Theory

Introduction

An ArUco marker is a 5x5 grid that is black and white in color. ArUco markers are based on Hamming code. In the grid, the first, third and fifth columns represent parity bits. The second and fourth columns represent the data bits. Hence, there are ten total data bits. So the maximum number of markers that can be encoded are-

2^10 = 1024

Encoding

Let us consider the number 650. Its binary representation is 1010001010. There are two data bits in each row.

Now, each row is encoded separately using slightly modified Hamming code. The first and third parity bits are calculated using even parity while the second parity bit uses odd parity. We get the following encoded values-

Data bit 2	Parity bit 3	Data bit 1	Parity bit 2	Parity bit1
0	0	1	0	1
0	0	1	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	0	1

We rearrange the bits in each row and get the following result: If the cell value is 0, color it black; if value is 1, color it white. This will give us the ArUco marker

	1		0	
	1		0	
	0		0	
	1		0	
	1		0	

Figure 1: Data rows

Parity1	Data1	Parity3	Data2	Parity2
0	1	0	0	1
0	1	0	0	1
1	0	0	0	0
0	1	0	0	1
0	1	0	0	1

Figure 2: ArUco bits for 0b1010001010

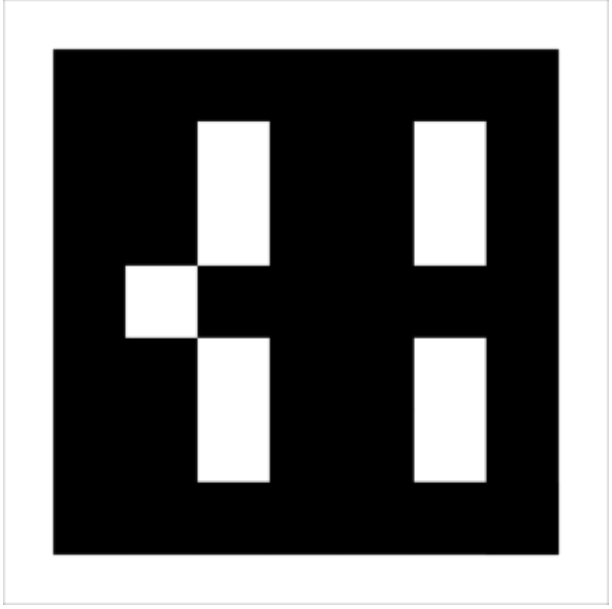

Figure 3: ArUco marker

The above marker is padded with one layer of black cells.

Decoding

After understanding the above section, decoding is an extremely simple process. The following steps are to be followed while decoding a perfect, computer-generated image of an ArUco marker.

Step 1: Extract the ArUco from the image.

Step 2: Remove the extra padding.

Step 3: Divide the resulting image into a 5x5 grids and check the color in each cell of the second and fourth columns (in that order) in a top to bottom manner.

Step 4: If the color is white, write 1; else, write it 0.
Step 5: The resulting number will be in binary. Convert it into decimal and it gives the ID of that marker.

Pose Estimation:

Pose of an object refers to its position and orientation with respect to the camera. To determine the pose, we need to solve the perspective-n-point problem. In perspective-n-point problem, we are given a set of object points and their corresponding image points and we should determine the pose of the image. Trigonometric operations are applied to find the solution. Refer to **Resources** to learn in detail.
Learn about camera calibration in ROS here
The pose of an object is represented in the form of its rotational vector and translational vector. We can obtain these two vectors using the camera calibration. Camera calibration has a lot of processing, so it consumes more time. Instead, we save the camera matrix and distortion coefficients obtained from camera calibration and use them in other functions to find pose.

Now, after finding the translational vector (tvec) and rotational vector (rvec), we need to extract the required information from them. The distance from camera is acquired from the Z-coordinate in the tvec. This is because we assume the object is always on Z=0 plane and the camera is put at a distance from it. The x-coordinate in tvec denotes the displacement between the origin object and camera axis in x-direction. The angle is obtained from rvec. But first, we need to convert it into a rotation matrix(R). This is done using open cv functions. After obtaining the matrix, we compare it to the rotational matrix of y-axis.

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Figure 1: Rotation matrix in y-axis

We then take the sine inverse of R[0][2]. This gives us the angle of inclination. The following image depicts this-

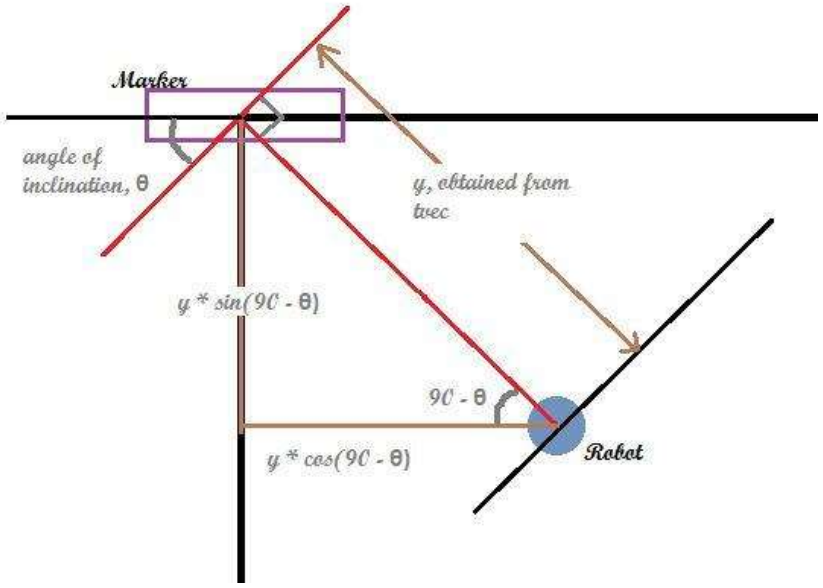

Figure 2: Pose estimation

Applications

- These markers are utilized in augmented reality applications.
- They can be used in robot navigation and localization algorithms.

Resources

- http://www.uco.es/investiga/grupos/ava/node/26
- http://iplimage.com/blog/create-markers-aruco/
- http://terpconnect.umd.edu/~jwelsh12/enes100/markergen.html