Algorithms and Applications in Computer Vision 046746

Homework #1

Vladimir Tchuiev, 309206795,
vovatch@campus.technion.ac.il

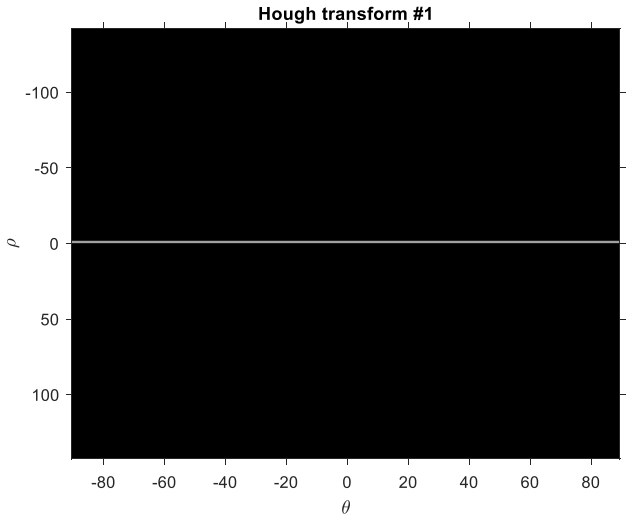
Alexander Shender 328626114,
aka.sova@gmail.com

Task 1 – Hough Transform

(a)

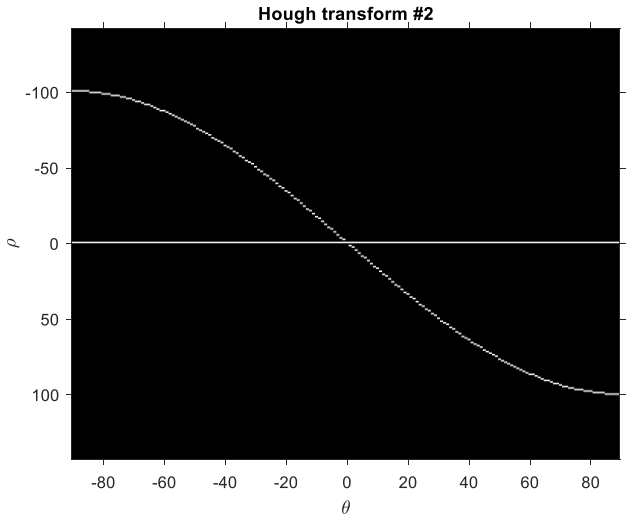
In order to make the explanation simpler and more understandable, we will expand the 'imshow' function, as in the MATLAB example for the Hough function. Thus, we obtain:

a. Black image with dot in LEFT TOP corner returns the Hough map:



As we can understand, the point is placed on the origin of the place, from which all the values are being calculated. For all the θ values, we obtain than the line, which originates in the top left corner and is perpendicular to any line crossing the (1,1) dot has a length of 0.

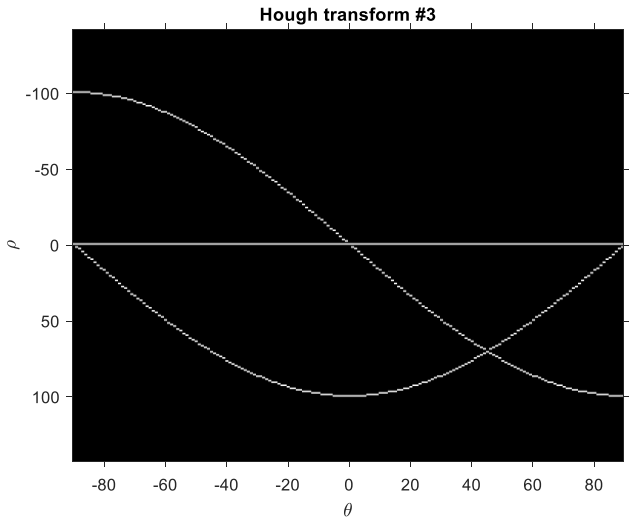
b. Black image with dots in (1,1) and (101,1) results in a Hough Map:



From this image we can see that there is 1 line which connects those 2 dots, and this line has a parameters: $\begin{cases}\rho = 0 \\ \theta = 0\end{cases}$; which gives us an understanding as to where the 0 angle

of θ is placed – it is the angle to the line, which goes from the origin (1,1) straight down. The graph is indeed correct, since the line, which has those ρ, θ parameters, passes through the (101,1) point, and is perpendicular to the origin.

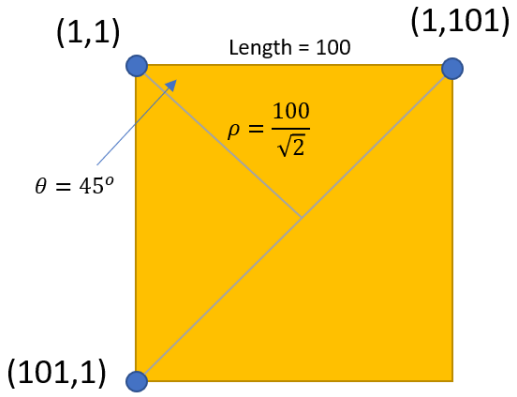
c. Adding the third dot at (1,101) gives us



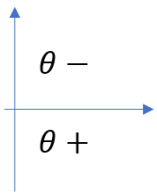
We can observe 2 things:

1. Is has intersection with a Hough line of the second dot precisely at: $\begin{cases} \rho = \frac{100}{\sqrt{2}} \\ \theta = 45^o \end{cases}$

   Visualizing it:

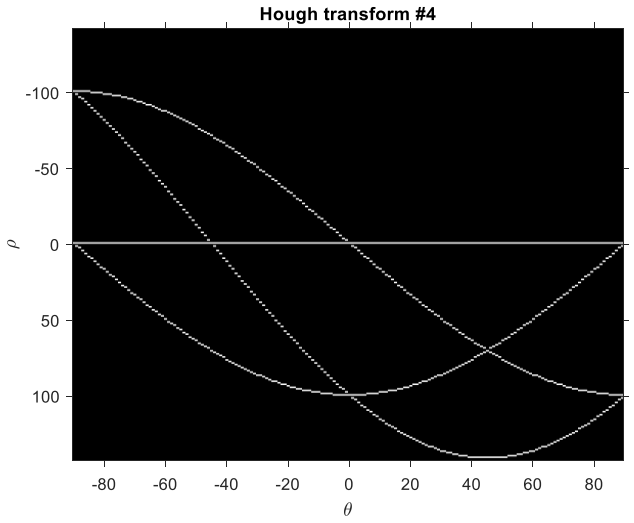


   Now we also know the θ positive direction:



2. We can see 2 intersections with the straight line (which corresponds to the (1,1) dot), at parameters: $\begin{cases} \rho = 0 \\ \theta = 90^o \end{cases}$ and $\begin{cases} \rho = 0 \\ \theta = -90^o \end{cases}$; Those 2 sets of parameters describe the same line which passes through both of the points (which connects (1,1) and (1,101)).

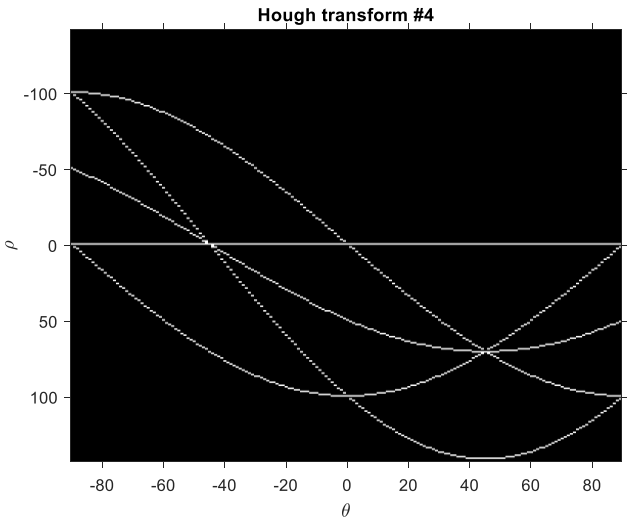
d. Adding the dot at (101,101) gives us:



We can identify the line by its biggest ρ value of $100 \cdot \frac{2}{\sqrt{2}} \approx 141.4$, since it is the largest distance from the origin (1,1). We can observe 4 intersections, describing 3 lines which connect dots:

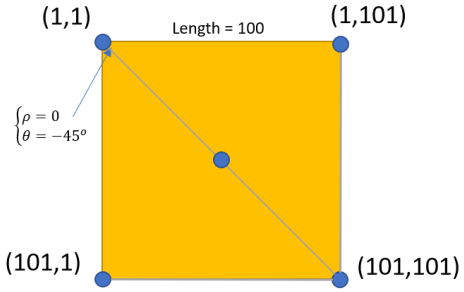
1. With (1,1) : intersection at $\begin{cases} \rho = 0 \\ \theta = -45^o \end{cases}$ - ρ is 0 because the line crossing the origin has a distance of 0 from the origin. θ is negative 45 degrees because the perpendicular line will be placed in the upper plane (the region, outside of the borders of the image, to the top, where the θ values are negative).
2. With (101,1): intersection a$\begin{cases} \rho = -100 \\ \theta = -90^o \end{cases}$t and intersection at $\begin{cases} \rho = 100 \\ \theta = 90^o \end{cases}$ - it has 2 intersections, since the loop goes over θ: $[-90\ 90]$ degreees. So, both first and the last line drawn will correspond to the same line (in every case). Since, in our case, both dots (101,1) and (101,101) lie on that same line – they get 2 intersections!
3. With (1,101): intersection at $\begin{cases} \rho = 100 \\ \theta = 0 \end{cases}$ – the perpendicular line to the line, which connects those 2 points, is at $\theta = 0$ and has a length of 100.

e. Adding the last dot at (51,51) gives:



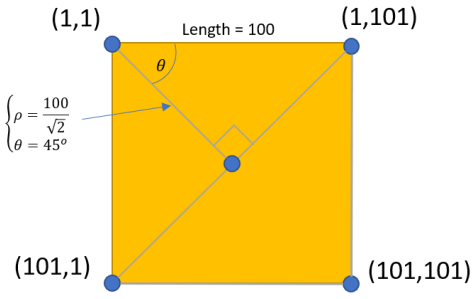
It is easy to identify the line, since its biggest ρ value is 50. It has 2 major intersections, or more precisely, each one contains 3 lines:

1. At parameters $\begin{cases} \rho = 0 \\ \theta = -45^o \end{cases}$ with dots (1,1), (101,101). Visualizing the case:



As we can see, indeed, the intersection with those parameters describes the line which connects those 3 dots.

2. At parameters $\begin{cases} \rho = \frac{100}{\sqrt{2}} \\ \theta = 45^o \end{cases}$ with dots (101,1), (1,101). Visualizing the case:

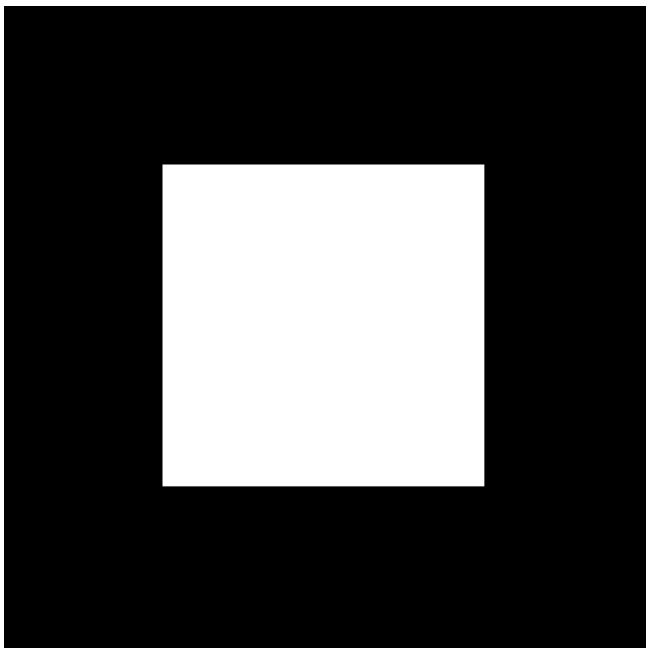


As we can see, indeed, the intersection with those parameters describes the line which connects those 3 dots.

(b)

Creating the binary image of a square (we used padding) results in:

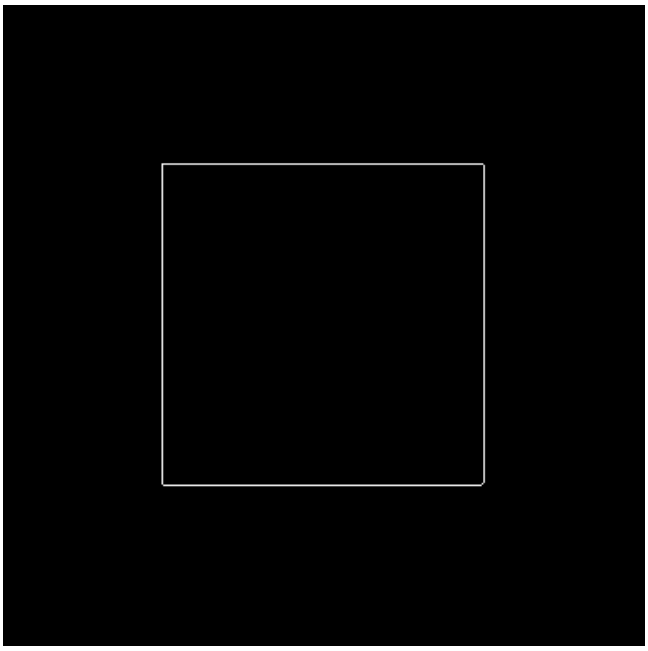
binary image of a square



(c)

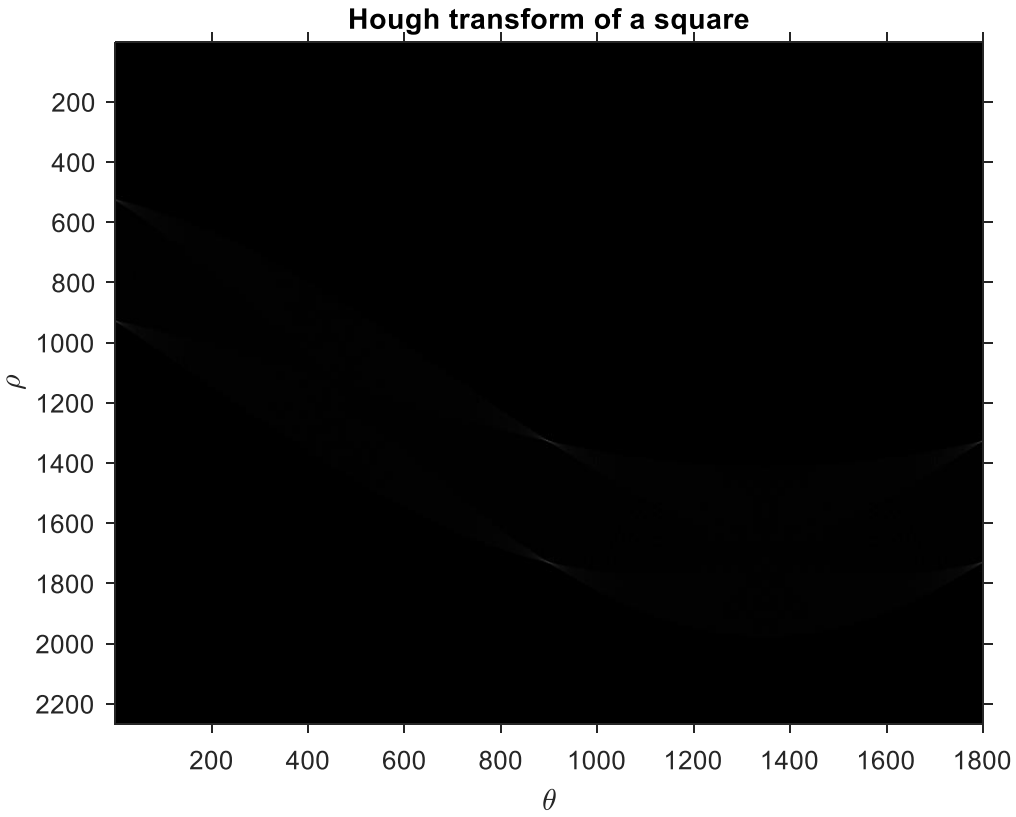
Finding the edges using the Canny edge detection results in:

Edges of a binary square

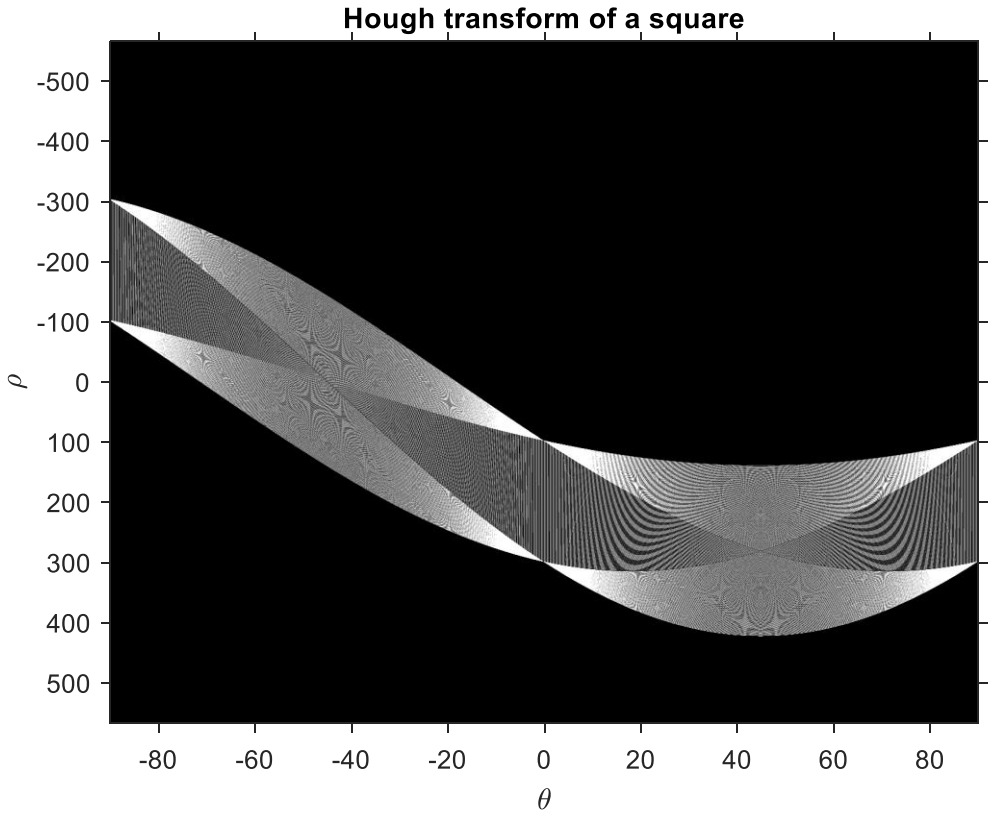


(d)

The Hough transform of the image looks in the following way:



Since we can hardly see anything here (any intersections) with a regular eye, we use the proposed MATLAB functions rescale & imadjust, which will increase the contrast of the image. The result is the following:



We can indeed spot 4 lines in the image, which correspond to the following parameters (intersections):

1. $\begin{cases}\rho = 100 \\ \theta = 0\end{cases}$, which is the left border of the square
2. $\begin{cases}\rho = 300 \\ \theta = 0\end{cases}$, which is the right border of the square
3. $\begin{cases}\rho = -100 \\ \theta = -90^0\end{cases}$ & $\begin{cases}\rho = 100 \\ \theta = 90^0\end{cases}$, which BOTH correspond to the upper border of the square
4. $\begin{cases}\rho = -300 \\ \theta = -90^0\end{cases}$ & $\begin{cases}\rho = 300 \\ \theta = 90^0\end{cases}$, which BOTH correspond to the lower border of the square

   Explanation on why lower and upper border create 2 intersections each: because the initial and the final line which are generated for any point on it is the same exact line, and is the line on which they lay! Same case was with the line in the (a) subsection of the question.

(e)

The lines are represented as the maximum point of the Hough transform. This is true, because it shows, that a certain line, describes by some certain parameters $[\rho, \theta]$, passes through many points. Which means that those points lay on the same line.

MATLAB provides the function *houghpeaks.* From the MATLAB helpfile:

`peaks = houghpeaks(H,numpeaks)` locates peaks in the Hough transform matrix, `H`, generated by the `hough` function. `numpeaks` specifies the maximum number of peaks to identify. The function returns `peaks` a matrix that holds the row and column coordinates of the peaks.

(f)

Using the function *hougheaks* on our image results in the following output (after specifying number of peaks > 4):

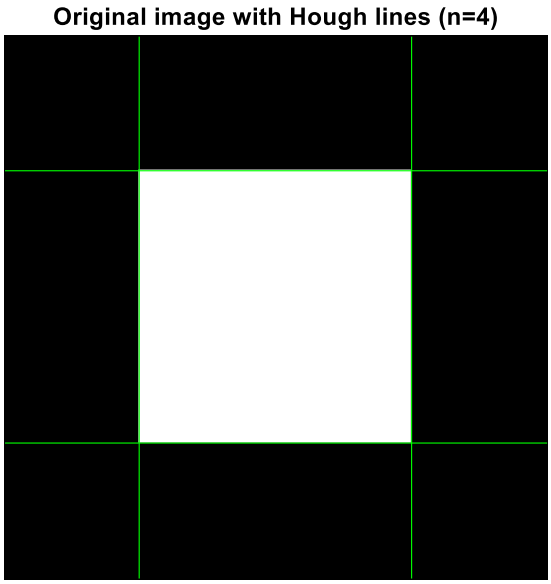
933	1
1333	901
531	1
1735	901

Those are the indexes of the $[\rho, \theta]$ values in the Hough matrix. Finding the exact values of those parameters through those indexes results in:

ρ	θ
-100	-90
100	0
-301	-90
301	0

(g)

Plotting the lines over the image results in:



We can clearly see that the reproduced lines exactly match the lines of the edges of the original figure (square).

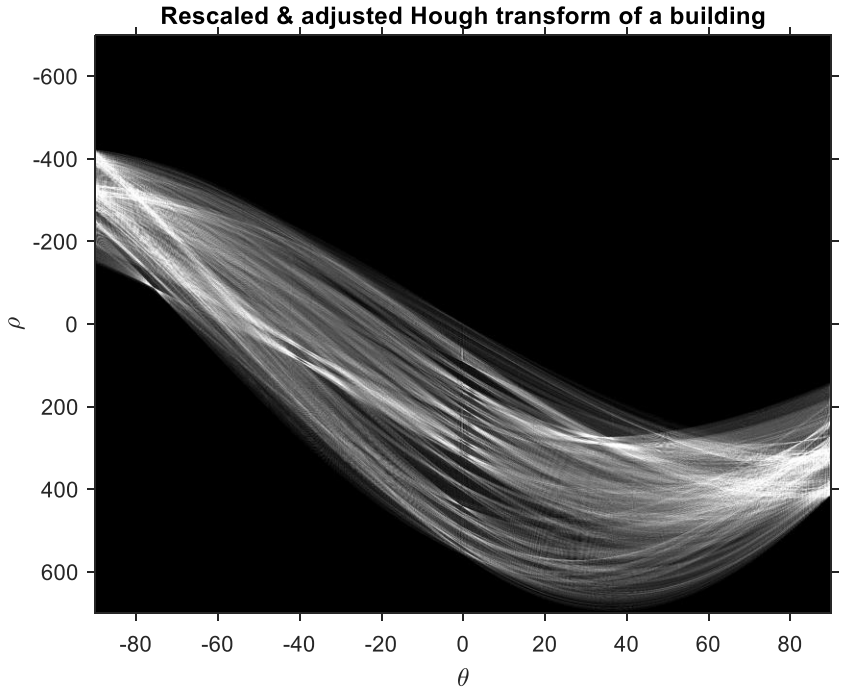
(h)

Reproducing the steps for the building.jpg:

1. Loading the image:

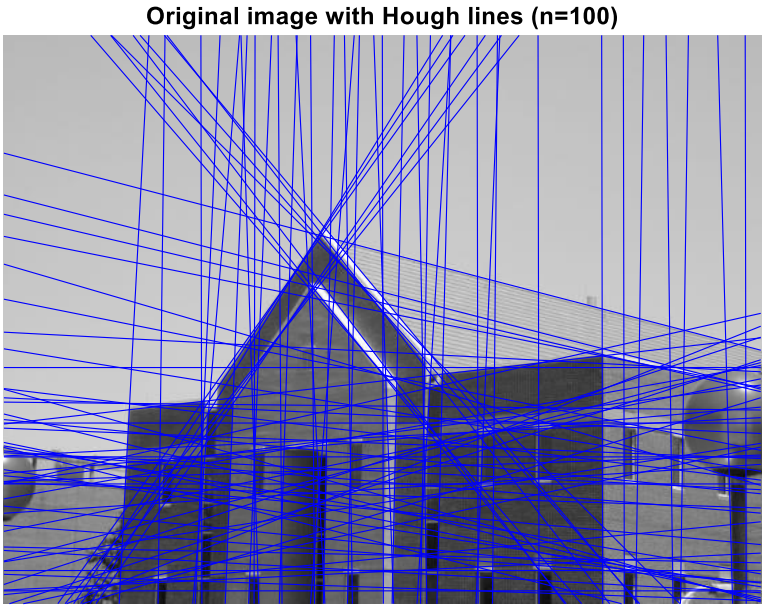
2. Adjusting the Canny edge detector (threshold parameters used: [0.05 0.1] to get the image of the lines:

3. Using the Hough transform to get the Hough map, which will look like this after the rescaling and contrast extension:



We can clearly see many edges here. Sanity check – many of the lines are parallel to the X axis or to the Y axis (it's a building!), so we have many intersections where θ value is close to 0 or 90 degrees.

4. Using the hough peaks to determine the values of the $[\rho, \theta]$ of the intersections. Setting a threshold to be high enough extracts only the most visible likes. In our case, we limit the number of lines to 100, and the results were found to be satisfying. Plotting those lines, like in (g), results in:



The results were found to be satisfying for us, only several False lines were found. Of course, it is possible to improve the performance by adjusting the Canny Edge detector (thresholds), and the parameters of the Hough Peaks function (threshold or maximum number of points).