Generating Aruco marker

ArUco markers are binary square fiducial markers that can be used for camera pose estimation. Their main benefit is that their detection is robust, fast and simple.Aruco

1) Script:loading the need packages

```
# import the necessary packages
import numpy as np
import argparse
import cv2
import sys
```

2) Script: handling argument from terminal.with running the program with this format.*python opencv_generate_aruco.py --id 24 --type DICT_5X5_100 --output tags/DICT_5X5_100_id24.png*

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-o", "--output", required=True,
    help="path to output image containing ArUCo tag")
ap.add_argument("-i", "--id", type=int, required=True,
    help="ID of ArUCo tag to generate")
ap.add_argument("-t", "--type", type=str,
    default="DICT_ARUCO_ORIGINAL",
    help="type of ArUCo tag to generate")
args = vars(ap.parse_args())
```

3) Script: define aruco dictionary.an ArUco dictionary specifies the type of ArUco marker we are generating and detecting aruco marker.The majority of these dictionaries follow a specific naming convention, `cv2.aruco.DICT_NxN_M`, with an *NxN* size followed by an integer value, *M* .what does the format mean The `cv2.aruco.DICT_8X8_50` value implies that we want to generate a binary 8×*8* square AruCo marker which 64 bits. We'll be able to generate 50 unique ArUco marker IDs using this dictionary.

```
# define names of each possible ArUco tag OpenCV supports
ARUCO_DICT = {
    "DICT_4X4_50": cv2.aruco.DICT_4X4_50,
```

```
    "DICT_4X4_100": cv2.aruco.DICT_4X4_100,
    "DICT_4X4_250": cv2.aruco.DICT_4X4_250,
    "DICT_4X4_1000": cv2.aruco.DICT_4X4_1000,
    "DICT_5X5_50": cv2.aruco.DICT_5X5_50,
    "DICT_5X5_100": cv2.aruco.DICT_5X5_100,
    "DICT_5X5_250": cv2.aruco.DICT_5X5_250,
    "DICT_5X5_1000": cv2.aruco.DICT_5X5_1000,
    "DICT_6X6_50": cv2.aruco.DICT_6X6_50,
    "DICT_6X6_100": cv2.aruco.DICT_6X6_100,
    "DICT_6X6_250": cv2.aruco.DICT_6X6_250,
    "DICT_6X6_1000": cv2.aruco.DICT_6X6_1000,
    "DICT_7X7_50": cv2.aruco.DICT_7X7_50,
    "DICT_7X7_100": cv2.aruco.DICT_7X7_100,
    "DICT_7X7_250": cv2.aruco.DICT_7X7_250,
    "DICT_7X7_1000": cv2.aruco.DICT_7X7_1000,
    "DICT_ARUCO_ORIGINAL": cv2.aruco.DICT_ARUCO_ORIGINAL,
    "DICT_APRILTAG_16h5": cv2.aruco.DICT_APRILTAG_16h5,
    "DICT_APRILTAG_25h9": cv2.aruco.DICT_APRILTAG_25h9,
    "DICT_APRILTAG_36h10": cv2.aruco.DICT_APRILTAG_36h10,
    "DICT_APRILTAG_36h11": cv2.aruco.DICT_APRILTAG_36h11
}
```

4) Script:if tag argument is provided check if it exist in the aruco dictionary.if not notify the user and exit the program

```
# verify that the supplied ArUCo tag exists and is supported by
# OpenCV
if ARUCO_DICT.get(args["type"], None) is None:
    print("[INFO] ArUCo tag of '{}' is not supported".format(
        args["type"]))
    sys.exit(0)
```

5) Script:

```
# load the ArUCo dictionary

arucoDict = cv2.aruco.Dictionary_get(ARUCO_DICT[args["type"]])
```

6) Script:

```
# allocate memory for the output ArUCo tag and then draw the ArUCo
# tag on the output image
print("[INFO] generating ArUCo tag type '{}' with ID '{}'".format(
    args["type"], args["id"]))
tag = np.zeros((300, 300, 1), dtype="uint8")
cv2.aruco.drawMarker(arucoDict, args["id"], 300, tag, 1)
```

7) script:Writing the created tag to disk with the file name specified and showing it to the the user

```
# write the generated ArUCo tag to disk and then display it to our
# screen
cv2.imwrite(args["output"], tag)
cv2.imshow("ArUCo Tag", tag)
cv2.waitKey(0)
```

Result