

3.2.1 main.m

```
clear;
clc;

warning('off','all');

% Desired Diameter
[desireddiameter] = getdata();

% Set up variables
URL1 = 'ece480group5-cam.dhcp.egr.msu.edu';
URL2 = 'plazacam.studentaffairs.duke.edu';

% Set up camera
camera = cv.VideoCapture;
camera.open(['http://' URL1 '/mjpg/video.mjpg']);
pause(1); % Necessary in some environment. See help cv.VideoCapture

% Set up display window
window = figure('KeyPressFcn',@(obj,evt)setappdata(obj,'flag',true));
setappdata(window,'flag',false);

% Start main loop
while true
    tic

    % Grab and process an image
    im = camera.read;
    subplot(2,2,1); subimage(im); title('Original Image');

    % Segment the original image
    [bw_done, bw_thres] = segmentimage(im);

    % Calculate the diameter of the blob within the inputted image
    [pixeldiameter, actualdiameter] = measurediameter(bw_done);

    % Changes the color of the bounding box if the desired diameter is achieved
    if pixeldiameter >= desireddiameter
```

```

        boxcolor = 'g'; else boxcolor = 'r'; end

    % Displays a Bounding Box around the blob in the most recent image created
    [x,y] = createboundingbox(bw_done, boxcolor); % Function must follow an image to
display bounding box

    subplot(2,2,2); subimage(bw_thres); title('Thresholded Binary Image');
    subplot(2,2,3); subimage(bw_done); title('Result of Segmentation and Noise Removal');

    % Displays a ruler that automatically updates
    dispruler(x,y,pixeldiameter);

    clc;

    % Contains TCP/IP code for communicating with L2 Computer
    senddata(actualdiameter);
    disp(desireddiameter)

    % Terminate if any user input
    flag = getappdata(window,'flag');
    if isempty(flag)||flag, break; end

    pause(.000000000000000001);

    toc
end
% Close
close(window);

```

3.2.2 cv.videocapture.m

```

classdef VideoCapture < handle
    %VIDEOCAPTURE VideoCapture wrapper class
    %
    % Class for video capturing from video files or cameras. The class

```

```

% provides Matlab API for capturing video from cameras or for reading
% video files. Here is how the class can be used:
%
%   cap = cv.VideoCapture;
%   pause(3); % Note below
%   for t = 1:30
%       imshow(cap.read);
%       pause(0.1);
%   end
%
% ## Note
% In some environment, there is a concurrency issue during camera
% initialization. To avoid unexpected crash, pause for seconds after
% the initialization of VideoCapture object.
%
% See also cv.VideoCapture.VideoCapture cv.VideoCapture.read
% cv.VideoCapture.get cv.VideoCapture.set
%

```

```

properties (SetAccess = private)
% Object ID
id
end

```

```

methods
function this = VideoCapture(filename)
%VIDEOCAPTURE Create a new VideoCapture object
%
%   cap = cv.VideoCapture
%   cap = cv.VideoCapture(devid)
%   cap = cv.VideoCapture(filename)
%
% VideoCapture create a new instance. With no argument, it
% connects to the default camera device found in the system.
% You can specify camera devices by devid, an integer value
% starting from 0. You can also specify a filename to open a
% video file.
%
% See also cv.VideoCapture

```

```

%
if nargin < 1, filename = 0; end
this.id = VideoCapture_(filename);
end

function delete(this)
%DELETE Destructur of VideoCapture object
VideoCapture_(this.id, 'delete');
end

function frame = read(this)
%READ Grabs, decodes and returns the next video frame
%
% frame = cap.read()
%
% The method captures the next video frame and return it. If
% capturing fails, empty array will be returned instead.
%
% See also cv.VideoCapture
%
frame = VideoCapture_(this.id, 'read');
end

function value = get(this, key)
%GET Returns the specified VideoCapture property
%
value = cap.get('FPS');
%
% The method returns a Property value of VideoCapture.
% PropertyName can be one of the followings:
%
% 'PosMsec'          Current position of the video file in milliseconds or video capture
timestamp.
% 'PosFrames'       0-based index of the frame to be decoded/captured next.
% 'AVIRatio'        Relative position of the video file: 0 - start of the film, 1 - end of
the film.
% 'FrameWidth'      Width of the frames in the video stream.
% 'FrameHeight'     Height of the frames in the video stream.
% 'FPS'             Frame rate.

```

```

% 'FourCC'      4-character code of codec.
% 'FrameCount'  Number of frames in the video file.
% 'Format'      Format of the Mat objects returned by retrieve() .
% 'Mode'        Backend-specific value indicating the current capture mode.
% 'Brightness'  Brightness of the image (only for cameras).
% 'Contrast'    Contrast of the image (only for cameras).
% 'Saturation'  Saturation of the image (only for cameras).
% 'Hue'         Hue of the image (only for cameras).
% 'Gain'        Gain of the image (only for cameras).
% 'Exposure'    Exposure (only for cameras).
% 'ConvertRGB'  Boolean flags indicating whether images should be converted to
RGB.
% 'Rectification' Rectification flag for stereo cameras (note: only supported by DC1394
v 2.x backend currently)
%
% See also cv.VideoCapture
%
value = VideoCapture_(this.id, 'get', key);
end

function set(this, key, value)
%SET Sets a property in the VideoCapture.
%
    cap.set('FPS', value)
%
% The method set a Property value of VideoCapture.
% PropertyName can be one of the followings:
%
% 'PosMsec'      Current position of the video file in milliseconds or video capture
timestamp.
% 'PosFrames'    0-based index of the frame to be decoded/captured next.
% 'AVIRatio'     Relative position of the video file: 0 - start of the film, 1 - end of
the film.
% 'FrameWidth'   Width of the frames in the video stream.
% 'FrameHeight'  Height of the frames in the video stream.
% 'FPS'          Frame rate.
% 'FourCC'      4-character code of codec.
% 'FrameCount'  Number of frames in the video file.
% 'Format'      Format of the Mat objects returned by retrieve() .

```

```

% 'Mode'          Backend-specific value indicating the current capture mode.
% 'Brightness'    Brightness of the image (only for cameras).
% 'Contrast'      Contrast of the image (only for cameras).
% 'Saturation'    Saturation of the image (only for cameras).
% 'Hue'          Hue of the image (only for cameras).
% 'Gain'         Gain of the image (only for cameras).
% 'Exposure'      Exposure (only for cameras).
% 'ConvertRGB'    Boolean flags indicating whether images should be converted to
RGB.
% 'Rectification' Rectification flag for stereo cameras (note: only supported by DC1394
v 2.x backend currently)
%
% See also cv.VideoCapture
%
VideoCapture_(this.id, 'set', key, value);
end
end

methods (Hidden = true)
function successFlag = open(this, filename)
%OPEN Open video file or a capturing device for video capturing
%
% ## Output
% * __retval__ bool, true if successful
%
% The methods release() to close the already opened file or camera.
% Parameters are the same as in the constructor.
%
% See also cv.VideoCapture.isOpened
%
if nargin < 1, filename = 0; end
successFlag = VideoCapture_(this.id, 'open', filename);
end

function flag = isOpened(this)
%ISOPENED Returns true if video capturing has been initialized already.
%
% ## Output
% * __retval__ bool, return value

```

```

%
% If the previous call to constructor or open() succeeded, the method returns true.
%
% See also cv.VideoCapture.open
%
flag = VideoCapture_(this.id, 'isOpened');
end

function release(this)
%RELEASE Closes video file or capturing device.
%
% The methods are automatically called by subsequent open() and by destructor.
%
% See also cv.VideoCapture.open
%
VideoCapture_(this.id, 'release');
end

function successFlag = grab(this)
%GRAB Grabs the next frame from video file or capturing device.
%
% ## Output
% * __successFlag__ bool, success flag
%
% The function grabs the next frame from video file or camera and returns
% true (non-zero) in the case of success.
%
% The primary use of the function is in multi-camera environments, especially
% when the cameras do not have hardware synchronization. That is, you call
% grab() for each camera and after that call the slower method retrieve() to
% decode and get frame from each camera. This way the overhead on demosaicing
% or motion jpeg decompression etc. is eliminated and the retrieved frames
% from different cameras will be closer in time.
%
% See also cv.VideoCapture.read cv.VideoCapture.retrieve
%
successFlag = VideoCapture_(this.id, 'grab');
end

```

```

function frame = retrieve(this)
%RETRIEVE Decodes and returns the grabbed video frame.
%
% The function decodes and returns the just grabbed frame. If no frames has
% been grabbed (camera has been disconnected, or there are no more frames
% in video file), the function returns an empty matrix.
%
% See also cv.VideoCapture.read cv.VideoCapture.grab
%
    frame = VideoCapture_(this.id, 'retrieve');
end
end
end

```

3.2.3 createboundingbox.m

```

function [x,y] = createboundingbox(inputimage,boxcolor)
% inputimage is of type mat
% boxcolor is of type str
% n is the number of bounding boxes of type num

st = regionprops(inputimage,'BoundingBox');
    for k = 1:length(st)
        thisBB = st(k).BoundingBox;
        x = thisBB(1);
        y = thisBB(2);
        w = thisBB(3);
        h = thisBB(4);
        rectangle('Position', [x,y,w,h],'EdgeColor',boxcolor,'LineWidth',2 );
    end
end
end

```

3.2.4 senddata.m

```

function [] = senddata(actualdiameter)

```

```

% actualdiameter is of type float

[YYYY,DD,MM,HH,MN,SS] = gettime();

actualdiameterstring = num2str(actualdiameter,'%1f');
dataout = ['$CAD,' YYYY '-' DD '-' MM ',' HH ':' MN ':' SS ',2,' actualdiameterstring '<CR>'];

disp('Data being sent to the L2 Comp:')
disp(dataout)

end

```

3.2.5 gettime.m

```

function [ year,day,month,hour,min,sec ] = gettime()
% All outputs of type str

t = fix(clock);
    year = num2str(t(1)); %Year
    day = num2str(t(2)); %Day
    month = num2str(t(3)); %Month
    hour = num2str(t(4)); %Hour
    min = num2str(t(5)); %Minute
    sec = num2str(t(6)); %Second

end

```

3.2.6 getdata.m

```

function [desirediameter] = getdata()

tcpipClient = tcpip('localhost',55000,'NetworkRole','Client');
set(tcpipClient,'InputBufferSize',320);
set(tcpipClient,'Timeout',30);
fopen(tcpipClient);
rawData = fread(tcpipClient,40,'char');
fclose(tcpipClient);

```

```

reshapedData = reshape(rawData,1,40);
temp=char(reshapedData(1, 28:31));
desireddiameter = str2double(temp);

```

```

end

```

3.2.7 tcpserver.m

```

data = '$L2D,2014-01-05,15:49:03,2,68.3,1237<CR>';
s = whos('data');
tcpipServer = tcpip('0.0.0.0',55000,'NetworkRole','Server');
set(tcpipServer,'OutputBufferSize',96);
fopen(tcpipServer);
fwrite(tcpipServer,data(:),'char');

```

3.2.8 dispruler.m

```

function [] = dispruler(x,y,length)
% x is of type int
% y is of type int
% length is of type int

imdistline(gca,[x x+length],[y-100 y-100]);

end

```

3.2.9 measurediameter.m

```

function [pixeldiameter,actualdiameter] = measurediameter(image)
% image is of type mat
% pixeldiameter is of type float
% diameter is of type float

temp = regionprops(image,'MajorAxisLength');
pixeldiameter = temp.MajorAxisLength; % segmentimage func should guarentee only 1 blob in
image at all times
% bwarea could be used for measurement

```

% MinorAxisLength could also be used

% Calibration code goes here

actualdiameter = pixeldiameter*.0517-0.002639;

end

3.2.10 segmentimage.m

function [bw3,bw1] = segmentimage(im)

% im is of type mat

% Remove the top black border. Necessary for Duke Live Cam

% im = im(40:end,:,:);

% Thresholding

level = 0.6;

bw1 = im2bw(im,level);

bw1(20:100,20:100) = 1;

% Noise removal

bw2 = imclearborder(bw1);

se = strel('disk',20);

bw3 = imopen(bw2,se);

% bwpropfilt is another option

% bwselect is another option

bw3 = bwareafilt(bw2,1); %This code is not supported with 2014a.

% Forces only largest blob to be displayed.

end