AprilTag tag recognition

Introduction to the results of routine experiments

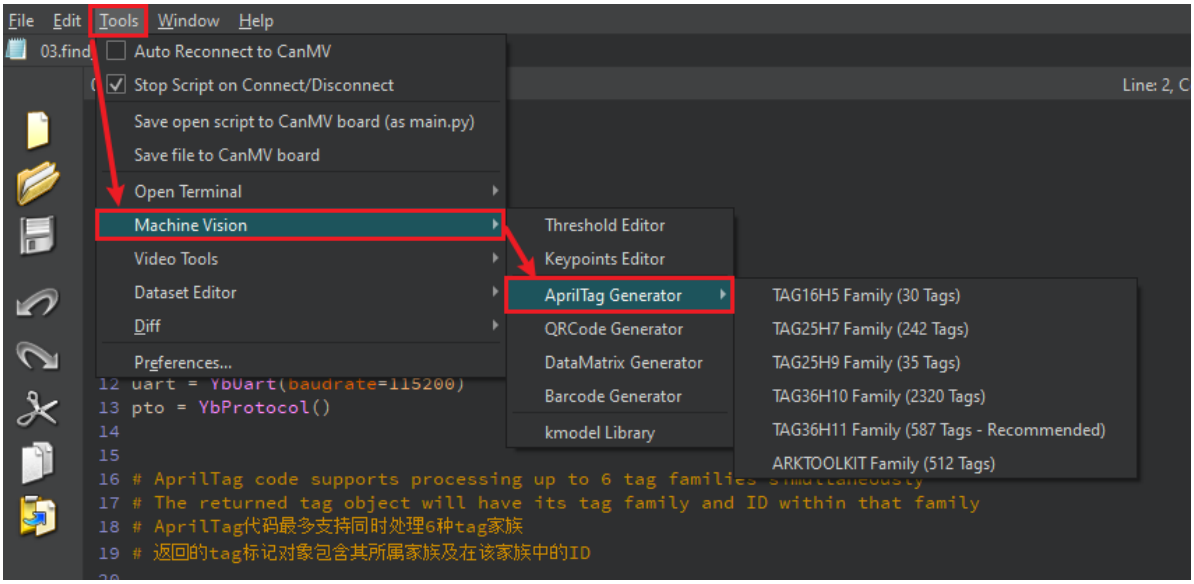
In this section, let's identify the AprilTag tag.

> AprilTag is a two-dimensional barcode for visual recognition, which is widely used in robot navigation, augmented reality, and camera calibration. It was originally developed by researchers at Georgia Institute of Technology. AprilTag is designed to provide a reliable and easy-to-recognize tagging system that can work efficiently in complex environments.
>
> Here are some of the key features of AprilTag:
>
> 1. **High-precision recognition** : AprilTag uses image processing technology to accurately identify and locate tags under a variety of lighting conditions and backgrounds.
> 2. **Information encoding** : Each AprilTag can encode different information, usually a unique integer ID, allowing the system to distinguish between multiple tags.
> 3. **Fast detection** : The system can quickly detect the tag position and orientation, which is particularly important for real-time applications such as robot localization and navigation.

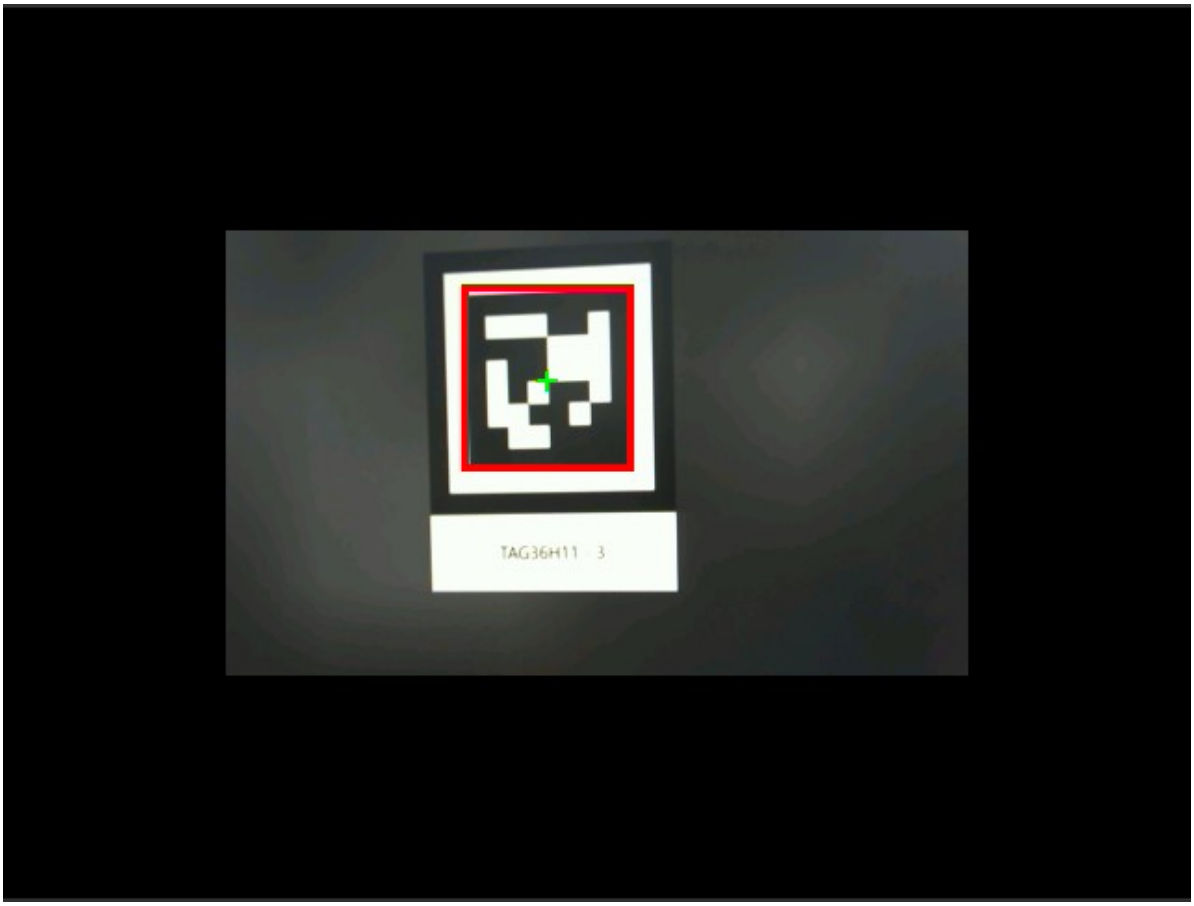
4. **Anti-interference** : AprilTag has strong anti-interference ability and can maintain a high recognition rate in the presence of noise or other interfering objects.
5. **Open source support** : AprilTag is open source and developers can use and modify it freely, which promotes its application in various projects.
6. **Easy to make** : Users can easily print out AprilTag and make labels for experiments or applications.

Due to these advantages, AprilTag is widely used in many fields such as robots, drones, AR/VR, and has become a popular choice for visual positioning and object recognition.

We have generated 587 tags of the tag36h11 family and placed them in the directory [K230 Tutorial\0. Development Board Data\AprilTags], which can be used directly for identification. You can also use the AprilTag generator that comes with CanMV IDE to generate AprilTag tags of other families.



Run the sample code and point the camera at the AprilTag code to be identified. You can see that the AprilTag code is marked with a red frame and a green cross arrow.

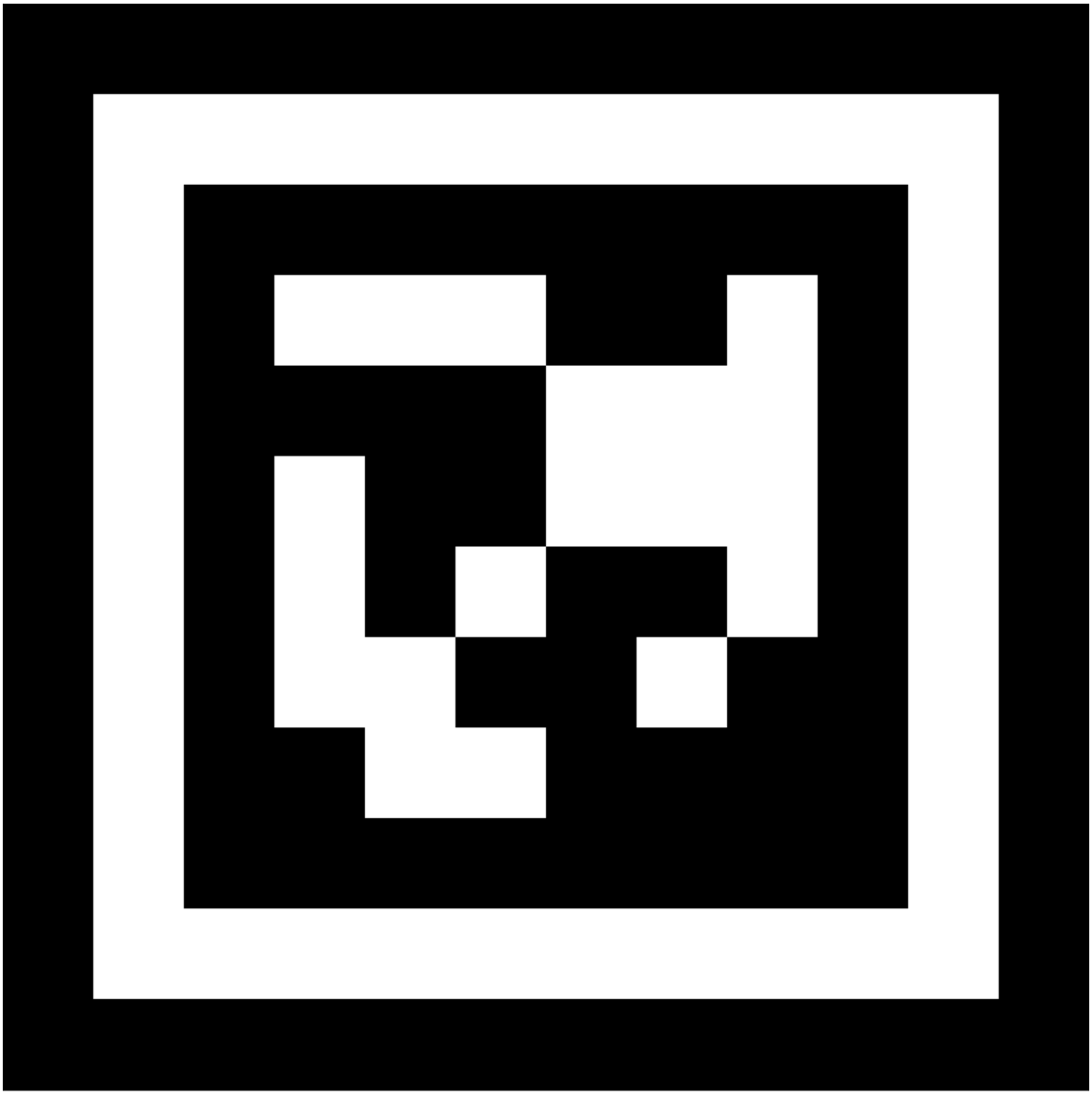


In the serial terminal output of CanMV IDE, you can see the following information output:

> Tag Family TAG36H11, Tag ID 0, rotation 2.835588 (degrees)

Where rotation represents the rotation offset

Example images



TAG36H11 - 3

Code Explanation

The peripherals we will use in this section are mainly camera modules

AprilTag detection and recognition is implemented by the find_apriltags() method in K230, which belongs to the image module

Complete code

```
# Import required modules
# 导入所需模块
import time, math, os, gc

from media.sensor import *
```

```
from media.display import *
from media.media import *

from libs.YbProtocol import YbProtocol
from ybUtils.YbUart import YbUart
# uart = None
uart = YbUart(baudrate=115200)
pto = YbProtocol()


# AprilTag code supports processing up to 6 tag families simultaneously
# The returned tag object will have its tag family and ID within that family
# AprilTag代码最多支持同时处理6种tag家族
# 返回的tag标记对象包含其所属家族及在该家族中的ID

# Initialize tag families bitmask
# 初始化tag家族位掩码
tag_families = 0
tag_families |= image.TAG16H5    # Comment out to disable this family / 注释掉以禁用此家族
tag_families |= image.TAG25H7    # Comment out to disable this family / 注释掉以禁用此家族
tag_families |= image.TAG25H9    # Comment out to disable this family / 注释掉以禁用此家族
tag_families |= image.TAG36H10   # Comment out to disable this family / 注释掉以禁用此家族
tag_families |= image.TAG36H11   # Comment out to disable this family (default) / 注释掉以禁用此家族(默认)
tag_families |= image.ARTOOLKIT  # Comment out to disable this family / 注释掉以禁用此家族

def family_name(tag):
    """
    Get the family name of a tag
    获取tag的家族名称

    Args:
        tag: AprilTag object / AprilTag对象
    Returns:
        str: Name of the tag family / tag家族名称
    """
    family_dict = {
        image.TAG16H5: "TAG16H5",
        image.TAG25H7: "TAG25H7",
        image.TAG25H9: "TAG25H9",
        image.TAG36H10: "TAG36H10",
        image.TAG36H11: "TAG36H11",
        image.ARTOOLKIT: "ARTOOLKIT"
    }
    return family_dict.get(tag.family())

# Initialize camera sensor
# 初始化摄像头传感器
sensor = Sensor()
sensor.reset()
sensor.set_framesize(width=400, height=240)
sensor.set_pixformat(Sensor.RGB565)
```

```
# Initialize display
# 初始化显示
Display.init(Display.ST7701, width=640, height=480, to_ide=True)
#Display.init(Display.VIRT, sensor.width(), sensor.height())

# Initialize media manager and start sensor
# 初始化媒体管理器并启动传感器
MediaManager.init()
sensor.run()

# Create clock for FPS calculation
# 创建时钟用于FPS计算
clock = time.clock()

# Main loop
# 主循环
while True:
    clock.tick()

    # Capture image
    # 捕获图像
    img = sensor.snapshot()

    # Find and process AprilTags
    # 查找和处理AprilTags
    for tag in img.find_apriltags(families=tag_families):
        # Draw rectangle and cross on detected tag
        # 在检测到的tag上绘制矩形和十字
        img.draw_rectangle(tag.rect(), color=(255, 0, 0), thickness=4)
        img.draw_cross(tag.cx(), tag.cy(), color=(0, 255, 0), thickness=2)

        # Print tag information
        # 打印tag信息
        print_args = (family_name(tag), tag.id(), (180 * tag.rotation()) /
math.pi)
        print("Tag Family %s, Tag ID %d, rotation %f (degrees)" % print_args)

        x, y, w, h = tag.rect()
        pto_data = pto.get_apriltag_data(x, y, w, h, tag.id(),
(180*tag.rotation())/math.pi)
        uart.send(pto_data)
        print(pto_data)
        break

    # Display image centered on screen
    # 在屏幕中央显示图像
    Display.show_image(img, x=120, y=120)

    # Print frames per second
    # 打印帧率
    print(clock.fps())
```

Code structure

The main functions and structure of this section's routine are as follows:

Basic settings section:

- First, import the necessary modules, including time, mathematical calculations, operating system, and garbage collection functions.
- Six different AprilTag tag families are initialized, and you can enable or disable specific families through comments.
- A family_name function is defined to convert the numeric ID of the tag family into a readable string name

Hardware initialization:

- Set the camera sensor parameters: the resolution is 1280x960, the actual working resolution is set to 320x240, and the RGB565 color format is used
- Initialize the display: Use ST7701 display, resolution 640x480
- Initialize the media manager and start the camera sensor

Main loop function:

- Continuously capture image frames
- Find the AprilTag in each frame
- For each tag found:
    - Mark the tag position with a red rectangle
    - Mark the center of the tag with a green cross
    - Print out the tag's family name, ID number, and rotation angle
- Display the processed image in the center of the screen
- Display the current frame rate (FPS)

find_apriltags()

```
image.find_apriltags([roi[, families=image.TAG36H11[, fx[, fy[, cx[, cy]]]]]])
```

This function finds all AprilTags within the specified ROI and returns a `image.apriltag` list of objects. For more information, see `image.apriltag` the documentation for the object.

Compared to QR codes, AprilTags can be effectively detected at greater distances, in poor lighting conditions, and in more distorted image environments. AprilTags are able to cope with various image distortion issues, while QR codes cannot. Therefore, AprilTags only encode a digital ID as their payload.

In addition, AprilTags can also be used for positioning. Each `image.apriltag` object will return its 3D position information and rotation angle. The position information is determined by `fx`, `fy`, `cx` and `cy`, which represent the focal length and center point of the image in the X and Y directions respectively.

> AprilTags can be created using the tag generator tool built into OpenMV IDE. This tool generates AprilTags in a printable 8.5"x11" format.

- `roi` is a rectangle tuple specifying the region of interest `(x, y, w, h)`. If not specified, ROI defaults to the entire image. The operation is limited to pixels within this region.

- `families` is a bitmask of tag families to decode, expressed as a logical OR of:
  - `image.TAG16H5`
  - `image.TAG25H7`
  - `image.TAG25H9`
  - `image.TAG36H10`
  - `image.TAG36H11`
  - `image.ARTOOLKIT`

The default setting is for the most commonly used `image.TAG36H11` tag families. Please note that each tag family enabled will slightly reduce `find_apriltags` the speed of .

- `fx` is the focal length of the camera in pixels in the X direction. The value for the standard OpenMV Cam is ((2.8 / 3.984) \times 656) which is calculated by dividing the focal length in millimeters by the length of the sensor in the X direction, multiplied by the number of pixels in the sensor in the X direction (for an OV7725 sensor).
- `fy` The focal length of the camera in pixels in the Y direction. The value for the standard OpenMV Cam is ((2.8 / 2.952) \times 488) which is calculated by dividing the focal length in millimeters by the length of the sensor in the Y direction, multiplied by the number of pixels in the sensor in the Y direction (for an OV7725 sensor).
- `cx` is the center of the image, that is `image.width()/2`, not `roi.w()/2`.
- `cy` is the center of the image, that is `image.height()/2`, not `roi.h()/2`.

Note: Compressed images and Bayer images are not supported.

kind `AprilTag`

The AprilTag object is returned by `image.find_apriltags` the function.

Constructor

```
class image.apriltag
```

Please use `image.find_apriltags()` the function to create this object.

`corners`

```
apriltag.corners()
```

This method returns a list of tuples containing the four corners of AprilTag, each tuple is in the format of (x, y). The order of the four corners is usually starting from the upper left corner and arranged in a clockwise direction.

`rect`

```
apriltag.rect()
```

This method returns a rectangle tuple (x, y, w, h) that can be used in other image processing methods, such as `image.draw_rectangle` the AprilTag bounding box in .

x

```
apriltag.x()
```

This method returns the x coordinate of the AprilTag's bounding box (int). You may also get this value using the [0] object.

y

```
apriltag.y()
```

This method returns the y coordinate of the AprilTag's bounding box (int). You may also get this value using [1] on the object.

w

```
apriltag.w()
```

This method returns the width of the AprilTag's bounding box (int). You may also get this value using [2] on the object.

h

```
apriltag.h()
```

This method returns the height of the AprilTag's bounding box (int). You may also get this value using [3] on the object.

id

```
apriltag.id()
```

This method returns the numeric ID of the AprilTag.

- TAG16H5 -> 0 to 29
- TAG25H7 -> 0 to 241
- TAG25H9 -> 0 to 34
- TAG36H10 -> 0 to 2319
- TAG36H11 -> 0 to 586
- ARTOOLKIT -> 0 to 511

You can also get this value doing [4] on the object.

family

```
apriltag.family()
```

This method returns the numeric family of AprilTag.

- image.TAG16H5
- image.TAG25H7

- image.TAG25H9
- image.TAG36H10
- image.TAG36H11
- image.ARTOOLKIT

You can also get this value doing [5] on the object.

`cx`

```
apriltag.cx()
```

This method returns the center x coordinate of the AprilTag (int). You may also get this value using [6] on the object.

`cy`

```
apriltag.cy()
```

This method returns the center y coordinate of the AprilTag (int). You may also get this value using [7] on the object.

`rotation`

```
apriltag.rotation()
```

This method returns the rotation angle of the AprilTag in radians (int). You may also get this value using [8] on the object.

`decision_margin`

```
aprilt

ag . decision_margin()
```

This method returns the decision margin of the AprilTag, reflecting the confidence in the detection (float). You may also get this value using [9] on the object.

`hamming`

```
apriltag.hamming()
```

This method returns the maximum acceptable Hamming distance (that is, the acceptable digit error) of AprilTag. The details are as follows:

- TAG16H5: Up to 0 bit error is acceptable
- TAG25H7: Up to 1 bit error can be tolerated
- TAG25H9: Up to 3 bit errors are acceptable
- TAG36H10: Up to 3 bit errors are acceptable
- TAG36H11: Up to 4 bit errors can be tolerated
- ARTOOLKIT: Up to 0 bit errors are acceptable

You can get this value doing [10] on the object.

goodness

```
apriltag.goodness()
```

This method returns the color saturation of the AprilTag image, ranging from 0.0 to 1.0, where 1.0 represents the best state.

> Currently this value is usually 0.0. In the future we plan to enable a feature called "Tag Refinement" to support detection of smaller AprilTags. However, currently this feature may cause the frame rate to drop below 1 FPS.

You can get this value using [11] on the object.

x_translation

```
apriltag.x_translation()
```

This method returns the camera's displacement in the x direction in unknown units.

This method is useful for determining the position of an AprilTag that is far from the camera. However, factors such as the size of the AprilTag and the lens you are using can affect where the x units belong. For ease of use, we recommend that you use a lookup table to convert the output of this method into information that is applicable to your application.

Note: The direction here is from left to right.

You can get this value using [12] on the object.

y_translation

```
apriltag.y_translation()
```

This method returns the camera's displacement in the y direction in unknown units.

This method is useful for determining the position of an AprilTag that is far from the camera. However, factors such as the size of the AprilTag and the lens you use can affect where the y units belong. For ease of use, we recommend that you use a lookup table to convert the output of this method into information that is applicable to your application.

Note: The direction here is from top to bottom.

You can get this value using [13] on the object.

z_translation

```
apriltag.z_translation()
```

This method returns the camera's displacement in the z direction in unknown units.

This method is useful for determining the position of an AprilTag that is far from the camera. However, factors such as the size of the AprilTag and the lens you are using can affect where the z units belong. For ease of use, we recommend that you use a lookup table to convert the output of this method into information that is applicable to your application.

Note: The direction here is from front to back.

You can get this value using [14] on the object.

`x_rotation`

```
apriltag.x_rotation()
```

This method returns the rotation angle of the AprilTag in the x-plane, measured in radians. For example, this method can be applied if the camera is moving from left to right to view the AprilTag.

You can get this value doing [15] on the object.

`y_rotation`

```
apriltag.y_rotation()
```

This method returns the rotation angle of the AprilTag in the y plane, measured in radians. For example, this method can be applied if the camera is moving from top to bottom to observe the AprilTag.

You can get this value using [16] on the object.

`z_rotation`

```
apriltag.z_rotation()
```

This method returns the rotation angle of the AprilTag in the z plane, measured in radians. For example, this method can be applied if the camera is rotated to view the AprilTag.

Note: This method is `apriltag.rotation()` a renamed version of .

You can get this value using [17] on the object.