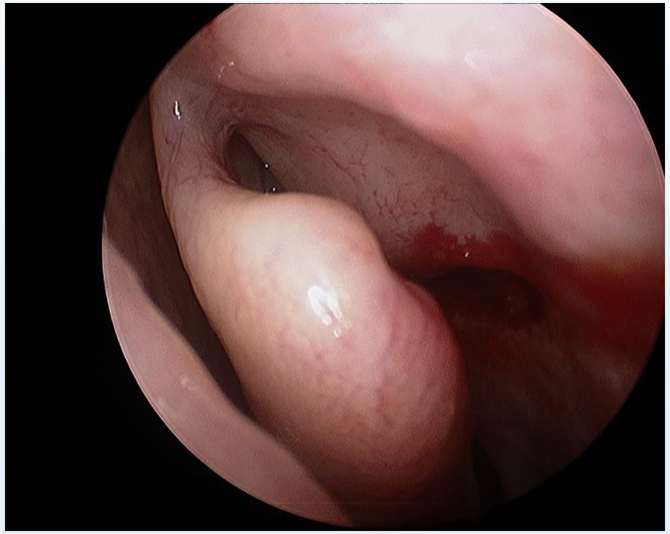
Applying New Edge Detection Methods to Sinus Surgery Image Processing

Kyle Xiong

Group 11 Seminar Presentation

Project Overview

- Using computer vision algorithms to detect occluding contours in sinus surgery videos to facilitate tool tip positioning with magnetic trackers.
- Using detected occluding contours from endoscope video, and registering them to CT data, we can accurately track the camera in the body.



Problem Summary

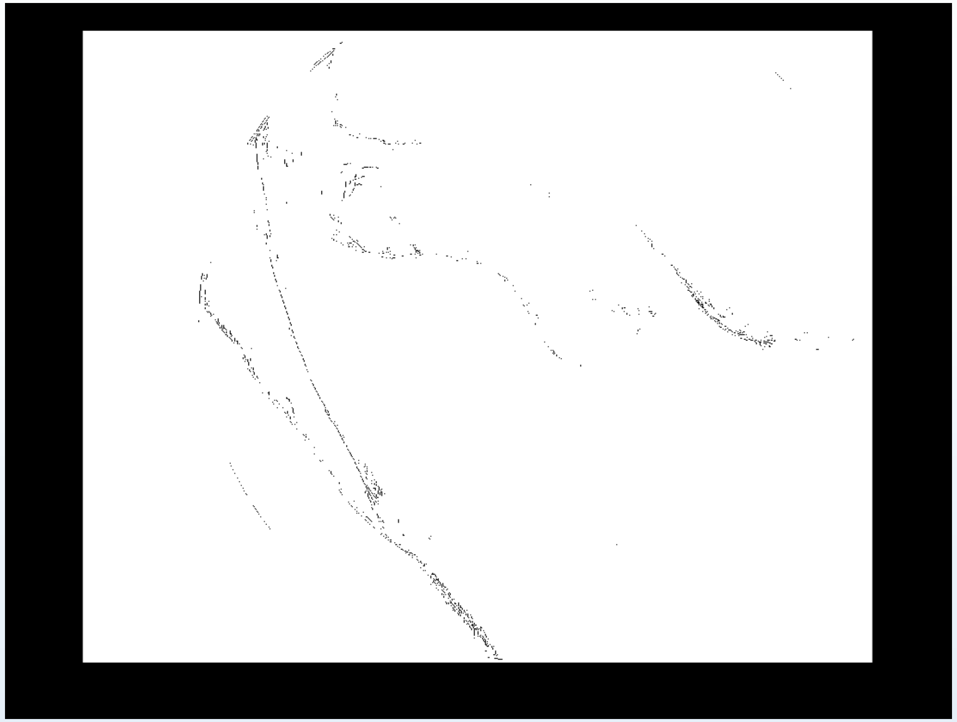
- Current limitations: efficiency and accuracy
  - Existing method by a group at Berkeley is accurate but slow
  - Faster methods tend to be inaccurate or return too much noise.
- The paper aims to accurately *and* efficiently detect edges from images by using new criteria – a predetermined set of edges classifications and few intensity comparisons and calculations for each pixel.
- Edges inherently have higher contrast, so it's possible to distinguish non-edge points from edge points by detecting a sharp change in pixel intensity.

Problem Summary

- Current attempted methods include:
  - Canny edge detection
  - Sobel edge detection
  - Horn-Schunk optical flow
  - Lucas-Kanade optical flow
  - Smoothness filtering
  - Intensity filtering
- Some combinations of the above methods have worked reasonably well, but not optimal yet.

Expected Results

- Contour detection similar to the figure below:
- Key edges are detected with little noise.
- Figure generated with static parameters and not as accurate in other frames.
- Need to increase accuracy and find a dynamic method of contour detection applicable to all frames.
- Upon completion, accurate edge locations will be used in video-CT registration algorithm.

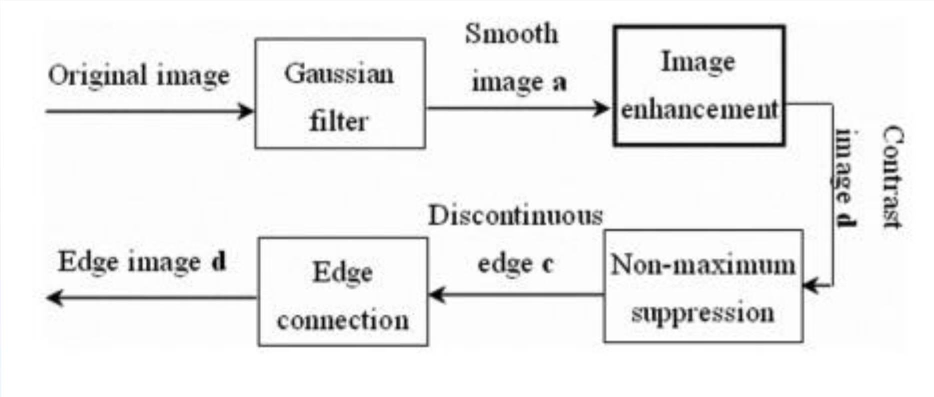


Paper Selection

- Z Li, Y Liu, J Yun, F Huang, “A New Edge Detection Method Based on Contrast Enhancement,” IEEE, December 2009.
- This paper introduces the contrast enhancement edge detection method (CEED).
- Relevant to contrast profiling method suggested by Balazs and Dr. Reiter, which uses the different intensity values at edges to detect edges and distinguish between edges and texture.

Method Overview

- Contrast is evaluated for each pixel according to intensity of surrounding pixels.
- Looks for a neighborhood of pixels around each pixel and stores contrast for each pixel in a matrix.
- Edges can be detected through threshold computation.
- CEED workflow:



Workflow – Gaussian Filtering

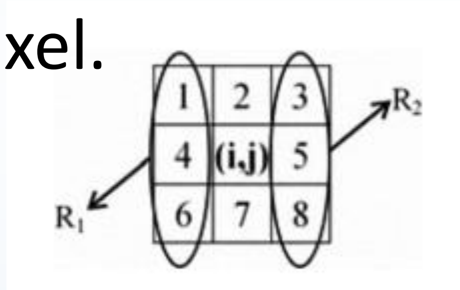
- Gaussian filter is used to blur the image, important in removing textures that could be misinterpreted as edges from the image.
- By convoluting the original image and a Gaussian filter, we can obtain a smooth image:

$$S(i,j) = G(i,j) * I(i,j)$$

- Using the preprocessed image the authors continued on to contrast calculations.

Workflow – Contrast Calculation

- Used a M x N grayscale image $I(i, j)$ with values within [0, 255].
- For each pixel in I, find the adjacent eight pixels and let $W(i, j)$ be a 3x3 window including the eight pixels and the original pixel.
- Figure to the right assumes $\{2, (i, j), 7\}$ as its edge.
- Separates into two regions: $R_1(1,4,6), R_2(3,5,8)$.
- Contrast function:



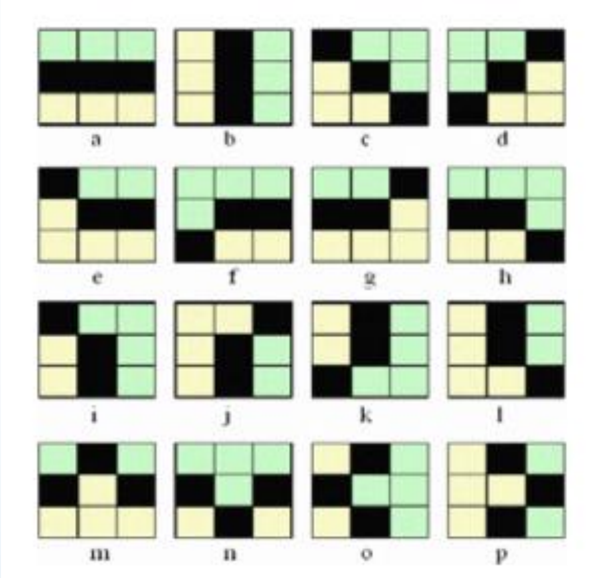
$$f(R_1, R_2) = \max\left(\frac{R_1}{R_2}, \frac{R_2}{R_1}\right)/3$$

Contrast Calculation Math

- Let S_1, S_2 be the average intensities in regions 1 and 2, respectively.
- $S_1 = \frac{I(i-1,j-1)+I(i,j-1)+I(i+1,j-1)}{3}$
- $S_2 = \frac{I(i-1,j+1)+I(i,j+1)+I(i+1,j+1)}{3}$
- $R_1 = \sqrt{(I(i-1,j-1)-S_1)^2 + (I(i,j-1)-S_1)^2 + (I(i+1,j-1)-S_1)^2}$
- $R_2 = \sqrt{(I(i-1,j+1)-S_2)^2 + (I(i,j+1)-S_2)^2 + (I(i+1,j+1)-S_2)^2}$

Contrast Calculation

- The paper introduces sixteen candidates that an edge can be categorized as:

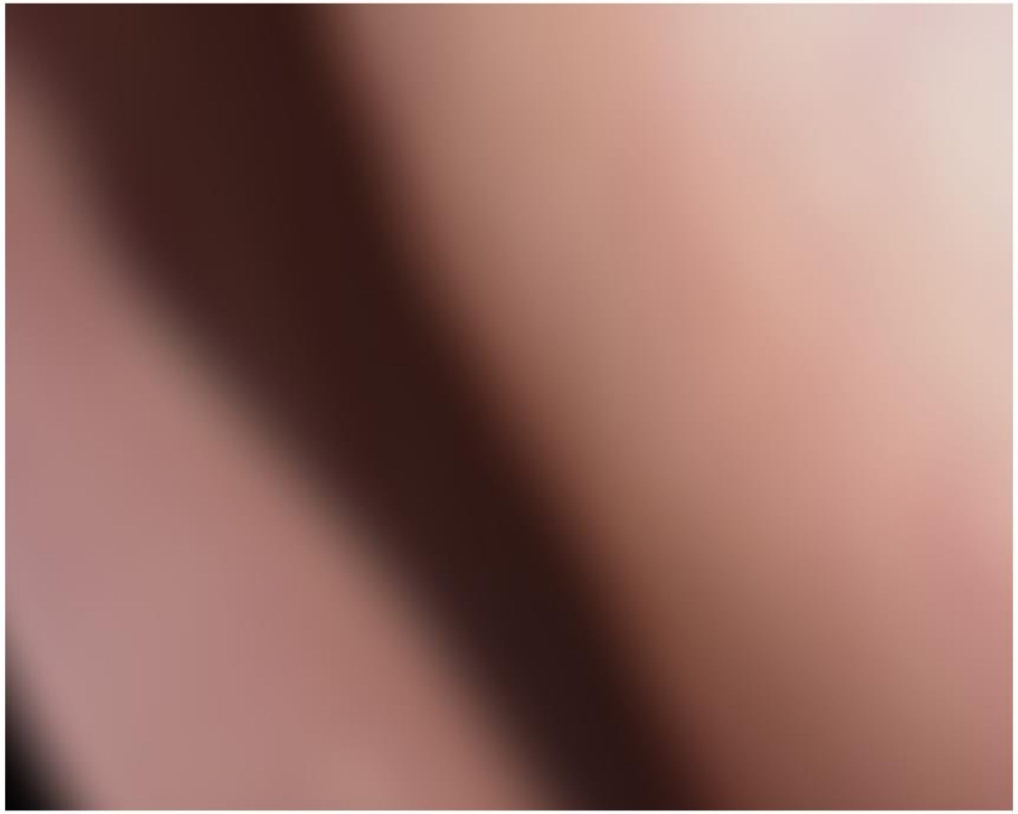


- The method compares each edge structure to the window W and the structure that yields the maximum $f(R_1, R_2)$ value is selected.

Workflow – Non-maximum Supression

- Eliminates any false edges by comparing calculated contrast with adjacent windows shifted one pixel vertically and horizontally.
- If contrast isn't higher, then detected edges are removed and the method remembers the contrast of adjacent windows.

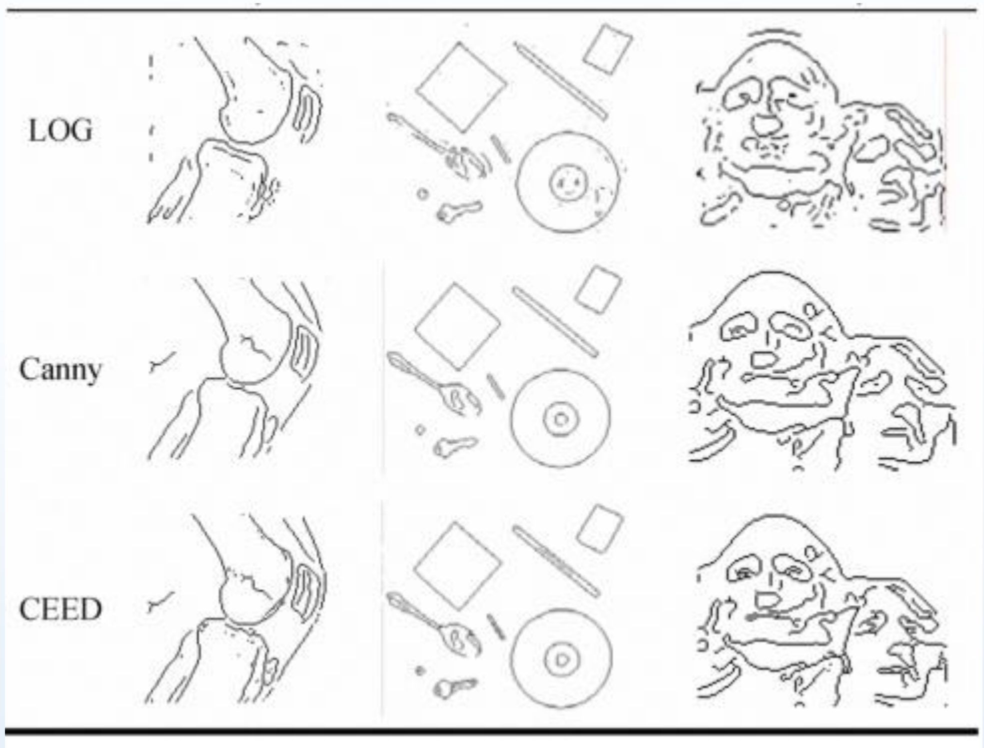
Non-maximum Supression



Workflow – Edge Connection

- Double threshold method to select and connect discontinuous edges.
- Set two thresholds δ_1 (low) and δ_2 (high).
- Compare intensities in all detected edges to these thresholds to produce edge images E_1 and E_2.
- The method first connects edges in E_2, then searches for connecting points in E_1, the less filtered image.
- Authors did not explain why they did this. Possibly to better filter out false edges. E_2 only contains very high contrast edges and E_1 serves to interpolate edges onto E_2.

Results



Evaluation

- Useful paper in introducing a rigorous edge detection technique.
  - Can eliminate false edges through non-maximum suppression.
  - Can accurately return true edges through use of a lower and higher intensity threshold when connecting edges.
  - Relatively fast compared to our current method – fewer comparisons and stores, and less image processing.

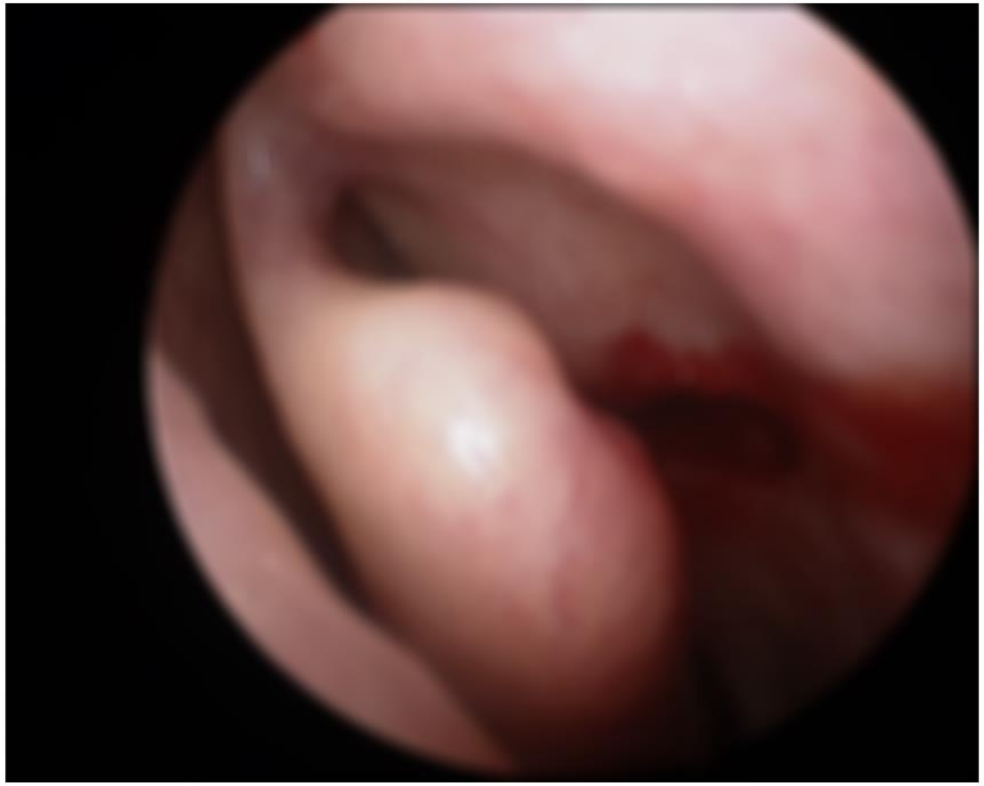
Possible Problems

- Gaussian blur inaccuracies

- Comparing the unprocessed and processed image, it’s easy to see how a 3x3 window size may detect a contrast peak away from a true edge.
- However, the method worked for the authors and we’re eager to implement it to see its effectiveness with our sample data.

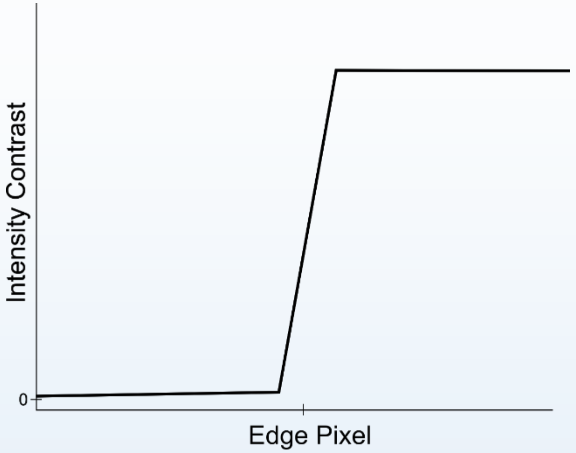
Possible Problems

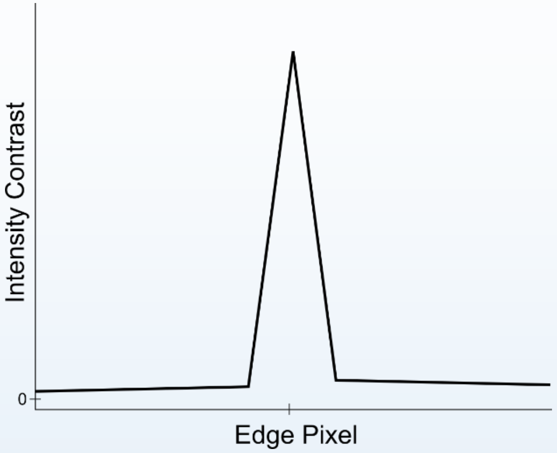
- Even after applying Gaussian blur, we see textures with contrast.
- Like Canny edge detection, we could still see too much noise from this method.



Next Steps

- Use the math and general shapes of edges in contrast-based edge detection.
- Contrast profiling that measures consistency of intensity across edges.
- Intensity plateaus vs. intensity peaks/valleys:





- Currently considering machine learning – hand labeling edges and using SVMs to help differentiate between a texture and an edge.