1) Download and view the web cutter video “wc.mov” from the class web page. Run MATLAB video analysis code "wc\_dig.m" provided below. It will save the first video frame as image "wc\_first\_frame.jpg" and digitize x-y pixel locations for red and green dots on revolutes B and C respectively. Output will be provided in text file "wc\_keep.txt".

Use Microsoft Paint to digitize column-row locations for the centers of revolutes A and D from "wc\_first\_frame.jpg". Remember to convert column-row locations to x-y pixel locations as shown at the bottom of the MATLAB code.

2) Plot angles  and  as functions of angle  as defined below.

3) Plot  as a function of time using simple finite difference and using Savitsky-Golay floating interpolants. Be certain to use units of rad/sec. Determine the mean value of  and root-mean-square (RMS) **about the mean value.**

mean \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ RMS about mean  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXTRA CREDIT**

Fit a circle to x-y data for revolute B and compare the center to your manual measurement.

Ax,Ay manual \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ax,Ay circle \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**C**

**B**

****

**P**

**Web**

**Q**

**VWEB**

**A**

**D**

****

****

****

**2**

**3**

**4**

AB = 4.00 cm

BC = 14.23 cm

CD = 20.32 cm

AD = 13.36 cm

% wc\_dig.m - digitize red/green dots on web cutter from MOV video

% HJSIII, 19.12.03

clear

% video file name

fn\_input = [ 'wc.mov' ];

% create video file reader object

VR\_obj = VideoReader( fn\_input );

% get video information

video\_fps = VR\_obj.FrameRate; % frames per second

video\_duration = VR\_obj.Duration; % sec

%video\_frames = VR\_obj.NumberOfFrames; % must recreate object to rewind after using NumberofFrames

%video\_width = VR\_obj.Width;

%video\_height = VR\_obj.Height;

% expected number of frames

nframe = round( video\_duration \* video\_fps );

fps = round( video\_fps );

disp( ' ' )

disp( [ 'reading ' num2str(nframe) ' frames recorded at ' num2str(fps) ' frames per second' ] )

disp( ' ' )

disp( 'see Figure 1' )

disp( ' ' )

% step through video

iframe = 0;

keep = [];

while hasFrame( VR\_obj )

a\_rgb = readFrame( VR\_obj ); % "readFrame" returns class uint8

[ nr, nc, nk ] = size( a\_rgb );

iframe = iframe + 1;

% save first image

if iframe==1,

imwrite( a\_rgb, 'wc\_first\_frame.jpg' )

end

% convert to CIE L\*a\*b\*

% L\* intensity 0=dark, 100=bright - a\_lab(:,:,1)

% a\* green<0, red>0 - a\_lab(:,:,2)

% b\* blue<0, yellow>0 - a\_lab(:,:,3)

a\_lab = rgb2lab( a\_rgb ); % size (nr,nc,3) - class double

% find red pixels

bw\_r = ( a\_lab(:,:,2) > 40 ); % size (nr,nc) - class logical

% find green pixels

bw\_g = ( a\_lab(:,:,2) < -30 ); % size (nr,nc) - class logical

% find centroid of one object in each black/white image

s\_r = regionprops( bw\_r, 'Centroid' ); % class structure

s\_g = regionprops( bw\_g, 'Centroid' );

% column and row stored in structure.Centroid

cr\_r = s\_r.Centroid; % size (1,2) - class double

cr\_g = s\_g.Centroid;

% new figure

figure( 1 )

clf

warning( 'OFF', 'images:initSize:adjustingMag' ) % disbale warning for large images

% RGB image in UL

subplot( 2, 2, 1 )

imshow( a\_rgb )

title( [ 'frame ' num2str(iframe) ] )

% BW image for red in LL

subplot( 2, 2, 3 )

imshow( bw\_r )

title( 'red a\*>40' )

hold on

plot( [ 0 cr\_r(1) ], [ 0 cr\_r(2) ], 'r' ) % line from origin to centroid

% BW image for green in LR

subplot( 2, 2, 4 )

imshow( bw\_g )

title( 'green a\*<-30' )

hold on

plot( [ 0 cr\_g(1) ], [ 0 cr\_g(2) ], 'g' ) % line from origin to centroid

% update graphics

drawnow

% save centroids

keep = [ keep ; [ cr\_r cr\_g ] ];

end % bottom - while hasFrame

% row number increases in negative y direction

keep(:,2) = nr - keep(:,2);

keep(:,4) = nr - keep(:,4);

% show x-y results

figure( 2 )

clf

plot( keep(:,1),keep(:,2),'r', keep(:,3),keep(:,4),'g' )

axis equal

% save to TXT file - x\_red y\_red x\_green y\_green

save( 'wc\_keep.txt', 'keep', '-ascii' )

% bottom - wc\_dig

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**circle fit for unknown radius**

 

  



**circle fit for known radius**

  