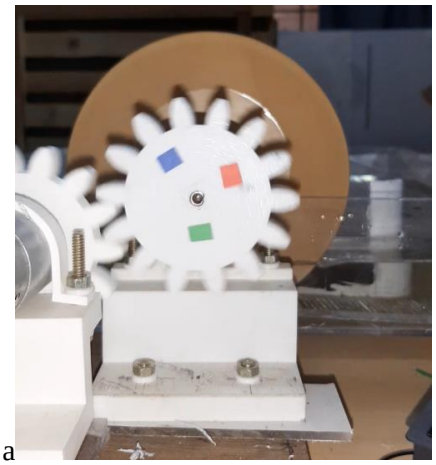
VAMSI KRISHNA M

DETERMINING THE RPM OF A ROTATING GEAR

MATLAB IMAGE PROCESSING

The RPM of the gear must be modelled as a function of the duty cycles/second supplied to the motor from the controller under constant load and constant power supply, using image processing techniques in MATLAB.

Colour strips were stuck to the gear surface as they would aid in segmentation of the images, as shown below.

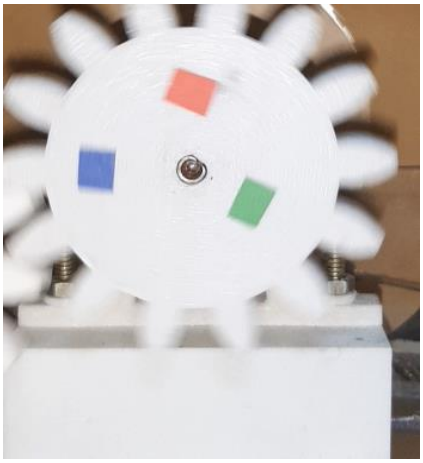


a

Videos were shot at 120 frames per second at duty cycles/second ranging from 20 to 100, with an interval of 10.

- First, the video files were loaded into MATLAB and cropped such that the focus was only on the gears.

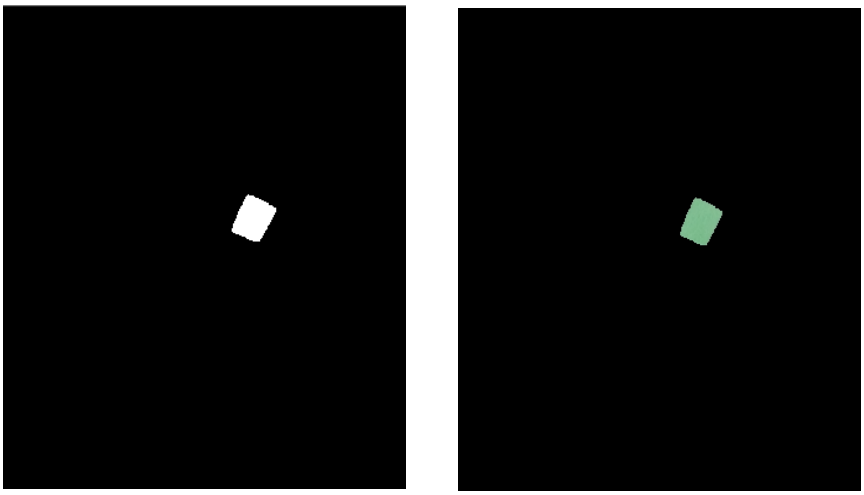
```
pic = read(v,i);
pic = imcrop(pic,[160 520 350 400]);
```



- The videos were parsed frame by frame.

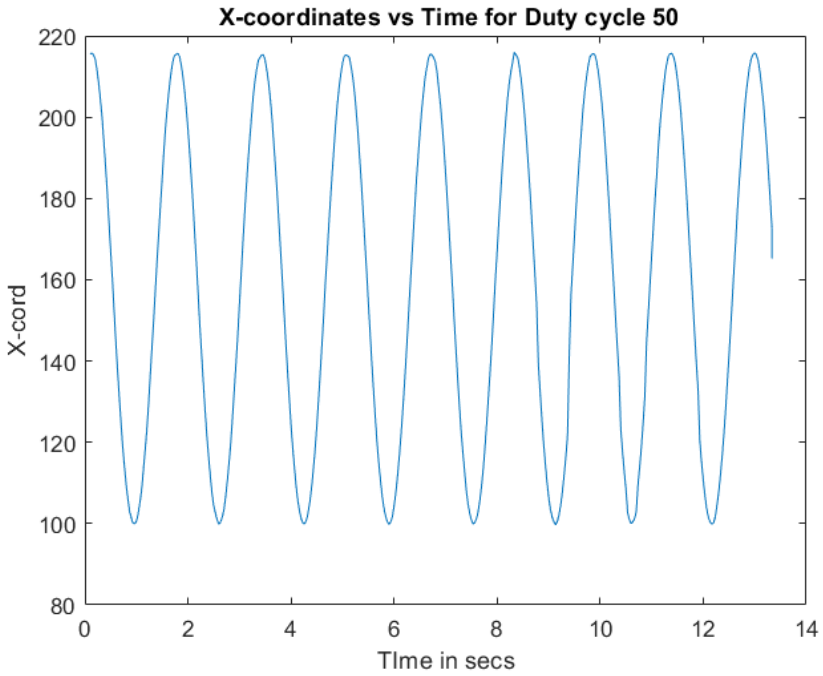
- Each frame was segmented into foreground and background elements. The green strip being the foreground object and the rest were considered as background.
  The mask and the masked image are shown below.

```
[mask,maskedpic] = masking(pic);
```

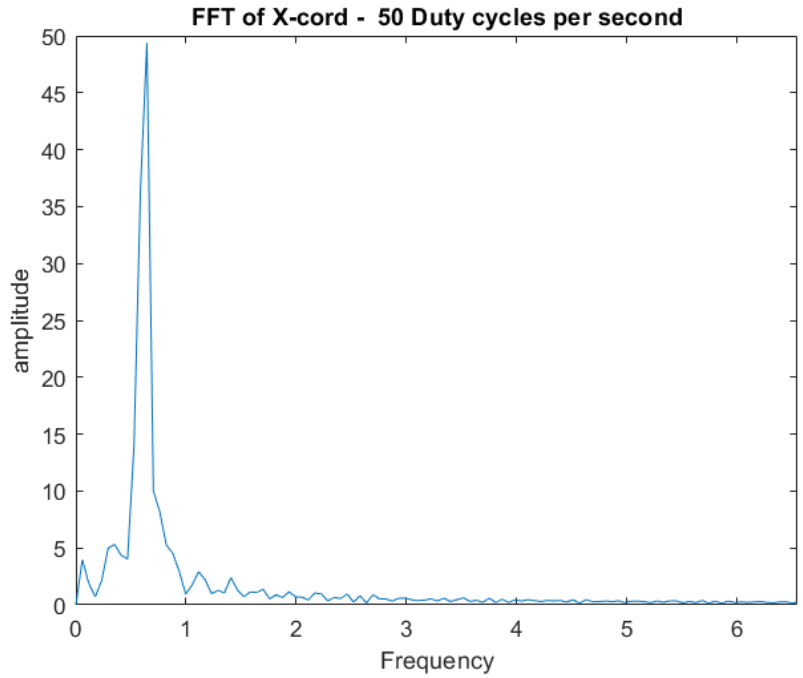


- The centroid of the foreground object was obtained and then the x coordinate of the centroid was stored along with the time taken to reach that specific coordinate.

```
snew = struct2array(regionprops(mask,"Centroid"));
x_cord = [x_cord,snew(1)];
time = [time,v.CurrentTime];
```

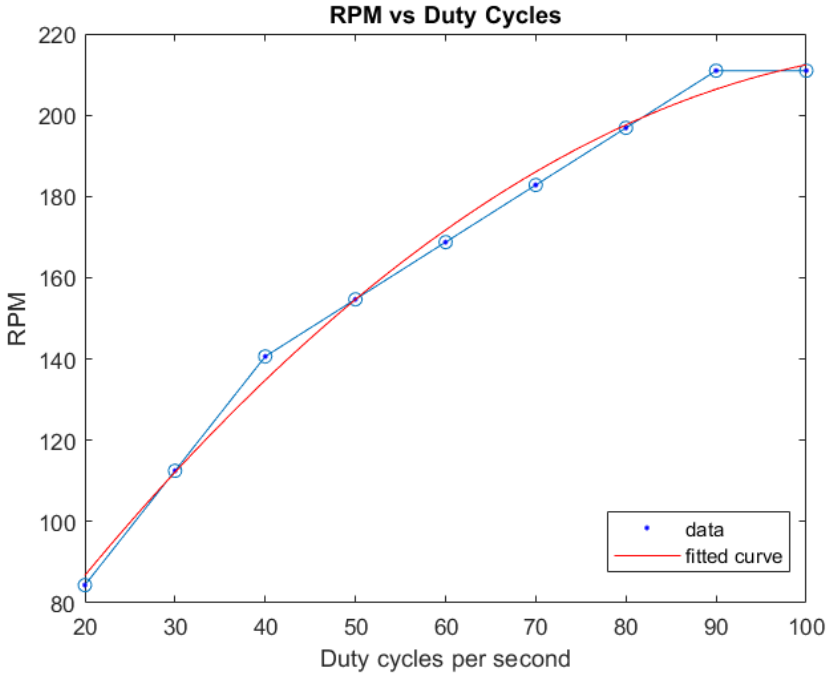


- Fast Fourier transform (FFT) was done on the x coordinate dataset to extract the corresponding frequency values.



- These values are adjusted based on the frame rate.

- A quadratic curve is fit based on the rpm values and the corresponding duty cycles/second values.



- The relation between RPM and Duty cycles/second is obtained.

$$RPM = -0.01385Du^2 + 3.232Du + 27.66$$

where, RPM is Revolutions per minute

Du is Duty cycles per second

MATLAB Code:

```
clear
clc
x_cord = [];
time =[];
rpm = zeros(1,9);
k = 1;
duty = 20:10:100;


for j = 20:10:100

v =VideoReader(sprintf("%d_.mp4",j));
string  =  j +"_bw";
vout = VideoWriter(string,"MPEG-4");
open(vout)

       for i = 1:400

              pic = read(v,i);
              pic = imcrop(pic,[160 520 350 400]);
              [mask, maskedpic] = masking(pic);

              snew = struct2array(regionprops(mask,"Centroid"));

              x_cord = [x_cord,snew(1)];
              time = [time,v.CurrentTime];

              if(i==v.NumFrames)
                     break;
              end

       end
       f = find_freqency(x_cord,length(x_cord),1/(time(2)-time(1)));
       rpm(k) = f;
       k = k+1;
       x_cord =[];
       time =[];
end

rpm = 60*rpm*4;
cur = fit(duty',rpm',"poly2");
plot(20:10:100,rpm,"-o");
hold on
plot(cur,duty,rpm)

writematrix([(20:10:100)',rpm']," Duty_vs_Rpm.txt","Delimiter",'\t');
```

```
function f = find_freqency(signal,Length,sampling_Freq)

NFFT = 2^nextpow2(Length);   % Next power of 2 from length of var
% removing DC component from signal
% if your signal has an offset, you want to get rid of that offset before
% taking the fft so that you do not get a max at the origin representing
% the DC component

DC = mean(signal);
signal = signal - DC;

Y = fft(signal,NFFT)/Length;
freq = sampling_Freq/2*linspace(0,1,NFFT/2+1);

amp = 2*abs(Y(1:NFFT/2+1));
P = amp;

[temp, I] = max(amp);

f = freq(I);
end

function [BW,maskedRGBImage] = masking(RGB)

% Convert RGB image to chosen color space
I = RGB;

% Define thresholds for channel 1 based on histogram settings
channel1Min = 103.000;
channel1Max = 145.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 168.000;
channel2Max = 217.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 124.000;
channel3Max = 185.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```