Camera based positioning system for the CCOM/JHC and OE engineering test tank.

Exploring the idea of using a camera to track small (RC boat sized) autonomous vessels in the test tank.

Existing technologies

OpenCV: https://opencv.org/ (https://opencv.org/)

ArUco: https://www.uco.es/investiga/grupos/ava/node/26 (https://www.uco.es/investiga/grupos/ava/node/26)

OpenCV does contain an implementation of the ArUco decoding library.

Test tank

According to http://ccom.unh.edu/facilities/test-tanks/engineering-tank (http://ccom.unh.edu/facilities/test-tanks/engineering-tank), the tank size is 60 by 40 feet.

In [1]:
```
tank_size_feet = (60,40)
tank_size_meters = tuple(x*0.3048 for x in tank_size_feet)
print tank_size_meters
```

```
(18.288, 12.192)
```

ArUco markers

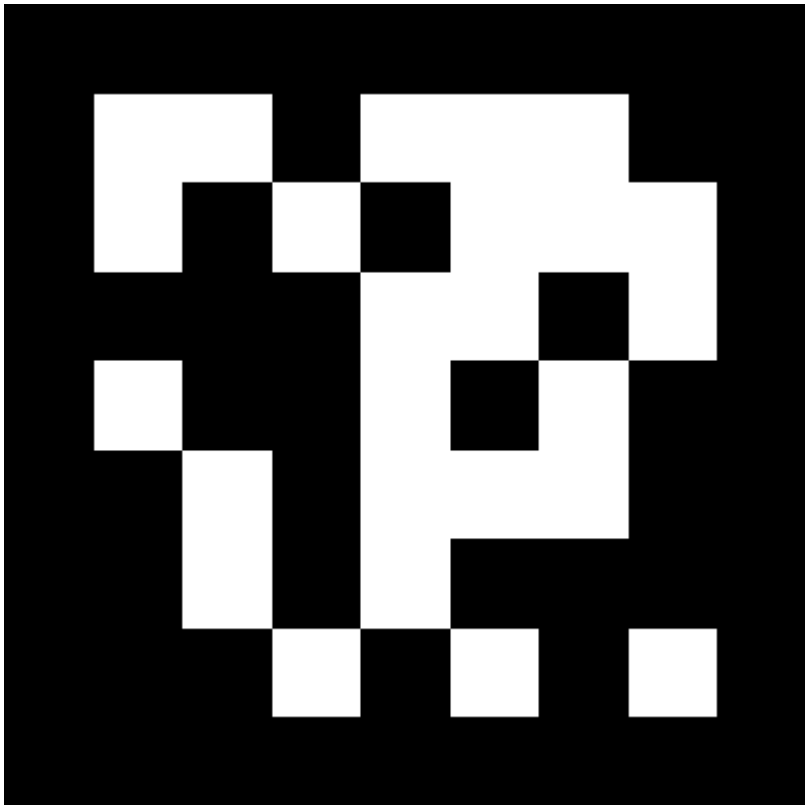
For testing, the ArUco DICT_7X7_50 dictionary was chosen. It was chosen due to it's larger number of bits (7x7) versus the number of markers in the dictionary (50) which is supposed to help supress errors.

Quick testing seems to suggest that markers should span at least 20 pixels in order to be reliably detected.

In [2]:
```
marker_size_pixels = 20
```

The marker is to be placed on top of a floating target, facing the ceiling with a camera looking down over the tank from the ceilng.

Example ArUco marker



Marker sizes

Let's explore various sizes ranging from 30 centimeters to 10 centimeters.

In [3]:
```
marker_sizes_meters = (.3,.2,.1,.05)
```

Minimal resolution necessary to reliably resolve markers:

In [4]:
```
import math

for msize in marker_sizes_meters:
    pixel_size = msize/marker_size_pixels
    resolution = tuple(int(math.ceil(x/pixel_size)) for x in tank_siz
e_meters)
    print 'marker size:',msize,'meters, pixel size:',pixel_size,'mete
rs, resolution:',resolution,'pixels'
```

```
marker size: 0.3 meters, pixel size: 0.015 meters, resolution: (1220,
813) pixels
marker size: 0.2 meters, pixel size: 0.01 meters, resolution: (1829,
1220) pixels
marker size: 0.1 meters, pixel size: 0.005 meters, resolution: (3658,
2439) pixels
marker size: 0.05 meters, pixel size: 0.0025 meters, resolution: (731
6, 4877) pixels
```

Camera geometry

Distance from top edge of tank to ceiling supports is 14' 5".

In [5]:
```
camera_distance_meters = (14+(5/12.0))*0.3048
print camera_distance_meters, 'meters'
```

```
4.3942 meters
```

In [6]:
```
field_of_view_degrees = tuple((180/math.pi)*2*math.atan(x/(2*camera_d
istance_meters)) for x in tank_size_meters)
print field_of_view_degrees,'degrees'
```

```
(128.6662586875929, 108.42935981879275) degrees
```

Two camera system

Two cameras independantly monitoring half the tank. Markers would get extracted separately instead of trying to merge the two images. Fixed markers on the tank edges where the images overlap could be used to fix the frames of reference of each camera.

In [7]:
```
half_tank_size_meters = (tank_size_meters[1],tank_size_meters[0]/2.0)
print half_tank_size_meters,'meters'
```

```
(12.192, 9.144) meters
```

Field of view calculations:

In [8]:
```
half_fov_degrees = tuple((180/math.pi)*2*math.atan(x/(2*camera_distance_meters)) for x in half_tank_size_meters)
print half_fov_degrees,'degrees'
```

```
(108.42935981879275, 92.27205607937618) degrees
```

Resolution calculations:

In [9]:
```
for msize in marker_sizes_meters:
    pixel_size = msize/marker_size_pixels
    half_resolution = tuple(int(math.ceil(x/pixel_size)) for x in half_tank_size_meters)
    print 'marker size:',msize,'meters, pixel size:',pixel_size,'meters, resolution:',half_resolution,'pixels'
```

```
marker size: 0.3 meters, pixel size: 0.015 meters, resolution: (813, 610) pixels
marker size: 0.2 meters, pixel size: 0.01 meters, resolution: (1220, 915) pixels
marker size: 0.1 meters, pixel size: 0.005 meters, resolution: (2439, 1829) pixels
marker size: 0.05 meters, pixel size: 0.0025 meters, resolution: (4877, 3658) pixels
```

Four camera system

Similar to two camera system, but spliting the tank into four quadrants.

In [10]:
```
quarter_tank_size_meters = (half_tank_size_meters[1],half_tank_size_meters[0]/2.0)
print quarter_tank_size_meters,'meters'
```

```
(9.144, 6.096) meters
```

In [11]:
```
quarter_fov_degs = tuple((180/math.pi)*2*math.atan(x/(2*camera_distance_meters)) for x in quarter_tank_size_meters)
print quarter_fov_degs, 'degrees'
```

```
(92.27205607937618, 69.49357368428157) degrees
```

In [12]:
```
for msize in marker_sizes_meters:
    pixel_size = msize/marker_size_pixels
    quarter_resolution = tuple(int(math.ceil(x/pixel_size)) for x in
quarter_tank_size_meters)
    print 'marker size:',msize,'meters, pixel size:',pixel_size,'mete
rs, resolution:',quarter_resolution,'pixels'
```

```
marker size: 0.3 meters, pixel size: 0.015 meters, resolution: (610,
407) pixels
marker size: 0.2 meters, pixel size: 0.01 meters, resolution: (915, 6
10) pixels
marker size: 0.1 meters, pixel size: 0.005 meters, resolution: (1829,
1220) pixels
marker size: 0.05 meters, pixel size: 0.0025 meters, resolution: (365
8, 2439) pixels
```

Warning

The above calculations are theoretical limits. They do not take into account overlap between the sections and with the tank edges. For practical puposes, cameras should be over-speced to account for overlap and inprecision in installation location and orientation.

In [ ]: