Consistent Multi-robot Object Matching via QuickMatch

Zachary Serlin(✉), Brandon Sookraj, Calin Belta, and Roberto Tron

Boston University, Boston, MA 02446, USA
zserlin@bu.edu

Abstract. In this work, we present a novel solution and experimental verification for the multi-image object matching problem. We first review the QuickMatch algorithm for multi-image feature matching and then show how it applies to an object matching test. The presented experiment looks to match features across a large number of images and features more often and accurately than standard techniques. This experiment demonstrates the advantages of rapid multi-image matching, not only for improving system performance, but also for use in new applications, such as object discovery and localization.

Keywords: Computer vision · Feature matching · Object matching

1 Motivation

In this paper, we propose a solution to the following problem: given a set of images taken from a team of robots (or camera network), match unique object features, as they enter and exit the images from multiple perspectives. This problem is fundamental to both computer vision and robotics applications, where feature matching can be used in object detection, localization, and tracking [2,21], homography estimation [15], structure from motion [6], and formation control [10]. Solutions to this problem are traditionally computationally complex, and often mismatch features when considering more than two images [2,8]. Multi-image correspondences allow for greater match reliability, and a more accurate representation of objects in the universe. The proposed solution leverages a relatively recent algorithm, QuickMatch [17], to quickly and reliably discover correspondences across multiple images. The experiments presented in this paper benchmark QuickMatch's performance by implementing an object matching framework under realistic conditions (i.e. images with clutter, repeated structures, and poor image quality); a target object is matched across a network of cameras, and then these matches are used to generate the target's trajectory.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-33950-0_64) contains supplementary material, which is available to authorized users.

J. Xiao et al. (Eds.): ISER 2018, SPAR 11, pp. 751–761, 2020.
https://doi.org/10.1007/978-3-030-33950-0_64

2 Problem Statement

Given a set of images $\mathcal{I} = \{1, 2, \ldots, i, \ldots, N\}$ and a set of K_i feature vectors, x_{ik}, extracted from each image, determine matches $(x_{i_1 k_1} \leftrightarrow x_{i_2 k_2} : i_1 \neq i_2)$ between features from separate images, such that matched features represent the same point in the scene.

3 Related Work

Feature matching is a basic process in many computer vision algorithms. *Pairwise matching* is the classical approach to this task, where features between two images are compared based on a distance metric (e.g. Euclidean or Manhattan distance), and declared a match if this distance is below some threshold [8,18]. This method is used in two standard algorithms, *Brute Force* (BF) matching, and *Fast Library for Approximate Nearest Neighbors* (FLANN) matching. Pairwise matching has difficulties matching entities with repetitive structure or similar appearance (e.g. windows) because the distance metric alone does not consider the *distinctiveness* (smallest distance between features from the same image) of the features. Including distinctiveness of features during matching has been shown to be beneficial [8]. For multi-image matching, pairwise matches scale poorly with the number of images and across multiple images, match correspondences often do not belong to the same ground truth object. *Graph matching* has also been used for pairwise matching. This approach attempts to match vertices (features) and edges (matches) simultaneously to determine better pairwise matches [20], but it cannot handle the multi-image setting.

Beyond pairwise matching, a number of other approaches exist for multi-image matching (where multiple images are directly considered) that are based on optimization, graphs, and clustering. *Optimization* based approaches are based upon non-convex problems where optimization constraints must often be relaxed to reliably obtain solutions [12,20]. Moreover, these approaches require to know *a priori* the number of objects, which is often not available, and do not consider distinctiveness of the features. *Cycles in graphs* are early predecessors to the QuickMatch algorithm and have largely been used to remove inconsistent matches [7]. *Clustering* can be cast as finding clusters of similar features. Algorithms such as k-means [9] and spectral clustering [11] have been explored to this end, but also often require a predefined number of objects, and do not consider that a unique feature only occurs once in an image.

QuickMatch uses *density-based clustering* algorithms [4,19], which find clusters by estimating a non-parametric density distribution of data [13,14]. These approaches do not require prior knowledge of the number or shape of clusters, and can be modified to include feature distinctiveness by construction. This paper is an experimental extension of [17], where QuickMatch was initially introduced.

4 Contribution

The primary contribution of this paper is the testing and experimental validation of the QuickMatch algorithm under more realistic conditions, as opposed to previous evaluations using standard datasets. This experiment tests the algorithm for computational efficiency and match accuracy by employing a distributed camera network to localize a moving target.

5 Technical Approach

A two-stage, offline, centralized solution is implemented on a system of distributed ground robots and a central computer. Features are first extracted using off-the-shelf feature extraction methods (SIFT), and the features are then matched using the QuickMatch algorithm to find a given reference object. These matches are used to perform homography estimation between the reference object and the camera network to generate target trajectories.

5.1 Feature Extraction

Feature extraction aims to find and describe representative points from high dimensional data, such as an image [1,5,8,21]. Features themselves are also high dimensional vectors. In this experiment, the *scale invariant feature transform* (SIFT) feature is used, which extracts K_i 128-dimensional vectors that represent the appearance of each feature point. See [8,18] for more details on this standard feature extraction algorithm. Other feature types can be used and we also tested with Oriented FAST and Rotated BRIEF (ORB) features and Speeded-Up Robust Features (SURF), however SIFT was the most reliable.

5.2 QuickMatch

The QuickMatch algorithm is a density based clustering algorithm. It begins by calculating the distance between all features (we use Euclidean distance in our application). For each image, the minimum distance, σ_i, between any two features is used as the distinctiveness of features for that image. Recall, from above, x_{ik} is a point in the high dimensional feature space. The feature density $D(x_{ik})$ is then calculated for each point using the formula

$$D(x) = \sum_{i=1}^{N} \sum_{k=1}^{K_i} h(x, x_{ik}; \sigma_i), \tag{1}$$

$$h(x_1, x_2; \sigma) = \exp(-\frac{\|x_1 - x_2\|}{2\sigma^2}), \tag{2}$$

with kernel function h, and distinctiveness σ_i. With this feature density, the features are organized into a tree structure, with parent nodes being the nearest neighbor with a higher density.

$$\text{parent}(x_{ik}) = \underset{i'k' \in J}{\arg\min}\, d(x_{ik}, x_{i'k'}), \tag{3}$$

$$J = \{i'k' : k \neq k', D(x_{i'k'}) > D(x_{ik})\}. \tag{4}$$

Edges are directed to parents along the gradient of feature density, and ultimately toward the center of the parent cluster or to another distant cluster. Once the tree has been constructed, edges are broken if either of two criteria are met; (1) parent and child groups have nodes from the same image, or (2) the edge is larger than a user defined threshold (ρ) times σ_i. This method results in a forest of trees, where each tree is a cluster representing a unique entity in the universe. In practice, each tree represents a point that is common among images, meaning the algorithm discovers common features among very similar objects. Feature discovery will be explored further in Sect. 7.3, where groups of matching points are organized into object detections and homography transformations.

5.3 Homography and Localization

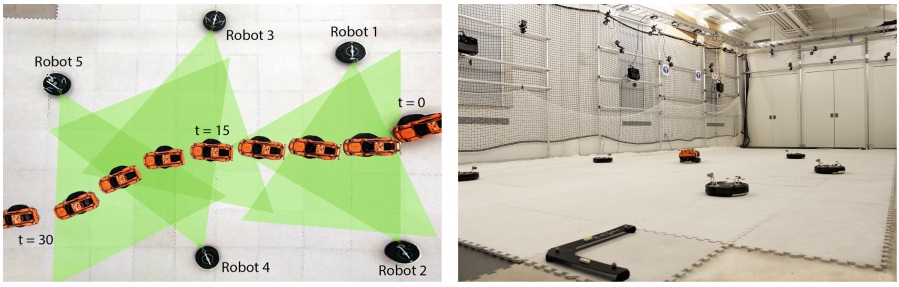
An homography is a perspective transformation between the view points of two images that can also be used to determine the relative position of an object given a reference image. Given a reference viewpoint $\widetilde{x}$, a new viewpoint can be found given the homography matrix $\overline{H}$ as $\overline{H}\widetilde{x}$. The $\overline{H}$ matrix can be estimated with a set of known relative points (or matched features) between the two images. Once $\overline{H}$ is estimated, it is possible to compare the position of objects in each image in a relative coordinate system. To improve the estimate of $\overline{H}$, random sample consensus (RANSAC) is used to remove match outliers by randomly sampling the matches, finding a fit of the data, and then removing any matches that fall outside of a user defined region [15].

Using the homography transformations between each image and a target reference image, the object can be localized up to a distance scale factor, as shown in Fig. 1(a). Given a known parameter of the target object, in this case the object's height, this ambiguity can be resolved, and the relative position can be determined. When taken together with other cameras in the network and a known global camera pose, the target object can be accurately positioned in the global reference frame, allowing for generating a target's trajectory (e.g., Fig. 4).

Homography and localization are limited by the reference images used for matching, and are prone to noisy and inaccurate measurements. Firstly, the system can only identify the known side of the object, unless the target is symmetric. To overcome this, multiple reference target images are used here. Secondly, inaccurate measurements in distance and bearing are common. These inaccuracies arise from extreme sensitivity to object height estimate errors when calculating target distance. To account for these errors in practice, multiple measurements can be used to estimate each position, and then a filter can be used to smooth the target's trajectory.

6 Experiment

The experiment consists of a team of five iRobot Create2 ground robots, each with a forward facing camera, distributed throughout the experimental area shown in Fig. 1. Each camera has a $62° \times 48°$ field of view, and takes a 640×480 px image at 2 Hz. Through the center of the area, the target object is driven along the trajectory shown in Fig. 1(a) over approximately thirty seconds. All cameras are triggered simultaneously and the images are sent to a central computer for feature extraction and matching. The central computer has an Intel i7-7800x 3.5 GHz processor, and runs Ubuntu 16.04 LTS and ROS Kinetic. Features are extracted using SIFT with an octave layer of 6, a contrast threshold of 0.10, an edge threshold of 15, and sigma of 1.0. The matches from QuickMatch (using $\rho = 1.1$) are used to determine which cameras observe the target object at each time step, based on the number of matches with a target image (in this experiment 10 matches are required). The matches between each reference images and the current images are used to determine the homography between them, using RANSAC with a threshold of 10.0. The homography is used to generate a bounding box around the target object using a perspective transformation on the target image corners. The relationship between pixel height of this box and distance from the camera is calibrated beforehand using an object of known size (in this experience a checkerboard pattern of know dimension). The localization points are recorded to build a target trajectory, which is then compared to ground truth measurements from an OptiTrack© motion capture system (Fig. 1(b)).



(a) (b)

Fig. 1. (a) Overhead view of experimental area with trajectory of the target object, position of the robots, and the approximate field of view for the camera network (shown in yellow). (b) Prospective view of experimental area with modified iRobot Create2 platform, target object, and overhead OptiTrack© motion capture system.

7 Results

QuickMatch is evaluated in two ways: pure matching performance, and in the context of a target localization application. The QuickMatch algorithm is first

compared to standard matching algorithms in the OpenCV Software Package [2], Brute Force (BF), and FLANN. Both algorithms use the Euclidean distance metric and a threshold match distance of 0.75 [2,8]. Unlike QuickMatch, both algorithms cannot consider matches across more than two images but do have very low execution times.

QuickMatch is implemented in Python and takes 5.6 s to find matches between 6254 SIFT features (from 115 images), while BF and FLANN are both implemented in C++, and both take approximately 0.05 s to find the matches between the reference image features, and the same 6254 features. This time difference arises from two factors: the inherently slower runtime of Python compared to C++ [3], and the extra comparisons done by QuickMatch to solve the entire Multi-match problem. If BF and FLANN compared all images with all other images combinatorially (as QuickMatch implicitly does) their computation times would be ∼5.75 s, which is comparable to QuickMatch's slower Python implementation. This time also does not account for the post processing time necessary to reconcile inconsistent matches from both BF and FLANN, is not required in QuickMatch.

7.1 Precision Versus Recall

Although QuickMatch is slower, it outperforms both BF and FLANN in the number of matches correctly found, and generally in terms of precision vs. recall (PR) and precision-recall area under the curve (PR AUC), which are common metrics for evaluating matching algorithms [16]. Figure 2(a) shows the precision (fraction of correctly matched images) versus recall (fraction of possible matches

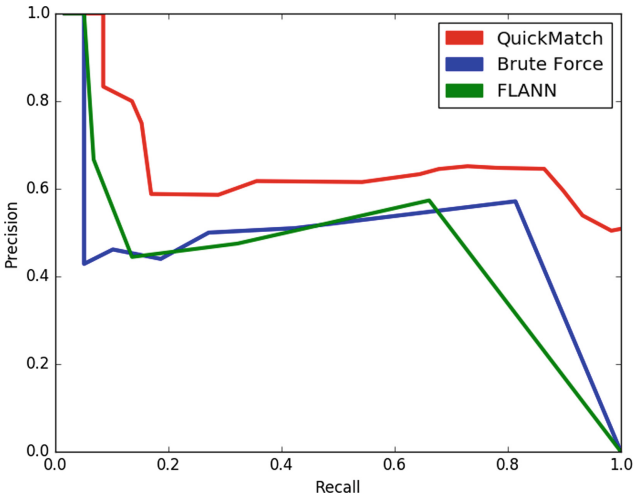
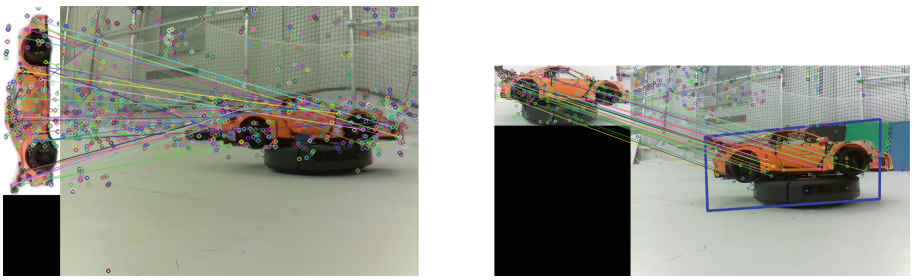


Fig. 2. (a) Precision vs. recall curves for the QuickMatch, Brute Force, and FLANN algorithms. All algorithms are run on the same feature vectors. A match is considered to exist if the number of matched features is above a threshold.

found) curves for QuickMatch, BF, and FLANN. For any recall level, QuickMatch maintains a higher precision level than either BF or FLANN. These curves are non-monotonic because mismatched features appear at a higher rate than correctly matched features at higher thresholds. PR AUC is a threshold agnostic metric used for comparing overall performance of matching algorithms [16]. In terms of PR AUC, QuickMatch achieves 0.64, while BF and FLANN reach 0.49 and 0.45 respectively. The overall increase in precision stems for QuickMatch's ability to consider more instances of the reference object, by matching cycles of features across multiple images. It is therefore able to find the reference object not only more consistently, but with many more matched features. An example of these matches is shown in Fig. 3.



(a) (b)

Fig. 3. (a) Example image matches between the reference object image (left) and an experimental image (right): Circles represent features, and lines indicate matches. (b) Homography and localization of car with prospective transform of bounding box.

7.2 Homography and Localization

In order to further demonstrate the utility of the QuickMatch algorithm, matches were used to localize a target object in relation to the camera network, and then estimate its global trajectory. This was done using all three above algorithms with again an identical set of SIFT features. QuickMatch considers multi-image matches between the set of target images and the set of five robot images at each time step, while BF and FLANN consider matches between each target image and the robot image individually. Once feature matches are generated, RANSAC is used to estimate the homography matrix $\overline{H}$ for each pair of images while also removing outliers from the matches. The homography between the reference image and each robot image is used to generate a bounding box around the target in the robot image as shown in Fig. 3(b). This bounding box, given a known camera calibration, provides bearing and height information for the target. The target height is known and is used to find the relate distance to the target with the bounding box height. With these two values, a distance and a bearing, the object can be localized with respect to each robot.

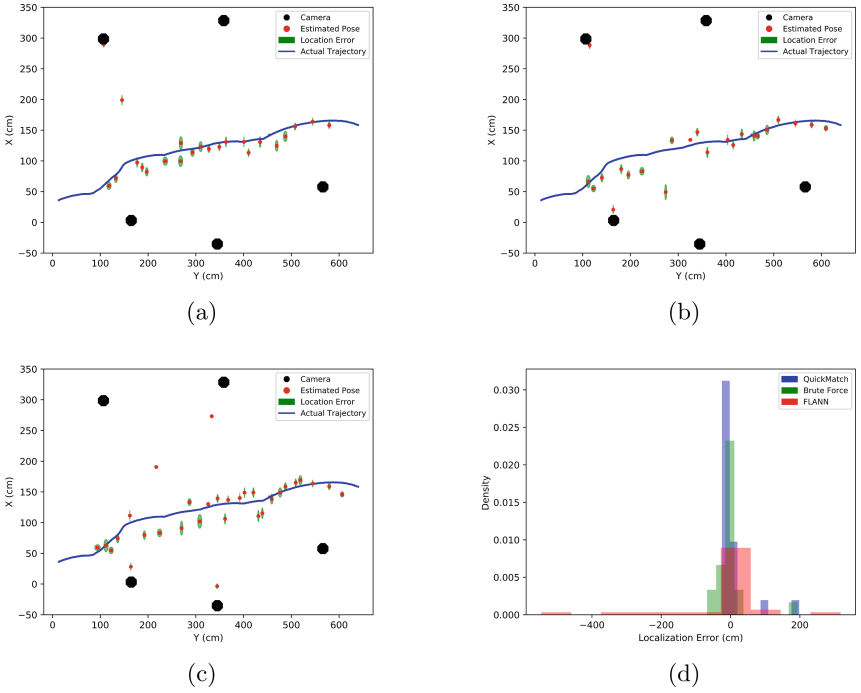


Fig. 4. (a) QuickMatch trajectory estimate. (b) BruteForce trajectory estimate. (c) FLANN trajectory estimate. (d) Histogram of estimate error for each algorithm.

The above steps are performed using the match data from each of the three above algorithms. Figures 4(a–c) show the results of the localization estimation for each algorithm. Red points are estimate target poses for each time step, blue points denote the ground truth measurements, black octagons are the camera network positions, and the green regions are the one standard deviation error between all localization estimates at each time step. The localization error was found by taking the absolute distance between the estimated and ground truth position at each time step. QuickMatch had an error of 0.2118 ± 0.4254 m, BF had an error of 0.2349 ± 0.4027 m, and FLANN had an error of 0.6232 ± 1.1722 m. QuickMatch outperforms both BF and FLANN in terms of accuracy, which is indicative of its higher match quality. BF matcher also performs well and maintains a low variance, however it is not as accurate. FLANN is the worst performing of the three, and has a number of extremely erroneous estimates. Generally, monocular camera distance measurements are very sensitive to match errors, meaning target localization error is an indirect method for testing the overall accuracy of each method. Figure 4(d) shows a histogram of the localization error, which is found by comparing the localization estimate to the ground truth pose at each time step. The histogram makes it clear that QuickMatch maintains a higher number of accurate matches and has a small number of highly erroneous estimates. In practical applications, a Kalman

filter would be employed to smooth the estimates, but the values are left unaltered here to demonstrate the algorithm's output.

7.3 Feature Discovery

The QuickMatch algorithm implicitly discovers common features among images by creating clusters of similar features. These clusters correspond to specific locations in the universe, and therefore can be used to find both targets and landmarks across images. Landmarks, although not used in this paper, are points that occur commonly across all images (except when occluded), and are useful for multi-agent localization tasks. In the images collected, landmarks were the clusters with the largest number of features, because many of this images did not contain the target object. An example landmark cluster is shown in Fig. 5(a). Features belonging to the target object are generally smaller than the landmark clusters, but can still be extracted, and show key features of the target. Figure 5(b) shows one such cluster, which is the front hood of the car model. Feature discovery is one attribute of QuickMatch that does not exist in either BF or FLANN and can be useful for discerning what features are most descriptive of images from the network.

(a)

(b)

Fig. 5. (a) Landmark feature cluster. (b) Target feature cluster.

8 Conclusion

This experiment highlights the utility of QuickMatch multi-image matching for object matching. QuickMatch is able to find many more object feature matches than standard methods by considering matches across all images, not just pairwise matches. The presented experiment tests the QuickMatch algorithm in an experimental setting with realistic conditions, and shows that multi-image matching is superior to standard methods at matching the reference object (even as it enters and exits images across the entire camera network). Quickmatch is also tested with a target object localization and again outperforms both the BF and

FLANN algorithms. Beyond testing Quickmatch, this experiment also demonstrates its feature discovery ability by showing a characteristic landmark and target feature cluster from the test images. This approach is the precursor to an online, distributed, and decentralized approach. Our future work will focus on a distributed version of object discovery and localization and multi-camera homography. We also plan to use these extracted trajectories for higher level tasks. Overall, QuickMatch is shown to be a versatile multi-feature matching algorithm that outperforms standard pairwise matching algorithms.

Acknowledgements. This work was supported by the National Science Foundation under grants NRI-1734454, and IIS-1717656.

References

1. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speededup robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
2. Bradski, G.: The openCV library. Dobb's J. Softw. Tools **25**, 120–125 (2000)
3. Fourment, M., Gillings, M.: A comparison of common programming languages used in bioinformatics. BMC Bioinform. **9**, 82 (2008)
4. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans. Inf. Theory **21**(1), 32–40 (1975)
5. Hariharan, B., Arbelaez, P., Girshick, R., Malik, J.: Hyper-columns for object segmentation and fine-grained localization. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
6. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004)
7. Huang, Q., Guibas, L.: Consistent shape maps via semidefinite programming. Comput. Graph. Forum **32**(5), 177–186 (2013)
8. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
9. MacKay, D.J.: Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge (2003)
10. Montijano, E., Cristofalo, E., Zhou, D., Schwager, M., Sagues, C.: Vision-based distributed formation control without an external positioning system. IEEE Trans. Robot. **32**(2), 339–351 (2016)
11. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. Neural Inf. Process. Syst. **2**, 849–856 (2002)
12. Oliveira, R., Costeira, J., Xavier, J.: Optimal point correspondence through the use of rank constraints. In: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1016–1021 (2005)
13. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**(3), 1065–1076 (1962)
14. Rosenblatt, M.: Remarks on some nonparametric estimates of a density function. Ann. Math. Stat. **27**(3), 832–837 (1956)
15. Szeliski, R.: Computer vision: algorithms and applications. Springer, Heidelberg (2010)
16. Ting, K.M.: Precision and recall. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning. Springer, Boston (2011)

17. Tron, R., Zhou, X., Esteves, C., Daniilidis, K.: Fast multi-image matching via density-based clustering. In: The IEEE International Conference on Computer Vision (2017)
18. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms (2008). http://www.vlfeat.org/
19. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: IEEE European Conference on Computer Vision, pp. 705–718. Springer (2008)
20. Yan, J., Cho, M., Zha, H., Yang, X., Chu, S.: Multi-graph matching via affinity optimization with graduated consistency regularization. IEEE Trans. Pattern Anal. Mach. Intell. **38**, 1228–1242 (2015)
21. Zhou, X., Zhu, M., Daniilidis, K.: Multi-image matching via fast alternating minimization. In: The IEEE International Conference on Computer Vision (2015)