# An Introduction to WhyCon Markers

WhyCon is a vision-based localization system that can be used with low cost web cameras. It achieves millimeter precision with very high performance. These markers consist of a dark outer ring and a concentric white circle as shown in Figure 1.1.

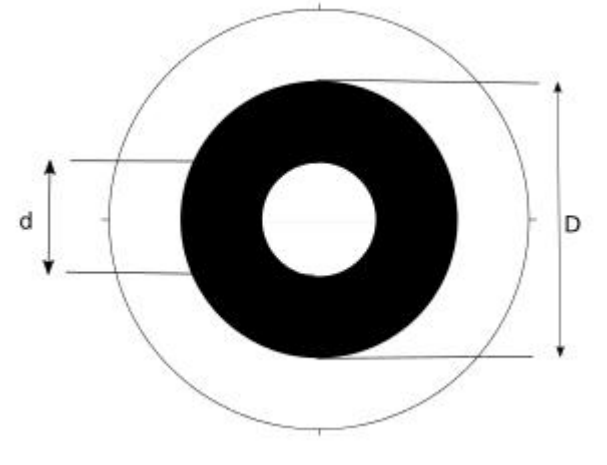


Figure 1.1 WhyCon Marker

WhyCon detection Process:

1. When a camera starts capturing frames, the algorithm starts scanning each frame pixel by pixel. Let us start from a pixel $p_0$. The algorithm checks the first pixel to see if the pixel is white or black using a threshold. On detection of a white pixel it jumps to the next pixel and repeats the process.
2. On finding a black pixel the algorithm employs a queue based flood fill technique to detect contiguous segment of dark circle.
3. The contiguous segment is then tested for possible match for the pattern of the outer ring. The detection of the outer ring is validated by taking minimum number of pixels in the segment and roundness tolerance limit as input from the user.
4. If the segment is found to be invalid (not part of the outer ring), it is assigned with an unique identifier so as to avoid redundant computation. The detection for further black segment (to detect the outer ring) continues starting from the next pixel position.
5. If a black segment is found to be valid (part of the outer ring), the detection resumes from the centroid pixel of the black segment to find another continuous area of white or bright pixels (segment of the inner ring). Flood fill algorithm is performed again to find a concentric white or bright circle.
6. If the minimum size and roundness test are passed, further validation is carried out by testing the centroid position, ratio of the area of black and white pattern according to the user parameters, and complex circularity measures like max-eccentricity parameters etc.
7. If the segment passes all tests, the WhyCon marker is considered as found and the centroid pixel is considered as $p_0$ for the next detection.

8. Multiple WhyCon markers are detected by running the detector several times consecutively depending upon the number of targets set by the user.
9. In each iteration the color of the inner circle pixels is changed to black to prevent multiple detection of the same target.

**WhyCon Localization:**

1. Considering the detected pattern as ellipse, the center and semi axes are calculated from the covariance matrix eigen-vectors and transformed to a canonical camera coordinate system.
2. Canonical form refers to a pin-hole camera model with unit focal lengths and no radial distortions.
3. The transformed parameters are then used to establish coefficients of the ellipse characteristic equation. These coefficients are subsequently utilized to calculate the patterns' spatial orientation and their positions within the camera coordinate frame.

**How to use WhyCon ROS-package ?**

**Follow the steps given below:**

1. Change the directory to your catkin workspace and clone the github repository in "src" folder of workspace by typing the following command:

   ***git clone https://github.com/lrse/whycon.git***

   The above command will clone the whycon package into your workspace. To build this package navigate to catkin workspace and run the following command:

   ***catkin_make***
   ***source devel/setup.bash***

2. Navigate to "launch" folder of whycon package. You will find a file "whycon.launch". This launch file initializes the run, all the required nodes and algorithm. This launch file subscribes and publishes the topics as follows:
   a) **Subscriber:**
      i. **/camera/camera_info:** This topic contains the necessary information about camera like frame_id, height, width etc.
      ii. **/camera/image_rect_color:** This topic contains the image on which the algorithm performs.

   **Map the above topics to their corresponding requirements.**

b) **Publisher:**
   i. **/whycon/image_out:** This topic publishes the image after performing the algorithm.
   **ii.** **/whycon/poses:** This topic has the information of the position of each individual whycon marker within the camera frame.

**NOTE: The above discussed topics are relevant to the task you are working on. Whycon also publishes other topics like /whycon/context and /whycon/projection. You may check for all whycon topics using the command: "*rostopic list*"**

As discussed in step 2, whycon subscribes to topics "/camera/camera_info" and "/camera/image_rect_color". WhyCon package provides output only when it receives the information on these topics. Now the question is - who will publish the information on these topics so that whycon can take that topic and apply the algorithm to detect the WhyCon markers? To know the answer, let us take a look at the "*whycon.launch*" in the launch folder of whycon. **Do not make any changes to the *whycon.launch* when you go through the following details. Just understand what are the different steps involved. The changes mentioned below will already be made for you in the repository which you will be downloading for Task 0.**

3. Let us take a look at "*whycon.launch*" of *whycon* package line by line.

These are the arguments that are passed to the whycon node.
I. Topic will start from the name given in the argument
II. Number of markers to be detected. In above case only one marker will be detected.
III. You will have to add two more parameter explicitly as you have to specify the outer and inner diameter of the whycon marker

```
1  <launch>
2    <arg name="name" default="whycon"/>
3    <arg name="targets" default="1"/>
4    <arg name="outer_diameter" default=".55"/>
5    <arg name="inner_diameter" default=".20"/>
6    <arg name="input_queue_size" default="50"/>
```

Make sure you set the "outer_diameter" and "inner_diameter" arguments in the whycon node as shown below using param tag inside the node tag of whycon node.

```
9    <group ns="camera">
10     <node pkg="image_proc" type="image_proc" name="image_proc"/>
11   </group>
12   <node name="whycon" type="whycon" pkg="whycon" output="screen">
13     <param name="targets" value="$(arg targets)"/>
```

```
14      <param name="outer_diameter" value="$(arg outer_diameter)"/>
15      <param name="inner_diameter" value="$(arg inner_diameter)"/>
16      <param name="input_queue_size" value="$(arg input_queue_size)"/>
17      <param name="name" value="$(arg name)"/>
18    </node>
```

This is another node running in the launch file. This node "imag_proc" takes the camera input and provides the output: image into mono image, RGB image and color image etc.
All the topics are published to "camera/topic_name" as the group tag name is "camera"
*For more information about image_proc visit here*
The above node will run the algorithm by taking the image input on topic "camera/image_rect_color". "camera/image_rect_color" topic is published by "image_proc" node as we have given the namespace as "camera".

4. Let us suppose that the image is published on topic "task_0/image_raw". The algorithm is to be performed on this topic. Remaining information about the image is published on "task_0/camera_info". Both these topics should be given to the whycon package in order to perform the algorithm on the image. Use *rqt_graph* for debugging.Then you have to use the remap parameter between the topics to connect them according to our requirement as you have learned from the video tutorial. More information about remap is explained *here*
5. After a successful remapping between topics you can check the output of whycon package by running the following command:
   `>> rosrun image_view image_view image:=/aruco_marker_publisher/result`
   The above command will show the image published by "/whycon/image_out" topic. This topic is the output image of whycon node. image_view is basically a ROS node inside a ROS package named image_view and takes image as the argument.

6. As you will have to run the node each time to see the output, it will be better if you call the node inside the launch file itself. In order to do that, you can add the following lines into the launch file to do same:

```
19    <node ns="whycon_display" name="image_view" pkg="image_view"
    type="image_view" respawn="false" output="screen">
20      <remap from="image" to="/whycon/image_out"/>
21      <param name="autosize" value="true" />
22    </node>
23  </launch>
```

**All The Best!!**