Detection of vanishing points in unstructured images

Chris Willcocks 10142939
Plymouth University, Plymouth, UK

Abstract

An algorithm was developed detect vanishing points within unstructured images. Edges were detected via the Canny filter, lines were extracted using the Hough lines transform and the crossing points were calculated as line crossings. Efforts were made for automated thresholding, with some success. Testing was done to establish the error between actual and calculated vanishing points for a set of images. Finally, observations are made for potential causes of errors, and potential solutions put forward for improved performance.
Keywords: Canny, Hough, vanishing point, edge detection

I Introduction

This report details efforts made to develop a program for vanishing point detection. A brief introduction to the background and history of perspective is presented. The main body of this report details the steps taken to detect a vanishing point: automated edge detection, efforts at automated line extraction, calculation of crossing points of convergence lines, and finally targeting the estimated vanishing point on the source image. The results are detailed and discussed, with a number of recommendations are made for algorithm improvement.

II Background

Prior to the 15th century and the renaissance, most artistic depictions tended to look skewed. This was due to the lack of understanding of perspective, which was not used extensively until Filippo Di Ser Brunellesco (Carter, 1998). Guidobaldo del Monte, building on the work of Alberti (Andersen, 2007), published the theorem of perspective in 1600 in his book *Perspectivae libri sex*, which is stated as:

"The image l_i of any line l that is not parallel to the picture plane is determined by the vanishing point V_l and the intersection $I_{r''}$"

The proofs that Guidobaldo details show that all straight geometry converges towards one or more vanishing points, which as a repeatable result has a number of potential applications.

Currently, a number of approaches to finding vanishing points have been identified. (Nieto, 2011), has extracted vanishing points in road scenes, for "camera calibration, perform … planar homography transformation (and) determine ROI inside the image".

The approach of this project is to retrieve a frame of an image, detect all the edges within said image, find the convergence lines, and detect the vanishing points as intersection of those convergence lines.

III Method

Development on a test image was identified as the most effective approach. A video feed would provide additional complexity, which would provide a greater range of opportunities for testing, but would also present many more challenges for a non-robust system. Because of this, a static jpeg image was chosen as the test data. As the final program is expected to work in the real world, an image with a clearly definable vanishing point was desired, without the image being overly structured, ie. a set of cubes in a laboratory setting.

To fulfil this specification, two images were chosen; one image of a set of train tracks, and another of a street in New York. Plenty of noise was present in the form of foliage, background hills, pedestrians and clouds with easily definable straight lines in the form of the train tracks, roads and buildings, and a readily definable vanishing point roughly central in the image, making any issues with the image processing much easier to spot.

The test image was loaded into a Matrix object, and converted to greyscale. The additional colour channels were deemed unnecessary for edge detection, and would simply have contributed to greater processing times. Additionally, the edge detection algorithms available only work on a single colour channel at a time.

As the Hough lines transform is dependent on the number of pixels in a line, and therefore dependent on image resolution, every test image was scaled to a width of 1000 pixels, maintaining aspect ratio before edge detection. This is performed using the following code:

```
float width = image.cols;
float scaleFactor = 1000 / width    //1000 can be set to any pixel value
cv:resize(image, image, cv:Size(), scaleFactor, scaleFactor, CV_INTER_LINEAR);
```

(Nieto, 2011) makes use of a custom edge detection algorithm, which the author reports good results with. However, this algorithm relies on specifying the width of

the edges (in this case lane markings) in pixels. As this program is intended to work in a variety of environments where the width of lane markings may not be known, or where lane markings may not be present, a Canny edge detector was chosen.
Adapting the example code from the OpenCV documentation, slider bars were used to set the threshold values for the Canny filter. After good edge detection was achieved (maximising the possible vanishing points ie. train tracks) the threshold values were hard-coded into the program.
The Hough lines filter was chosen to detect straight lines from the resulting edge detection, owing to its simplicity of use and integration, and good reports of functionality. The existing OpenCV examples were adapted to detect what should now be the convergence lines in the test image.

IV Automatic threshholding

At this point, the constant necessary tuning of parameter values for the Canny filter and Hough filter was found to be time consuming, and repeatability due to human error was not acceptable. An approach using the mean value of the grey channel was attempted, with some success.
(Wong, 2009) reports good results taking the median grey value of the imported image and setting the Canny threshold values to be one standard deviation from the average. Building on this work, the median value of the grey channel was calculated, the thresholds set to one standard deviation away, and a Hough line transform applied.
For both test images, convergence lines were found, although some manual tuning was still necessary to modify the standard deviation. Using the median over the mean did not result in substantial improvement for edge detection, however, convergence lines were reliably detected with minimal tuning effort.
An attempt was also made at automating the Hough lines threshold. As the only Hough threshold parameter is the number of votes in rho/theta space, and any convergence point should theoretically be somewhere close to the centre of the image, the threshold was set to half the diagonal size of the image. This did reduce the tuning required, however a divisor is still found to be necessary to produce acceptable Hough lines. In general, this divisor must be larger for larger images. Although a different approach is required for more robust line detection, this approach does make tuning the Hough lines threshold more intuitive.

IV Filtering and crossing points

With the Canny filter and Hough filter applied, convergence lines were successfully detected, however many false positives were also present. Investigating the train tracks test image, a lot of false positives were caused by the railway sleepers, resulting in horizontal lines. Theoretically, these horizontal lines would also converge on the horizon, a long way outside the image window, which would be used with a more developed, robust program. However, at present these horizontal lines would present an issue detecting the solitary vanishing point in the test image. The second test image had a similar issue; the zebra crossing was being detected as having vertical lines.
As the program is intended to detect the vanishing point within the image, only diagonal lines were desired. To achieve this, after the Hough filter, the lines were checked for orientation. To achieve this, the rho/theta space was divided into four quadrants of 0 – 90, 90 – 180, 180 – 270 and 270 – 360. Any lines with a theta value of 0, 90, 180, 270 or 360, plus or minus a threshold value, were rejected, using the code below:

```
if ((theta1 > (0 + theta_thresh)) && (theta1 < CV_PI/2 – theta_thresh)) ||
      ((theta1 > (CV_PI/2 + theta_thresh)) && (theta1 < CV_PI – theta_thresh)))
```

This effectively removed any lines not deemed diagonal enough.
Now that the correct convergence lines have been detected and isolated from the image, the convergence points would be calculated. Every line should pass through the convergence point as a single X/Y coordinate, however with noise and errors during detections some error is expected. Since the error cannot be known ahead of time, the crossing of every pair of lines must be evaluated.
As the Hough lines filter outputs a rho and theta value, this was converted into an x/y intersection coordinate by the following:

```
float determinative = (costheta1 * sintheta2) – (sintheta1 * costheta2);
intersection.x = ((sintheta2*rho) – (sintheta1 * rho2)) / determinative;
intersection.y = ((-costheta2 * rho1) + (costheta1*rho2)) / determinative;
```

This was calculated for every pair combination of lines. These intersections were then drawn on the test image as cross-hairs, using the drawMarker function.
The crossing points were found to primarily cluster roughly in the centre of the image, however many false positives were also found along the length of the detected convergence lines. Upon further investigation, the cause was found to be the lines overlaying each other, which the program detected as a crossing. As any lines lying over each other would have similar gradients, in this case theta values, a comparison of theta values was performed before calculating intersection. Any two theta values with insufficient difference were then rejected.
With all these steps completed, the program was found to successfully detect the vanishing point on the train tracks image. This is shown in figure 1.

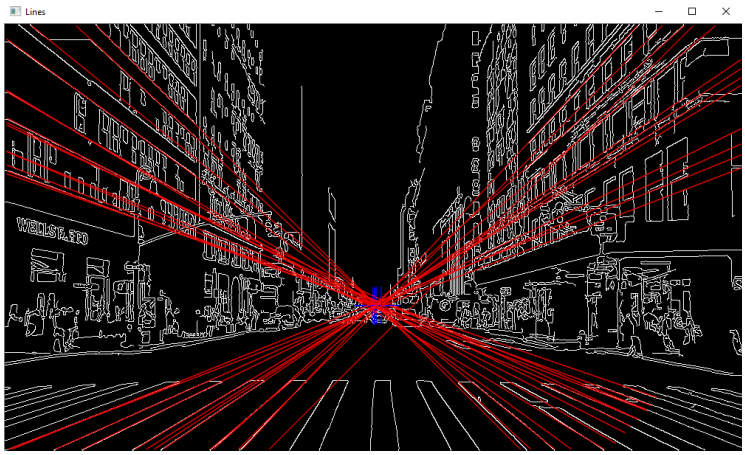


Figure 1 - nystreet vanishing point

A flowchart of the complete algorithm is shown in appendix 1.

V Testing

Based on the success of automatically setting Canny values with image parameters, the Hough lines threshold was set using the size of the image. However, the threshold value still needed tuning for each individual image.

The goal of the testing procedure was to establish the algorithm's error when detecting vanishing points.

To achieve this, three test images were selected with readily definable vanishing points. As no tools were available to automatically define a vanishing point, each vanishing point coordinate was manually selected.

The Hough line divisor was varied across each image, and the number of crossing points were logged. The median of these values was taken as the detected vanishing point, and the difference between detected and manually selected points were calculated. The results are shown in figures 1, 2 and 3.

VI Discussion

The vanishing point detection error is found to increase as more crossings are detected, which implies the rejection algorithm is not robust to scaling.

Preliminary testing found that the Canny filter parameters were highly sensitive to the content of the image, and required time consuming manual tuning. To resolve this, automatic threshholding based on the median grey scale and standard deviation was found to produce reasonable results, although issues with bright regions in the image causes some edges to not be detected.

The current testing method relies on manually specifying a vanishing point, which introduces additional uncertainty. A more thorough future test could be performed as follows:

For a given camera type and lens, the vanishing point could be calculated. A precision test-bed can be used to place the camera such that the vanishing point lies on a specific pixel, and elements of controlled geometry, such as precision ground granite cubes, could be placed to produce a structured scene. This would have the benefit of removing the human error of manually specifying a vanishing point from a photo, and define a true, repeatable vanishing point.

The current program needs to reject lines that do not converge; ie. horizontal and vertical lines. This has the drawback that an image must contain a horizon perpendicular to the top and bottom of the image, preventing use with tilted images

Detecting the crossing points of lines currently produces many false positives, increasing proportionally with the number of detected lines. A better rejection technique than angle between lines is needed. A better approach may be to set a parameter proportional the camera lens, which would define the focal point, and the vanishing point.

The automatic threshholding of the Canny filter produces a large number of edges without manual tuning, but does not work for all situations, ie. heavy "blooming" in bright images. Possible solutions could include the use of other edge detectors.

The Hough filter currently detects many false positives, particularly amongst dense, clustered lines, such as with foliage. A better method to set the threshold parameters is necessary.

During testing, the Hough lines threshold was found to require tuning for different size and aspect ratios of images, which was expected. Rescaling each image, the Hough filter threshold still required tuning, and the cause was determined to be the geometry of each image, particularly the length of lines within the image. As the geometry of the image, which cannot be known ahead of time, must be accounted for when setting the Hough lines parameters, some other method to establish suitable line length must be applied. From the test results, there likely exists some minimum Hough threshold function which produces the most adequate results.

Finally, the assumption for this automatic thresholding is that the vanishing point will lie somewhere close to the centre of the image; as this cannot be relied on, this approach often fails.

However, despite these issues, this project shows potential for detecting vanishing points.

As shown by Nieto (2009) vanishing points have applications with self-driving cars and other autonomous vehicles. This function could be extended for autonomous or semi-autonomous path finding behaviour and obstacle avoidance, by having a robot navigate towards detect vanishing points.

Zhou et al (2011) and Cipolla et al. (1999) have also successfully used vanishing points for automatic calibration of cameras, showing that this can be a robust and effective technique, with the possibility of establishing all of a camera's intrinsic parameters.

Other uses for a successful algorithm could include horizon estimation without line of sight, and pitch/roll calculation.

VII Conclusions

An algorithm using a Canny edge filter and Hough transform were used to detect perspective lines in real world images.

Automatic threshholding was implemented based on existing literature.

Rejection of detected lines was performed based on angle of the perspective lines, and the intersection of the remaining lines was calculated.

Finally, the crossing points were marked on the original image.

Improvements in performance for the Hough lines transform were identified as necessary, as were better rejection algorithms to improve image handling.

References

Carter, P. (1998). Geometry in Art and Architecture Unit 11. [online] Available at: https://www.dartmouth.edu/~matc/math5.geometry/unit11/unit11.html [Accessed 17th March 2018]

Andersen, K. (2007). *The Geometry of An Art*. 1st ed. [ebook] Copenhagen: University of Copenhagen. Pp245-248. Available at: https://books.google.co.uk/books?id=8B_JeMxNUIkC&pg=PA245&lpg=PA245&dq=vanishing+point+theorem&source=bl&ots=IqUy5GfP8z&sig=J8ElEdZGflC14_lwo3FTLjWts-o&hl=en&sa=X&ved=0ahUKEwiIhur7xLnaAhWkL8AKHab0AfoQ6AEIygEwGw#v=onepage&q=vanishing%20point%20theorem&f=false [Accessed 17th march 2018]

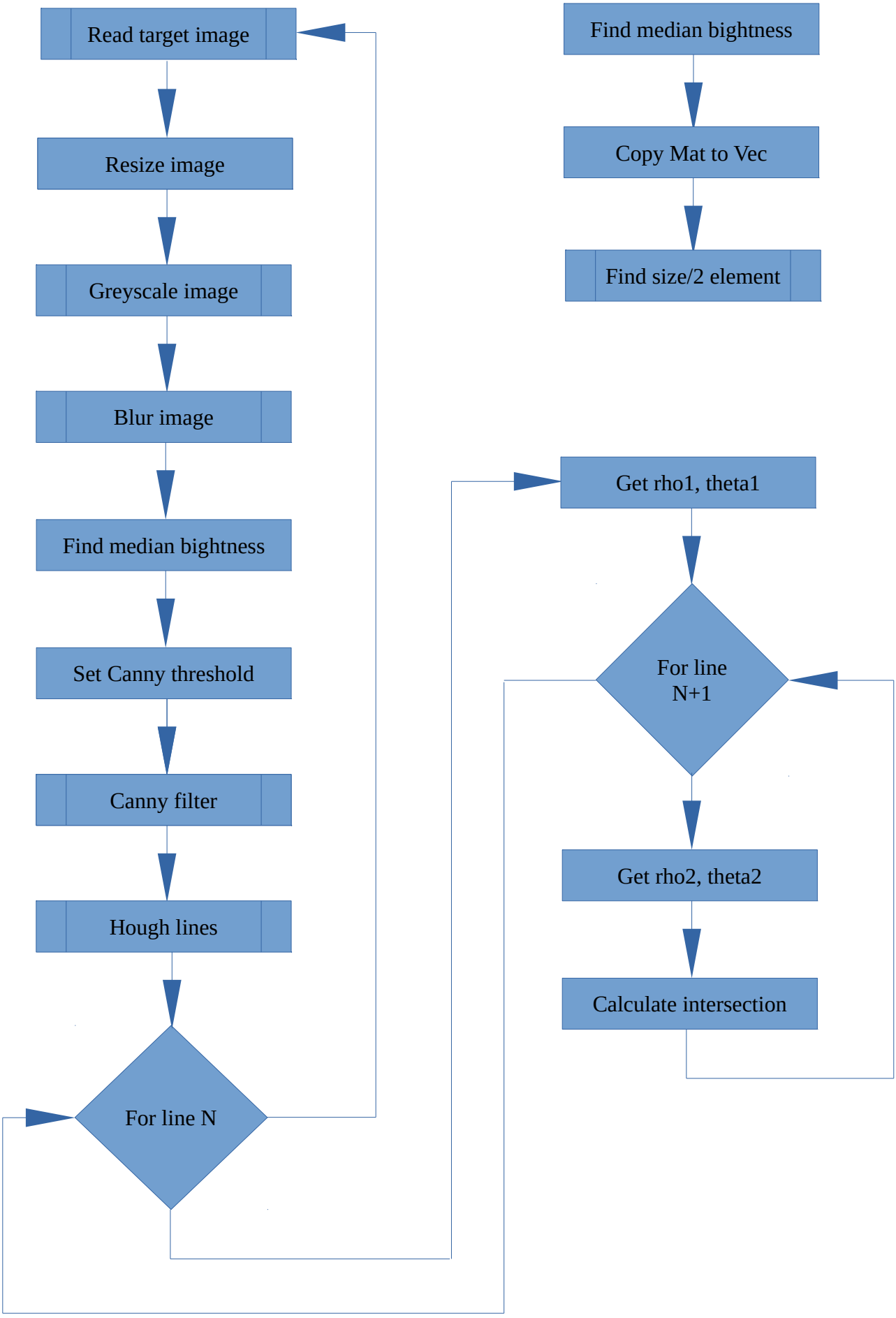
Nieto, M. (2011). *Lane markings detection and vanishing point detection with OpenCV*. [Blog] marcosnietoblog. Available at: https://marcosnietoblog.wordpress.com/2011/12/27/lane-markings-detection-and-vanishing-point-detection-with-opencv/ [Accessed 22nd March 2018]

Wong, K. (2009). Canny Edge Detection Auto Thresholding. [online] Available at: http://www.kerrywong.com/2009/05/07/canny-edge-detection-auto-thresholding/ [Accessed March 30th 2018]

He, B.W, Zhou, X, L, Li, Y, F. (2011). A new camera calibration method from vanishing points in a vision system. In: *Transaction of the Institute of Measurement and Control*. [online] Hong Kong: City University of Hong Kong, pp.806-822. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.920.2887&rep=rep1&type=pdf [Accessed 3rd April 2018]

Cipolla, R, Drummond, T, Robertson, D. (1999) camera calibration from vanishing points in images of architectural scenes. In: *The tenth British Machine Vision Conference*. [online] Cambridge, University of Cambridge. Available at: https://pdfs.semanticscholar.org/7782/2ed97da9f8d7deb59aafb869c8234803f05a.pdf [Accessed 8th April 2018]

Appendix 1


Read target image
Resize image
Greyscale image
Blur image
Find median bightness
Set Canny threshold
Canny filter
Hough lines
For line N
Find median bightness
Copy Mat to Vec
Find size/2 element
Get rho1, theta1
For line N+1
Get rho2, theta2
Calculate intersection

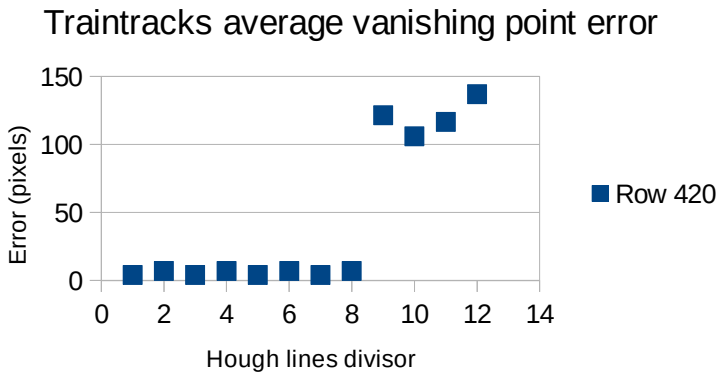


Figure 2 - Traintracks testing

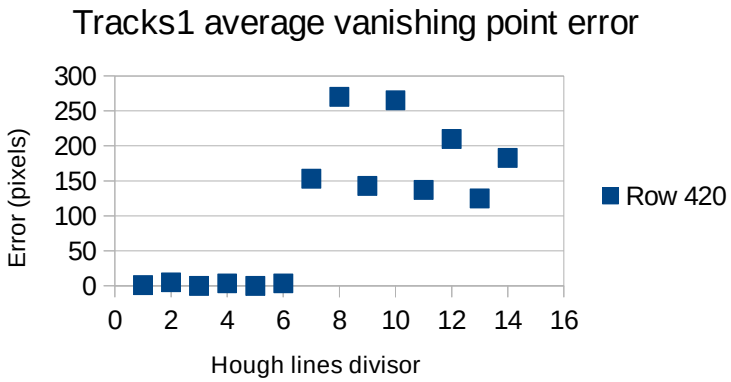


Figure 3 - Tracks1 testing

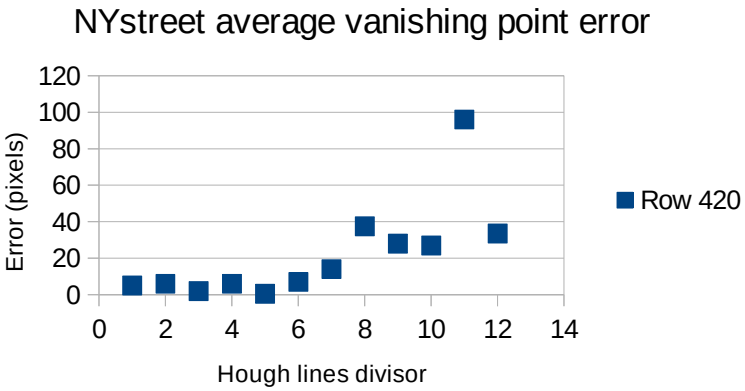


Figure 4 - NY Street testing