南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

计算机科学与工程系
Department of Computer Science and Engineering

计算机科学与工程系
CS413-创新实践 III

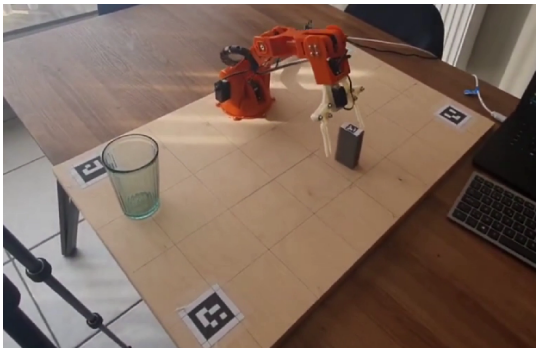
Optimization of ArUco functions in OpenCV

Author: Tian LunShuo & Bai LianJi
Supervisor: Yu ShiQi
Time: 2024/12/25

I. ProjectBackground

Pose estimation is of great importance in many computer vision applications: robotics navigation, augmented reality, and many more(1). This process is based on finding correspondences between points in the real environment and their 2D image projection. This is usually a difficult step, and thus, it is common to use synthetic or fiducial markers to make it easier. Developed by Rafael Muñoz and Sergio Garrido, one of

(a) ArUco is used during the takeoff and landing of drones.



(b) ArUco is used in the process of robot localization.

Figure 1. Examples of ArUco applications.

the most popular approaches is the use of the ArUco module, based on binary square fiducial markers(2). The main benefit of these markers is that a single marker provides enough correspondences (its four corners) to obtain the camera pose. In addition, the inner binary codification makes them particularly robust, allowing the possibility of applying error detection and correction techniques. Due to its good performance and simplicity of

Figure 2. Example of an ArUco marker.

use, the ArUco module was initially integrated into the OpenCV_contrib library as an additional feature during the early stages of development. It continued to evolve and was officially incorporated into the main OpenCV module in December 2022. This version of ArUco already has good performance, which led the development team to shift their focus from further engineering updates to enhancing additional features and optimizing algorithms. Consequently, several new features and algorithmic improvements have been introduced in ArUco but have not yet been synchronized with OpenCV.

Recently, the OpenCV 4.11 version has been scheduled for an update. As one of the most frequently used modules, ArUco' s debugging and optimization have been given high priority. Through constructive discussions between ArUco developers and OpenCV leaders to promote project cooperation, this project is committed to

evaluating and quantifying ArUco' s official algorithm updates that are not incorporated into OpenCV, with the aim of upgrading the ArUco algorithm module in OpenCV.

II. Project Preliminary Discussion

After determining the project direction, we got in touch with Professor Rafael Munoz, the developer of ArUco, and had an online discussion, specifically understanding the update progress of ArUco in recent years, including:

1) ArUco3: An algorithm optimization provided by Professor Rafael Munoz's student Francisco J.Romero-Ramirez, which mainly optimized detector and added to the OpenCV ArUco module in the form of additional parameters.. He has provided some meaningful updates to ArUco, such as the angular accuracy optimization algorithm ArUco3 and fractal labeling, a new method for remote marker pose estimation under occlusion. Since then, he has worked in the areas of 3D model tracking and human pose estimation. According to his paper "Speeded up detection of squared fiducial markers", this optimization can increase the detection speed of ArUco by up to about 4 to 17 times.

2) ArUcoNano: A microkernel file written by the Rafael Munoz development team that contains most of the common functions of ArUco, integrated in a simple.h file. This file is rewritten with simple processes for complex algorithms that previously required multiple dependencies, making it possible to simply call ArUco's common functions in a single file with little dependency.

III. Technological Pre-analyze

After communicating with the development team of ArUco, we conducted an extensive investigation into the actual use of ArUco. ArUco is usually called by developers from the OpenCV library, rather than the separate ArUco Lib library. At the same time, because ArUco code is often used as a real-time recognition scene, such as often used in robot image recognition, such as UAV accurate landing positioning, robotic arm pose estimation and other scenes, developers still have expectations for its higher processing accuracy and processing speed. But at the same time, even though OpenCV gives developers multiple parameters to adjust, these parameters are usually not used, people prefer to use the default parameters, which is considered to be the best solution for OpenCV pre-debugging.

Based on this information, we believe that it is necessary to evaluate ArUco's recognition performance of raw video streams in real situations, including some common low-quality camera images. At the same time, ArUco's uptime and accuracy are also the main metrics that we need to balance. We want to do this in a way that is based on adjusting default parameters and the internal architecture of the code, so that there is no additional development cost to the developer.

IV. Theoretical Pre-Analyze

Through the analysis of the algorithm paper experiment[footnote], we believe that the performance test of the experiment is too simple:

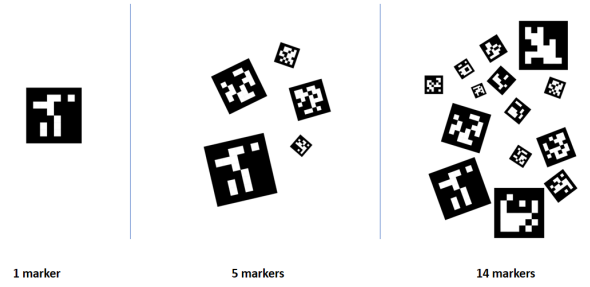
- This experiment only used video streams with varying distances between the camera and ArUco markers as test data. These images, captured from far to near, could not cover all common situations, such as angle distortion.
- At the same time, in common engineering applications, the camera is usually a depth camera or USB camera, and there are many unavoidable issues that affect picture quality, such as noise, jitter, and distortion. However, the dataset for this experiment was shot using a mobile phone, and the original video was optimized by an algorithm, which may have masked these problems and could not represent actual application scenarios.
- Compared with the theoretically optimal ARUCO_MIP_36h12 two-dimensional code [1], the Hamming distance in the two-dimensional code of the dictionary used in this experiment is relatively close, making the decoding process more prone to errors.

Therefore, we adjusted the main direction of the project, starting with the design of a comprehensive test scheme, and conducted a more detailed and realistic performance test of ArUco's algorithm update.

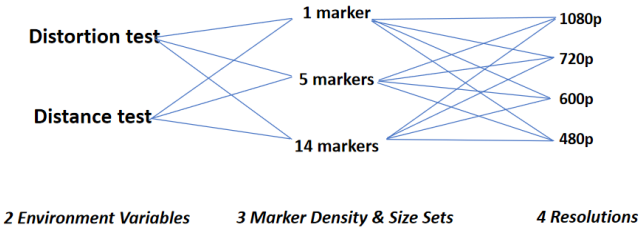
V. Dataset Preparation

In order to ensure that the dataset used for image detection has the same characteristics as the actual application, we used a 1080p resolution Jerui Micro USB camera for shooting, which is also the main camera type used in ArUco recognition. It was found that the video shot by mobile phones and the video shot by USB cameras have obvious differences in picture noise, angle distortion, and other issues.

In practical application scenarios, we comprehensively simulated the image characteristics that may arise in the use of ArUco markers and designed and collected several sets of data (3b). Each dataset contained three different ArUco cluster densities (1, 5, and 14 ArUco markers, 3a) within the same area. This approach allowed us to analyze performance under varying ArUco engineering requirements. For each dataset, the key controlled



(a) Three different ArUco cluster densities (1, 5, and 14 ArUco markers).



(b) Dataset Cross Classification

Figure 3. Dataset Preparation Process

variables were as follows:

1) The inclination angle between the marker and the camera, 4a: This variable was controlled by rotating the camera horizontally from left to right to alter the shooting angle.

2) The proportion of the marker in the image, 4b: This variable was controlled by adjusting the vertical distance between the camera and the marker (from far to near) to modify the marker' s relative size within the image.

[1]Generation of fiducial marker dictionaries using mixed integer linear programming

3) Image resolution: The effect of different resolutions on ArUco detection was assessed by compressing the original 1080p images to generate datasets with 480p, 600p, and 720p resolutions. This approach allowed us to

(a) The inclination angle between markers and camera.

(b) The proportion of the marker in the image.

Figure 4. Dataset Example Images

analyze the performance of ArUco under varying resolution conditions.

VI. ArUco3

ArUco3 has already been integrated into the OpenCV main library in previous updates.Compared to the traditional ArUco, ArUco3 introduces a more complex identification module and a new key parameter, minSideLengthCanonicalImg. This parameter controls the minimum side length of the marker in the standardized marker image, which helps to a certain extent in eliminating noise and false marker interference, thereby improving detection accuracy and robustness.

The default value of minSideLengthCanonicalImg is 32, which represents the minimum side length (pixels) of the marker in the normalized marker image. Its primary function is to exclude excessively small noise or false markers, thereby improving the accuracy and stability of detection.

If minSideLengthCanonicalImg is set too small, it may result in the misdetection of noise or false markers. Conversely, if it is set too large, smaller markers may be missed during detection.

Theoretically, the larger the value of minSideLengthCanonicalImg, the shorter the processing time required.

To conduct an in-depth analysis of the effect of minSideLengthCanonicalImg on the algorithm's performance, we performed experiments using three self-constructed datasets. Specifically, we set the value range of this parameter from 5 to 40 and systematically tested detection accuracy and runtime under different parameter conditions. The experimental results are presented below(5, 6, 7).

By minSideLengthCanonicalImg parameter adjustment to the more suitable values, 12, compared with the default value of OpenCV detection efficiency significantly increased(5, 6), the growth range of 35% to 300%, and consumed time almost remain the same(7), showing high optimization potential.

In the distortion test, although ArUco3 shows some robustness in some scenarios, its highest detection accuracy fails to surpass the traditional ArUco. Especially in scenes with large lens torsion Angle and serious viewing Angle distortion, the detection accuracy of ArUco is usually better than that of ArUco3, indicating that ArUco has stronger adaptability under extreme shooting conditions.

In scenes where the lens twist Angle is small and the identification code takes a low proportion in the image, especially when the image resolution is high (1080p or above), the detection accuracy of ArUco3 is slightly higher than that of ArUco, indicating that ArUco3 has certain advantages in high-resolution detail detection.

To sum up, the original ArUco3 default parameters detection efficiency significantly below ArUco, but in such key parameters as minSideLengthCanonicalImg optimized adjusted ArUco3 sure to better than ArUco performance in a specific application scenario, has certain actual application value and potential for improvement.

This evaluation underscores that the integration of ArUco3 has been unsuccessful. OpenCV's integrated ArUco consistently demonstrated greater robustness across a broader range of conditions. From an engineering perspective, we recommend removing ArUco3 from future OpenCV versions to maintain a streamlined and user-friendly codebase.

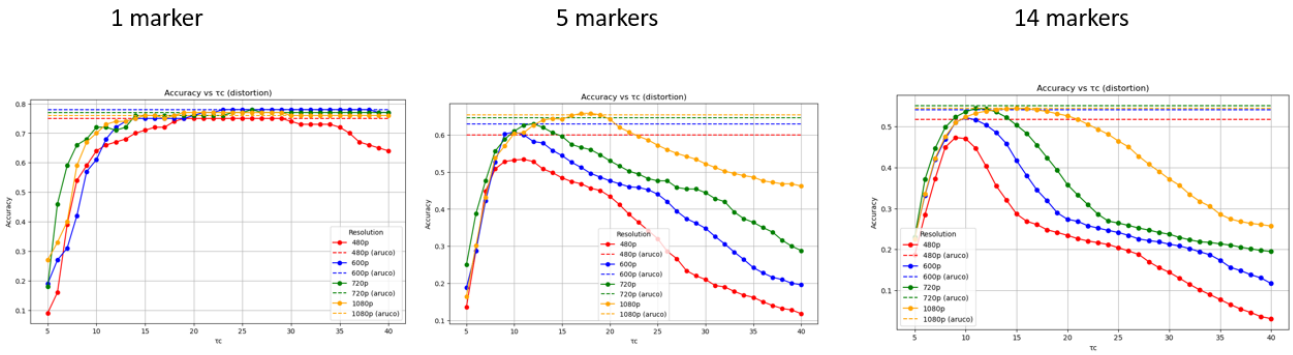


Figure 5. Distortion dataset

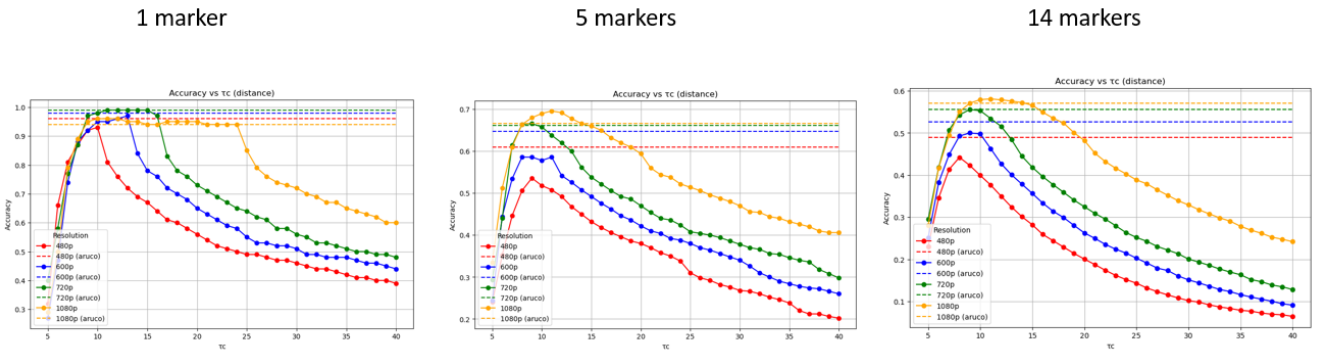


Figure 6. Distance dataset

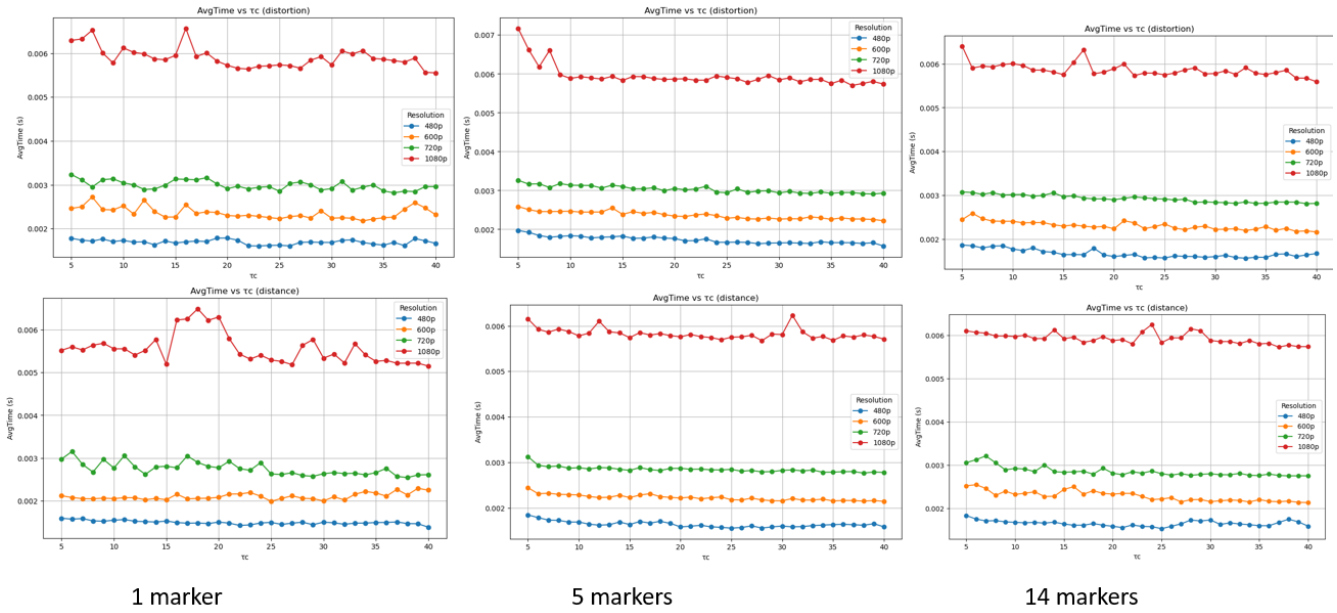


Figure 7. Time cost versus τc

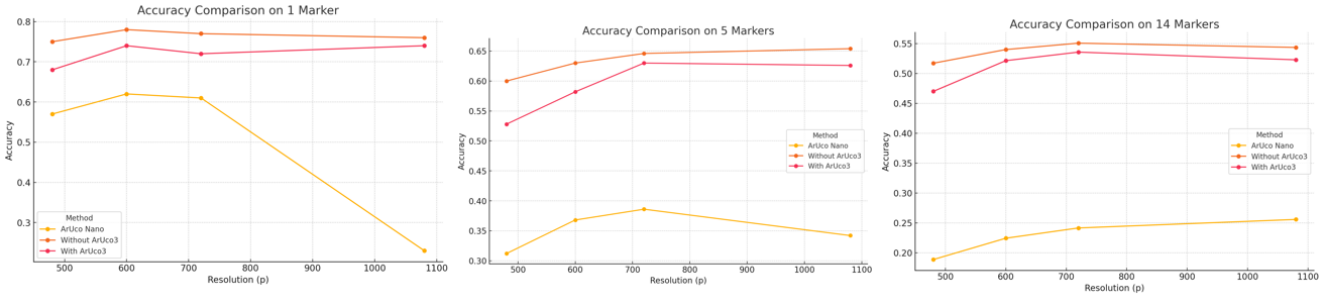


Figure 8. Nano, ArUco, ArUco3 on distortion dataset

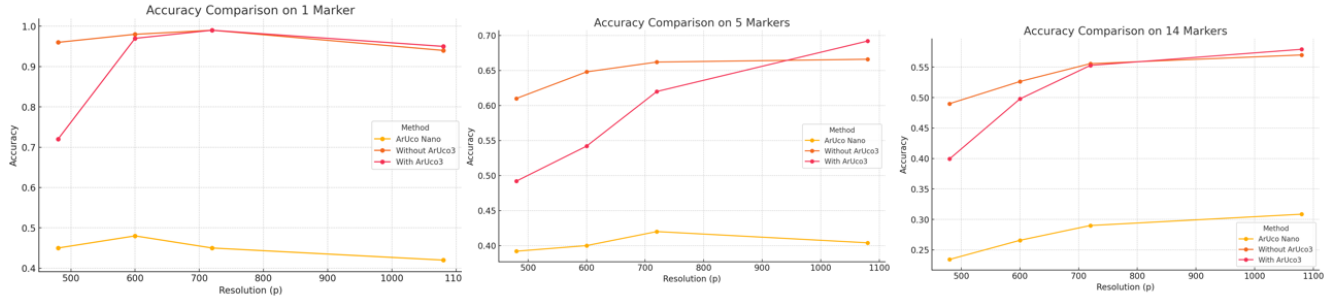


Figure 9. Nano, ArUco, ArUco3 on distance dataset

VII. ArUcoNano

We also tested the performance of ArUcoNano on the data set by comparing the performance of ArUcoNano at different resolutions.

ArUcoNano has a simpler algorithm process than other modules, so it is reasonable to assume that ArUcoNano is more efficient. The results of our efficiency tests confirm our view that ArUcoNano has a 10% to 20% uptime improvement compared to the OpenCV integrated ArUco module.

But tests of recognition accuracy (8, 9) showed that the accuracy performance of ArUcoNano module at all resolutions is about 50% lower than that of ArUco and ArUco3.

In conclusion, ArUcoNano is not as robust as the other two algorithms in the recognition of common low-quality images such as Angle distortion, image blur, and excessive image noise.

ArUcoNano's performance analysis shows that it can't meet developers' expectations for real-world applications. Therefore, after discussions with the developer team and OpenCV, we have decided not to integrate ArUcoNano into OpenCV.

VIII. Conclusion

This project conducted a detailed evaluation of different versions of the ArUco algorithm, revealing their performance differences in real-world applications. We tested the performance of ArUco, ArUco3, and ArUcoNano, focusing on their detection accuracy and efficiency under low-quality images, varying resolutions, and complex environmental conditions.

In the evaluation of ArUco3, although optimizing the minSideLengthCanonicalImg parameter can significantly improve detection efficiency, the traditional ArUco algorithm still outperforms ArUco3 under extreme shooting conditions (e.g., large-angle distortion). Therefore, we recommend continuing to use the traditional ArUco in future OpenCV versions, and we consider safely removing ArUco3 from OpenCV to maintain a simplified and user-friendly codebase.

As for ArUcoNano, although it excels in processing speed, its recognition accuracy in real-world applications, especially when dealing with low-quality images, is far lower than that of ArUco and ArUco3. After discussions with the development team, it was decided not to integrate ArUcoNano into OpenCV.

Additionally, the dataset and evaluation metrics tailored for testing ArUco algorithms in this experiment are more comprehensive than those previously provided by the UCO team. We have considered more complex scenarios that might arise in real-world applications. This customized testing approach provides more reliable results and valuable references for future ArUco developers.

References

[1]Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6), 2280-2292.

[2]Avola, D., Cinque, L., Foresti, G. L., Mercuri, C., & Pannone, D. (2016, February). A practical framework for the development of augmented reality applications by using ArUco markers. In International Conference on Pattern Recognition Applications and Methods (Vol. 2, pp. 645-654). SciTePress.

[3]Mráz, E., Rodina, J., & Babinec, A. (2020, October). Using fiducial markers to improve localization of a drone. In 2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR) (pp. 1-5). IEEE.

[4]Roos-Hoefgeest, S., Garcia, I. A., & Gonzalez, R. C. (2021, October). Mobile robot localization in industrial environments using a ring of cameras and ArUco markers. In IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-6). IEEE.

[5]Romero-Ramirez, F. J., Muñoz-Salinas, R., & Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. Image and vision Computing, 76, 38-47.

[6]Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. Pattern recognition, 51, 481-491.