The International Journal Of Engineering And Science (IJES)
|| Volume || 6 || Issue || 1 || Pages || PP 39-50|| 2017 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Distance Estimation to Image Objects Using Adapted Scale

M. Zuckerman(1) Intel Corp., Israel
E. Kolberg(2)* Bar-Ilan University, Israel

ABSTRACT

Distance measurement is part of various robotic applications. There exist many methods for this purpose. In this paper, we introduce a new method to measure the distance from a digital camera to an arbitrary object by using its pose (X,Y pixel coordination and the angel of the camera). The method uses a ***pre-data*** *that stores all the information about the relation between the pose and the distance of an object to the camera. This process designed for a robot that is a part of a robotic team participating in RoboCup KSL competition.*

Date of Submission: 12 January 2017 Date of Accepted: 05 February 2017

I. INTRODUCTION

Distance measurement from a camera to an arbitrary object is widely needed. We use the fact that an object looks larger as it gets closer to the camera. That means that the same object when close to the camera will have more pixels representing it than if it was located further. It allowed us to develop a robust mapping method for measuring a distance to an object.

Among the common methods used to yield distance measurement from image processing we find Pinhole camera model [1, 2], Stereo vision [3, 4] and object volume [5, 6]. In [2] distance measurements are based on footprint size. In [1] the distance to an object is derived using the laws of perspective and depends on the road geometry and the point of contact between the vehicle and the road. This method is used in order to estimate a distance to a car or a truck that are an order of magnitude bigger than the objects we deal with. Measuring a distance of 90 meters, the error is 10%. The object size and the dependency of the road geometry using a pinhole camera is therefore less adequate in our case. In [2] the method is again deals with vehicles and focus on the detection of preceding vehicles. It takes an area in an image bounded by lane marks, (taking advantage that lane marks are at 45° when they appear in the image), and execute a logical-AND operation with the binary road image with no lane marks. It also takes advantage of the non symmetrical shape of a vehicle in order to verify the vehicle footprint. These two last methods while adequate for vehicle identification are less useful for the case described in this paper. In [3] the distance measurement is based on stereoscopic pictures. It depends on horizontally aligned cameras, taking pictures at the same time, parallel axis of the two cameras. As the distance is longer, the error grows in proportion to the squared distance. In the case we describe here, the robot moves so a static infrastructure is not relevant here. In addition, we need to reduce the error even in long distances. In [4] there is a comparison between camera and human depth accuracy. They concluded that the focal length affected the depth resolution remarkably. Wide-angle lens deteriorated the focal resolution as well. The calibration errors added to the overall error. In [5] the authors describe a method for measure distance to thin and high pole like objects. Each pole needs three alignment measurements for each placement. While this is fine for a pole positioned in a single static position, it is less adequate for our case. In [6] the method for distance calculation has two steps. First, calculating an interpolation function based on the height and the horizontal angle of the camera. Second, using this function to calculate the distance of the object from the camera. This method relies upon the camera height and horizontal angle. The paper shows results of distances up to 1m. It is based on interpolation, which in our case might increase the error. Using laser like the one presented in [7] integrated with a camera can increase accuracy on a static system as described there. Besides the fact that it is forbidden for use in the competition, it will require substantial design and additional hardware and software for using it on a walking robot. In [8] plate size of car number is used to estimate distance to a car. While this method might be adequate for distance from a car, it proved to be less useful in our case. The method described in [9], deals with a flat board of size 750mm x 750 mm, with target points on it. This target is too big for our case. The focal length and camera's lens diameter along with object base size were used to derive the distance to an object. The limitations of this method include relevancy for exact camera model in combination with particular lens, and target should be in optical axes.

In this paper we will present the volume solution for deriving a distance to an object. The *Mapping method* is based on the idea of the volume solution.

II. VOLUME SOLUTION

When dealing with known objects, it will be useful to measure the object size with different distances, as it appears on the image. As close the object to the camera as bigger it appears. The object, which appears on an image, is called *reflected object*. This information needs to be recorded and saved as a function [5] [6] . We took measurements of distance versus radius of a ball with a radius of 5 cm. We shot images out of Logitech C905 camera with 1280x720 pixels. Figure 1 presents a graph of the measurements we took. The graph looks similar to the one presented by Sant'Ann et. al [7] . In our example, the software program identified the ball. Since the ball was a known object, the program could use pre-stored data about ball diameter, ball color, and ball center. While these parameters are known it is possible to use a scale tool that measures the different radius sizes as appear on the image.

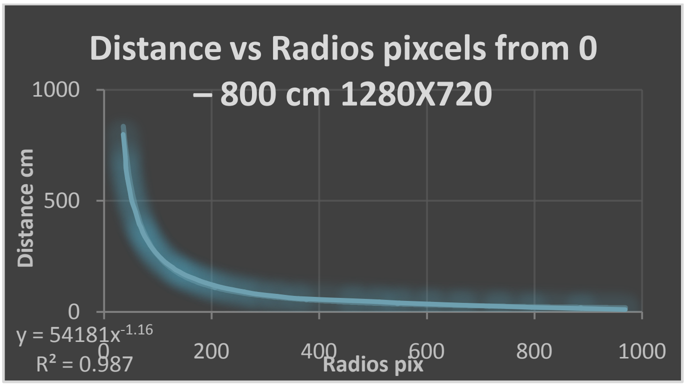


Figure 1: Distance Vs Radios

From figure 1 we see that the reflected object distance is an exponential function of the number of its radius pixels. This means that when the ball is far from the camera from a certain distance, it will have relatively less changes in number of pixels. On the other hand, the sensitivity will increase as the object is getting close to the camera, since the reflected object radius will have substantially less pixels in comparison to a closer distance when the ball is close to the camera.

From figure 1, we can see that there is a correlation between the number of pixels of the object radius in the image and the distance of the object from the camera in the real world.

In this particular graph, the correlation is presented by the function $y = 54181x^{-1.164}$, when y stands for the distance and x stands for the number of pixels. Next step is to determine the accuracy of the correlation. It will be presented by the error between the real distance and the computed one.

III. VOLUME SOLUTION ACCURACY

The function accuracy depends on the following factors.

3.1 Camera's angle aimed to an object

Objects in image might look different from various angles and different lighting conditions. One common method of identifying objects is first saving an image as a binary picture and then count the white pixels of an object like the radius of a ball or the width of a goal in a soccer game. The accuracy of the function will depend on the way the object is presented. Figure 2 presents the error ratings obtained when we tested the distance from the camera to a 10cm diameter orange ball on a green field.

Table 1 presents the average distance error of short and long distance areas in the graph.

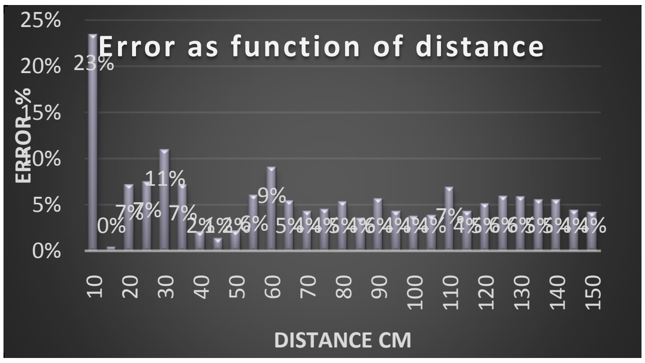


Figure 2: Error as function of distance orange

Table 1: Average error values of distance from a camera to a 10 cm dia. ball, calculated from B/W image

Distance	Error values
Short distance 0-55 cm	7%
Long distance 60-150 cm	5%

The values presented in table 1 are reasonable for our purpose of a soccer game since they are close enough to the real world. When we tried the same technique on the identification of the goal, it was not as robust. One reason to this phenomenon is that the ball looks round from almost any angle and light conditions. However, the goal might look different from different angles. The gap between the goal's poles will look different when the camera looks at the goal from different angles. The shape of the goal in these cases might look as a part of a parallelogram, or a triangle or a trapezoid, etc.
It leads to a conclusion that volume solution by counting white pixels is valid only for a perfect rounded shape or a simple shape like an equilateral triangle. In case of different objects, the error might increase substantially.

3.2 Light conditions

Different light conditions might change objects' edges as appears in the reflected image. With brighter illumination, white objects like white lines or white poles may look larger in the reflected image than they are in reality. Shadows or hiding a part of an object might also considered as different light conditions.

3.2 Filter type

The type of filter will have also an impact on the accuracy of the distance. In some cases, HSV will be used in others, RGB, etc. The filter types of edge detection will also have an impact on the accuracy.
3.3. Approximation function. Tinnachote and Pimprasen [5], suggest using a polynomial approximation function in order to calculate the distance, in a case of exponential function. We found that while it improves the results, it still was not proper to our purposes. We came out with the idea of further dividing the graph into smaller sections, which makes it more delicate for better approximation. In this specific example, we divided the graph into three sections. In each section, we used a different type of approximation function:
1. For ball distance of 0 to 55 cm – linear function as presented in figure 3.
2. For ball distance of 60 to 150 cm – 2nd order polynomial function as presented in figure 4
3. For ball distance of 160 cm and above – exponential function as presented in figure 5
The resulted distance error is presented in figure 6. The error decreased. The average distance error of the divided function was 3% compared to 6% average distance error presented in figure 1.

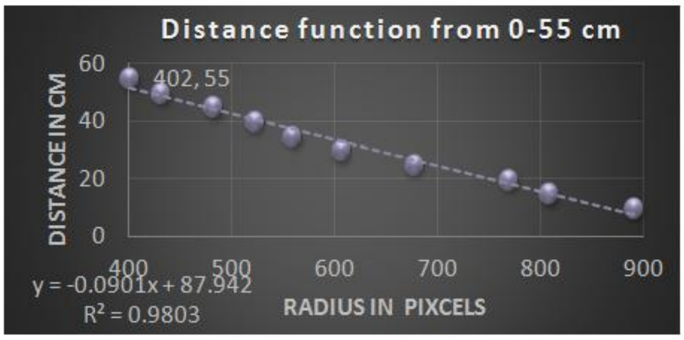


Figure 3: linear function of section 1

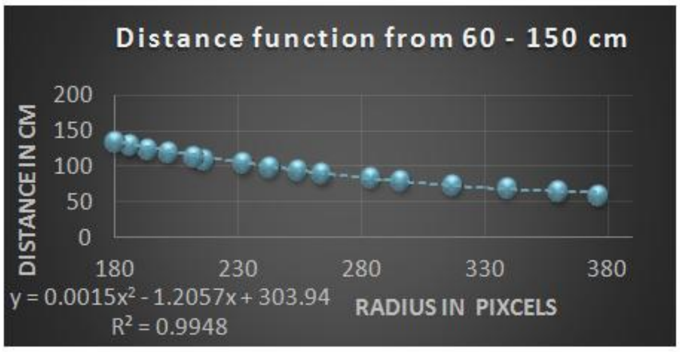


Figure 4: 2nd order polynomial function of section 2

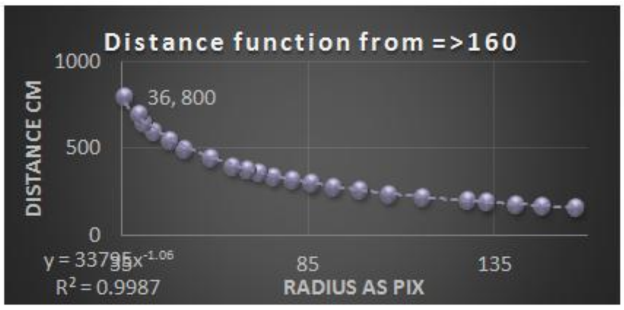


Figure 5: exponential function of section 3

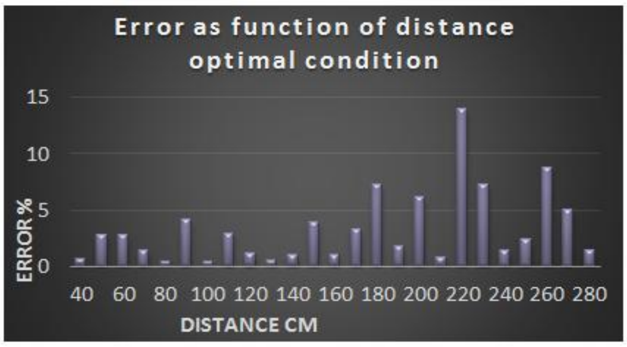


Figure6: Error as function of distance: average error of 3%

3.4 Color/filter calibration

Before analyzing an image, it is translated from a colored image into a B/W image. One of the tools used for this purpose is HSV/RGB threshold. With a proper threshold, it is possible to identify object types. Notwithstanding, it is somewhat problematic. For instance, if we allow for a low threshold, the object will be seen larger compared to the real object. Increasing or decreasing the value of the HSV threshold, will change the object's area.

IV. NEW METHOD'S IDEA

Since distances to objects are important values as inputs to a localization system in general and for RoboCup soccer game in particular, it was required to develop a reliable distance measuring method that is independent as much as possible of the factors described above.

After consideration of various options like using more expensive cameras, developing more complex algorithm, etc., we decided upon an algorithm that take as an input the location (x,y) of the object in 2-D Cartesian coordinate system. The (x,y) coordinates is the only needed input to the algorithm. In addition, the algorithm requires building in advance, a lookup table that transfers these coordinates into distances. Using such a lookup table makes it unnecessary to use an approximation function or count white pixels. Actually, the algorithm is independent of the binary picture. The only input needed is the (X, Y) coordinates of the object.

In order to implement the lookup it is necessary to find a correlation between any point coordinates (X, Y) and the distance to the point. For this purpose, we chose to use a scale.

After creating the scale, it is easy to build the lookup table. Then it is straightforward to implement the algorithm that deduce the distance to a point with its Cartesian coordinates. Next we will describe the two scales we built during the development process.

V. SCALE VERSION 1.0 (Y DEPENDENT)

The scale is made of black and white stripes. We developed a specific software in order to identify the black and white scale. We added numbers on the scale, which correlates to the distance between the camera and the object.

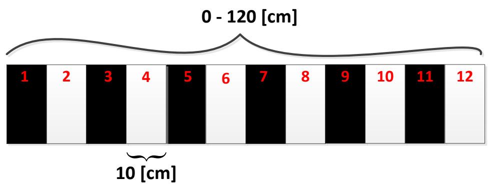


Figure 7: Scale 1.0 with is 10[cm] stripes

VI. THE LOOKUP TABLE

The lookup table saves the output information coming from the software in the preparation test mode. The Algorithm of creating to the lookup table appears in table 2.
First, variables and constants are initialized (Lines 1-10). The camera is placed in the middle of first stripe. Then, the algorithm count the current stripe pixels' number (Lines 12-16). Each stripes pixels count is saved (Line 18). In addition the algorithm takes care of saving the accumulated pixel counts (Line 19). J presents the stripe number. Since the scale reflected image looks like a trapezoid, the middle stripe's x value changes as y value advances to next stripe. Figure 8 presents the geometric scheme of this state.

Here α is the left side slope and $\theta = \frac{\pi}{2} - \alpha$.

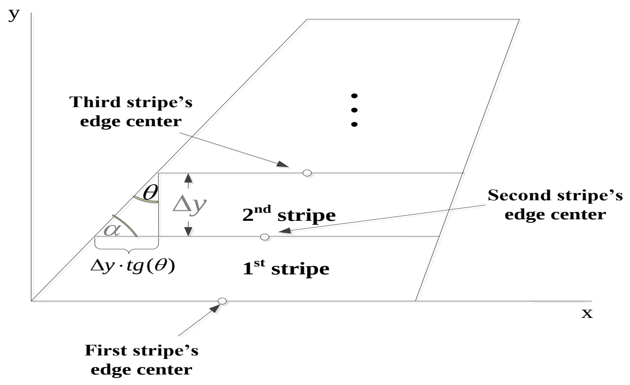

y
x
Third stripe's edge center
Second stripe's edge center
First stripe's edge center
θ
α
Δy
$\Delta y \cdot tg(\theta)$
2nd stripe
1st stripe

Figure 8: x values change

Table 2: Creation of lookup table algorithm

```
1:  Algorithm Lookup_Table(N,T):
2:      place the camera in front of middle scale's first stripe
3:      y = 0
4:      count = 0
5:      j = 0
6:      x of stripe's lower left corner = 0
7:      x = x value of first stripe's center
8:      read color(x, y)
9:      color_temp = color(x, y)
10:     sec = 0
11:     while $j \le N$ do
12:         $T_{1,j}$ = sec
13:         y++
14:         read color(x, y)
15:         if color(y) = color_temp
16:             count++
17:         else
18:             $T_{2,j}$ = count
19:             $T_{3,j} = T_{2,j} + T_{3,j-1}$
20:             j++
21:             sec = sec+10
22:             count = 0
23:         endif
24:         $x = x + \Delta y \cdot tg(\theta)$
25:     end while
26: return T
```

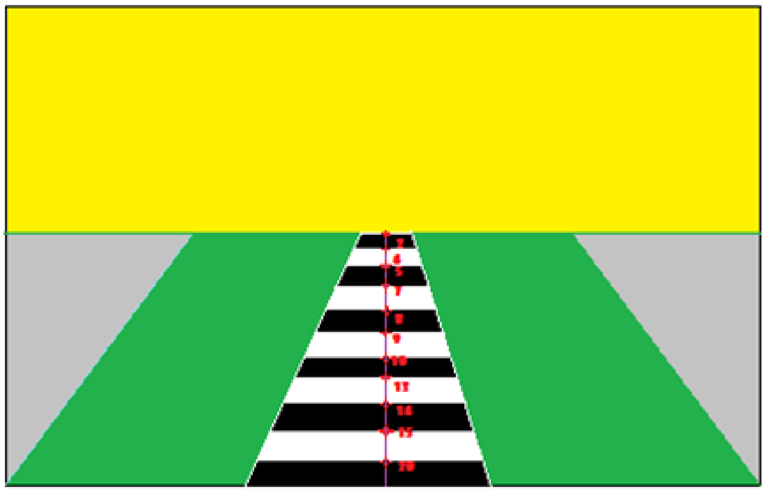


Figure 9: the measurements" environment

The result is an array similar to the one presented in table 3. We see that as further the stripe from the camera, the less pixels it has.

Table 3: Lookup table of pixel numbers as a function of the distance

0	10	20	30	40	50	60	70	CM
0	20	15	14	13	10	9	8	#PIX
0	20	35	49	62	72	81	89	ACC

The red line presents the measurement points along the y-axis. The green areas present the field and the yellow and gray areas presents the area outside the count range. The table (like table 3), can be used to correlate number of pixels to a point (x,y) further from the camera in the field and then derive the distance to this point. In order to calculate a distance from the camera to a certain point we will usually have to perform interpolation. Let us take an example. Suppose we need to know the distance to point (320, 75). We will look at the y-axes value, which is 75 in this case. From table 3 we see that the appropriate region end-points in terms of accumulated pixel values are 72 and 81. The related distance value lies between 40 and 50 cm. The distance value is calculated by interpolation:

$40 + \frac{75-72}{81-72} * 10 = 43.3333$ cm. In general the distance function will be:

$$distance\ (cm) = y(i) + \frac{y_{pix} - y(i)}{y(i+1) - y(i)} \cdot 10$$

Where $y(i)$ is the lower end point and $y(i+1)$ is the higher end point in the region. y_{pix} is the pixel's y value of the desired point.

VII. MAPPING SOLUTION ACCURACY

The method we described use the scale solely. When the lookup table is complete, there is no need to process the binary image. The only input data needed is the (x,y) object location.

Figure 10 presents the error as a function of the distance.

Table 4: Lookup table of pixel numbers as a function of the distance and stripe width type.

0	5	15	20	25	35	45	65	CM
0	20	15	14	13	10	9	8	#PIX
0	20	35	49	62	72	81	89	ACC
0	0	0	0	0	1	1	2	type

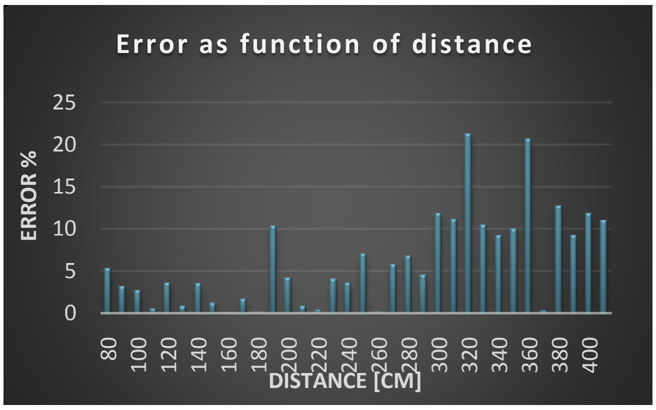


Figure 10: error as a function of distance

There are areas in figure 10 that show higher error compared to other areas. In order to have a distinction among the different areas we divided the area into three distance ranges: short range, middle range and long range. Table 4 presents the average error ratings for each range. The error in the long distance range is higher by more than three times compared to the short distance range and more than double compared to the middle distance range. Clearly, there is a need to adapt the method such that the error will be satisfactory for all ranges.

Table 5: The average error as a function of the distance ranges

Range	Distance	Error
Short	80-190[cm]	2.68%
Middle	200-300[cm]	4.95%
Long	310-410[cm]	11.6%

Looking closely at figure 9 and table 3 reveals that in the long distance there is a gap, which is realized by a jump in the pixels count. It means that the algorithm miscount the long or far distance. For example, if the real distance was 30 [cm] the algorithm calculates it as a distance of 10 [cm] due to the jump in pixels count. We will call this error as *Jump error* meaning miscount pixels error.

In order to take care about this problem we will design a new scale that solves the issue of Jump error. The new scale will take into account the different pixels count of stripes relative to their distance from the camera. After testing several options, we concluded that the following measures of the different ranges on the scale would be adequate: stripes' width of be 5 cm, 10 cm, and 20 cm for short, middle, and long distances accordingly.

VIII. SCALE VERSION 2.0

Based on the above discussion the 2nd version of the scale looks like the one presented in figure 11.

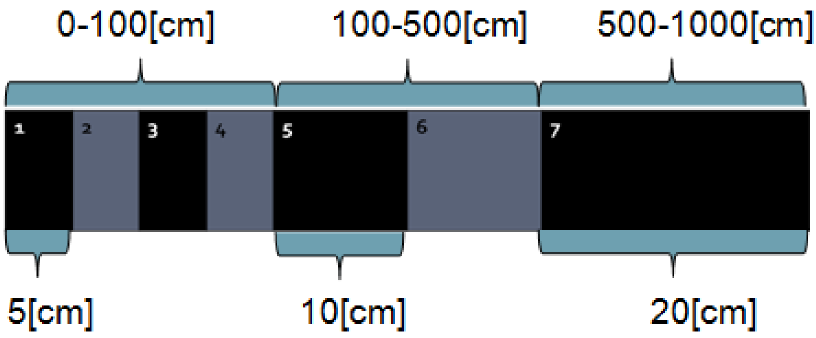


Figure 11: Scale 2.0 with different stripe's size

IX. THE LOOKUP TABLE ALGORITHM FOR SCALE 2.0:

The new lookup table will contain the information of each stripe width. The digit 0 will designate the stripes with width of 5 cm. The digit 1 will designate the stripes with width of 10 cm, and the digit 2 will designate the stripes with width of 20 cm. Thus, the lookup table might look as the one presented in table 5.

Reading the new ribbon requires the use of a new algorithm. This new algorithm will be similar to the one presented in table 2 with additional part for the information about different stipe sizes (5, 10, or 20 cm). Actually, since the ribbon structure is known, it is easy to manually add the last row of the lookup table.

Table 6 presents the algorithm for scale 2.0. The additional inputs compared to the algorithm of table 2 are n1 and n2, which presents the first stripe number with width of 10 cm and 20 cm respectively.

Table 6: Creation of lookup table 2.0 algorithm

```
1:  $Algorithm\ Lookup\_Table\_2.0\ (N, T, n_1, n_2):$
2:      $place\ the\ camera\ in\ front\ of\ middle\ scale's\ first\ stripe$
3:      $y = 0$
4:      $count = 0$
5:      $j = 0$
6:      $x\ of\ stripe's\ lower\ left\ corner = 0$
7:      $x = x\ value\ of\ first\ stripe's\ center$
8:      $read\ color(x, y)$
9:      $color\_temp = color(x, y)$
10:     $\sec = 0$
11:     $while\ j \le N\ do$
12:         $T_{1,j} = \sec$
13:         $y++$
14:         $read\ color(x, y)$
15:         $if\ color(y) = color\_temp$
16:             $count++$
17:         $else$
18:             $T_{2,j} = count$
19:             $T_{3,j} = T_{2,j} + T_{3,j-1}$
20:             $if\ \ j < n_1$
21:                 $T_{4,j} = 0$
22:             $else$
23:                 $if\ \ j < n_2$
24:                     $T_{4,j} = 1$
25:                 $else$
26:                     $T_{4,j} = 2$
27:                 $endif$
28:             $endif$
29:             $j++$
30:             $\sec = \sec + 10$
31:             $count = 0$
32:         $endif$
33:         $x = x + \Delta y \cdot tg(\theta)$
34:     $end\ while$
35: $return\ T$
```

The part that takes care of writing the stripe width index (0,1,2) is presented in lines 20-28 of the algorithm. When the lookup table is complete, a distance to a certain point (x,y) is calculated using the algorithm in table 7.

Table 7: Distance calculation algorithm

```
1 :  Algorithm Distance_Calculation_2.0 $(x, y, T, DIS_y)$:
2 :     $count[1] = count[2] = count[3] = 0$
3 :     $index = 0$
4 :     $type = 0$
5 :     $j = 1$
6 :     $last\_width = 5$
7 :     while $(y > T(3, j))$
8 :     if $T(4, j) = 0$
9 :         $count[1]++$
10 :        $last\_width = 5$
11 :    else if $T(4, j) = 1$
12 :            $count[2]++$
13 :            $last\_width = 10$
14 :        else
15 :            $count[3]++$
16 :            $last\_width = 20$
17 :        endif
18 :    endif
19 :    $j++$
20 :    endwhile
21 :    $DIS_y = 5 \times count[0] + 10 \times count[1] +$
       $+ 20 \times count[3] + last\_width \times \frac{y - T(3, j-1)}{T(3, j) - T(3, j-1)}$
22 : return $DIS_y$
```

For example if we need the distance to a point with y=85, using the data in table 5, we will get:

$$5 \times 5 + 10 \times 2 + 10 \times \frac{85 - 81}{89 - 81} = 50\ cm$$

X. THE ACCURACY OF SCALE VERSION 2.0

Figure 12 presents the error of the distance value for distances up to 410 cm, using the scale version2. Table 8 presents the average error in short, middle and long distance ranges of the two scale versions.

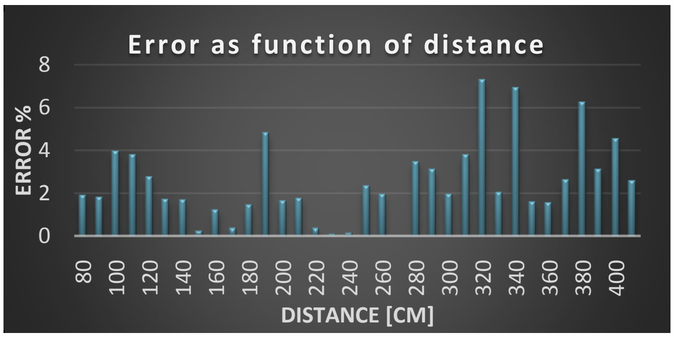


Figure 12: Error as function as distance in Scale ver 2.0

Table 8: error as a function of distance for the two scale versions

Range	Distance	Error ver 1.0	Error ver 2.0
Short	80-190[cm]	2.68%	2.11%
Middle	200-300[cm]	4.95%	1.69%
Long	310-410[cm]	11.6%	3.81%

It is clear from table 8, that using scale version 2, significantly improved the accuracy of the distance measurement.For example at the maximum distance presented in table 8, the error will be about 410 cm*3.81%=15.6cm. Since the robot feet size is more than 20 cm this is accurate enough for all practical purposes.

XI. ADDING THE X-AXIS INTO CALCULATIONS

Up until now, we used only the y-axis for calculation of the distance. Although the x-axis value has less effect on the distance, it is still good practice to use its value for better distance estimation.
We can calculate the x-axis value using the scale version 1.0. The number of pixels inside each cell doesn't change much, meaning that the number of pixels in the middle cell are more or less equal to those on the edge cell, as can be seen in Table 9.

This fact led us to an additional design. We sketched two vertical white lines, with a distance of 6m between them as presented in figure 15. Adding the horizontal lines creates a trapezoid of the 2-D field.
Then, we took measurements at every 20[cm] of the y-axis, starting from the camera position (figure 13). In each slice that created by the scale 1.0, we counted the number of pixels entering to 10cm segments, starting from one edge of the trapezoid and finshe in the other edge (figure 14).

Table 9 shows the data we got using Logitech C905 camera. Each column in Table 9 presents a different slice of y-axis with header y_{end}/y_{begin} (Top and bottom of the cell [figuer 7] in terms of pixels) [the image produced by this camera is an inverse image].
Each row presents different segment of 10[cm] (width x-axis). The numbers in Table 9 cell represents the

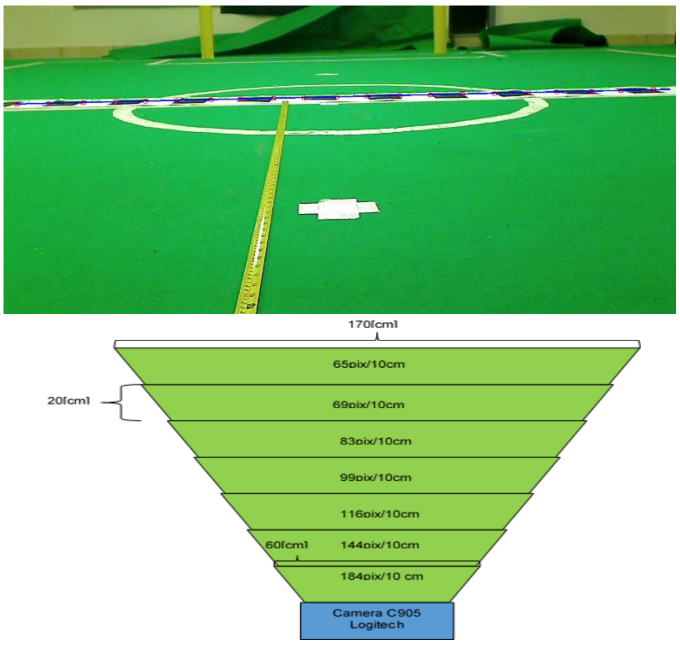
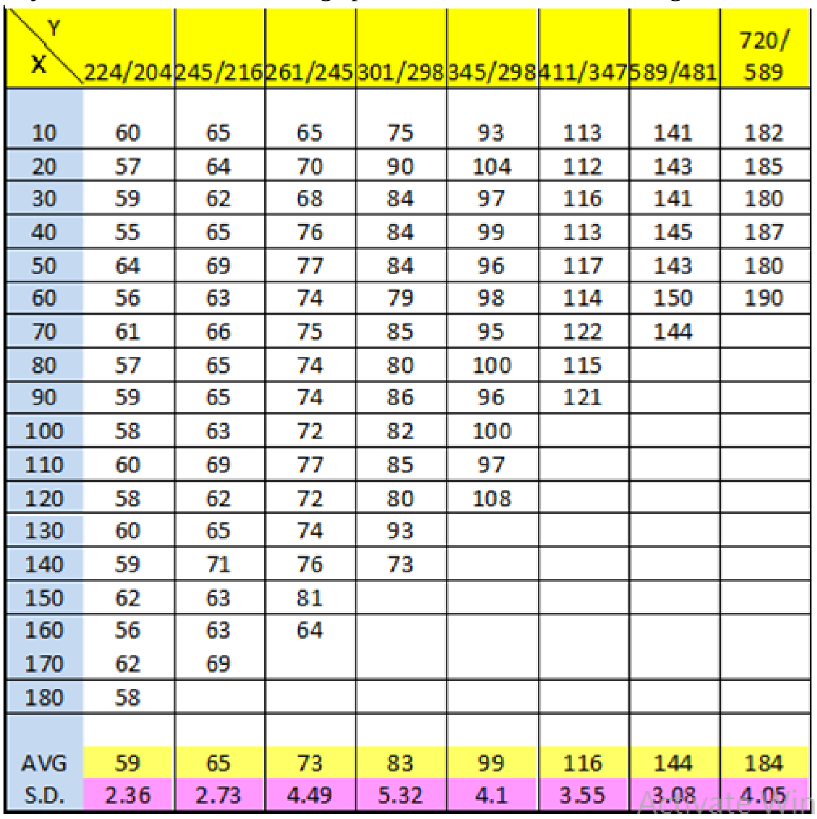


Figure 13: x-axis width measurement using Scale version 1.0

Table 9: presents the y-axis slices with the average pixels count for each 10cm segment in x-axis in each slice



X \ Y	224/204	245/216	261/245	301/298	345/298	411/347	589/481	720/ 589
10	60	65	65	75	93	113	141	182
20	57	64	70	90	104	112	143	185
30	59	62	68	84	97	116	141	180
40	55	65	76	84	99	113	145	187
50	64	69	77	84	96	117	143	180
60	56	63	74	79	98	114	150	190
70	61	66	75	85	95	122	144	
80	57	65	74	80	100	115		
90	59	65	74	86	96	121		
100	58	63	72	82	100			
110	60	69	77	85	97			
120	58	62	72	80	108			
130	60	65	74	93				
140	59	71	76	73				
150	62	63	81					
160	56	63	64					
170	62	69						
180	58							
AVG	59	65	73	83	99	116	144	184
S.D.	2.36	2.73	4.49	5.32	4.1	3.55	3.08	4.05

number of pixel entering to 10[cm] of width. For example in 100[cm] from the right side of the picture. In high of 240[pixel] (the second column) enter 63 pixels.

X- In terms of cm | Y-In terms of pixels
**The data presented in table 9 might be different for a different camera, but the calculation will be the same.*

We can see from table 9 that each column standard deviation is in most cases less than 5% and in the worst case is less than 6.5%. We can conclude that it is safe to use the average number of pixels for distance calculations.

Next, we will explain how to receive the distance from x_i to $x_{middle.}$ For that, let us define x_{middle} as the middle point value of the horizontal line (figure 15) with y_i as its y-axis value. x_{middle} is known for every y. First , we recorded the two left corners of the trapezoid ((x_1,y_1) and (x_2,y_2)) (This is true for the right side as well).

Suppose that we are interested in the distance to the point (x_i,y_i). We know that the distance of any point (x_3 , y_i) on the white line to the parallel point (x_{middle} ,y_i) on the middle line equals to 3[M].
The algorithm will find the x_3 using the method proposed in the papers [8] and [9].
It is easy to prove that the x-axis value (x_{wl}) of a point on the left vertical white line, with y_i as its y-axis value, can be calculated to be:

Figure 14:number of pixels presents 10cm segments in y-axis in avg

$$x_{wl} = f(y_i) = \frac{y_1 - y_i}{m} + x_1$$

Where m is the left white line slope ((y_1-y_2)/(x_1-x_2)).
We define two more variables: pix_1 and pix_2 as follows:

$$pix_1 = x_3 - x_{middle}$$

$$pix_2 = x_{wl} - x_{middle}$$

The algorithm will compute the x-axis distance (*DISx*) with metric units according to the following equation:

$$DIS_x = \frac{pix_1}{pix_2} \times 3[m]$$

This gives the X distance from the center to the desired point (x_i,y_i)

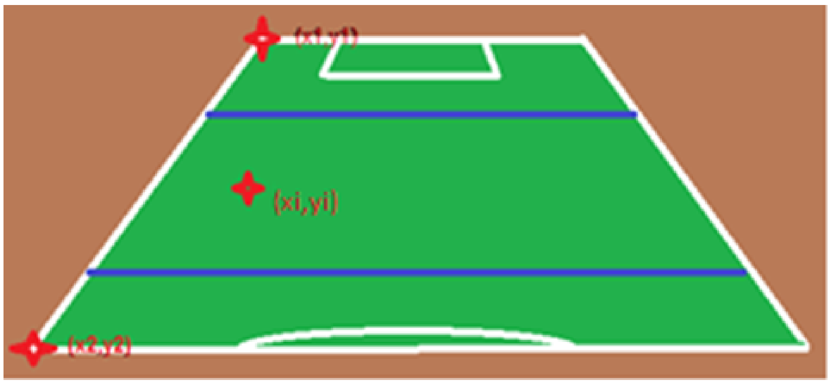


Figure 15: Field configuration for calculating x-axis value of a certain point

XII. RETRIEVE DISTANCE

The last step required to compute the real distance is by use the right triangle hypotenuse formula:

$$distance = \sqrt{(DIS_x)^2 + (DIS_y)^2}$$

XIII. SUMMARY

This paper demonstrates an easy and efficient method to calculate a distance from a camera to an arbitrary object in the playing field of a KSL competition in the frame of RoboCup soccer league. In fact, this method is more accurate and faster than *volume solution*, in other words, it reducing the error mistake in each step and have no obligation to count the white pixel in the binary threshold picture.
The base of this idea relies on initial setup, which includes adapting the pixel numbers into the distance. At first, there was an exponential function, as we proceeded, we decided to divide the graph into small sections, achieving a better result. Apparently, the problem with long distance remained. Another obstacle was different objects on the field, different shapes are acting differently, what made us create different function for each shape (Goal, Ball, White strip, etc.)
The database was prepared by using a suitable scale and merged all of the different functions into one database. Then, an algorithm with only (x,y) coordination was created. These solved both problems. Additionally, this method can be adapted to different scenarios and different cameras.

REFERENCES

[1]. G. P. Stein, O. Mano and A. Shashua, "Vision-based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy", Proc. Intelligent Vehicles Symposium, pp. 120-125, 2003.
[2]. Y.C. Kuo,N.S. Pai and Y.N. Li, "Vision-based Vehicle Detection for a Driver Assistance System," Computers and Mathematics with Applications (61) ,pp. 2096–2100, 2011.
[3]. J. Mrovlje and D. Vrančić, "Distance Measuring Based on Stereoscopic Pictures", 9th Int. PhD Workshop on Systems and Control: Young Generation Viewpoint, Slovenia, 2008.
[4]. M. Kyto, M. Nuutinen, and P. Oittinen , "Method for Measuring Stereo Camera Depth Accuracy Based on Stereoscopic Vision", Proc. SPIE 7864, Three dimensional imaging, Interaction and Measurement, 2011.
[5]. C. Tinnachote and K. Pimprasan, "Distance Measurment from Digital Photograpg Using 3rd Order Polynomial Equation", Proc. ACRS ,2012..
[6]. M. Mirzabaki, "Depth Detection Through Interpolation Functions", The 12th int. conf. WSCG, pp. 105-108, 2004..
[7]. S.V.F Barreto R E.S. Anna, and M. Feitosa, "A Method for Image Processing and Distance Measuring Based on Laser Distance Triangulation", IEEE 20th int. conf. on Electronics, Circuits and Systems, pp. 695-698, 2013.
[8]. Y. Lim and H. Lim, "A Method for Measurement of Distances using License Plate Detection", Advanced Science and Technology Letters, pp. 13-16, 2014.
[9]. A. de La Bourdonnaye, R. Doskocil, V. Krivanek, and A. Stefek, "Practical Experience with Distance Measurement Bsed on Single Visual Camera," Advances in Military Technology, (7) 2, pp. 49-56, 2012