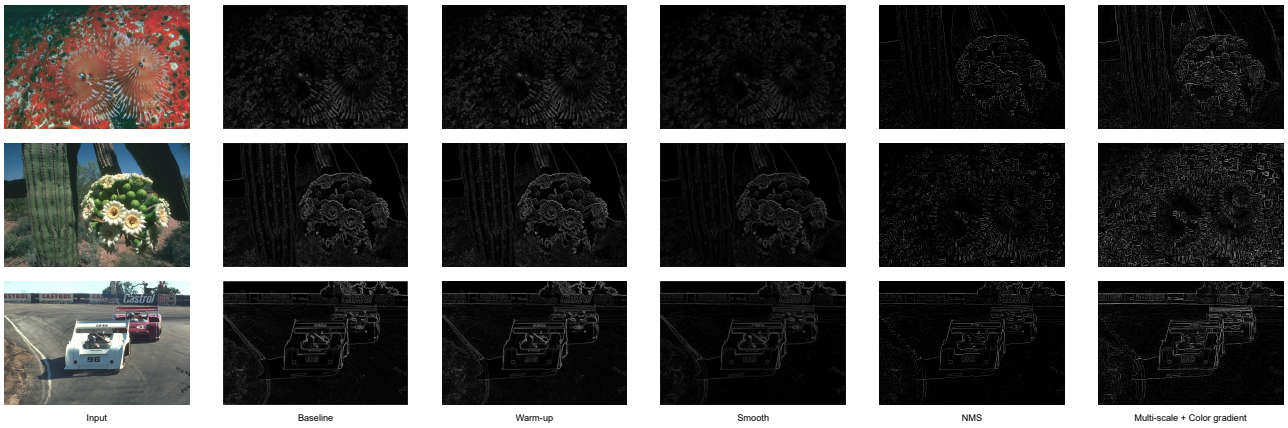

Figure 2: Result Visualization

taking maximum gradients for each color channels, the best result is reached with sharp and detailed contours.

1.5 Bells and Whistles

There are two methods that eventually contributes to the benchmark improvement in the Bells and Whistles.

- **Multi-scale gradient**: Detecting edges at multiple scales can help capture both fine details and broader structure. Implementing this involves applying Gaussian blurring with different σ values (scales) before edge detection. Then, you can combine the edge maps from each scale to get a more comprehensive edge map.
- **Color Channel gradient**: Edge detection can be improved by considering color information. Our approach is to use the gradient of color models that are more perceptually uniform, like Lab or HSV. Here, we take the maximum gradients among Lab channels.

Table 2: Results

Method	overall max F-score	average max F-score	AP	Runtime (seconds)
Multi-scale gradient = [1, 2]	0.563703	0.610959	0.563743	0.229228
Multi-scale gradient = [1, 2] + Color Channels	0.607736	0.635121	0.607354	0.678846

2 Corner Detection

2.1 Method Description

In the implemented Harris contour Detection, First, the gradients on each points are calculated along the x and y directions using Scharr operators. Secondly, for each pixel, the algorithm compute elements of the structure tensor (also known as the second moment matrix), which involves products of derivatives and their Gaussian-weighted sums. The Harris corner response R is computed for each pixel using the determinant and trace of the structure tensor as it is in Original Harris Algorithm. After normalizing the corner response, the implementation iterates through each

pixel in the response map and apply a threshold to identify potential corners. Then, for each potential corner, it can only become a corner when it's a local maximum in its neighborhood.

There are a few further improvements which will be discussed in the 2.5.

2.2 Precision Recall Plot

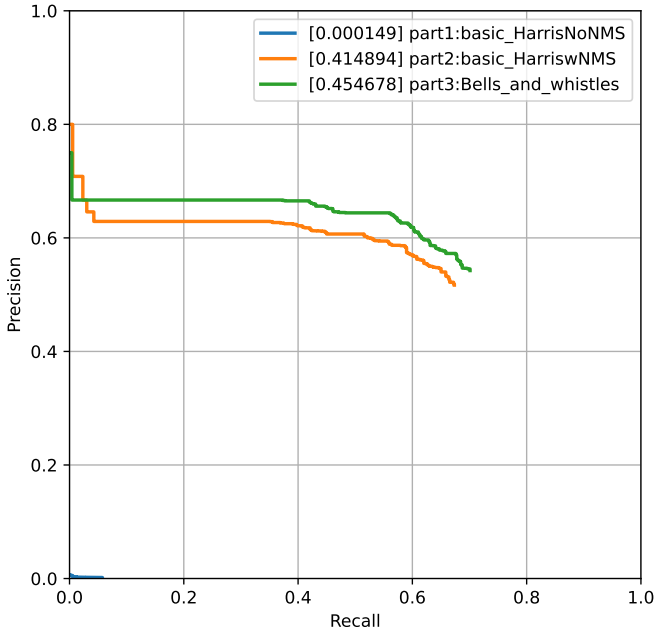

Figure 3: Corner Benchmarking

2.3 Results Table

Table 3: Results

Method	Average Precision	Runtime
Random	0.002	0.001
Harris w/o NMS	0.000149	0.131228
Harris w/ NMS	0.414894	0.081793
k = 0.02, sigma = 0.8 (On Gradescope)	**0.454678**	**0.082969**
k = 0.02, sigma = 1	0.418846	0.100993

2.4 Visualizations

As we can see from the visualization of results, we notice that the model is generally good at detecting corners encircled by black edges while it did not notice any corners under different colors. This is suspected to be caused by the way of loading and precessing the images when calculating the gradients in current pipeline. Currently, we are using grayscale version of images to calculate the gradients which will automatically result in the black edged corners are more emphasized in the prediction. A possible solution is to include different color channels in computation as it is implemented in the Contour detection.

Figure 4: Beckman Visualization

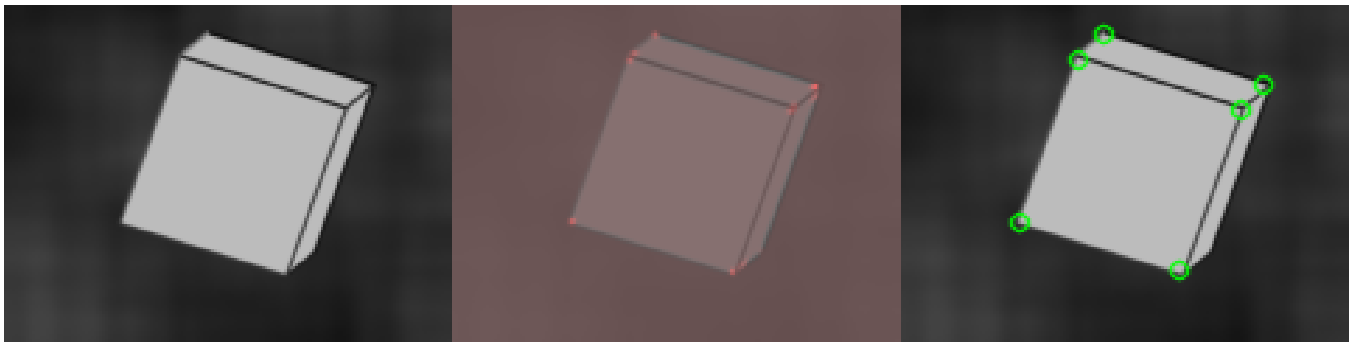

Figure 5: Cube Visualization

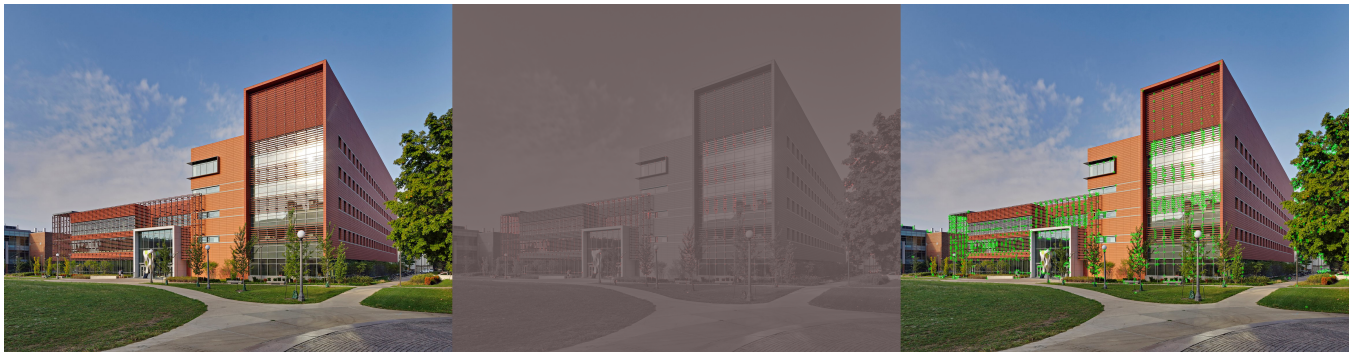

Figure 6: ECE Building

2.5 Bells and Whistles

During the early experiment, the author noticed that there are false positive predictions near each of the correct predictions. It is suspected that both the detected corners have relatively high gradients resulting double prediction for one corner.

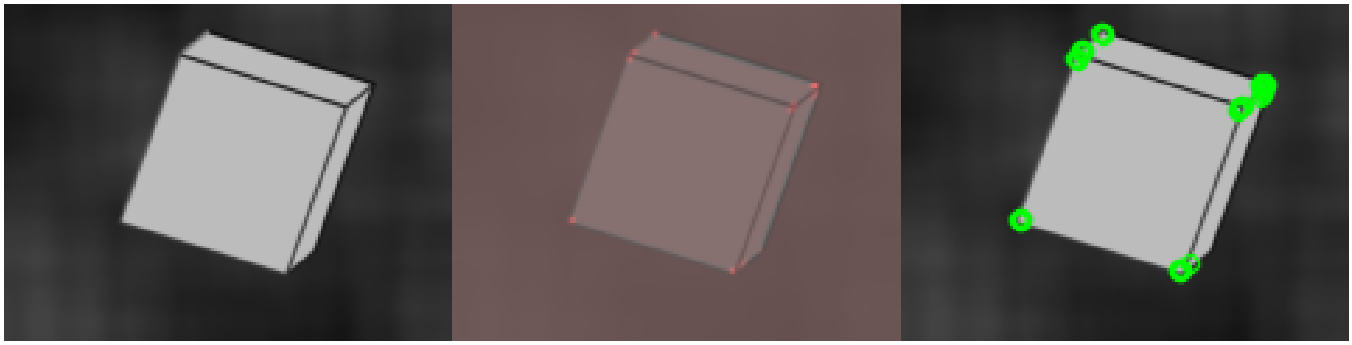

Figure 7: False Positive

The solution to this phenomena is by using a window post-process the gradient output to ensure there can only be one entry larger than threshold in a sliding window. Ensuring each of the corners can only be detected once.

3 Blend

3.1 Method Description

The Gaussian Pyramid based Blending requires two pyramids:

- First, Generate **Gaussian Pyramids** for both images and the mask. The Gaussian pyramid reduces the image size by half at each level, effectively capturing the image's coarse to fine details.
- With Gaussian Pyramids, the **Laplacian pyramid** is created by subtracting the expanded version of the Gaussian pyramid level from its previous level, capturing the image's detail at various scales.

By element-wise multiplication of the mask with one image's Laplacian pyramid and the inverse of the mask with the other image's Laplacian pyramid, followed by summing these results. We can reconstruct the final image by collapsing the blended Laplacian pyramid, progressively adding the details of different layers back. The submitted solution is a 5-Layer Gaussian/Laplacian Pyramid Blending code.

3.2 Oraple

3.3 Other Blends

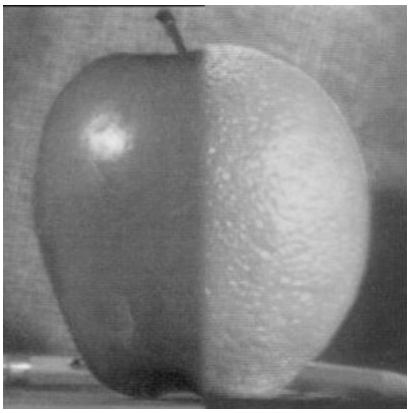

Figure 8: Oraple

Figure 9: Side eye Cat Blend

Figure 10: Huh cat