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 case. 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 existing algorithms, 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, from multiple perspectives, as they enter and exit the images. This problem is fundamental to both computer vision and robotics applications, where feature matching can be used for object matching, localization, and tracking [2, 22], homography estimation [16], structure from motion [7], and formation control [11]. Solutions to this problem are classically computational complex and often mismatch features when considering more than two images [2, 9]. Multi-image correspondences allow for greater match reliability, and a more accurate representation of objects in the universe. The proposed solution leverages the novel QuickMatch algorithm [18], to quickly and reliably discover correspondences across many images. The presented experiment tests QuickMatch's utility by implementing an object matching framework in a realistic scenario (i.e. images with clutter, repeated structures, and poor image quality).

2 Problem Statement

Given a set of images $\mathcal{I} = \{1, 2, \ldots, i, \ldots, N\}$, and a set of K_i feature vectors extracted from each image, with each feature denoted x_{ik}, determine matches $(x_{i_1k_1} \leftrightarrow x_{i_2k_2} : i_1 \neq i_2)$ between features from separate images, such that matched features represent the same entity in the universe.

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 [9, 19]. Two standard algorithms, *Brute Force matching* (BF), and *Fast Library for Approximate Nearest Neighbors matching* (FLANN) both determine pairwise matches in this way. 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* of the features. Including distinctiveness of features has been explored in [9] to mitigate the effect of repeated structures. 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.

Beyond pairwise matching, a number of other approaches exist for feature matching that are based on optimization, graphs, and clustering. *Optimization* based approaches solve a global non-convex problem where optimization constraints must often be relaxed to reliably obtain solutions [13, 21]. These approaches require an a priori known number of objects, which is often unknown, and do not consider distinctiveness of the features. *Cycles in graphs* and *graph matching* are early predecessors to the QuickMatch algorithm and have largely been used to remove inconsistent matches [8]. *Clustering* approaches such as k-means [10] and spectral clustering [12] have been explored, but also often require a predefined number of objects and do not consider that a unique object only occurs once in an image.

QuickMatch builds primarily on *density-based clustering* algorithms [5, 20], which find clusters by estimating a non-parametric density distribution of data [14]. These approaches do not require prior knowledge of the number or shape of clusters, and can include feature distinctiveness by construction. This paper is an experimental extension of [18], where QuickMatch is initially introduced.

4 Technical Approach

A two part, offline and 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, on a centralized computer, using the QuickMatch algorithm to find a given reference object. This approach is a precursor to an online, distributed, and decentralized approach.

4.1 Feature Extraction

Feature extraction aims to extract representative points from high dimensional data, such as an image [1, 9, 22]. In this experiment, the *scale invariant feature transform* (SIFT) feature is used, which extracts K_i 128 dimensional vectors that

represent the change in pixel value over small patches of each image. This is a standard technique and more detail can be found in [9, 19].

4.2 QuickMatch

The QuickMatch algorithm is a density based clustering algorithm. It begins by calculating the Euclidean distance between all features. For each image, the minimum distance between any two features is used as the distinctiveness of features for that image σ_i. Recall, from above, x_{ik} is a point in a high dimensional feature space. The feature density $D(x_{ik})$ (see (1)) is then calculated for each point based on a kernel function h (see (2)), distinctiveness σ_i, and density contributions from all other features.

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

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

With this feature density, the features are organized into a tree structure, with parent nodes being the nearest neighbor with a higher density.

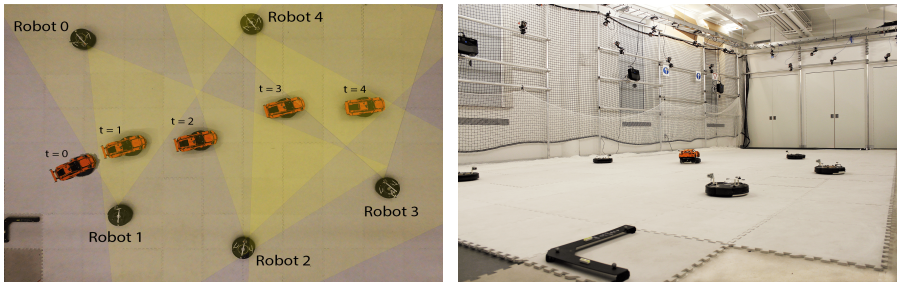
$$parent(x_{ik}) = \underset{i\prime k\prime \in J}{\arg\min}\, d(x_{ik}, x_{i\prime k\prime}), \tag{3}$$

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

This causes parent edges to be directed up the gradient of feature density, and ultimately toward the center of its parent cluster or to another distant cluster. Once the tree has been constructed, edges are broken if either of two criteria are true; 1) if the parent and child groups of nodes are in the same image, or 2) if 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 object in the universe. Objects can also be discovered from these trees.

5 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 pixel image every two seconds. Through the center of the area, the target object is driven along the path shown in Fig. 1a over ten 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 3.5GHz processor, and runs Ubuntu 16.04 LTS and ROS Kinetic. Features are extracted using the SIFT algorithm with an octave layer of 12, a contrast threshold of 0.1, an edge threshold of 3, and sigma of 1. The matches from QuickMatch (using $\rho = 1.5$) are used to determine which cameras see the target object at each time step, based on the number of matches with a target object image.



(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 motion capture system.

5.1 Planned Experiment Modifications

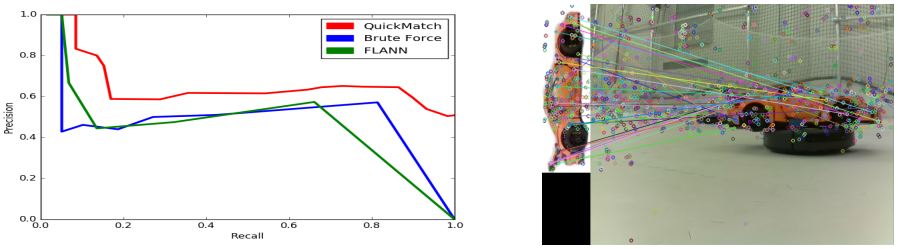
The current experiment takes a matching centered approach to the multi-image matching problem. Four extensions to this experiment are planned in the coming weeks. First, feature matches will be used to also localize the reference object in each image. With object localization, an approximate trajectory can be generated for the target object's motion and compared to ground truth measurements taken from the motion capture system shown in Fig. 1b. Second, the algorithms will be tested for individual feature match accuracy. Finally, time permitting, we plan to both implement QuickMatch in C++ to reduce its runtime and to extract features via hyper-column feature extraction to tailor features to specific applications [6].

6 Results

The QuickMatch algorithm is compared to the standard matching algorithms in the OpenCV Software Package [2], Brute Force (BF), and FLANN. Both algorithms use Euclidean distance and a threshold match distance of 0.75 [2, 9]. Unlike QuickMatch, both algorithms cannot consider matches across more than two sets of features but do have very low execution times.

QuickMatch is implemented in Python and takes 5.6 seconds 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 seconds 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++ [4], and the extra comparisons done by QuickMatch to solve the entire Multi-match problem. If BF and FLANN compared all images combinatorially (as QuickMatch implicitly does) their computation times would be $\sim 5.75s$ seconds, which is comparable to QuickMatch's slower Python implementation.

Although QuickMatch is slower, it out performs 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 [17]. Fig. 2a shows the precision (fraction of correctly matched images) versus recall (fraction of possible matches found) curves for QuickMatch, BF, and FLANN. Note 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 [17]. 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 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. 2b.



(a) (b)

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. (b) Example image matches between the reference object image (left) and an experimental image (right). Each circle is the location of a SIFT feature, and lines indicate a match between features.

7 Insights

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 tested the QuickMatch algorithm in a realistic setting, 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. Our future work focuses on using multi-image matching for online and distributed object discovery and localization, multi-agent localization and formation control, and online multi-camera homography.

References

1. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speededup robust features (SURF). Computer Vision and Image Understanding, 110(3):346359, 2008.
2. Bradski, G.: The OpenCV Library. Dr. Dobbs Journal of Software Tools, 2000.
3. Dollar P., Zitnick, C.L.: Structured forests for fast edge detection. In International Conference on Computer Vision, 2013.
4. Fourment, M., Gillings, M.: A comparison of common programming languages used in bioinformatics. BMC Bioinformatics, vol. 9, p. 82, Feb 2008.
5. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory, 21(1):3240, 1975.
6. 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.
7. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, second edition, 2004.
8. Huang, Q., Guibas, L.: Consistent shape maps via semidefinite programming. Computer Graphics Forum, 32(5):177-186, 2013.
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91110, 2004.
10. MacKay, D.J.: Information theory, inference and learning algorithms. Cambridge university press, 2003.
11. Montijano, E., Cristofalo, E., Zhou, D., Schwager, M., Sagues, C.: Vision-based Distributed Formation Control without an External Positioning System. IEEE Transactions on Robotics, vol. 32, no. 2, pp. 339-351, April 2016.
12. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. Neural Information Processing Systems, 2:849856, 2002.
13. Oliveira, R., Costeira, J., Xavier, J.: Optimal point correspondence through the use of rank constraints. In IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 10161021, 2005.
14. Parzen, E.: On estimation of a probability density function and mode. The annals of mathematical statistics, 33(3):1065 1076, 1962.
15. Rosenblatt, M.: Remarks on some nonparametric estimates of a density function. The Annals of Mathematical Statistics, 27(3):832837, 1956.
16. Szeliski, R.: Computer vision: algorithms and applications. Springer Science & Business Media, 2010.
17. Ting, K.M.: Precision and Recall. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA, 2011.
18. 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
19. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. http: //www.vlfeat.org/, 2008.
20. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In IEEE European Conference on Computer Vision, pages 705718. Springer, 2008.
21. Yan, J., Cho, M., Zha, H., Yang, X., Chu, S.: Multi-graph matching via affinity optimization with graduated consistency regularization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.
22. Zhou, X., Zhu, M., Daniilidis, K.: Multi-Image Matching via Fast Alternating Minimization. In: The IEEE International Conference on Computer Vision, 2015