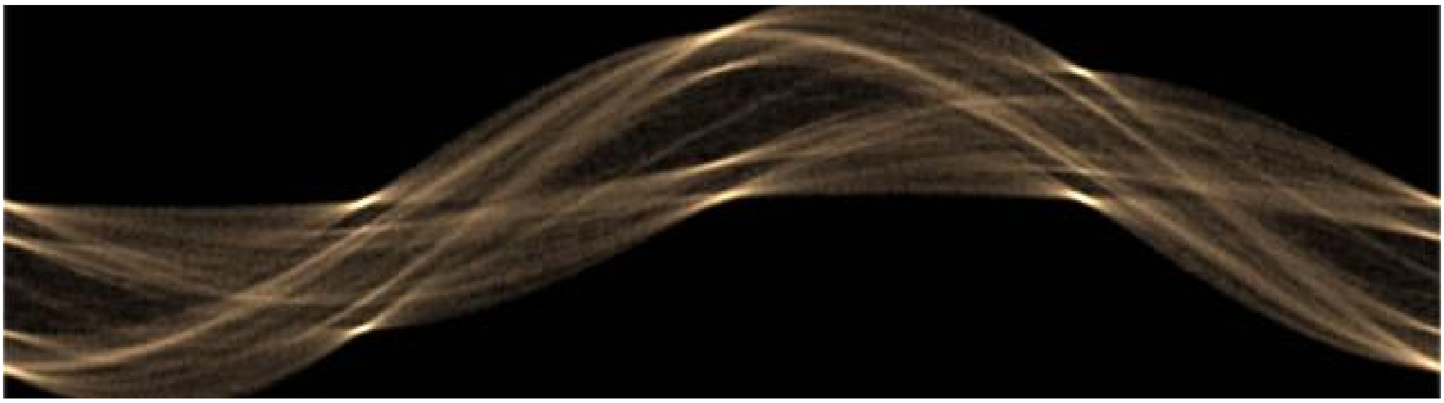
Home Tutorials Blog About Contact

Tutorials Computer Vision Hough transform in OpenCV



Hough transform in OpenCV

By Utkarsh | Published: April 11, 2010

OpenCV already comes with a function to perform hough transforms. It lets you choose between different variants of the transform: the standard hough transform, the probabilistic hough transform and the multi-scale hough transform. Here I'll get into the technical details of getting the command, cvHoughLines2, to work. The command expects and returns parameters in a certain format.

The command

The cvHoughLines2 command goes like this:

```
CvSeq* cvHoughLines2(CvArr* image, void* line_storage, int method, double rho, double theta,
```

We'll go into each parameter in detail.

image is the image you want to do the hough transform on. This has to be an 8-bit single channel binary image. Though you can supply a grayscale image, it will be treated as a binary image (non-zero pixels are used).

line_storage is the place where this function stores its result. This can be either a *CvMemoryStorage* structure or a matrix with N rows. More on this parameter later.

method is either CV_HOUGH_STANDARD, CV_HOUGH_PROBABILISTIC, or CV_HOUGH_MULTI_SCALE. And you can guess they're for the standard hough transform, the probabilistic hough transform and the multi-scale hough transform.

rho and *theta* set the desired accuracy of the transform. *rho* is in pixels and *theta* is in radians. The smaller the value, the better the transform will be... but it'll also take more time. Usually, values 1 and 0.01 should be sufficient.

threshold determines which lines are returned. Each line has a particular number of "votes". This parameter sets the minimum number of "votes" in order to qualify as a potential line. You might want to read about The Hough Transform for more information on this.

param1 and *param2* are used by the different transforms.

- For the standard hough transform, these are not used
- For the probabilistic hough transform, param1 is the minimum line segment length and param2 is the separation between collinear points to split them into two segments (instead of merging into a single one).
- For the multi-scale hough transform, *rho/param1* and *theta/param2* is the final resolution of the for refining results.

Extracting results

Getting results out of this command depends on the *line_storage* parameter. You have two options: supply a CVMat matrix or supply a CvMemoryStorage stucture.

The matrix approach

This one is straight forward. You give it a matrix, and the function will populate this matrix with its results. For different *method* values, this matrix must have different formats:

- Standard Hough transform and multi-scale hough transform: The matrix must be N rows by 1 column, and 2 channeled (CV_32FC2). It stores the p and θ values
- Probabilistic hough transform: The matrix must be N rows by 1 column, and 4 channeled (CV_32FC4). It stores the two end points of the line segments ((x,y) twice).

The function will set the number of rows of the matrix to the number of lines detected. Also, it will return a NULL.

The CvMemoryStorage approach

If you provide a memory storage, the function will return a CvSeq* sequence. Using this sequence, you can access the various parameter of the detected lines:

```
float* currentLine = (float*) cvGetSeqElem(line_seq , index);
```

- For the standard hough transform and the multi-scale hough transform, you can access the p and θ values using *currentLine[0]* and *currentLine[1]* (both *float*)
- For the probabilistic hough transform, the returned sequence is a sequence of CvPoint. So you can access the end points of line segments using *currentLine[0]* and *currentLine[1]* (both *CvPoint*)

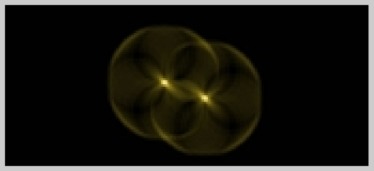
Done!

This should be enough to get you started with using the cvHoughLines2 command! Got questions? Suggestions or criticism? Let me know or leave a comment below!

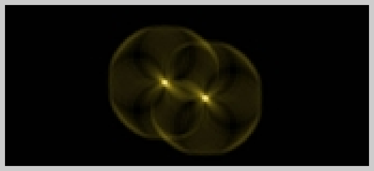
Issues? Suggestions? Visit the Github issue tracker for AI Shack

Back to top

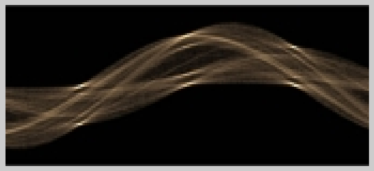
Suka 2.575 orang menyukai ini.



Hough circles in OpenCV



Circle Hough Transform



The Hough Transform

18 Comments

JorgeSG
Posted December 2, 2010 at 8:49 am | *Permalink*

Hi I'm working in line detection of an image.

Do you know how to detect lines of a certain thickness using #OpenCV?

Reply

Utkarsh
Posted December 9, 2010 at 12:48 pm | *Permalink*

Hi Jorge! I'm not sure how you'd do that. But I can think of doing some edge detection followed by the hough transform. If two lines are almost parallel and at a certain distance, you have your line.

Reply

AruniRC
Posted January 7, 2011 at 3:32 pm | *Permalink*

hey Utkarsh
this was a really helpful guide. Could you pls help me out with a problem:
I've drawn Hough Lines in red on the input Image (with help from the sample given in OpenCV). However I want to keep ONLY those lines that are roughly horizontal and not keep any vertical lines.
How do i go about finding the degree with the X-axis that the lines is making?

Reply

Utkarsh
Posted January 11, 2011 at 7:13 pm | *Permalink*

Did you have a look at this: Cropping Robotics Arena Boundaries. In a part of it, I go into the details of finding horizontal and vertical lines. Did this help?

Reply

AruniRC
Posted January 12, 2011 at 12:23 pm | *Permalink*

thanks. i managed to get the line angle from the saequence which stores the hough lines and filtered out the reqd angles.

However I have another question regarding Hough Lines. When used in probabilistic mode how come some lines don't appear even when I've decreased threshold? LIke i'm currently working on Devnagari text locating in any image. Trying to do this by first using horizontal hough lines to identify possible 'matras' and then seeing if there are vertical hough lines below any of the horizontal lines obtained (thus filtering out horizontal lines from other objects like billboards etc.)
any suggestions to resources on the net would be really helpful. 🙂

Reply

Utkarsh
Posted January 13, 2011 at 9:50 pm | *Permalink*

Just to confirm, the probabilistic mode returns line segments – two endpoints for each line. This is different from the standard hough transform (where you get theta and distance). Did you consider this while writing code?

Reply

AruniRC
Posted January 14, 2011 at 3:05 pm | *Permalink*

yes, i had considered that. found the theta using

atan() of the slope of the two CvPoints returned by the hough function.
however often the line segment of hough (prob) line is only a part of a full line that is clearly visible on the image.

Reply

Rafi
Posted January 17, 2011 at 10:19 am | *Permalink*

hi…..
i am using hough transform for detection of lanes… i got the result with grey scale image…

now i want the result to be in RGB… i want those lines to be drawn on rgb image…

plz help……

Reply

Utkarsh
Posted January 19, 2011 at 9:42 pm | *Permalink*

Well… do the hough transform on the grayscale image. And draw the line on a BGR image. Simple as that.

Reply

Rafi
Posted January 17, 2011 at 10:28 am | *Permalink*

and how to fit the hyperbola in that lane….?

Reply

Utkarsh
Posted January 19, 2011 at 9:42 pm | *Permalink*

For that, better read some Numeric Analysis textbook.

Reply

Pavan
Posted March 16, 2011 at 5:04 pm | *Permalink*

Hey i’m working on detection of lanes using opencv. i’ve been using the probabilistic hough transform which gives the sequences. From this, I have been getting a sequence of cv point. But this sequence has only the starting pixel values of the lines.
I wanted to know how to get the starting and ending pixel values of the lines??
Thanks in advance

Reply

Utkarsh
Posted March 21, 2011 at 9:49 am | *Permalink*

The sequence you get is actually a sequence of arrays. Each array has 2 elements – the start and the end. You can access the start as [0]and the end as [1]. Did you do this?

Reply

Marek
Posted April 4, 2011 at 10:15 pm | *Permalink*

Hi,
How would you try to get the pattern describing each line – I’m searching a method to compute the points where these lines cross each other.

Besides – quite good article.

Reply

Utkarsh
Posted April 29, 2011 at 9:40 pm | *Permalink*

What do you mean? A pattern describing each line?

Reply

Kagan

Posted April 24, 2011 at 6:54 pm | *Permalink*

Hi,I'm trying to find road lanes in a video.I can find roadl lines with hough transform in opencv but i need to find vanishing points after that step to determine if the car is on lane or not.I could not find anything.Can u Help me on that please ?

Reply

Utkarsh
Posted April 29, 2011 at 9:04 pm | *Permalink*

Isn't the vanishing point the intersection of the road's two lines?

Reply

ANAND
Posted May 18, 2011 at 3:57 pm | *Permalink*

hi, I have been given a as stupid as it sounds but i wud say equally challenging job that is to extract hairs and count them as wll as find the thickness of hair from a scalp image, I have done the initial part but I am stuck at finding the thickness of a strand, any ideas for that?? or in general how do u find the thickness of hair at all

Reply

Post a Comment

Your email is *never* published nor shared. Required fields are marked *

Name *

Email *

Website

Comment

You may use these HTML tags and attributes `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

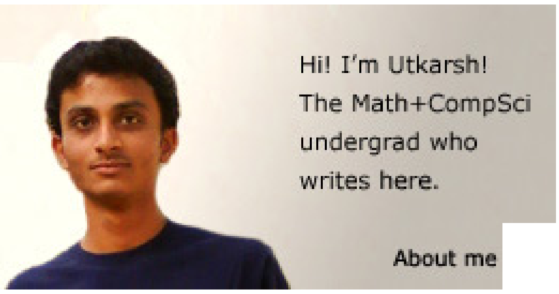
Post Comment

Free updates

Email Address

More info

1000+ subscribers!



Hi! I'm Utkarsh! The Math+CompSci undergrad who writes here.

About me

Popular Posts

- Scanning QR Codes
- The Canny Edge Detector
- Implementing Canny Edges from scratch
- Image Moments
- Predator: Tracking + Learning

More on AI SHack

Popular **Beginners** **Utkarsh's favs**

- Scanning QR Codes
- The Canny Edge Detector
- Implementing Canny Edges from scratch
- Image Moments
- Predator: Tracking + Learning

About me

My name is Utkarsh Sinha, and I'm an undergraduate student, pursuing B.E. Computer Science + M.Sc. Mathematics. Here, I help you understand ideas in Artificial Intelligence, using a not so techy and mathematical language. And in the process, learn more about Artificial Intelligence myself.

Read more at the about page