Camera Pose Estimation: What is the computationally efficient approach?

Emad W. William*, Hossam E. Abd El Munim*, Sherif Hammad† and Maged Ghoneima†
*Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University
†Mechatronics Engineering Department, Faculty of Engineering, Ain Shams University

***Abstract*—In this paper we present a system that is suitable for real time camera pose estimation over a video stream, Our approach proves to be computationally efficient for real time applications to calculate the new extrinsic parameters for a moving camera. The system depends only on the camera without the aid of any external sensors or fixed points. Our analysis for the problem shows that the most computationally intensive task is in the feature detectors/descriptors used, for that we will be comparing between 2 feature detectors/descriptors; AKAZE and ORB, that are well known for their good performance in terms of computational cost. Although AKAZE has managed to achieve better results, it comes with higher computational cost yet with substantial gain in terms of quality, this will provide more robust results for our system if compared to ORB. On the other hand, ORB still manages to give enough quality to satisfy our needs and with real-time performance.**

I. Introduction

Camera Pose Estimation is essential in many computer vision applications like Head pose estimation, Path planning and localization, Driver monitoring systems, Moving object detection and autonomous vehicles/robots. While all of these applications rely on having a calibrated camera with a known camera matrix, the intrinsic parameters will be the same as long as the camera is not altered. However, for the extrinsic parameters, It will be changeable in case of a movable camera as these extrinsic parameters describe the new pose for the camera.

There are many ways to get the new camera pose for a constraint movable camera in a closed environment, One is by calculating the camera's Yaw, Pitch and Roll using external sensors, Another is by having some fixed points to trace them in each image and detect the new pose by calculating the orientation of these fixed points. While all of these algorithms provide efficient solutions to our problem, The major advancements in computer vision algorithms as well as camera sensors and processors help us to depend solely on the camera and the computer vision without any support from other external sensors or having additional aiding points of any means. Thanks to stereo vision and epipolar geometry, One can find a solution to estimate the phase difference between 2 stereo cameras for 3D reconstruction. One way is by providing the correspondences of at least 8 good points between the 2 images captured from both camera sensors. By knowing the intrinsic parameters we can use these 8 correspondences to decompose the Essential matrix to estimate the extrinsic parameters as described in [1].

The same idea could be done for Camera pose estimation by using a single camera, knowing it's intrinsic parameters as well, we can easily calculate the new extrinsic parameters. Firstly, we get to capture 2 images but this time using the same camera, we get also the correspondences between this 2 images in at least 8 points, Then, we use them to solve the Essential matrix and get the new extrinsic parameters relative to the old parameters. Hence, we get the direction of motion and the new estimated camera pose.

Since to estimate the new camera pose we will always need to solve this Essential matrix, We will need to automatically detect some points in the 2 images to show correspondences between these 2 images and to match the points in the 2 images and eliminate any outlier points. All of these steps will add to the computational effort until we reach the correctly estimated parameters.

Here in this paper in the Methods section we will present the System and Algorithm description, the proposed feature detectors and descriptors while providing the motivation behind them, we will also be showing how each step contributes to the computational effort of the algorithm and which of these steps is the bottle neck that needs to be optimized to get the best computationally efficient approach.

A. Related work

As we described earlier, the way to estimate the camera pose relys mainly in the solution of the Fundamental matrix, Many optimizations and contributions have been made to this step that further leads to a minimal case for such problem which is based only on five points to solve the Fundamental matrix and get the estimated pose based on the state of the art Nisters algorithm[2], other contributions to this solution as in Hongdong Li and Richard Hartley's[3] that provides another way of implementation for the so called Five-point solution by eliminating many unknowns all at once instead of eliminating the unknown variables one by one sequentially as in [2]. Their contribution also shows another further improvements in the equation solving stage that makes it easier for implementation than the original algorithm.

Another contribution made to the Five-Point algorithm as described in "Fast Iterative Five point Relative Pose Estimation" [4], This proposes a new iterative method which is based on

Powells method[5] for finding a local minimum of a function. This improvement returns the same precision yet it's much faster than Nisters algorithm.
On the other side, In the part of feature detectors and descriptors that aims to find the required 5 points correspondance to match between the two images, many algorithms have been developed. The most famous or these are SIFT[6] and SURF[7]. Other algorithm is proposed which was highly recommended for our application which is ORB described in "Orb: An efficient alternative to sift or surf"[8]. The work discussed in this paper proposes a new binary descriptor based on BRIEF [9] but is rotation invariant and has better performance dealing with imposed noise, It also shows how it compares to the well known for speed SIFT or SURF as well. Another performance comparison later on is done by"Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images"[10] where the authors describe how all of these algorithms compare to each other in terms of performance and speed. Lastly "A comparison of object detection algorithms using unmanipulated testing images"[11] which adds another highly recommended algorithm AKAZE [12]. Here we will be focusing mainly on AKAZE and ORB showing how these algorithms perform in terms of speed and whether this will translate into significant gain for the performance of our application or not.

II. Methods

A. System Description

Our System is constructed of a monochrome single camera sensor with 2 IR Leds for illumination without any additional sensors, This camera sensor supports up to 1,920x1,080 resolution and 30 frames per second. The camera shall be fixed on a holder and has a defined range of motion around all axes so that there will always be a correlation between the 1st frame captured (center frame) and any captured frame later on.
The camera is connected directly to the ECU where it contains an ARM cortex CPU; however, for simulation purpose we used the video feed directly from the camera to a laptop with an Intel 6th gen Core-i5 6300U 2.4GHz processor and 8GB of RAM. OpenCV C/C++ library for Windows is used to implement our algorithm and perform all the needed tests. Although our simulation videos were fed as RGB format, they are converted to grey scale to be used in our algorithm.
Our expected output would be the new rotation matrix which indicates the Roll, Pitch and Yaw which describes how much the rotation around the X,Y and Z axes respectively and constitutes the 3D rotational matrix, this is in addition to a vector of motion indicated through the estimated translation matrix.

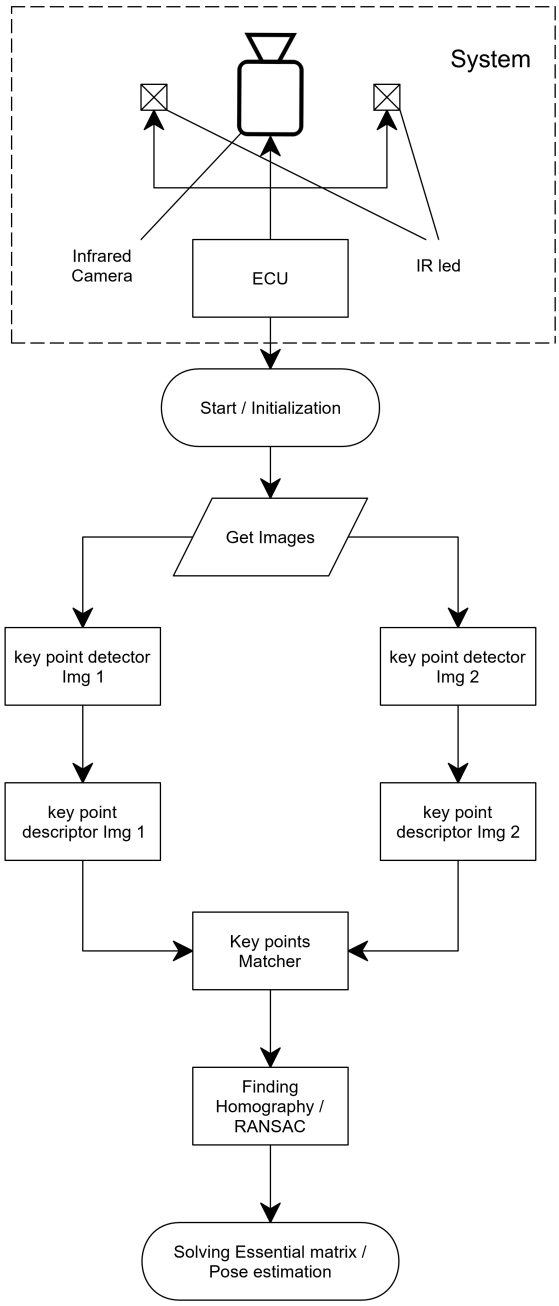

Fig. 1. Block Diagram showing the steps of our Pose estimation algorithm and the construction of the system used

B. Initialization

For the initial use of our camera we firstly have to define the intrinsic parameters, this is done using Zhang's well know checkerboard calibration method [13], This method will get us the 2 main parameters the focal length and the principal point. Other camera parameter could be obtained like camera distortion matrix but they are not mandatory. If the focal length and principal point of the camera are given, then this step is not necessary; however, it's still recommended to be done in the initial setup.
The calibration matrix will only be computed at the first time use only and then stored in memory, For every time we use

the application this matrix shall be loaded again and used for the computation of pose estimation.

C. Algorithm Steps and Approach

The main algorithm steps as shown in figure 1 starts by having this calibrated camera, Processing the frames is done by capturing the frames and comparing one to another according to that shown in the block diagram, Our captured image will be compared either to the previously computed image or a fixed image in case of constraint movement environment like in our implementation as we described before, a fixed camera with a pre-defined range of motion. We will have the steps as following:

1) Start by Image acquisition.
2) Convert image from RGB to Grey scale.
3) Detect Key points and calculate their description.
4) Matching the descriptions of the scene's features.
5) Computing Homography and excluding outliers using RANSAC.
6) Decomposition of the essential matrix for Pose Estimation.

Our aim is to optimize the computational cost, By analyzing all of these steps to get to the bottleneck of the algorithm and show good optimized performance which was found to be mainly in the key point detection and description. For that we will focus on two algorithms that are well know for the speed of computation; AKAZE and ORB with AKAZE provides good compromise between speed and performance in terms of correct matches compared to SIFT while ORB provided the best computational speed but not as accurate results as AKAZE, accordingly will be used if it's sufficient for the application's intended usage and will perform correctly or not.

D. AKAZE

While KAZE is based on solving non-linear diffusion filtering which in terms requires solving a series of PDEs, This PDEs was known to have an issue as solving such is a computationally expensive task, The AKAZE (Accelerated KAZE) came to solve this issue by using a faster method to create the non-linear scale-space called the Fast Explicit Diffusion (FED). These FEDs are used to compute a nonlinear Scale Space using a fine to coarse pyramidal framework. This leads to fast nonlinear scale space construction which constitutes the main difference from KAZE, This leads to a speed up the overall operation of the algorithm. With this nonlinear scale space filtered image, The determinant of Hessian is computed for each then, this will be followed by concatenated Scharr filters to find Maximas on two levels and discard the non-maxima responses. More details for the full algorithm could be found in "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces"[12].

For key point description, AKAZE is using binary descriptor (same as BRIEF and ORB) which is also optimized for speed of operation yet still very efficient. While there are many approaches for the binary descriptor for ordering the pixels of a patch using Hamming distance, locality sensitive hashing (LSH) or other algorithm. AKAZE used a Modified-Local Difference Binary (M-LDB) based on their previously computed nonlinear scale space. These can get the gradient and intensity information of the images, reducing the number of operations needed for computing the key point description. Another gain is through using various grids of smaller patches so that the operations over these subdivisions works faster.

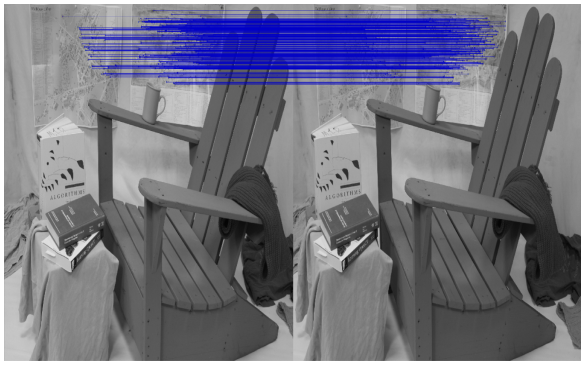

Fig. 2. Figure Showing Matching features using AKAZE

Although binary descriptors suffers mainly from rotation variance, this has been solved by estimating the orientation of the key points and then the grid of the M-LDB is rotated accordingly, this new approach solves the problem of rotation variance found in binary descriptors like BRIEF.

E. ORB

Oriented Fast and Rotated Brief (ORB)[8] algorithm was developed to provide a better performance than the well known state of the art SIFT and SURF yet being a free form licensing. Despite this fact, It still provides better computational cost and almost matching them in performance. As its name implies, ORB is based on FAST as a key point detector and BRIEF as a key point descriptor which is very strong in having a high discriminative feature descriptions. Since both FAST and BRIEF are rotation variant, ORB came with modifications to the original algorithms so it uses oriented FAST detector and rotation-aware BRIEF as a descriptor so ORB became rotation invariant in both key point detector and descriptor while still having it's main feature as being very fast and computational efficient and it's key points description retains being highly discriminative. It also provides good performance in some applications that it can be used on low-power devices since both algorithms are computationally fast.

*1) **oFAST detector**:* ORB algorithm starts initially with using the oFAST detector to detect key points, It's based on FAST method[14] where it takes the intensity weight between the center pixel and the circular patch around, then Harris corner measure is applied to order the the top key points detected, so in order to get N key points, the intensity threshold must be adjusted to evaluate more then the N required points

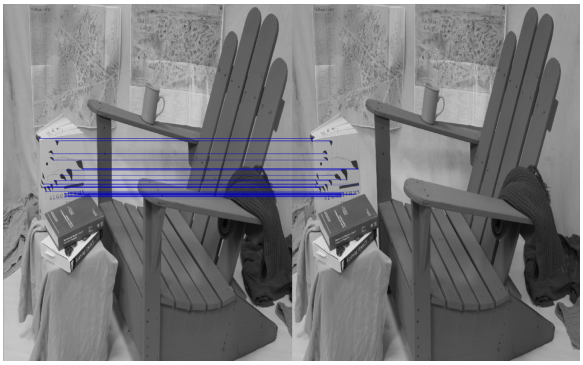

Fig. 3. Figure Showing Matching features using ORB

TABLE I
COMPUTATIONAL COST FOR EACH PART OF THE SYSTEM IN SECONDS AND SHOWING THE CONTRIBUTION FOR EVERY STEP TO THE TOTAL TIME CONSUMED BY THE WHOLE APPLICATION FOR BOTH AKAZE AND ORB IN PERCENTAGE

Aspect	AKAZE	ORB	AKAZE %	ORB %
Grey scale transformation	0.0007	0.0006	0.8%	1.8%
Feature Detection / Description	0.076	0.011	83.7%	32.3%
Feature Matching	0.00027	0.002	0.3%	5.9%
Homography RANSAC	0.00224	0.0064	2.5%	18.8%
Pose Estimation	0.0116	0.014	12.7%	41.2%

and later on ranked according to the Harris measure.
As a modification ORB also use a scale pyramid to produce a multi-scale feature as FAST features will be produced at the levels of that pyramid.

For Fast lacks the rotation invariance, oFast provides a solution by using corner orientation, It computes the centroid of the patch according to intensity and according to this centroid, the orientation of the corner is towards the centroid, Also for further improvement for rotation invariance, Moments are computed for x and y in a circular region with the size of the patch.
According to them, This approach provided a consistent results in terms of orientation detection regardless of noise level better than what was found in SIFT or SURF.

*2) **rBRIEF descriptor**:* While the original BRIEF descriptor [9] which is another binary descriptor like the one used in AKAZE, It is a bit string description that is constructed from a set of binary intensity tests on pixel level of image patch after smoothing using an integral image, It needs a good set of key points to train a set of classification trees using binary test, then this trained tress will be used to return a unique description for any keypoint. This descriptor performs well in many aspects compared to SIFT or SURF under different lighting conditions, distortion and noise immunity, However it's main problem was in rotation variance, for BRIEF as a descriptor was not able to show any good performance in in-plane rotated images, In a modified version called steered BRIEF they provided a solution for this limitation then later on ORB introduced another method which is called rotation-aware BRIEF (rBRIEF).
While Steered BRIEF came with a solution to the problem of invariance to in-plane rotation. so Steered BRIEF is to steer the BRIEF descriptor according to the orientation of the key points by detection the orientation of any given patch and then the correct set of points will be chosen to compute it's descriptor accordingly instead of having a random orientation in case of the original BRIEF.
This modification introduced another issue which is that sBRIEF's descriptors became less discriminative due to the face that the variance is decreased between features as we lost the randomness in BRIEF orientation as having an uncorrelated tests is an important aspect for such kind of descriptors.

rBRIEF presents a solution for the loss of variance problem introduced by steered BRIEF, to do so they implemented the Principal Component Analysis (PCA) along with a learning method and using large binary tests the algorithm will return highest variance features that are more likely to be uncorrelated features.

When using ORB in OpenCV we have some optional parameters to configure according to the performance needed:

1) nFeatures: Number that denotes maximum number of features to retain.
2) scaleFactor: Pyramid decimation ratio.
3) nlevels: The number of pyramid levels.
4) edgeThreshold: The size of the border where the features are not detected.(should match the patch size)
5) WTA_K: The number of points to produce each element of rBRIEF descriptor. For value more than 2, NORM_HAMMING2 will be used to define the matching distance otherwise, NORM_HAMMING distance is used.
6) scoreType: To choose either Harris score or FAST score to rank the features.
7) patchSize: Size of the patch used by the descriptor.

III. SIMULATION RESULTS

In this section we evaluate our system in 2 aspects. The first is what is the most computationally intensive tasks in our system, and the second quality generated after optimizing this task compared to other algorithms.
Firstly we consider which task of the whole algorithm contributes more to the efficiency of computation as depicted in table I, It has been shown that all the computationally exhaustive tasks are in feature extraction and description which the time consumed in this step might reach 4x more than the time taken in pose estimation in case of AKAZE while in other algorithms like SIFT and SURF this factor increases dramatically, also more than the time required for matching of the features computing the Homography and RANSAC.

TABLE II
AKAZE VS ORB COMPUTATIONAL PERFORMANCE WHILE USING INTEL VTUNE AMPLIFIER FOR PROCESSOR PERFORMANCE PROFILER

Aspect	AKAZE	ORB
CPU Time	0.94s	0.44s
Task Time	2.93s	0.531s
CPU Utilization on task	32%	83%
Best achieved runtime value	0.07	0.01
Frame rate	11 fps	28 fps

Further more by analyzing the CPU utilization by using INTEL VTune performance profiler that analyzes how the task works on the CPU as shown in table II, It illustrates that while a single AKAZE task for a single image takes almost 2.93 to finish, the actual CPU usage on task was only 32% of that time which shows that the algorithm is not as computationally efficient yet as ORB which gives 83%, according to the profiler, it indicates that AKAZE rely more on memory transference which leads to having lower CPU utilization time during task compared to ORB, needless to say that this test must be done in debug mode so that the timing might differ from the values provided in table I. It's also shown that the best achieved timing for both ORB and AKAZE during release mode using both of them is 0.01 and 0.07 seconds and the maximum frame rate that our application can reach is around 28 and 11 fps respectively.

In terms of quality, while all of our algorithms shows enough number of matched key points for our application to work, yet we need to further show how both of AKAZE and ORB compared to each other and compared to SURF and SIFT. In this part we use he Middlebury dataset[15] as a ground truth to compare all the algorithms. In table III we show the number of key points extracted for both images and the descriptor size used by each algorithm to describe each key point, the good matches returned after filtration using RANSAC and the matching rate indicating the quality of each algorithm to match the correct key points together. In our test while AKAZE is automatically computing the number of key points detected, ORB, SURF and SIFT gives us a way to control this part, as illustrated in the table while ORB using 500 key point (which is the least value used to train the descriptor) gives 18.8% , this percentage falls dramatically while increasing the number of key points which shows no much gain in the good matches with respect to increasing the number of key points, In case of SURF, regardless of the much more computation cost it still provides the most number of good matched but still the quality of detection is not superior as still it's 20.9% accurate out of the total number of key points detected, also when reducing the key points detected to match those of ORB, it's still slower and also provided worse quality and lower number of good matches only 66 with 12.5% of accuracy, lastly SIFT which is the same as SURF, provides higher accuracy of them all in the normal scenario while when selecting only the best 500 key points to be used, it provided lower accuracy than both ORB and AKAZE as well.

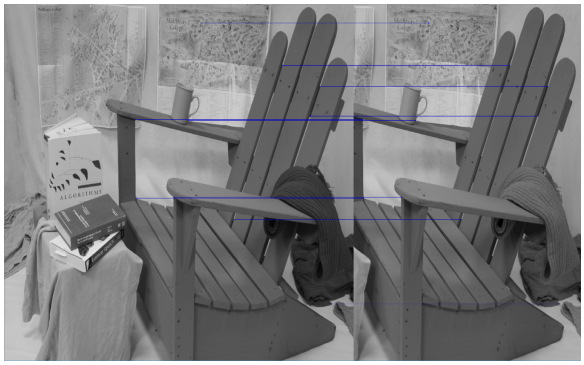

Fig. 4. Figure Showing the features detected using ORB while having cropped part of the picture

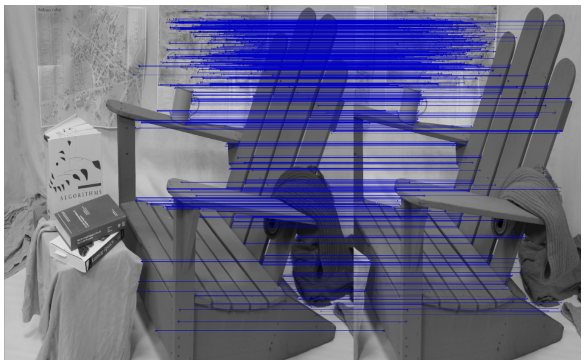

Fig. 5. Figure Showing the features detected using AKAZE while having cropped part of the picture

TABLE III
RESULTS OF ORB, AKAZE, SURF AND SIFT KEY POINTS DETECTION AND DESCRIPTION TO SHOW THE QUALITY OF DETECTION COMPARED VERSUS EACH OTHER

Aspect	AKAZE	ORB	SURF	SIFT
Keypoints image 1	2874	500-2000	11517-602	4132-500
Keypoints image 2	2600	500-2000	11162-527	3897-500
Descriptor size	61	64	64	128
Good Matches	476	94-134	2337-66	882-77
Match rate	18.3%	18.8%-6.7%	20.9%-12.5%	22.6%-15.4%

On other hand, while comparing both AKAZE and ORB extensively in regards of rotation, scaling, cropping and different illumination. It is shown that AKAZE provides more robust results in all the aspects, yet ORB did what the application needs, to provide more than 8 correct points for the worst case scenario in order to do any kind of essential matrix decomposition and to get the new camera pose. While ORB provided

TABLE IV
PERFORMANCE COMPARISON FOR AKAZE AND ORB USING DIFFERENT RESOLUTIONS AND IMPOSING ROTATION, SCALING, ILLUMINATION AND CROPPING VARIANCES

Aspect	Kpnts1	Kpnts2	Match%
ORB Rotation	500	500	307
AKAZE Rotation	2874	2796	2033
ORB Scaling 50%	500	500	66
AKAZE Scaling 50%	2874	1177	821
ORB Cropping	500	500	23
AKAZE Cropping	2874	1244	955
ORB Illumination	500	500	23
AKAZE Illumination	2874	2381	128

good results in the rotation part, it falls dramatically in the scaling, cropping and under different illumination conditions, In case of scaling, as shown in figure 3 most of the correct key points were located in the part that contains higher variation in grey levels, this is not the case for AKAZE as illustrated in figure 2 as most of the correct key points are located in the part with smaller details. While cropping these parts; as shown in figure 4, ORB failed to get more key points but kept relying on the large variations of grey levels, but in case of AKAZE; shown in figure 5, the algorithm succeeded to find other key points other than those in the cropped area which provided even more key points than those found in the original image.

IV. CONCLUSION

In this paper we have showed that for a computationally efficient approach for Camera Pose Estimation, The most computational intensive task is in the part of Key Point detection and description, It has been shown that the usage of ORB despite lower quality performance, It's still better and more suitable for a real-time application for camera pose-estimation than many other alternatives for it's simplicity and speed yet still has sufficient performance. This implementation using ORB is proven to reach 28 fps for a 640*480 frame picture with a Knn Matcher and RANSAC to filter out the outliers for correctly decomposing the essential matrix to get the new camera extrinsic parameters.

We also showed that both AKAZE and ORB came out of modifications done to other existing algorithms that suffered from higher computational cost and rotational variance. The usage of either of them ORB or AKAZE is much computational efficient than the well know SIFT and SURF and serving the free form licensing restrictions. Due to ORB's fast computation as we showed here, it can be used for applications on low-power and mobile devices for real-time applications as it provides sufficient performance depending on the application. While for robust performance AKAZE would be better but need to be optimized better to utilize the CPU load to give better computational performance.

REFERENCES

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2 ed., 2004.

[2] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[3] H. Li and R. Hartley, "Five-point motion estimation made easy," *18th International Conference on Pattern Recognition (ICPR'06)*, 2006.

[4] J. Hedborg and M. Felsberg, "Fast iterative five point relative pose estimation," *2013 IEEE Workshop on Robot Vision (WORV)*, 2013.

[5] M. J. D. Powell, "A hybrid method for nonlinear equations," in *Numerical Methods for Nonlinear Algebraic Equations* (P. Rabinowitz, ed.), Gordon and Breach, 1970.

[6] D. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," *2011 International Conference on Computer Vision*, 2011.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision ECCV 2010*, pp. 778–792, 2010.

[10] E. Karami, S. Prasad, and M. Shehata, "Image matching using sift, surf, brief and orb: Performance comparison for distorted images," *CoRR*, vol. abs/1710.02726, 2017.

[11] O. Andersson and S. Reyna Marquez, "A comparison of object detection algorithms using unmanipulated testing images : Comparing sift, kaze, akaze and orb," 2016.

[12] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *Procedings of the British Machine Vision Conference 2013*, 2013.

[13] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.

[14] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision ECCV 2006*, pp. 430–443, 2006.

[15] D. Scharstein, H. Hirschmuller, Y. Kitajima, G. Krathwohl, N. Nesic, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," *Lecture Notes in Computer Science*, pp. 31–42, 2014.