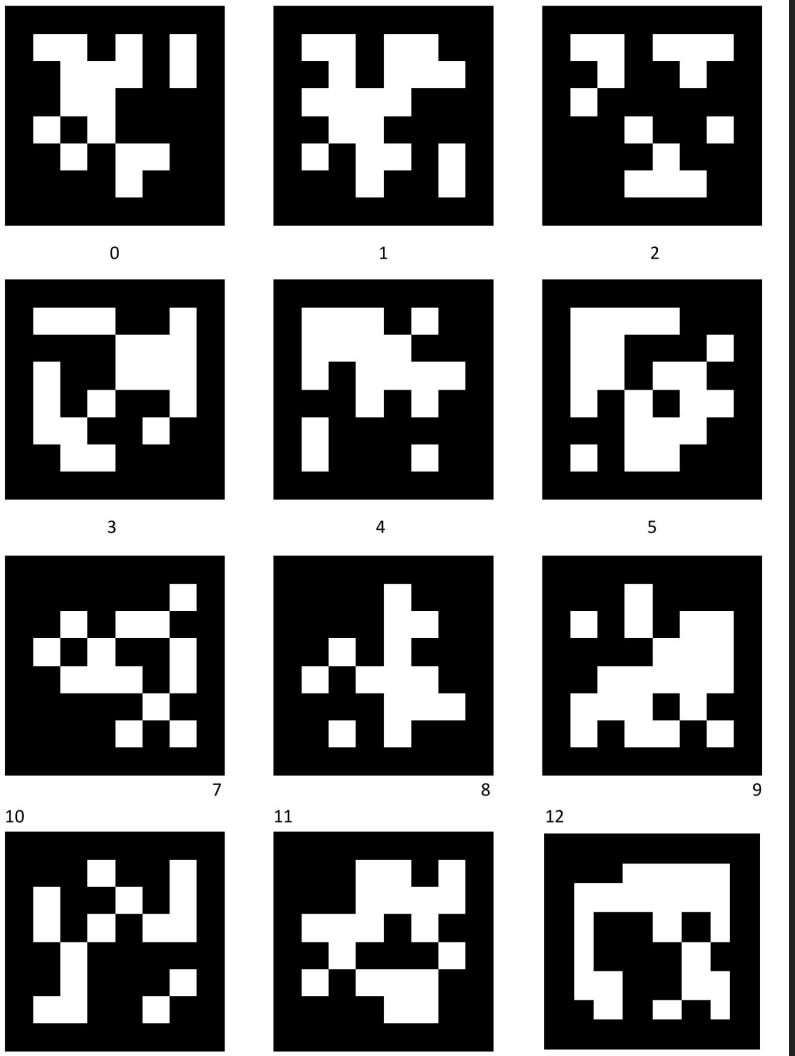


Reading and Tracking AprilTags on a Raspberry Pi

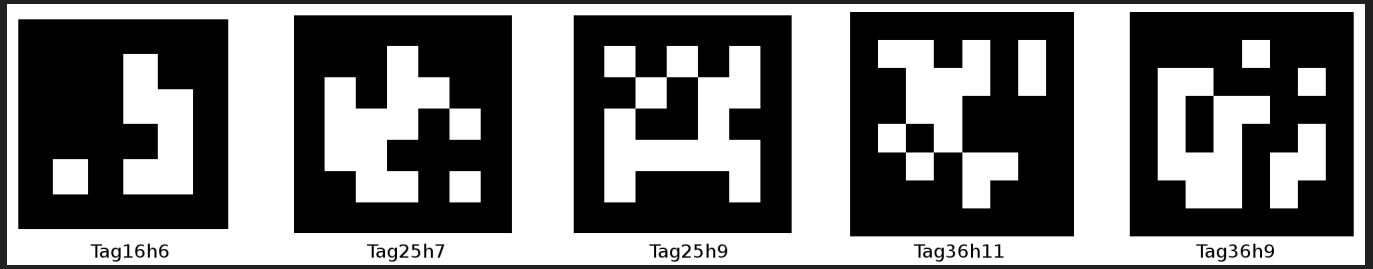
Presented by Fred Probst

INVERSE U 1100 POLARITY

Outline

1. April Tags
2. Solution Space
3. Chosen Solution
4. Solution Components
5. Python Application
6. AprilTag Integration
7. Questions
8. References

AprilTags



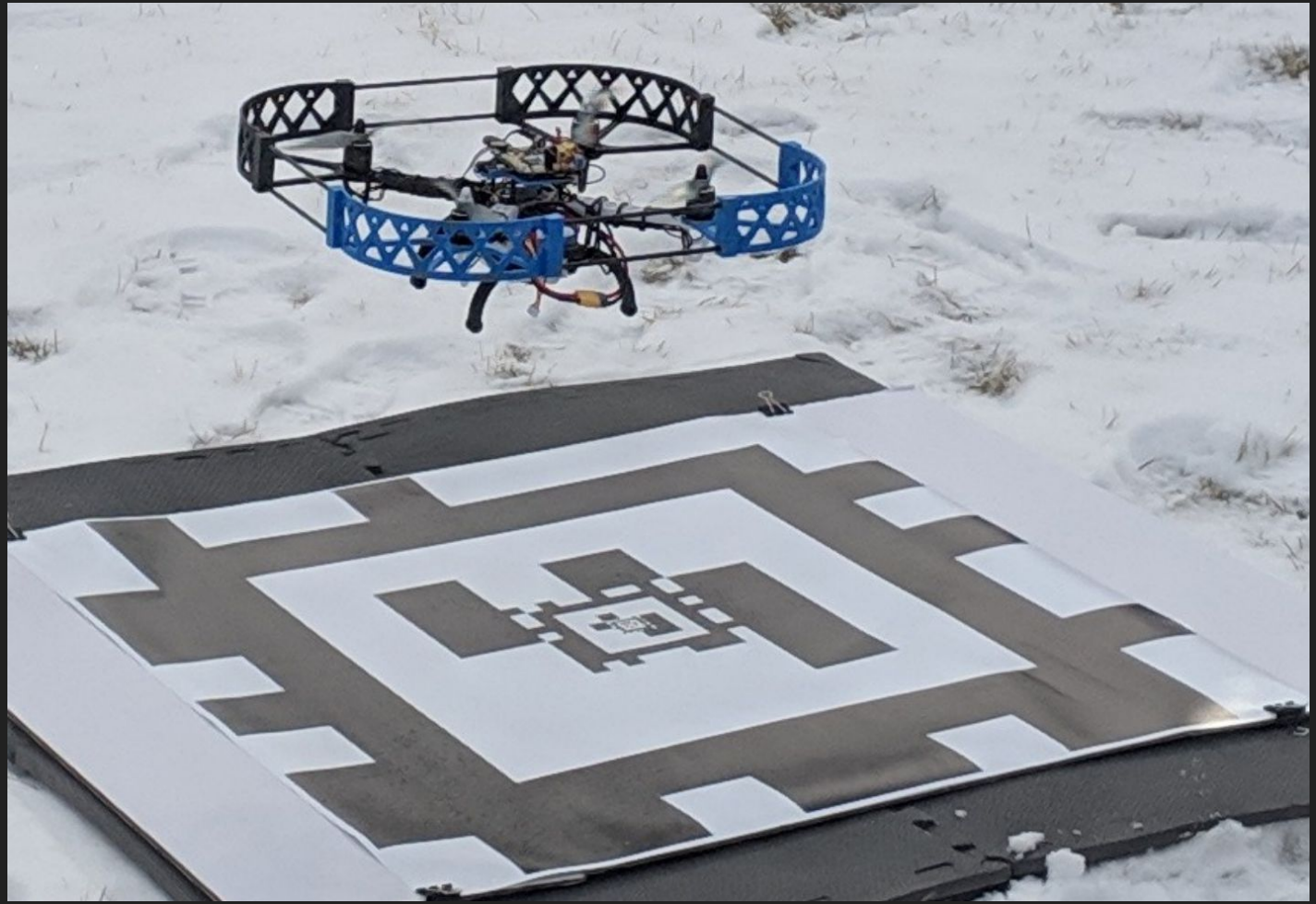
https://people.csail.mit.edu/kaess/apriltags/

What are AprilTags?

- AprilTags are a type of two dimensional bar code similar to a QR code.
- There are 3 versions of the AprilTag fiducial system
  - Variety of different families
- Classic AprilTag families are squares, but additional shapes are supported
- FIRST Robotics is using the 36h5 family of AprilTags, which are squares
- There are **30** tags in the 36h5 family.

Why AprilTags?

- Better detection in varying light conditions and from a wide variety of viewing angles
- Ability to encode additional information and easily detect different items
- Does not require targeting lights
- Can be printed on paper



https://april.eecs.umich.edu/media/pdfs/krogius2019iros.pdf

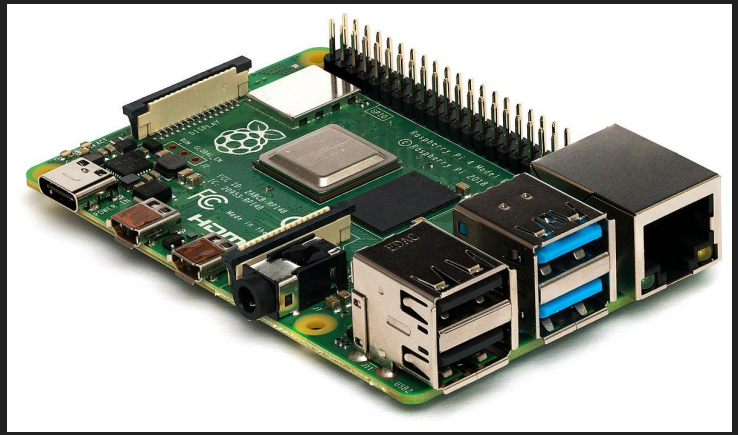
Solution Space

On roboRIO

- Integrated WPILib support
  - WPILib GitHub

On coprocessor

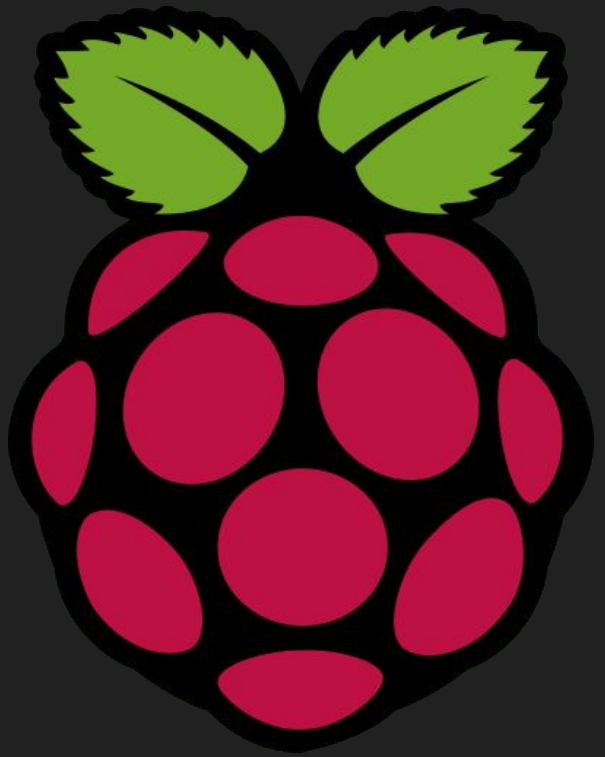
- Photovision
  - PhotoVision
- Direct use of AprilTag library
  - chosen solution
  - AprilTag GitHub



Chosen Solution

Direct use of AprilTag library on Raspberry Pi

- Coprocessor offloads vision processing
  - Leaves roboRIO free to complete other important work
- Direct use of AprilTag library provides the most flexibility
- Provides a more intimate understanding of how vision works



Solution Components

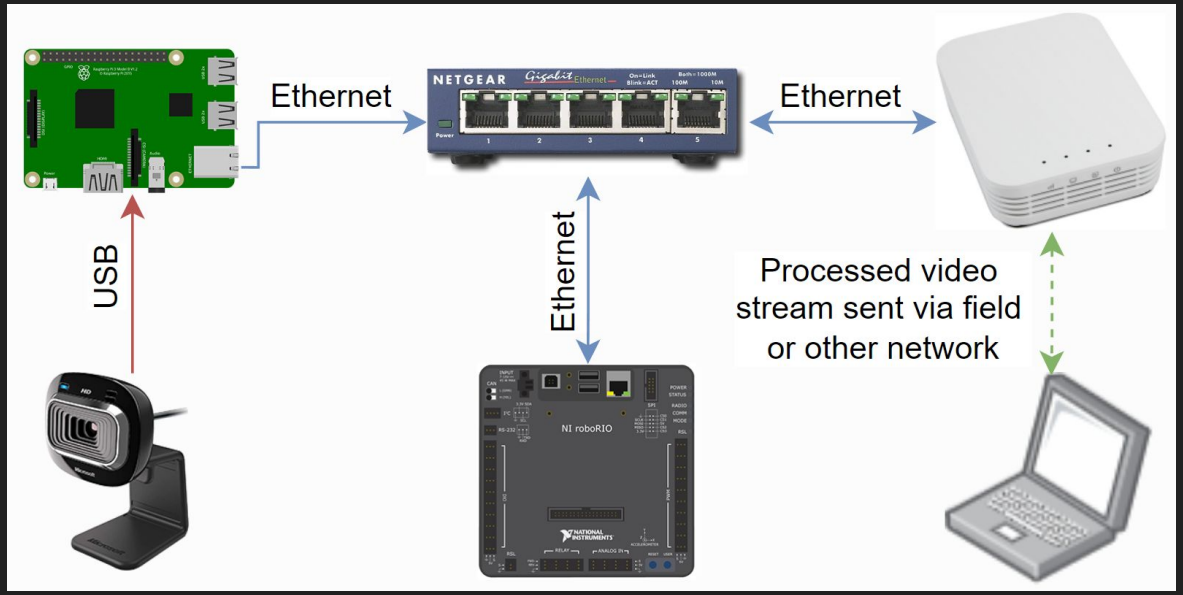
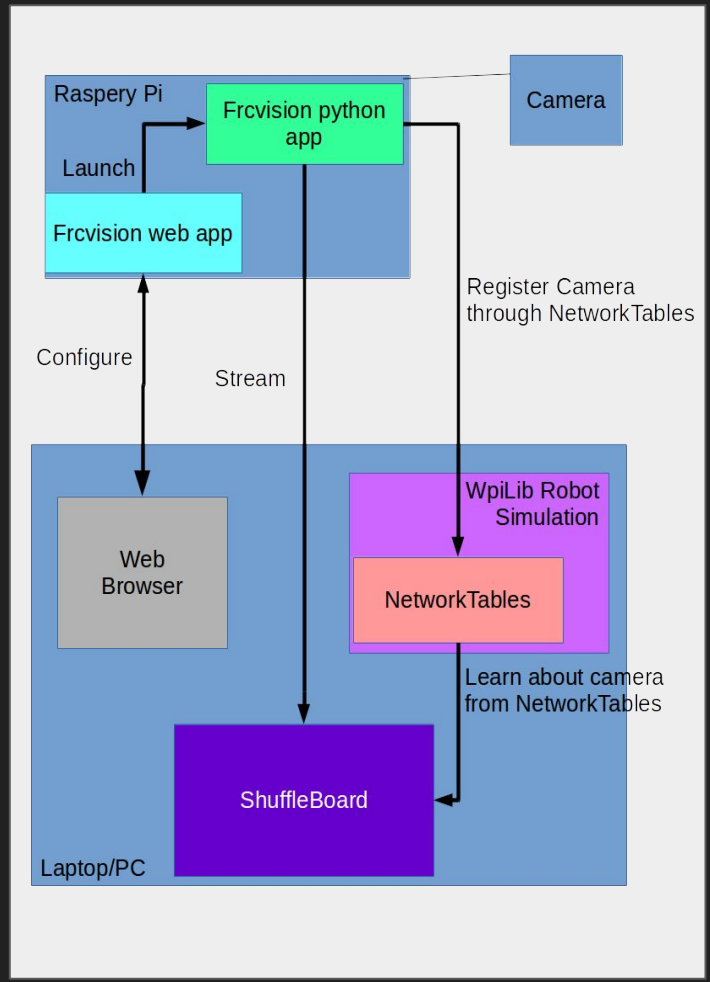
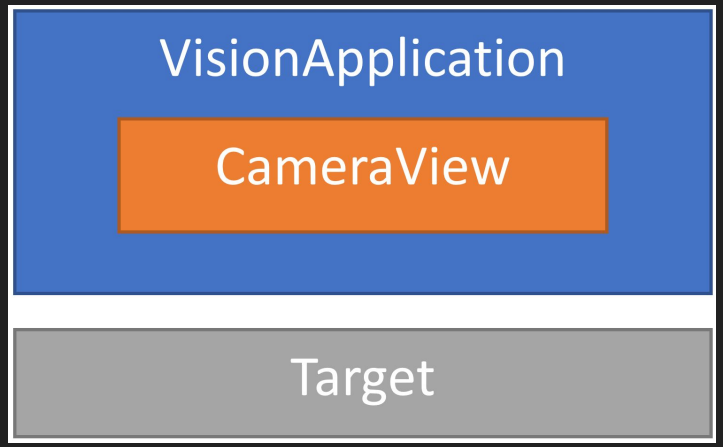


Diagram of the vision setup on the robot. Picture borrowed from docs.wpilib.org



Python Application



Main()

VisionApplication.__init__()

apriltag()

VisionApplication.readConfig()

CameraView.__init__()

VisionApplication.initialize CameraServer()

VisionApplication.initialize NetworkTables

```
VisionApplication.runApplication()
while True:
    grabs a frame from the camera
    gets potential tags
    for each tag:
        if minimum margins are met:
            draw bounding box
            mark tag with number
            create Target object and store it
    if no tags are detected:
        publish targetDetected as 0
    else:
        publish targetDetected as 1
    publish processed camera frame
```

April Tag Integration

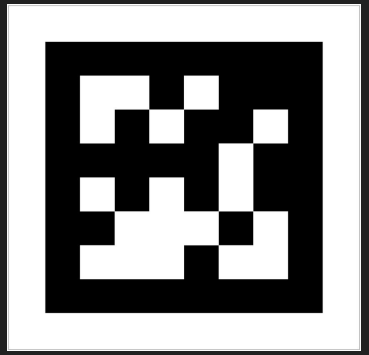


Image Source

```
from apriltag import apriltag
```

Imports the python AprilTag library

```
self.TAG = "tag36h11"
self.detector = apriltag(self.TAG)
```

Creates a detector object for the 36h11 tag family

```
greys = cv2.cvtColor(input_img1, cv2.COLOR_BGR2GRAY)
dets = self.detector.detect(greys)
```

Creates an array of possible AprilTags detected based off of a grayscale version of the current frame

```
for det in dets:
    if det["margin"] >= self.MIN_MARGIN:
        ident = str(det["id"])
        pos = det["center"].astype(int) + (-10,10)
```

Loops through the array of possible AprilTags and filters out tags that don't meet minimum confidence requirements (MIN_MARGIN). Then, both the AprilTag ID and coordinate position on the screen are recorded into variables.

Questions

References

- https://april.eecs.umich.edu/software/apriltag
- https://april.eecs.umich.edu/media/pdfs/olson2011tags.pdf
- https://april.eecs.umich.edu/media/pdfs/krogius2019iros.pdf
- https://github.com/AprilRobotics/apriltag-imgs
- https://github.com/AprilRobotics/apriltag
- https://github.com/AprilRobotics/apriltag/wiki/AprilTag-User-Guide
- https://github.com/AprilRobotics/apriltag-imgs/tree/master/tag36h11
- https://github.com/wpilibsuite/allwpilib/tree/main/apriltag/src/main/java/edu/wpi/first/apriltag
- https://docs.photonvision.org/en/latest/docs/getting-started/april-tags.html
- https://www.researchgate.net/figure/Exemplary-AprilTag-tag-42-from-the-36h11-family_fig5_336614405
- Fantastic resource for vision and other FRC programming concepts: https://docs.wpilib.org/en/stable/docs/software/vision-processing/apriltag/index.html