# Computer Vision

Computer vision is the area of artificial intelligence concerned with training of computers to extract useful information from the environment using visual sensors. Using digital images from cameras and videos and deep learning models, machines can accurately identify and classify objects. With this information computers can further run mathematical models and filters to improve what they can “see”. We can then react to external changes by observing them through the video.

Computer vision is a scientific field that deals with how computers can gain high-level understanding from digital images or videos by using mathematical models and equations to interpret the video frame by frame. It seeks to understand and automate tasks that the human eyes can do, example - tracking objects, face detection, line detection, etc.

## Standard Hough Transform

In general, any line can be represented using the expression ‘y = mx + c’. In Hough Transform the line can be associated with a parametric pair of (Θ, ρ), where ‘ρ’ represents the shortest distance from the origin to the straight line and ‘Θ’ is the angle between the x axis and the line connecting the origin to a straight line. According to Hugh transform, ‘ρ’ = xcos Θ + ysinΘ.

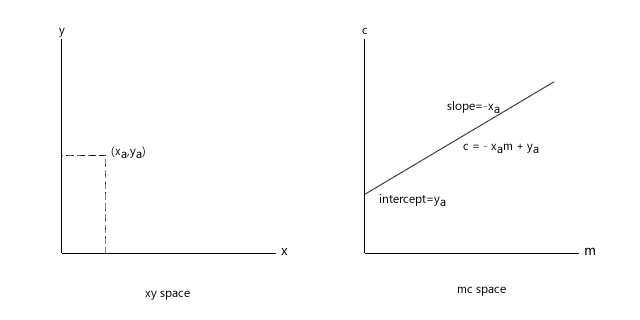
Consider a line that passes through a given set of points. After applying Hough transform on the line, we obtain a sinusoidal line passing through the same set of points in the Hough transform plane. Since many lines in an image intersect each other, its understandable that the sinusoidal lines intersect each as well. The number of sinusoids intersecting determines the strength of a particular line.

## Extracting Line segments using Hough Transform

This assignment requires for us to detect the angles created by the pendulum for our inverted pendulum assignment. This meant we had to use a video capture device to view the inverted pendulum and measure the angle of disturbance between the pendulum and the horizontal axis.

Using Hough Transform we can extract line segments from any frame of the video. Since in a Hough Transform plane, all points where many numbers of sinusoidal lines intersect represents a peak i.e. a detected line. We can then obtain these lines to identify where the detected line occurs in our original video source.

1. Any line passing through (xa, ya): ya = mxa + c
2. Rearranging: c = - xam + ya
3. The above is the equation of a line in the mc space.



So, a point in the xy space is equivalent to a line in the mc space.

## Using MatLab to Extract Line Segments Using Hough Transform

To extract the line segments from the image, we first convert it into black and white so we can apply the Matlab function ‘hough’. This function returns a Hough Transform matrix. We then need to locate the peaks as peaks represent detected lines, using the ‘houghpeaks’ function in Matlab. Now, we extract line segments corresponding to the peaks in the Hough Transform Matrix using the function ‘houghlines’.

## Methodology

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Function Name | Syntax | Description |
| 1. | hough | [[H,theta,rho] = hough(BW)](https://uk.mathworks.com/help/images/ref/hough.html#d120e80547)  [[H,theta,rho] = hough(BW,Name,Value)](https://uk.mathworks.com/help/images/ref/hough.html#d120e80588) | [[H](https://uk.mathworks.com/help/images/ref/hough.html" \l "buwew0l-1-H),[theta](https://uk.mathworks.com/help/images/ref/hough.html#buwew0l-1-theta),[rho](https://uk.mathworks.com/help/images/ref/hough.html#buwew0l-1-rho)] = hough([BW](https://uk.mathworks.com/help/images/ref/hough.html#buwew0l-1-BW)) computes the Standard Hough Transform (SHT) of the binary image BW. The hough function is designed to detect lines. The function uses the parametric representation of a line: rho = x\*cos(theta) + y\*sin(theta). |
| 2. | houghpeaks | [peaks = houghpeaks(H,numpeaks)](https://uk.mathworks.com/help/images/ref/houghpeaks.html#d120e81535)  [peaks = houghpeaks(H,numpeaks,Name,Value)](https://uk.mathworks.com/help/images/ref/houghpeaks.html#d120e81561) | [peaks](https://uk.mathworks.com/help/images/ref/houghpeaks.html#buw9d0p-1-peaks) = houghpeaks([H](https://uk.mathworks.com/help/images/ref/houghpeaks.html" \l "buw9d0p-1-H),[numpeaks](https://uk.mathworks.com/help/images/ref/houghpeaks.html#buw9d0p-1-numpeaks)) locates peaks in the Hough transform matrix, H, generated by the [hough](https://uk.mathworks.com/help/images/ref/hough.html) 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. |
| 3. | houghlines | [lines = houghlines(BW,theta,rho,peaks)](https://uk.mathworks.com/help/images/ref/houghlines.html#d120e81088)  [lines = houghlines(**\_\_\_**,Name,Value)](https://uk.mathworks.com/help/images/ref/houghlines.html#d120e81123) | [lines](https://uk.mathworks.com/help/images/ref/houghlines.html#buwgo_f-1-lines) = houghlines([BW](https://uk.mathworks.com/help/images/ref/houghlines.html" \l "mw_a6dd2036-53c1-4fc4-a63a-ac870658ff37),[theta](https://uk.mathworks.com/help/images/ref/houghlines.html#buwgo_f-1-theta),[rho](https://uk.mathworks.com/help/images/ref/houghlines.html#buwgo_f-1-rho),[peaks](https://uk.mathworks.com/help/images/ref/houghlines.html#buwgo_f-1-peaks)) extracts line segments in the image BW associated with particular bins in a Hough transform. theta and rho are vectors returned by function hough. peaks is a matrix returned by the houghpeaks function that contains the row and column coordinates of the Hough transform bins to use in searching for line segments. |

Table 1 : Matlab Functions used and Description

Figure 1 : Method to extract line segments

Hpeaks= hough peaks(H,10,’NHoodSize’,[55 11]);

Where,

10 – is the number of peaks starting from highest.

’NHoodSize’,[55 11] – is to suppress peaks within a neighboring region. Increased neighborhood size results in lesser overlap.

## Additional preprocessing

BW Thershold – this is required for hough transform as Hough Transform can be applied to binary images only.

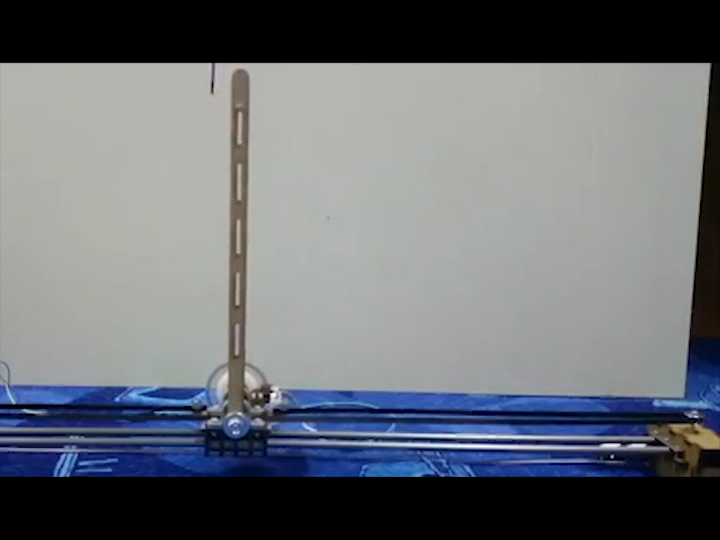
Moropological function – to convert thick lines into a single thin detected line, we use this function along with the Skeleton operation. It reduces any error in angle calculations. 

Figure 2 : Inverted Pendulum Sample Video

## Code developed for computer vision

%%

close all

clear

clc

%% Initialize System Objects

% Create a VideoFileReader System object to read video from a file.

vidFReader = vision.VideoFileReader('igvcVideo.avi',...

'VideoOutputDataType','double');

% Create a VideoPlayer System object to visualize video

vidPlayer = vision.DeployableVideoPlayer;

%% Loop Algorithm

idx = 1;

while(~isDone(vidFReader))

%% Preprocess

% Acquire frame

frame = step(vidFReader);

% Mask to get Lines of interest

BW = createLineMaskHSV(frameNoOrange);

% Apply a close morphology to make continuous lines

BM = imclose(BW,strel('disk',10));

% Apply a skeleton morphology to get the thinnest lines

BMSkel = bwmorph(BM,'skel',Inf);

%% Detect Lines

% Perform Hough Transform

[H,T,R] = hough(BMSkel);

% Identify Peaks in Hough Transform

hPeaks = houghpeaks(H,10,'NhoodSize',[45 45]);

% Extract lines from hough transform and peaks

hLines = houghlines(BMSkel,T,R,hPeaks,...

'FillGap',100,'MinLength',100);

%% View results

% Overlay lines

[linePos,markerPos] = getVizPosArray(hLines);

if(~isempty(hLines))

lineFrame = insertShape(frame,'Line',linePos,...

'Color','blue','LineWidth',5);

outFrame = insertObjectAnnotation(lineFrame,...

'circle',markerPos,'','Color','yellow','LineWidth',5);

else

outFrame = frame;

end

% Update video player

step(vidPlayer,outFrame);

end

%% Clean up

release(vidFReader)

release(vidPlayer)

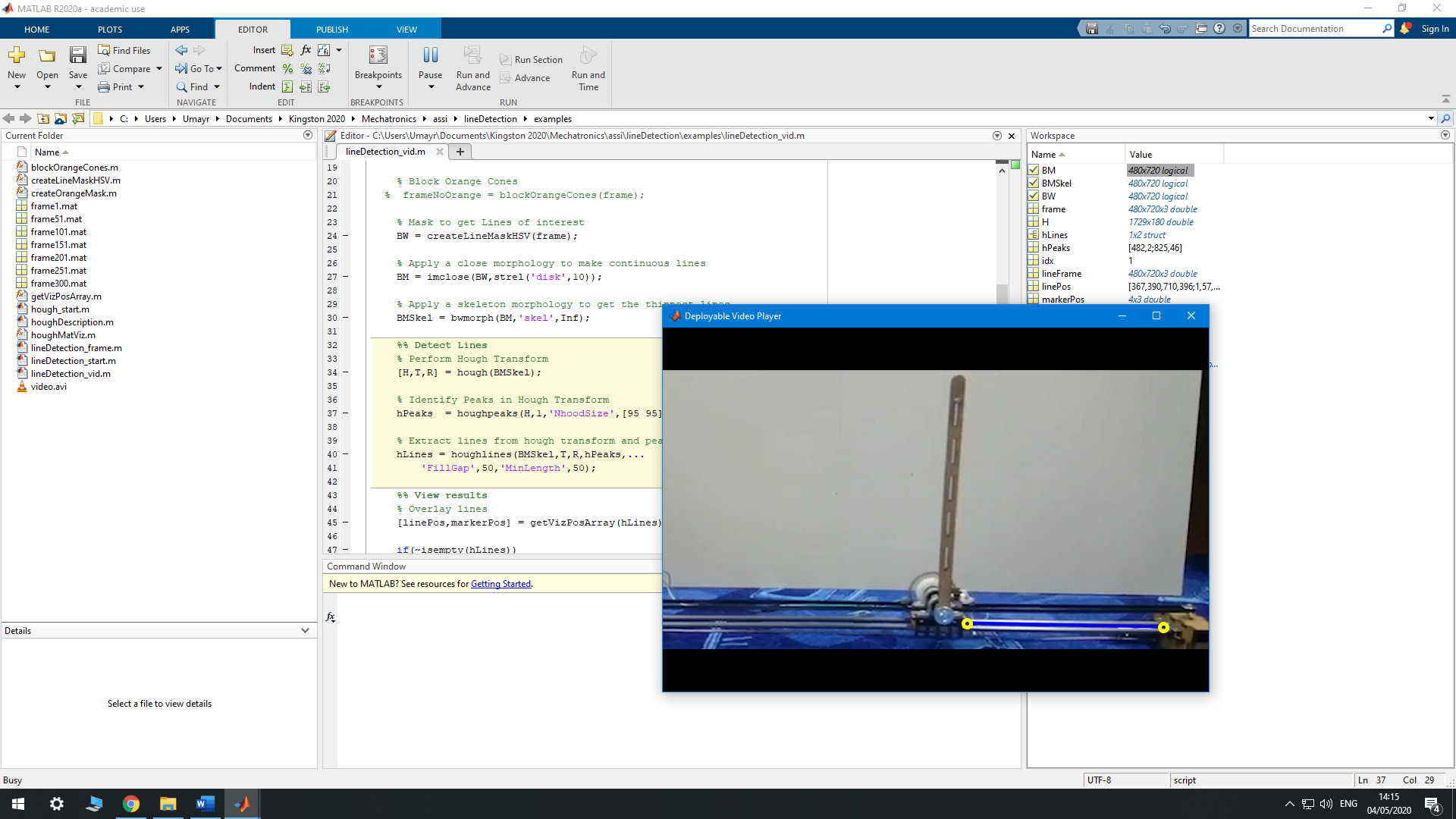


Figure 3 : MATLAB Result of Computer Vision Code