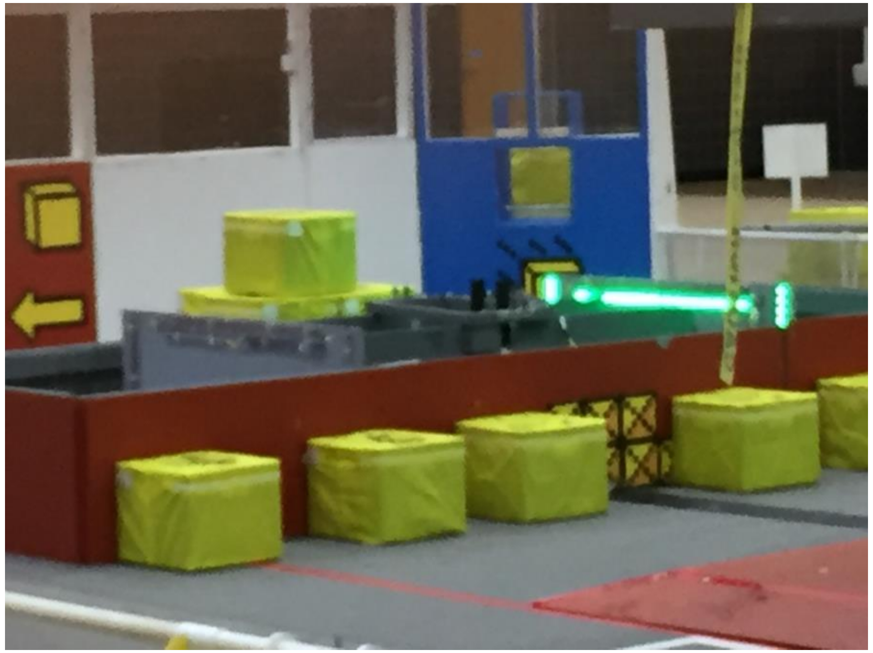
# 2018 Vision Design

Team 1259, Paradigm Shift

# Design Requirement

- Identify yellow cubic on images from camera;
- Determine the distance and angle of the cubic in front of robot
- Send the estimated distance and angle to RoboROI



# High Level Design

Step 1 - Capture images from a camera through Raspberry pi;

Step 2 - Identify Yellow Cubic based on color

Step 3 - Determine the location of Yellow Cubic on the image

Step 4 – Correct the distortion of the location caused by camera

Step 5 – Estimate actual distance and angle of the cubic in front of robot.

Step 6 – Send estimated information to RoboROI

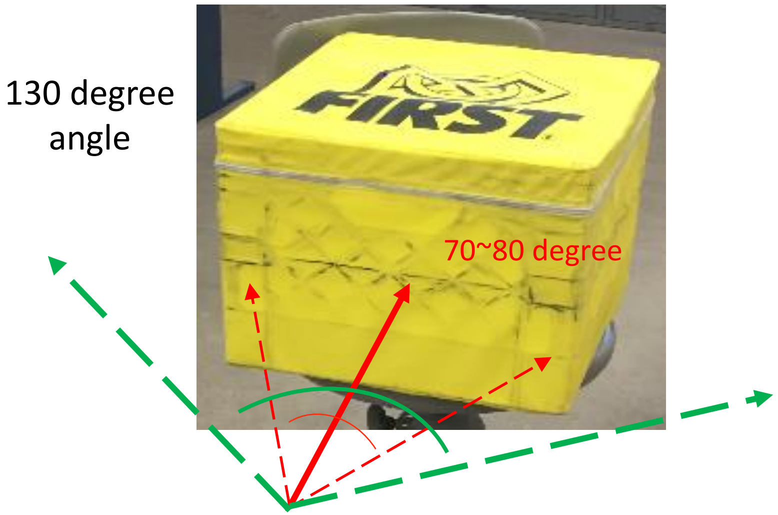
Other – Error handling

.

# Step 1 - Capture Images from A Camera

1. Determine the type of camera after mounting location is determined.

Decision: use wide angle camera to cover whole cubic on an image when the cubic in front of the Robot.



2. Determine frame rate of images to be captured by camera.

Decision: ~8 frames/sec which is faster enough to guide robotic motion

Concept:

- Motion speed = X cm /sec
- Required distance, Y cm, to be updated
- Frame rate = X / Y which determine how fast camera has to acquire images per second.

# Step 1 - Capture Images from a Camera

## Example

OpenCV function:

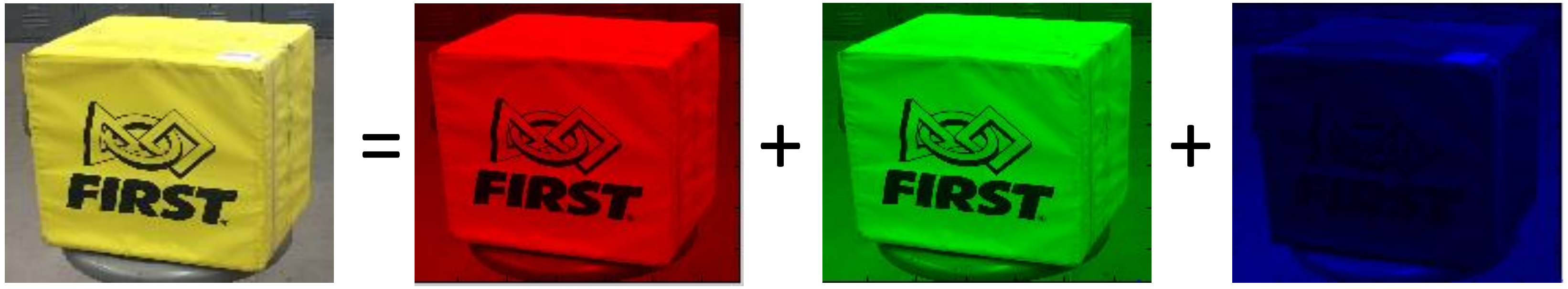
Camera.retrieve(image);

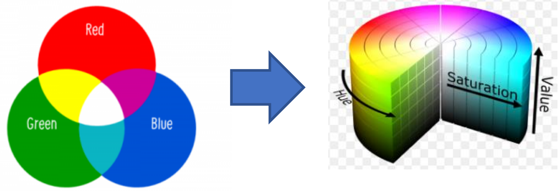
Default image size: 960x1280x3 where 3 indicates three images for Red, Green, and Blue.

# Step 2 - Determine Yellow Cube based on Color

- Images from camera have RGB (Red, Green, Blue) color.
- A lot of time, separating color per RGB image is not very easy due to different light brightness;
- To make color object be easily identified and more reliable detection, HSV (Hue, Saturation, Value) color system are used. It helps identify particular color at an environment with different light brightness.
- After identify the object with specified color, a binary image with (0, 1) value could be used to show the object for additional processing.

Images from camera



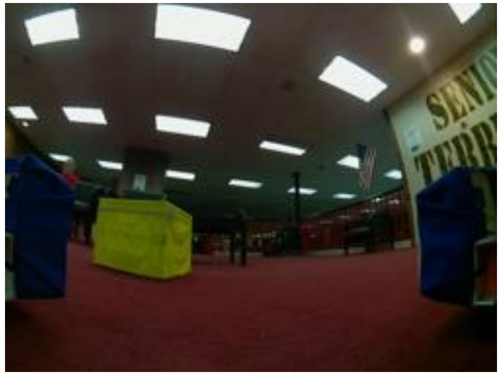


# Step 2 - Determine Yellow Cubic based on Color

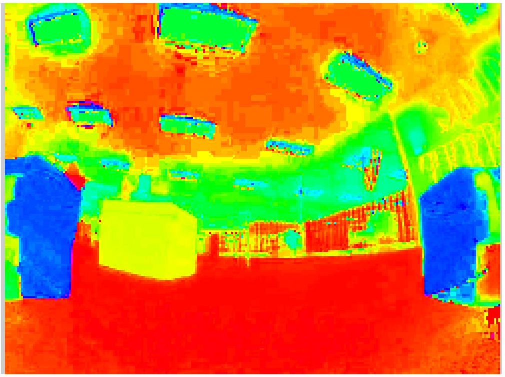
## Example

```
cvtColor(image, imageHSV, COLOR_BGR2HSV);     // Convert BGR to HSV
inRange(imageHSV, lower, upper, inrange);      // Identify color per HSV image
Where
 // HSV threshold to find all yellow
          const Scalar lower = Scalar(20, 200, 50);
          const Scalar upper = Scalar(40, 255, 255);
```

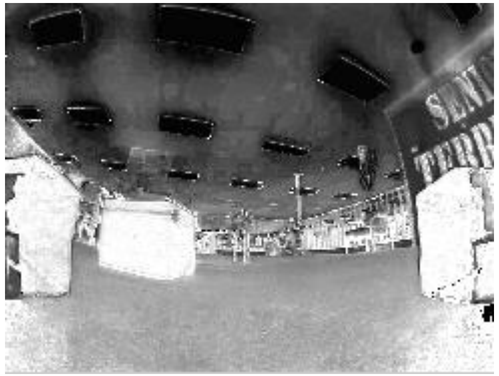
Original



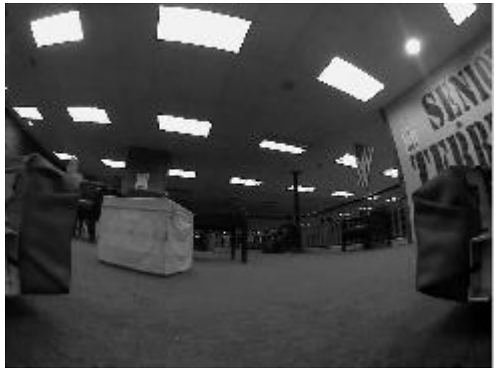
Hue



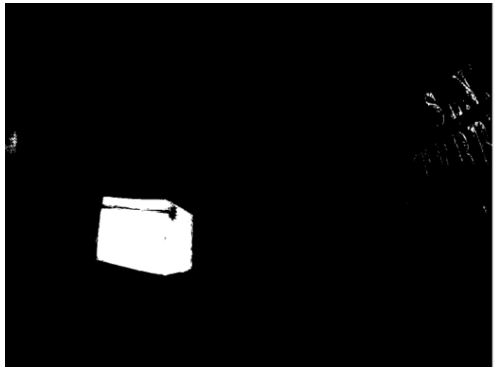
Saturation



Value



Binary image



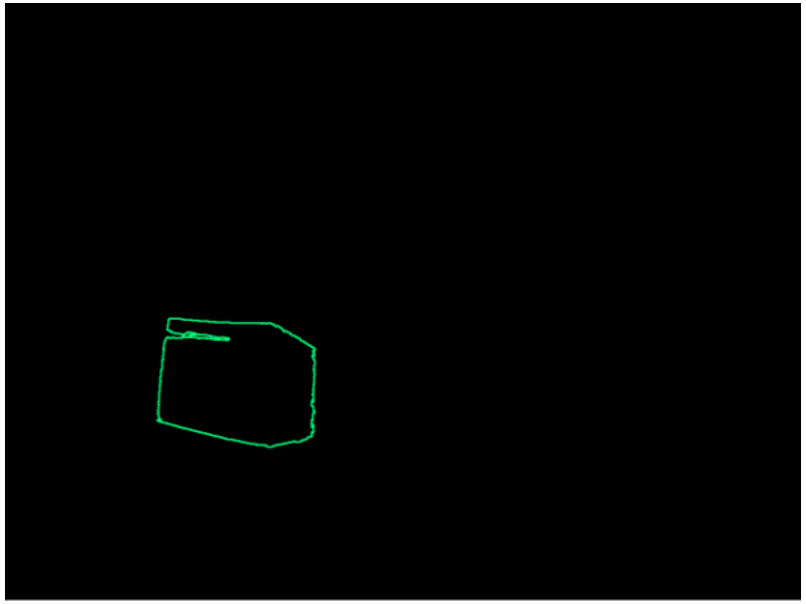
# Step 3 - Determine the Location of Yellow Cube on Image

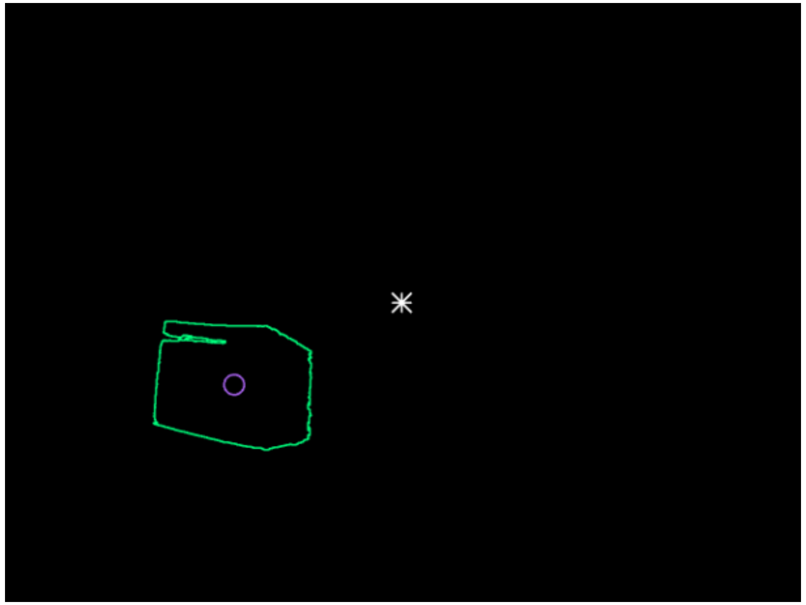
- Draw object contours
- Find a biggest contour
- Determine the center location of the Cubic contour per Moment function() & the center of image

**OpenCV functions**

```
findContours(inrange, contours, hierarchy, CV_RETR_LIST,CV_CHAIN_APPROX_NONE);
….
drawContours(drawing, contours,biggestContourLocation , Scalar(124, 252, 0), 2, 8,hierarchy,0);

Moments m = moments(contours[biggestContourLocation], true);
```

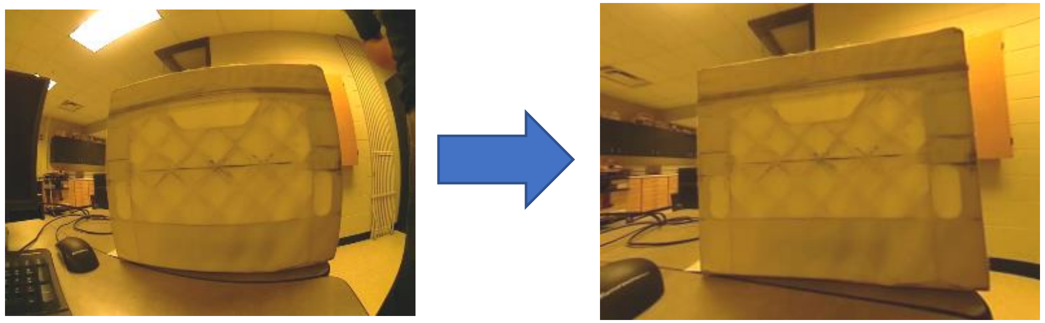




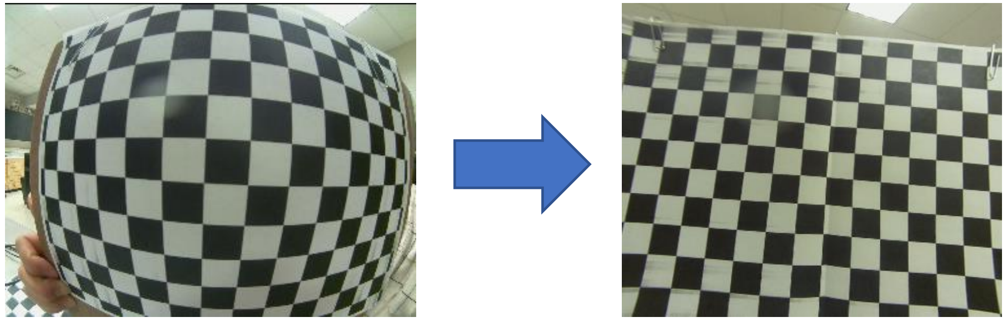
# Step 4 - Correct distortion of the location on Image

- By using a wide angle camera with Fisheye lens, distortion is typically occurred. This distortion looks like a sphere. So it is also named as barrel distortion. See examples
- Barrel distortion could be corrected based on following equation
  - Converts the x-y coordinates to polar coordinates (r, θ)
  - Calculate the maximum vector (image center to image corner) to be used  and then normalize polar coordinate r
  - Apply r-based transformation  s = r.*(1+k.*r), where k = 0.46 is determined per calibration w/ chess board image; Scale back to original unnormalized r and then convert (r, θ) back to cartesian coordinates (x, y)
- To speed up computation, only some coordinates (x, y) of the cubic on image are corrected instead of whole image to be corrected.

Test images



Calibration and test images



# Step 4 - Correct Distortion of the Cubic Location

## Example

Original Cube Center X: 372.523

Original Cube Center Y: 611.934

Original Cube Height[(*)]: 188

Image Center X: 640,

Image Center Y: 480

After distortion correction:

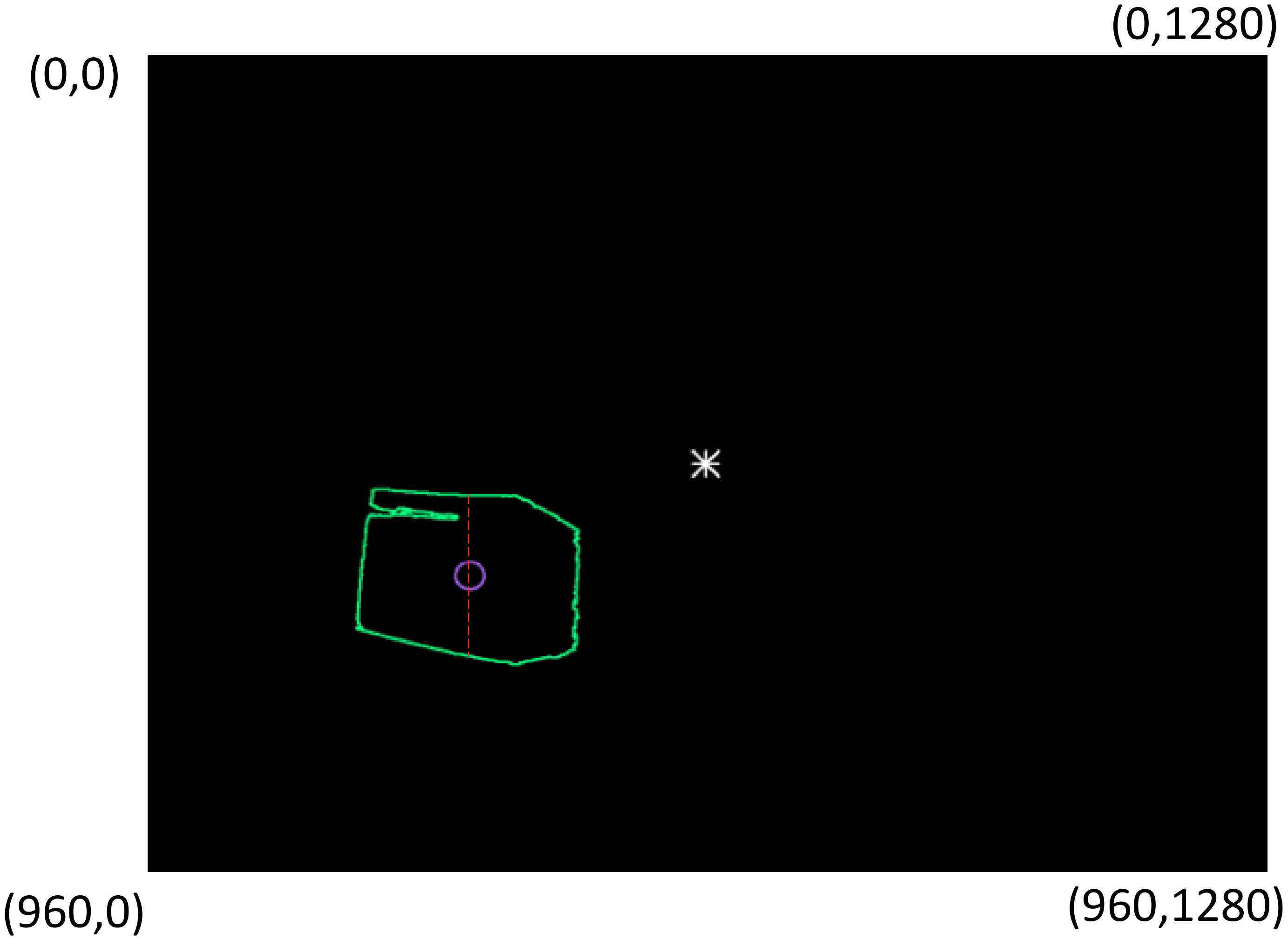
Corrected Cube Center X: 374.412

Corrected Cube Center Y: 611.002

Corrected Cube Height: 178.002

(*) Cubic height is determined by two coordinates on the contour

Center location of the Cubic contour and the center of image



# Step 5 - Estimate Actual Distance and Angle of the Cubic

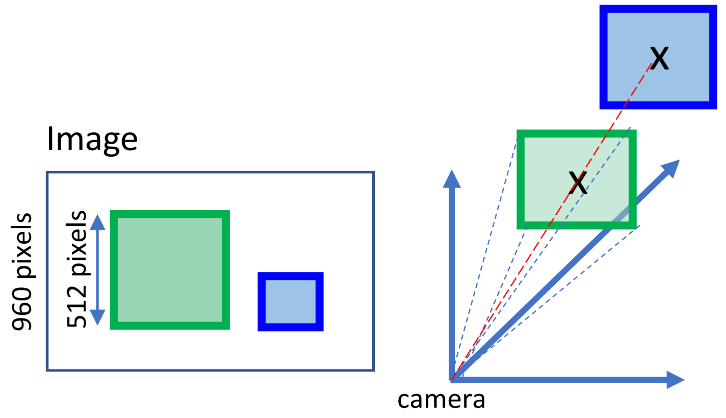
- Distance and angle of the cubic at different locations in front of camera could be reflected to the size of cube and its location on the image.
- At a fixed, known distance, the size of cube on image could be predetermined.

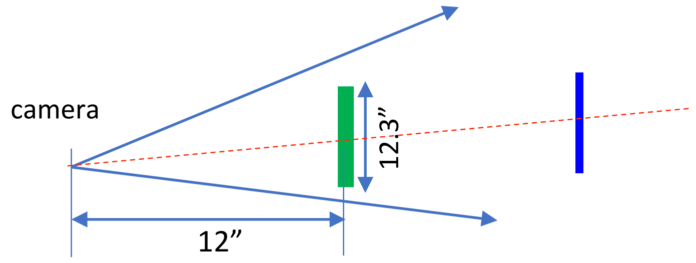
```
// Following settings is for camera calibrated values
DEFAULT_FOV_ROW_NUM            = 960;
DEFAULT_HEIGHT_PIXEL           = 510;
DEFAULT_PIXEL_PER_INCH         = 41.5;
CAL_DISTANCE_INCH              = 12;
…
```

Because we could configure camera to different size to trade of resolution vs. processing speed,
So, we could scale those calibrated value per image size. That is,

```
…
standard_height_p = DEFAULT_HEIGHT_PIXEL /(DEFAULT_FOV_ROW_NUM/drawing.size().height);
pixel_per_in = DEFAULT_PIXEL_PER_INCH/(DEFAULT_FOV_ROW_NUM/drawing.size().height);
```

Example:
If image size is 640x480, half of original image size for calibration,
(1) standard height of cube in pixel is 255 (= 510 / (960/480)) at calibration distance
(2) Pixel per inch is also half of original, 20.75, ( = 41.5 / (960/480))





# Step 5 - Estimate Actual Distance and Angle of the Cubic

- Determine the center location of the Cubic contour relative to the image center

**Total_Distance_Inch** = ((standard_height_p/cube_height)*CAL_DISTANCE_INCH);

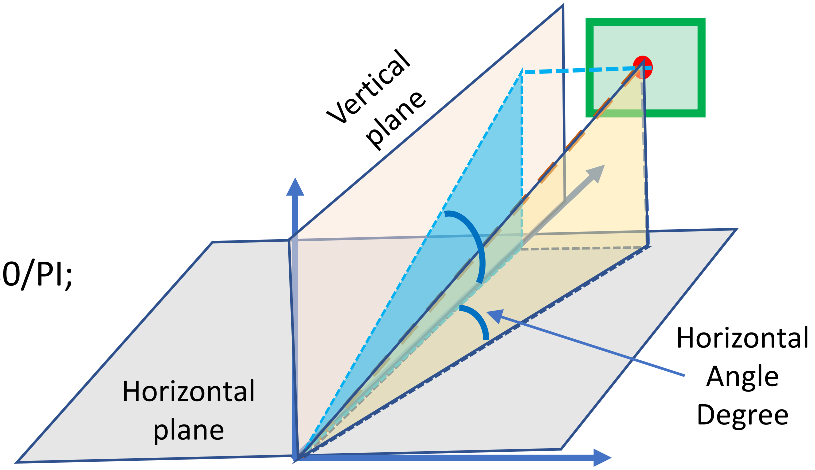
Horizontal_Distance_Pixel = cube_center_x - im_center_x;

Vertical_Distance_Pixel = im_center_y - cube_center_y;

Horizontal_Angle_Degree = atan(Horizontal_Distance_Pixel/(pixel_per_in*CAL_DISTANCE_INCH))*180/PI;

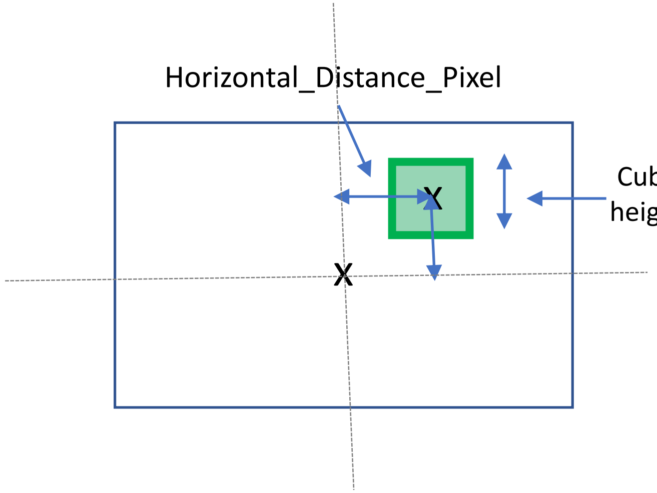
Vertical_Angle_Degree = atan(Vertical_Distance_Pixel/(pixel_per_in*CAL_DISTANCE_INCH))*180/PI;

Forward_Distance_Inch =
Total_Distance_Inch*cos(Vertical_Angle_Degree*PI/180)*cos(Horizontal_Angle_Degree*PI/180);



**“Forward Distance in inch” and “Horizontal angle in degree” are considered.**

**Another option is to send**
“Total_Distance_Inch*cos(Vertical_Angle_Degree*PI/180)”



**Note that angle and distance are updated per every image**

# Step 5 - Estimate Actual Distance and Angle of the Cubic Example

All computed parameters

| | |
|---|---|
| Horizontal Distance in Pixel: | -265.588 |
| Vertical Distance in Pixel: | -131.002 |
| Forward Distance in Inch: | 29.3389 |

| | |
|---|---|
| **Horizontal Angle in Degree:** | **-28.0714** |
| Vertical Angle in Degree: | -14.7381 |
| Total Distance in Inch : | 34.3817 |

| | |
|---|---|
| Horizontal Distance in Inch: | -16.179 |
| Vertical Distance in Inch: | -8.74675 |
| **Forward Distance in Inch:** | **29.3389** |

# Step 6 – Send Data to RoboRIO

- Initialize Network table on RoboROI
- Complete the calculation of angle and distance
- Send angle, distance, and vision counter to network table on RoboROI

- **Example of code**

```
nt_Inst = NetworkTableInstance::GetDefault();
nt_Inst.StartClientTeam(1259);
…
netTable->PutNumber("visioncounter", counter);
netTable->PutNumber("XOffAngle", Horizontal_Angle_Degree);
netTable->PutNumber("Forward_Distance_Inch", Forward_Distance_Inch);
…
```

# Other -- Error Handling

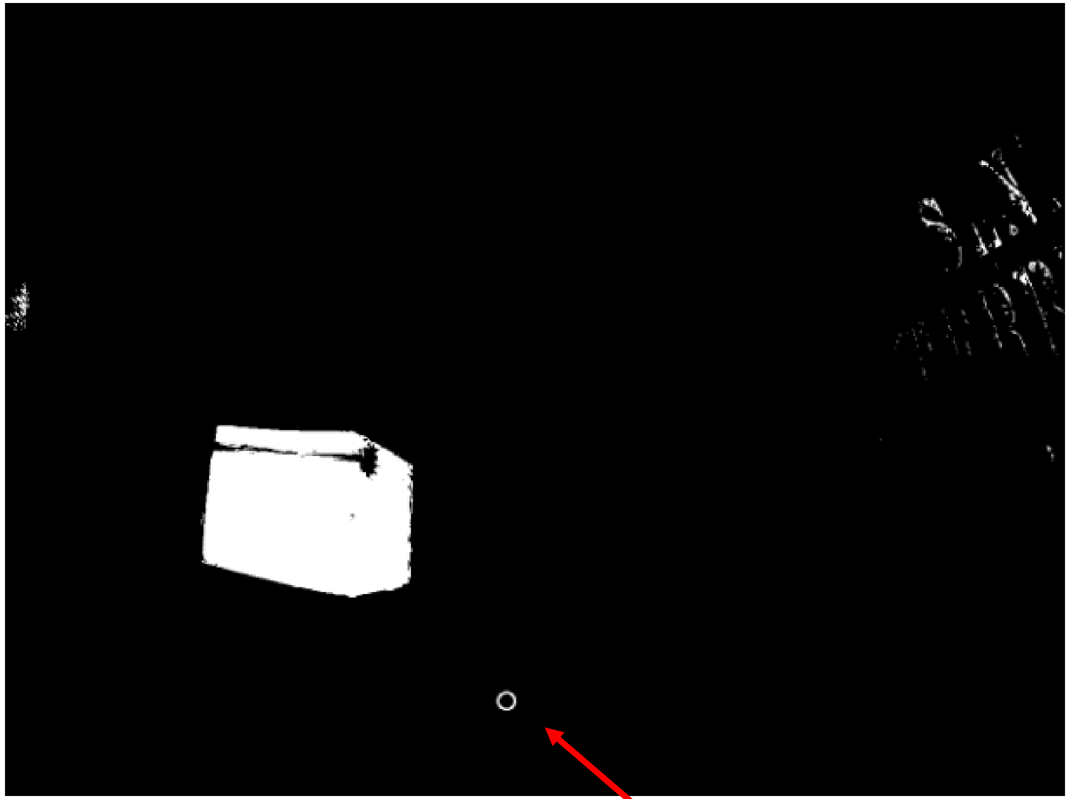
1. Most of time, camera might not get a yellow cube on the image or camera has a lens cover which will have a dark image.

   In that case, OpenCV function to find contour or its center location will be wrong.

   To avoid this error, small circle object is added to image to prevent calculation error. However, this circle object is too small as compared to yellow cube and will be removed later.

2. Camera needs to warm up and gets reasonable image brightness. Set 60 frames at the beginning to stabilize the brightness.

3. Add output check such as angle < 60 degree, distance < 20 feet, and sum of cube contour pixel > 100 pixels to make sure that a valid result is from the image.



Intended to add a small circle to prevent calculation error