iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** XII **Month of publication:** December 2022

DOI: https://doi.org/10.22214/ijraset.2022.47919

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XII Dec 2022- Available at www.ijraset.com

Comparison of Different Tracking Algorithms in OpenCV

Anshu Sarkar[1], Subhash Negi[2], Ashutosh Dangi[3]
[1, 2, 3]*Jagan Institute of Management Studies (Sec – 5 Rohini), New Delhi-110085*

Abstract: ***In today's world, mechanical processes long performed by human vision have been superseded by computer vision, which uses cameras and algorithms to automate processes. Object trackers [2] are an important part of this computer vision. In this article, I tried to compare different kinds of tracking algorithms [5] implemented in OpenCV based on different factors.***

I. INTRODUCTION

Object tracking is a deep learning application in which an object is detected and divided into a series of frames, with each frame recording its trajectory. This object can be a person, a bat, a vehicle, or anything in a series of frames. Object tracking is normally performed in two models. The first is the motion model, which tracks the speed and direction of object movement, and the second is the appearance model [4], which defines the selected object within a frame. Object tracking through OpenCV is one famous option in this area. OpenCV consists of a set of built-in functions designed for object tracking, so there are many different object trackers you can use [7], and choosing a particular tracker depends on the application you develop. Developing a universal tracking algorithm Is almost impossible. OpenCV includes interfaces such as Java, Python, C, C++, Windows, Mac, Linux, Android, etc. Object tracking can be accessed using any OpenCV application [11]. There are two concepts [3] of tracking and detection, when we track an object that was detected in the previous frame, we know a lot about the object's appearance, its position in the last frame, its speed and direction of movement, so tracking is detection is faster than So with all the information you can easily predict the position of the object in the next frame [4]. Tracking is also useful when discovery fails. Few tracking algorithms available. Each has its own purpose, specific strengths, and weaknesses [1].

Some of the tracking algorithms are shown below [2] -

1) *Boosting Tracker:* Based on the algorithm using incorporation of machine learning in Haar Cascade [9], but this age over 10 years, unhurried and works satisfactorily. But it has mediocre tracking performance.
2) *MIL Tracker:* Works very well in various cases and is more accurate than Boost-her tracker, but bugs are poorly reported.
3) *KCF Tracker:* Kernelized Correlation Filter. More accurate and faster than Boost and MIL. Like MIL and KCF, but does not manage full occlusion very well.
4) *CSRT Tracker:* It is more accurate than KCF, but somewhat slower.
5) *Median Flow Tracker:* Great for reporting errors. But if the movement has too big a jump, like B. fast moving object, it will fail [6].
6) *TLD Tracker:* Incredibly false-positive and potentially unusable, but works best under multi-frame occlusion.
7) *MOSSE Tracker:* Good for tracking, but not as good as CSRT or KCF. If your tracking criteria are purely velocity-based, this is a very good choice.
8) *GOTURN Tracker:* The only deep learning-based object detector, it employs the "Caffe" model for tracking and needs extra model files to function [10].

II. ALGORITHM DESCRIPTION

OpenCV has a huge collection of libraries, but the available algorithms are limited. Here we use 3 feature detectors, 3 pure trackers and a tracking framework [7].

A. Feature Detector

Objects are detected in each frame. It helps identify different features in object images and find the best mapping within the frame. OpenCV mainly has his three feature detectors: SURF, SIFT and ORB [6].

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue XII Dec 2022- Available at www.ijraset.com

B. Pure Trackers

A pure tracker is designed to track an object by identifying its trajectory and predicting its future position. This includes fixing errors in the process. OpenCV has 3 pure trackers: - MIL, Boosting and MedianFlow [11].

C. Tracing Frameworks

This type of algorithm aims to provide the most complex responses to tracking activity. They regularly adapt to new situations and correct big mistakes. OpenCV has a tracking framework: TLD

III. MEASUREMENT METHODS

Performing complex and complex comparisons of tracking algorithms can be very difficult. One of the main reasons for this is that tracking itself can be used for different purposes (tracking faces, people, vehicles, etc.) in different environments and locations (airports, offices, streets, etc.). Low quality/high quality, near/far, relocation, etc. Tracking itself also consists of overcoming different types of problems, such as: For example, partial occlusion [9] of tracked objects, changing lighting conditions, and fast camera movements that cause blurry images. So, this requires a very large dataset.

A. Success Evaluation

We used the following formula for each frame: Csuc (f) = |rt∩rg|/ |rt∪rg| where Csuc(f) [4]is also the matching criterion function for frame f. rt is the bounding box and rg is the bounding box.

B. Precision Evaluation

We used the rectangular scale obtained from the tracker and used the bottom truth and the best accuracy is reasonably 1 using the formula: Cprec (f) = |rt|/ |rg| [4] where, Cprec(f) is the accuracy criterion function for the currently processed frame f [9].

C. Time Demands Evaluation

We propose to simply measure the time of each frame. Ctim(f) = t where Ctim(f) [4] is the algorithm time required criterion function. t is the time it took to process this frame.

IV. DIFFICULTIES

Many issues had to be resolved before the data could be collected and processed. The original idea was to test how much the bounding box provided by the tracker and the lower truth bounding box overlap, and measure the separation at the same time. The first problem was that sometimes the tracker would reject items from subpages on initialization, or lose items entirely during tracking. Data is generated at every frame, so even if tracking fails, the tracker may not be able to handle all tracking issues in the video currently being processed. In benchmarking, you can often fix this by triggering a reinitialization after a few failed frames. I decided to set the re-trigger threshold to 30 failed frames [10]. This could mean that some trackers look easier (ground truth may not be available). In very rare cases, another issue has been discovered where the tracker behaves unexpectedly and the output data is outside the expected range. Sometimes it took a few seconds to process a relatively small image, sometimes it took a few seconds to process almost the entire image, but the bottom truth-limiting rectangle was much smaller. Such performance is reclassified as an error and information is functionally restricted to achieve penalties and obtain reasonable statistical records [5].

V. RESULTS

A. Success

Count the number of frames that gave a result within a defined limit ($Csuc(f) \geq 0.5 \wedge Cscale(f) < 2.0 \wedge t(f) < 1$) [6], and denote this number by the total number of split frames. I split. Surprisingly, despite its complexity, TLD was not the most effective algorithm. Two reasons for this -

First, the algorithms MIL and Boosting classified as pure trackers were more successful thanks to reinitialization from Underside Truth., the TLD may be so self-correcting that the bounding box moves so much from frame to frame that it does not even follow the bottom truth bounding box (sometimes less than 50% intersection) [6].

It does not mean that TLD loses stuff. Select is true but does not center the element in its bounding box.

©IJRASET: All Rights are Reserved

B. Precision

I decided to use a scaling ratio to measure the accuracy of each algorithm. If the item is found exactly, the dimensions of the bounding rectangle should be about the same as the truth below than otherwise. This criterion excludes all failed data and suggests a maximum acceptable scaling ratio of 2 [9]. The extreme values of all algorithms were below this upper bound and the lower bound was of course 0. So, to determine which algorithms performed well according to this criterion, we need to see how close their mean is to 1.0 [1]. Precise minimal and maximal values

C. Time Demands

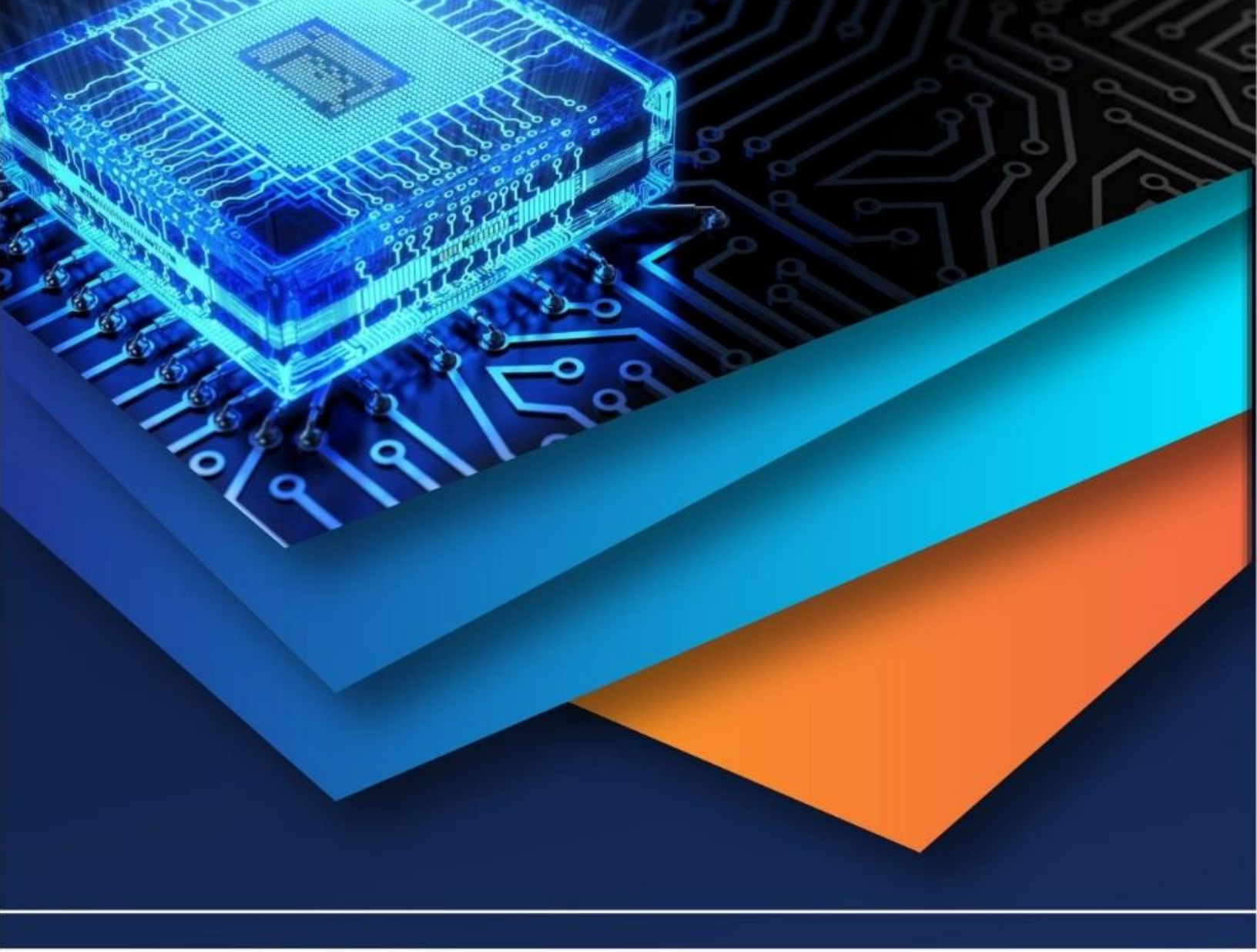
For this metric, we measured the time it takes each algorithm to process a frame. I had to remove not only the very short time but also the very long time, as the boxplot would be extremely overstretched. SIFT and SURF are very slow. This is because it deals with floating point values and internal calculations take time [8]. The rest of the algorithms are fast, but TLD is the slowest. The slowness of the TLD algorithm is due to its relative robustness and lack of optimization in the OpenCV implementation.

VI. CONCLUSION

The first part is that success brings unexpected results. Of course, the TLD algorithm, which was considered the best on the research base so far, was not all that great. Partly due to its complexity (I could not put the target blob in place due to customization), partly he was due to poor implementation in OpenCV. The second criterion - precision - gives consistent results in terms of standards [4][5]. On the other hand, significant differences are often seen between the minimum and maximum levels of this measure. These results are marked as predictable based on previous research. The long run work on this research is going to be focused of detailed analysis of implementation of those algorithms, including used programming techniques and memory and computing efficiency [8].

REFERENCES

[1] D. Li, B. Liang, aW. Zhang, „ Real- time moving vehicle discovery, shadowing, and reckoning system enforced with OpenCV", in 2014 fourth IEEE International Conference on scientific discipline and Technology, 2014, s. 631 – 634.
[2] Lowe, D.G.," Distinctive Image options from ScaleInvariant Keypoints", International Journal of laptop Vision, 60, 2, pp. 91- 110, 2004.
[3] OpenCV preface to SIFT (Scale- Invariant purpose transfigure)", recaptured Commonwealth Day, 2016, from http//docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html
[4] Bay, H. and Tuytelaars,T. and Van Gool,L.," suds accelerated sturdy Features", ninth European Conference on laptop Vision, 2006.
[5] preface to suds (Speeded- Up sturdy Features)", recaptured Commonwealth Day, 2016, from http//docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro. html
[6] Features2D Homography to search out a given object", recaptured Commonwealth Day, 2016, from http//docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html
[7] Ethan Rublee, Vincent Rabaud, Kurt Konolige, GaryR. Bradski ORB a good volition to SIFT or SURF. ICCV20112564-2571.
[8] B. Babenko, M-H. Yang, andS. Belongie," Visual trailing with on-line Multiple Instance Learning", In CVPR, 2009.
[9] Z. Kalal, K. Mikolajczyk, andJ. Matas," ForwardBackward Error Automatic Discovery of trailing Failures", International Conference on Pattern Recognition, 2010, pp. 23- 26.
[10] Milton Friedman,J.H., Hastie,T. and Tibshirani,R.," additive provision Retrogression a applied mathematics read of Boosting.", Technical Report,Dept. of Statistics, Stanford, 1998.
[11] Z. Kalal,K. Mikolajczyk, andJ. Matas," trailing literacy- Discovery," Pattern Analysis and Machine Intelligence 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)