



Vision estimation using dark channel prior and canny edge filtering methods

Author: Marcin Dudzik

Field of study: Electronics and Telecommunications

Subject: Design Laboratory

Supervisor: prof. dr hab. inż. Andrzej Dziech

Kraków, 2020

Table of contents

1. ABSTRACT	3
2. Dark Channel Prior method	4
2.1 Dark channel prior step by step	4
2.2 Assumptions	6
2.3 Atmospheric Scattering Model	6
2.4 Dark Channel	7
2.5 Estimation of atmospheric light	8
2.6 Transmission map	8
3. Canny Edge Detection method	9
3.1 Assumptions	9
3.2 Canny Edge Detection	9
3.3 Similarity of base and current image	10
4. Structure of program	11
5. Dark channel prior to estimate visibility	12
6. Canny edge detection as fog estimation method	18

1. ABSTRACT

Main goal of my project was to create an algorithm which can be used in order to estimate visibility which means fog and haze detection. Algorithms for haze and fog detection have been studied for many years, and countless papers were published. In this paper I will present two solutions I found very interesting and try to compare them.

Such solution can have a very wide scope of usage. It can be placed in some car systems to provide hints for driver about road conditions and how to behave in them. Some meteorology stations can use this type of algorithms to provide informations about state of weather.

Haze detection can find an application in monitoring of security in production halls when we want some early warnings before smoke sensors will detect any danger.

Air pollution affect the visibility of drivers on roads. When the weather is overcast, the sun's rays do not reach us to a large extent, which limits visibility and adds pollution, air visibility deteriorates significantly. By assessing visibility with traffic cameras we are able to send information in real time about whether driving a car can be done safely under normal conditions, whether speed should be limited and higher speeds should be used to estimate safety measures, which would significantly improve road safety and it would be possible to avoid many accidents.

With the rapid growing of image processing algorithms new methods to achieve our goal are developed and invented every day.

All over the network there are a lot of papers published about different approaches on haze detection on images and haze removal. For the scope of this project I have chosen two:

- Dark Channel Prior
- Canny Edge Detection

2. Dark Channel Prior method

2.1 Dark channel prior step by step

1. We take RGB color image (assumptions details discussed in 2.2)
2. We find it's dark channel (explained in 2.4)
 - a. split RGB image to three RGB channels
 - b. select one of the channels and start operating on its values of intensities, i.e. R (red) channel for start
 - c. define patch size 15×15
 - d. for every pixel x look for minimum value of intensity in a whole patch of size 15×15
 - e. write down this value to position of x
 - f. repeat operations b-e for other color channels
 - g. for every pixel x compare its value of intensity from above operations and look for minimal one
 - h. write that minimal value as intensity of pixel x for dark channel
 - i. we receive matrix named "**dark**" - this is our dark channel
3. Estimate atmospheric light (2.5)
 - a. take dark channel matrix **dark**
 - b. take input image, split into RGB channels
 - c. select one of the channels and start operating on it's values of intensities, i.e. R (red) channel for start
 - d. search for 0.1% of most brightest pixels in our dark channel and pick them
 - e. among those picked pixels search for the one with highest intensity in our input image
 - f. repeat operations c-e for other color channels
 - g. we receive atmospheric light A^c value for every channel - c stands for R, G or B channel

4. Estimate transmission map (2.6)

- a. take our estimated light value A^c
- b. take input image, split it into three RGB channels
- c. select one of the channels
- d. for every pixel x in selected channel divide its value of intensity by value of estimated atmospheric light A^c for this channel
- e. repeat c-d for other channels
- f. find dark channel (explained above in point 2.) on modified by division from point d. values of intensities of pixels
- g. perform calculation of transmission map which is **1-darkchannel** which was estimated in point f.
- h. we receive transmission map t

5. Visibility scale

On the basis of my observations I have developed visibility scale which uses transmission map as a parameter to estimate how well we can see in the image (on scale 1-5). Visibility scale is explained in page 14.

2.2 Assumptions

Dark Channel Prior idea was proposed by Kaiming He[1] and my work is based on this approach.

In order to make our algorithm work we need to fulfill following **Assumptions**:

- **Static Image** - it works on single images like screenshots or photos taken by smartphone, not video films.
- **Image in day-view** - my approach based on that method does not handle night-time images. For night-time conversion of colors algorithms are required[4].
- **Color Image** - it is required for image to consist of 3 RGB Color channels. Algorithm is not working properly on grayscale images.

2.3 Atmospheric Scattering Model

In image processing atmospheric scattering model is widely used to describe formation of haze images:

$$I(x) = J(x) t(x) + A[1 - t(x)]$$

Model is observed on three RGB color channels

where:

$I(x)$ - observed image, intensity of each pixel

$J(x)$ - scene radiance - illumination of our observed scene, in our computations

we omit that parameter, I assumed calculating it is not required for program

$t(x)$ - called transmission map, value at each pixel (explained in 2.6)

A - atmospheric light value (explained in 2.5)

My main goal of working with this method is to recover $t(x)$ and on the basis of that, estimate fog scale.

2.4 Dark Channel

Dark channel prior is based on the observations made by Kaiming He[1]. Most haze-free images with non-sky, consist of at least one RGB channel, which has very low value of intensity at each given area of pixels in our image. It was based on over 5000 images with manually cut off sky region. Dark channel at each pixel can be defined as:

$$dark(x) = \min_{c \in \{R,G,B\}} (\min_{y \in \Omega(x)} (J^c(y)))$$

where:

$dark(x)$ - value of intensity of dark channel in each pixel

$\min_{c \in \{R,G,B\}}$ - minimum value of intensity in local patch in R, G or B channel (we search for minimum between those 3)

$\min_{y \in \Omega(x)}$ - minimum value of intensity from all pixels y belonging to local patch $\Omega(x)$ (15x15) around our pixel x

$J^c(y)$ - intensity of pixel y which belongs to local patch $\Omega(x)$

For every pixel:

We take one of three RGB channels. We take a local patch 15x15 around current pixel, look for minimum intensity in this patch. Then we do comparison of this minimum intensity with other channels and we get our $dark(x)$.

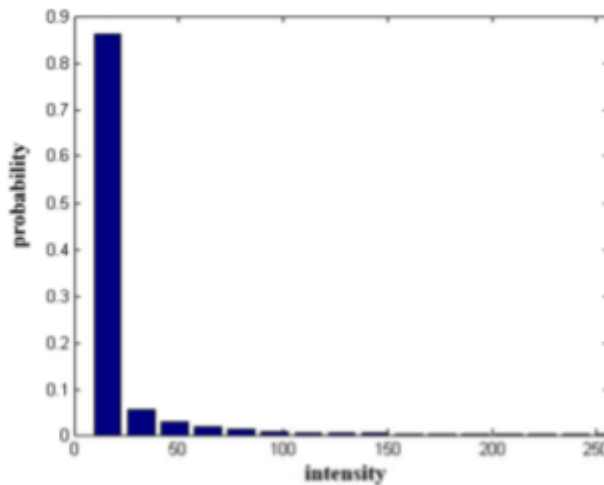


Figure 1. Histogram of intensity of pixels in dark channels (images tested by Kaiming). Each bar width is 16.

Conclusion of observation is that for non-sky regions intensity of dark channels is really small, tending to zero. As we can see in graph (Figure 1).

2.5 Estimation of atmospheric light

Next step is to estimate atmospheric light in our image. We will need it for calculation of transmission map on which we base our visibility scale. Formula for transmission map is provided in point 2.5.

Atmospheric light can be easily estimated using dark channel image. We estimate light by simply picking pixels which are closest to 1 in dark(x) so our dark channel. We pick 0.1% of brightest pixels. Those pixels are most “haze-opaque”. Then among those pixels we search for ones with highest intensity in our input image.

This method is more efficient than just picking the brightest pixels of image, because it can eliminate the problem of some objects which have white color like cars.

2.6 Transmission map

Transmission map[6] is describing the light that is not scattered and reaches the camera. Map is a continuous function of depth and reflects informations about depth of our image. In our visibility estimation we will be using that parameter to determine visibility factor of our image, scale determined from 1 to 5 where 5 states for best (no haze) image. Transmission map can be estimated using equation:

$$t(x) = 1 - \omega \min_{c \in \{R,G,B\}} \left(\min_{y \in \Omega(x)} \left(\frac{J^c(y)}{A^c} \right) \right)$$

where:

$t(x)$ - value of transmission in given pixel x

$\min_{c \in \{R,G,B\}}$ - minimum across three R, G, B channels

$\min_{y \in \Omega(x)}$ - minimum value of ratio of intensity (J^c) from all pixels y belonging to local patch $\Omega(x)$ (15x15) around our pixel x and A^c

$J^c(y)$ - minimum value of intensity in local patch y for given pixel x

A^c - value of atmospheric light for every R,G,B channel

ω - fixed value, 1 in my calculations

In case of haze removal algorithm[1] ω should be fixed to 0.95 (for purpose of leaving some haze on a picture) due to fact that haze is fundamental for humans to see depth[7] in image-dehazed pictures. In my case I simply set it to 1 because we do not care about seeing depth in our picture.

3. Canny Edge Detection method

3.1 Assumptions

Canny Edge Detection[5] method is based on detection of edges of objects in image and then comparing them with the same photo from different time of the day, when weather condition was different. This edge detection algorithm was developed by John F. Canny in 1986.

In order to make our algorithm work we need to fulfill following **Assumptions**:

- **Static Image** - it works on single images like screenshots or photos taken by smartphone, not video films.
- **Having base/ideal state of weather image** - it is required for us to have a reference pattern to which we will compare our photo.

3.2 Canny Edge Detection

Noise Removal

Canny edge detector is using filter based on first derivative of Gauss function. We before we attempt to detect any edges we have to remove noise. In effect we receive slightly blurred image which does not suffer from any big noise interruptions.

Looking for intensity gradient of the image

Edge on the image can be directed in different direction. Canny's Algorithm uses four filters to detect horizontal, vertical and diagonal edges on smoothed image. Operators like Prewitt and Sobel return values of first derivative for vertical (G_y) and horizontal (G_x) direction. Slope (gradient, speed of increase) of edges and direction can be specified using equations:

$$G = \sqrt{G_x^2 + G_y^2}$$

G - magnitude of gradient
 G_y - vertical derivative
 G_x - horizontal derivative

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Θ - slope of the gradient

Non-maximum suppression

Final image should have thin edges, because of that we perform this operation. We simply go through all the points on the gradient intensity matrix and find pixels with maximum value in edge directions.

To sum up. Each pixel has 2 main criteria (edge direction in radians, and pixel intensity (between 0–255)).

Edge tracking using hysteresis

Last step is to remove non-relevant edges which have slope below set threshold. This operation adds to already detected edges next pixels, and so on until it's slope reaches threshold. It prevents splitting edges in places where contrast is very low.

3.3 Similarity of base and current image

To estimate the level of fog we need to compare two canny edges filtered images (base and current image). We do that by performing bitand operation[2] between each pixels of those two images. As output we receive intersection of two images. Then we proceed to calculating visibility factor and developing scale for such factor.

Calculating our visibility factor:

Number of non-zero elements of intersection divided by number of non-zero elements of base image will be our visibility factor.

$$N_{inter}/N_{base} = visi$$

where:

N_{inter} - number of non-zero bits in intersection

N_{base} - number of non-zero bits in base image

$visi$ - visibility factor in scale 0-1

4. Structure of program

My algorithm is done in Matlab environment.

It contains functions and folders listed below:

- Dataset folder - Folder in which we place our tested images
- Results - Folder mainly used to save outcomes of the program, like transmission maps, dark channel images and canny edge filtering results.
- main.m - main function, used to call functions and perform operations on both canny edge and dark channel methods.

Dark channel prior functions:

- atmlight.m - function to estimate atmospheric light
- darkchannel.m - estimate dark channel
- readIm.m - read RGB values and change their range to 0-1.
- transmission.m - estimate transmission map using atmospheric light and dark channel
- visibility.m - dividing image into 9 parts and creating matrix for each
- scaling.m - calculate visibility factor for each of 9 parts of image
- overall.m - calculate visibility factor for whole image
- scaleonimage.m - write down results of visibility estimation on our image

Canny edge function:

- cannyresult.m - calculate visibility factor for whole image

Other canny edge functions and calculations are performed in main.m file.

5. Dark channel prior to estimate visibility

This method is working only on color images and we need to keep that in mind. Our starting point will be an image which we load to our file. In program there is variable called scale, use it to change image size. Personally I recommend using scale to adjust image to maximum of 1200x800 pixels or similar to make calculations shorter. Of course bigger scale results in better outcome, but also in longer calculation time. Max scale = 1.

This is example image on which I will be presenting how program works.
foggycar.png 350x700 pixels (included in dataset folder)



Using function `readIm.m` load image and divide RGB values by 255 to achieve range of values from 0 to 1.


Next step is calling `darkchannel.m` function which uses our RGB image. Dark channel prior is based on the observation of haze and fog free outdoor images. Most non-sky patches of this images have at least (can be more) channel of pixels with very low intensity at some pixels.

So we compare intensity of every RGB channel in every pixel and choose the minimal values for each. We do that by dividing image into 15x15 patches and finding its minima.

Performing this operation we obtain dark channel image:



Then, we attempt to estimate atmospheric light. Function `atmlight.m` does it by sorting image and taking out few last from the vector.

 1x3 double

	1	2	3
1	0.8237	0.9178	0.8864

In case of our `foggycar.png` those are the values of brightest pixels (for R, G, B channels).

Next step is estimation of transmission map which is done by `transmission.m` function



Such transmission map is obtained, darkest pixels stand for worst visibility possible = biggest haze/fog.

We come to last step, estimating visibility on image based on obtained transmission map.

To develop visibility scale I did simulations on many different pictures taken from databases and done using my smartphone.

I represent it in 1 to 5 scale, where 5 stands for best possible visibility.

I came to conclusion that:

visi = 0.7 to 1 equals to 5

visi = 0.55 to 0.6(9) equals to 4

visi = 0.4 to 0.5(4) equals to 3

visi = 0.2 to 0.3(9) equals to 2

visi = 0 to 0.1(9) equals to 1

Using functions visibility.m, scaling.m and overall.m im estimating visibility factor in each of 9 parts of my image, also overall for whole image.

I do that, by calculating average value of pixels in whole image and separately in each of 9 parts of image.

Then I use scaleonimage.m function to print results of scaling.m on our image.

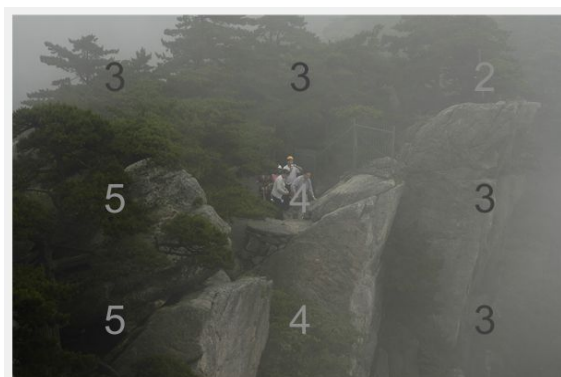


This is the outcome, we can see that program estimated dense fog as 1 and bottom right corner is estimated as 4 so exactly as we want it to work.

Overall visibility factor was estimated as: 2

Running program on other examples gives results:

(mountain.jpg)



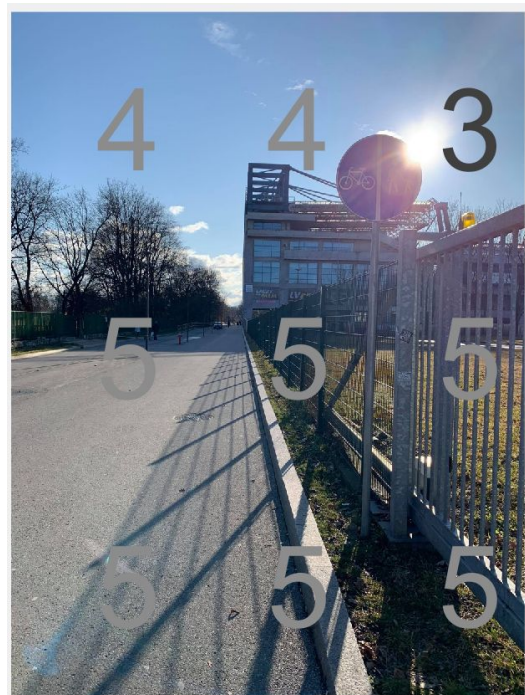
Overall: 4

(1.jpg)



Overall: 5

(2.jpg)



Overall: 5 - What is interesting here, this algorithm handles really strong light source flashing directly into camera as disturbed visibility.

It is very interesting because in car systems that could be used as one of the factors calculating bad conditions on the road. Not only visibility limited by haze and fog but also by sun which can distract driver.

(3.jpg)



Overall: 5

Dataset: For dark channel prior method I have used database HazeRD[3], did some own pictures using smartphone (1,2,3.jpg), used some pictures provided by Kaminghe and collected some random haze images from google graphics tool. All of them besides pictures from HazeRD (dataset was really huge so i deleted most of them to not waste space) are included in dataset folder in my project.

6. Canny edge detection as fog estimation method

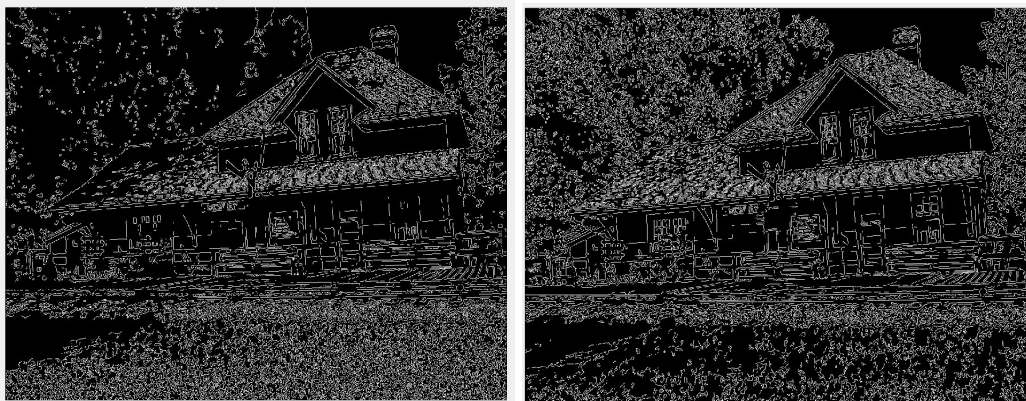
Second way to estimate haze/fog which I found interesting is canny edge detection method. For this method HazeRD[3] dataset will be used. It operates on very simple principle. Comparison of edges in good and bad condition of weather and by adjusted scale, we define visibility factor.



At start we provide our image (img_7411_100) and base image(img_7411_RGB).

We are using matlab built-in function for canny edge filtering so we follow these steps:

1. Convert both images using `rgb2gray()`; - they will be now grayscale images
2. Do canny edge filtering of both images - we are using matlab built in function `edge(imagerefgray, 'Canny');`



Left - image with haze, Right - image without any haze

3. Do `intersection[2]` between outputs of both canny edge filterings - we do that by performing `bitand()`; logic operation on pixels in matrices representing those images.



Intersection between imagebase and image with fog

4. Calculating our visibility factor:

Number of non-zero elements of intersection divided by number of non-zero elements of base image will be our visibility factor.

$$N_{inter}/N_{base} = visi$$

In this case based on my personal observations I developed slightly different scale, than in dark prior channel method.

For the reason they operate on different values extracted from the images and the principle of such operations differs we have to develop different scales for those two methods.

visi = 0.8 to 1 equals to 5

visi = 0.65 to 0.7(9) equals to 4

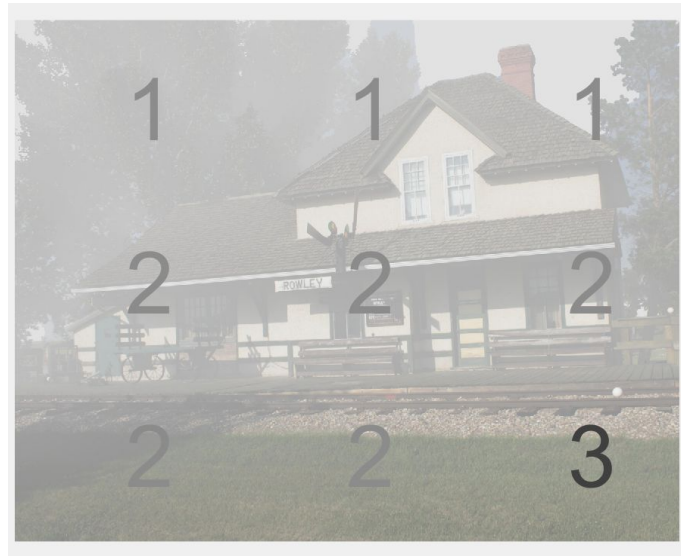
visi = 0.4 to 0.5(4) equals to 3

visi = 0.2 to 0.3(9) equals to 2

visi = 0 to 0.1(9) equals to 1

Overall visibility factor using canny edge obtained: 3

To compare it with previous method:



Overall: 2

6. Conclusions

Results are slightly different, but It is mainly because of scale adjustment for both algorithms. More tests performed on wider database and some optimization calculations will result in more accurate comparison, but the outcome is satisfying. Both methods handled well task of estimating visibility and calculating its factor.

Main advantage of proposed dark channel prior method over canny edge detection is that we do not need any source of base image to estimate visibility in uploaded to program.

Bibliography:

- [1] <http://kaiminghe.com/publications/cvpr09.pdf>
- [2] <https://www.mathworks.com/matlabcentral/answers/119432-intersection-between-2-images>
- [3] <https://ieee-dataport.org/open-access/hazerd-outdoor-scene-dataset-and-benchmark-single-image-dehazing>
- [4] https://www.researchgate.net/publication/261387111_Nighttime_haze_removal_using_color_transfer_pre-processing_and_Dark_Channel_Prior
- [5] <https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123>
- [6] <https://www.hindawi.com/journals/ijcgt/2014/308629/>
- [7] E. B. Goldstein. Sensation and perception. 1980. 4