Pattern Recognition ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fast matching of large point sets under occlusions

Julian J. McAuley [a,b,*], Tibério S. Caetano [a,b]

[a] Statistical Machine Learning Group, NICTA, Australia
[b] Research School of Computer Science, Australian National University, Australia

ARTICLE INFO

Article history:
Received 24 January 2011
Received in revised form
1 May 2011
Accepted 14 May 2011

Keywords:
Point-pattern matching
Graphical models
Global rigidity

ABSTRACT

The problem of isometric point-pattern matching can be modeled as inference in small tree-width graphical models whose embeddings in the plane are said to be 'globally rigid'. Although such graphical models lead to efficient and exact solutions, they cannot generally handle occlusions, as even a single missing point may 'break' the rigidity of the graph in question. In addition, such models can efficiently handle point sets of only moderate size. In this paper, we propose a new graphical model that is not only adapted to handle occlusions but is much faster than previous approaches for solving the isometric point-pattern matching problem. We can match point-patterns with thousands of points in a few seconds.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Identifying a correspondence between two sets of point features is a fundamental problem in pattern recognition, with many applications in computer vision, including object detection [1], image categorization [2], scene reconstruction [3], fingerprint recognition [4], and shape matching [5]. There are also several applications in other fields, such as molecular biology [6,7], computational chemistry [8,9], and astronomy [10].

A common setting of this problem is that of *isometric point-pattern matching*, where we assume that a 'copy' of a graph $\mathcal{G}$ (the 'template') appears isometrically transformed within $\mathcal{G}'$ (the 'target'). Although such a formulation allows for outliers in the target graph, it does not allow for outliers in the template graph, i.e., points that appear in $\mathcal{G}$ but not in $\mathcal{G}'$. In some scenarios, this is a realistic assumption, for instance if the user has manually specified a set of interest points in a template image. However, in many cases we are interested in identifying the *largest common isometric subgraph* between the two point sets, for instance where automatic interest point detectors have been used for a pair of images.

Amongst approaches concerned with isometric matching, there are further subtle differences in the formulations used. Fast algorithms exist for the case of *exact* isometric matching (i.e., no noise) [11,12], but achieving an optimal solution for even a small known amount of noise becomes a computationally difficult (albeit polynomial) problem [13]. Others seek solutions that are fast under certain conditions but may be slow or inaccurate in other cases (e.g. graphs such as grids) [4,14]. Here, we shall follow the line of work of [15], and later [16,17], which is concerned with algorithms that are *provably optimal and efficient* in the noise-free case, but which give empirically good performance in the case of noise.

Along these lines, we aim to make two contributions: firstly we specify a graphical model that allows us to solve the point-pattern matching problem more efficiently than the existing state-of-the-art. This is done by adapting previous results from [17] to use a different graph topology, which allows us to use efficient search algorithms. In practice, doing so reduces the running time from $O(|\mathcal{G}||\mathcal{G}'|^3)$ in [17] to $O(|\mathcal{G}||\mathcal{G}'|^2\log|\mathcal{G}'|)$. Furthermore, memory requirements are reduced from $O(|\mathcal{G}||\mathcal{G}'|^3)$ to $O(|\mathcal{G}'|\log|\mathcal{G}'|)$, which in practice allows the algorithm to be run on much larger point-patterns (e.g. thousands rather than dozens of points).

Our second contribution is to demonstrate that this model can easily be adapted to handle occlusions, simply by running it multiple times under different configurations. The increase in running time depends on the number of occlusions N_{occ} that we wish to handle. For $N_{\text{occ}} < \lfloor|\mathcal{G}|/2\rfloor$, we achieve a running time of $O(N_{\text{occ}}|\mathcal{G}||\mathcal{G}'|^2\log|\mathcal{G}'|)$, with no increase in memory requirements. We can handle *any* number of occlusions in $O(|\mathcal{G}|^3|\mathcal{G}'|^2\log|\mathcal{G}'|)$ time; in practice, this allows us to find the largest common isometric subgraph in graphs with several hundred nodes, requiring only a few seconds on a standard desktop machine.

2. Background

2.1. Point-pattern matching

The problem of *point-pattern matching* consists of finding an instance of a 'template' graph $\mathcal{G}$ within a 'target' scene $\mathcal{G}'$.[1] In this

* Corresponding author at: Research School of Computer Science, Australian National University, Australia.
E-mail address: julian.mcauley@gmail.com (J.J. McAuley).

[1] Strictly, we consider *embeddings* of graphs in the plane, i.e., the nodes of $\mathcal{G}$ and $\mathcal{G}'$ encode 2-D coordinates. Similarly, by 'globally rigid graph' we mean a

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.05.008

paper, we shall assume that this instance corresponds to an *isometric* transformation, meaning that $\mathcal{G}'$ can be formed by isometrically transforming $\mathcal{G}$, and possibly adding outlying points (and possibly applying some noise to the point coordinates).[2] More formally, we wish to find a total function $\hat{f} : \mathcal{G} \rightarrow \mathcal{G}'$ such that the distances between points in $\mathcal{G}$ are preserved by their image in $\mathcal{G}'$ under $\hat{f}$. That is, we wish to find

$$\hat{f} = \underset{f:\mathcal{G}\rightarrow\mathcal{G}'}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{G}} |d(g_i,g_j) - d(f(g_i),f(g_j))|, \tag{1}$$

where $d(\cdot,\cdot)$ is simply an Euclidean distance function. While this would appear to be an instance of a *quadratic assignment problem*, which is NP-hard in general [18], it has been shown in [15,16] that this problem can be solved in polynomial time by means of a *globally rigid* graphical model. In [16], a graphical model is presented which allows exact inference to be performed in $O(|\mathcal{G}||\mathcal{G}'|^3)$ memory and time; we will present this model and improve upon it in Section 3.

2.2. Largest common subgraph

The above problem can be adapted to allow for outliers in $\mathcal{G}$ *and* $\mathcal{G}'$, by allowing the function f to be a *partial function*. Of course, in the case of non-zero noise, the minimizing assignment of (Eq. (1)) then becomes the null-match (i.e., no points are mapped), so we must add some penalty κ for each point which is occluded in $\mathcal{G}$:

$$\hat{f} = \underset{f:\mathcal{G}\rightarrow\mathcal{G}'}{\arg\min} \sum_{(i,j)\in\mathcal{G}} |d(g_i,g_j) - d(f(g_i),f(g_j))| + \kappa \underbrace{(|\mathcal{G}| - |f(\mathcal{G})|)}_{\text{number of unmapped points}}. \tag{2}$$

Assuming that κ is equal to the maximum amount of noise we are likely to observe (so that points perturbed by noise always incur a cost less than κ), and that we always favor mappings over non-mappings in the case of equal costs, this encodes the problem of identifying the largest common isometric subgraph between $\mathcal{G}$ and $\mathcal{G}'$. To our knowledge, this problem has not previously been approached using globally rigid graphical models, since the absence of a single node will generally 'break' the rigidity of the graph in question.

2.3. Matching with globally rigid graphs

A graph $\mathcal{G}$ is said to be globally rigid if the pattern of distances for edges in $\mathcal{G}$ uniquely determine the distances in the graph complement $\overline{\mathcal{G}}$. In simpler terms, the only transformations that can be applied to the node coordinates while preserving the distances in $\mathcal{G}$ are isometries.

Several papers have proposed to solve the matching problem in (Eq. (1)) by using graphical models whose embedding in the plane is 'globally rigid' [15–17]. In these formulations, the nodes of the graphical model correspond to points in $\mathcal{G}$, while their assignments are points in $\mathcal{G}'$. If the globally rigid graph in question forms a *junction-tree* with small maximal clique size then *min-sum belief propagation* [19] results in an exact and efficient solution to the matching problem.

Essentially, if we have a rigid graph $\mathcal{R}$ (which includes the same nodes as $\mathcal{G}$, but only a subset of its edges), it is not necessary to solve the problem described in (Eq. (1)), but only

$$\hat{f} = \underset{f:\mathcal{G}\rightarrow\mathcal{G}'}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{R}} |d(g_i,g_j) - d(f(g_i),f(g_j))|, \tag{3}$$

(similarly for (Eq. (2)) in the case of occlusions). Preserving the lengths of the edges in $\mathcal{R}$ implies that the lengths of *all* edges $(i,j)\in\mathcal{G}$ are preserved, due to the global rigidity result (see Theorem 1 from [15]).

Research in this direction has consisted of producing globally rigid graphs with smaller maximal clique sizes. Ref. [15] proposed using a '3-tree' model, which had maximal cliques of size 4, allowing inference to be run in $O(|\mathcal{G}||\mathcal{G}'|^4)$ time and space. Ref. [16] reduced this to $O(|\mathcal{G}||\mathcal{G}'|^3)$ using an iterative algorithm based on loopy belief propagation. It was shown in [17] that this could be achieved in a single iteration, resulting in a faster algorithm in practice. Examples of such models are shown in Fig. 1.

3. Our model

We start from the model described in [17]. There, the authors start with a graph that is rigid to translations and rotations (but not reflections). To handle isometric transformations, a third-order term is added to (Eq. (1)) which either enforces or bans reflections:

$$\hat{f},\hat{r} = \underset{f:\mathcal{G}\rightarrow\mathcal{G}', r\in\{1,-1\}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{R}} |d(g_i,g_j) - d(f(g_i),f(g_j))| + \underbrace{\sum_{i=1}^{|\mathcal{G}|-2} r \times \det\begin{bmatrix}(g_i - g_{i+1})^T\\(g_i - g_{i+2})^T\end{bmatrix} \times \det\begin{bmatrix}(f(g_i) - f(g_{i+1}))^T\\(f(g_i) - f(g_{i+2}))^T\end{bmatrix}}_{r\ r\text{ 'selects' whether the determinants should agree or disagree in sign}}. \tag{4}$$

If the points (g_i,g_{i+1},g_{i+2}) are reflected, it will change the sign of the determinant above; thus the second term in (Eq. (4)) ensures that the points have a consistent orientation. Either they should all be reflected in the target ($r=1$), or none of them should be ($r=-1$).

We improve this model by making a simple modification to the topology found in [17]. Whereas the graph from [17] forms a chain, it allows for more efficient inference if this model is replaced by a 2-tree, such as the one from Fig. 1 (bottom right). We shall say that the variables g_1 and g_2 form the 'root' of the 2-tree. Although this graphical model still has maximal cliques of size 3, meaning that min-sum belief propagation would take $O(|\mathcal{G}||\mathcal{G}'|^3)$ time, we can achieve faster results by conditioning upon g_1, g_2, and r. By conditioning on these three variables, the expected locations of the remaining points are known; we can search for points close to these locations in logarithmic time using a KD-tree [20]. Thus we have $O(|\mathcal{G}'|^2)$ different points to condition on, $O(|\mathcal{G}|)$ points to search for, and an $O(\log|\mathcal{G}'|)$ search algorithm; this results in a final running time of $O(|\mathcal{G}||\mathcal{G}'|^2\log|\mathcal{G}'|)$. The algorithm requires $O(|\mathcal{G}'|\log|\mathcal{G}'|)$ space, as required by the KD-tree [20].

The KD-tree also allows us to search for K-nearest-neighbors. While this is not necessary in the exact case, it may prove beneficial under noise. For fixed K the asymptotic running time remains the same.[3]

Pseudocode for our method is shown in Algorithm 1. Although this algorithm will not necessarily produce the same solution as running the $O(|\mathcal{G}||\mathcal{G}'|^3)$ min-sum algorithm for point-patterns subject to noise, it will produce the same solution *if an isometric*

(*footnote continued*)
graph with a globally rigid embedding. Note that rather than using the common notation $\mathcal{G}=(\mathcal{V},\mathcal{E})$, we consider $\mathcal{G}$ to be a set of points for simplicity, and say $(i,j)\in\mathcal{G}$ when we want to refer to edges.

[2] Additional points in $\mathcal{G}'$ are referred to as 'outliers', while additional points in $\mathcal{G}$ are 'occlusions'.

[3] A KD-tree computes nearest-neighbors amongst n points in expected-case $O(\log n)$ for random points, though certain distributions of points may lead to worse performance.

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008

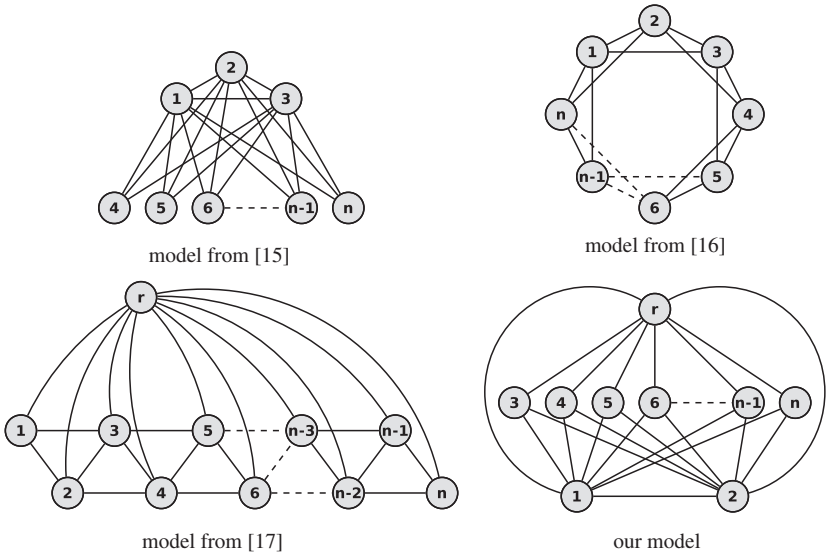

Fig. 1. Models for rigid matching from [15–17]; the proposed model is shown at the bottom right (the nodes labeled **1** to **n** represent the points $g_1,\ldots,g_n$ in $\mathcal{G}$; **r** is a boolean variable encoding whether or not the pattern is reflected). The models are globally rigid when embedded in the plane.

instance exists. Also note that under this assumption any arbitrary choice of root nodes g_1 and g_2 will lead to the same solution. Thus we replicate precisely the theoretical results reported in [15]; in the case of noise, we will show in Section 4 that the two algorithms perform comparably. Also, note that due to the `break`statement on Line 10, the algorithm will be much faster if a 'good' solution is found early on. Thus if *bestcost* is initialized using a reasonable search heuristic, we may observe running times closer to $O(|\mathcal{G}'|^2)$.

Algorithm 1. Find the function f resulting in the best match.

Input: two graphs $\mathcal{G}$ and $\mathcal{G}'$ ($|\mathcal{G}| \leq |\mathcal{G}'|$), and root nodes g_1 and g_2
Output: a function $f : \mathcal{G} \rightarrow \mathcal{G}'$
1. **Initialize:** $best = \varnothing$, $bestcost = \infty$
2. **for** each possible mapping $(f(g_1), f(g_2)) \in \mathcal{G}' \times \mathcal{G}'$ **do**
3. **for** $r \in \{1,-1\} = \{$reflected,not reflected$\}$ **do**
4. $cost = ||g_1 - g_2| - |f(g_1) - f(g_2)||$
5. **for** each node $g_i \in \mathcal{G} \backslash \{g_1, g_2\}$**do**
6. Find the expected position p of $f(g_i)$, given $f(g_1)$, $f(g_2)$, and r
7. Search for p's nearest-neighbor n_p using a KD-tree
8. $cost = cost + ||g_1 - g_i| - |f(g_1) - n_p|| + ||g_2 - g_i| - |f(g_2) - n_p||$
9. **if** $cost > bestcost$ **then**
10. break
11. **end if**
12. **end for**
13. **if** $cost < bestcost$ **then**
14. $best = f$
15. $bestcost = cost$
16. **end if**
17. **end for**
18. **end for**
19. **Return:** *bestcost*,*best*

3.1. Handling occlusions

At this stage, our model does not handle occlusions (i.e., outliers in the *template* pattern). However, the proposed 2-tree model does have the advantage that removing any nodes other than g_1 or g_2 (the 'root') will not 'break' the global rigidity of the model. Thus if we bound the distance in Algorithm 1, Line 8 by κ (the occlusion cost), this algorithm will solve the problem of matching with occlusions *assuming that we know two points that are not occluded*, which are used as the root of the tree.

Of course, we do not know in general any two points that are not occluded, so we would like to avoid such a requirement. Assuming that fewer than $\lfloor |\mathcal{G}|/2 \rfloor$ points are occluded, doing so is easy – we can just choose $\lfloor |\mathcal{G}|/2 \rfloor + 1$ independent edges (i.e., edges with no nodes in common), and use each of them as a root – at least one of these edges *must* contain a pair of non-occluded points. By rerunning Algorithm 1 with each of these edges as the root, the minimum cost solution amongst all reruns shall correspond to the desired solution. Thus for $N_{\text{occ}} < \lfloor |\mathcal{G}|/2 \rfloor$, we require $O(N_{\text{occ}})$ reruns, resulting in a running time of $O(N_{\text{occ}}|\mathcal{G}||\mathcal{G}'|^2 \log|\mathcal{G}'|)$.

It is not complicated to extend this result to $N_{\text{occ}} \geq \lfloor |\mathcal{G}|/2 \rfloor$. First, we note that if all edges within a maximal clique of size K are used as the root then we can handle up to $K-2$ occlusions, since the remaining two points will be used as the root during some rerun. Now, suppose that we have C such cliques; we only require that *one* of these cliques contains a pair of non-occluded points, which is guaranteed if $N_{\text{occ}} < (K-1)C$. Thus using cliques of size K, we can handle $N_{\text{occ}} < (K-1)\lfloor |\mathcal{G}|/K \rfloor$ occlusions, which requires a total of $\lfloor |\mathcal{G}|/K \rfloor \binom{K}{2} \in O(|\mathcal{G}|K)$ reruns. This scales until we have $|\mathcal{G}|-2$ occlusions, in which case we must consider *every* edge as the root, requiring precisely $|\mathcal{G}|^2$ reruns.

The behavior when $|\mathcal{G}|/K$ is not an integer is simple: to handle at most N_{occ} occlusions, we will need to use a combination of cliques of size $\lfloor |\mathcal{G}|/(|\mathcal{G}| - N_{\text{occ}} - 1) \rfloor$ and cliques of size $\lfloor |\mathcal{G}|/(|\mathcal{G}| - N_{\text{occ}} - 1) \rfloor + 1$. We will need $|\mathcal{G}| \bmod (|\mathcal{G}| - N_{\text{occ}} - 1)$ of the larger cliques, and $(|\mathcal{G}| - N_{\text{occ}} - 1) - (|\mathcal{G}| \bmod (|\mathcal{G}| - N_{\text{occ}} - 1))$ of the smaller cliques. Thus for $|\mathcal{G}| = 8$ and $N_{\text{occ}} = 4$ (for example), we would need two cliques of size 3, and one clique of size 2. A demonstration of the number of edges required for different values of N_{occ} is shown in Fig. 2.

Pseudocode for our algorithm is shown in Algorithm 2. Remember that N_{occ} is the number of occlusions we wish to be able to *handle*; the actual number of occlusions must be no greater than this value. Furthermore, we adapt Algorithm 1, Line 1 so that it uses the value of *bestcost* from Algorithm 2; this way

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008

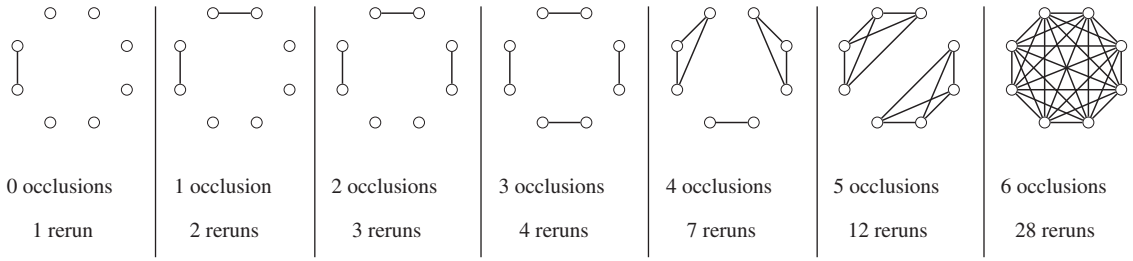

Fig. 2. The number of reruns required for different values of N_{occ} for a graph with $|\mathcal{G}| = 8$ nodes. Example choices of the edges to be used as the root are shown, though the nodes may be permuted in any order.

we do not waste time searching for solutions that have larger cost than the best solution already found.

Algorithm 2. Find the maximal common isometric subgraph.

Input: two graphs $\mathcal{G}$ and $\mathcal{G}'$, and $N_{\text{occ}} \leq |\mathcal{G}|-2$
Ouput: a function $f : \mathcal{G} \rightarrow \mathcal{G}'$
1. **Initialize:** $best = \varnothing$, $bestcost = \infty$
2. $K_{\text{small}} = \lfloor |\mathcal{G}|/(|\mathcal{G}|-N_{\text{occ}}-1) \rfloor$ (size of small cliques)
3. $K_{\text{large}} = K_{\text{small}} + 1$ (size of large cliques)
4. $N_{\text{large}} = |\mathcal{G}| \bmod (|\mathcal{G}|-N_{\text{occ}}-1)$
5. $N_{\text{small}} = (|\mathcal{G}|-N_{\text{occ}}-1)-N_{\text{large}}$
6. $ind = 1$
7. **for** size $\in$ {large,small} **do**
8. **for** $c \in \{1 \ldots N_{\text{size}}\}$ **do**
9. **for** $(i,j) \in \binom{\{1 \ldots K_{\text{size}}\}}{2}$ **do**
10. run Algorithm 1 using the root (g_{i+ind}, g_{j+ind}), returning $cost, f$
11. **if** $cost < bestcost$ **then**
12. $best = f$
13. $bestcost = cost$
14. **end if**
15. $ind = ind + K_{\text{size}}$
16. **end for**
17. **end for**
18. **end for**
19. **Return:** $bestcost, f$

A plot of the number of reruns required for different values of N_{occ} is shown in Fig. 3, for $|\mathcal{G}| = 10{,}000$. The function is bounded below by the linear function $y = N_{\text{occ}}$, and above by the quadratic function $y = \binom{N_{\text{occ}}}{2}$; it is not significantly worse than linear for reasonable values of N_{occ}.

Note that at this point the problem can be 'reversed', i.e., we can just as easily consider searching for $\mathcal{G}$ in $\mathcal{G}'$ as we can consider searching for $\mathcal{G}'$ in $\mathcal{G}$. Since our asymptotic running time has higher degree in $|\mathcal{G}|$ than in $|\mathcal{G}'|$ (at least for large values of N_{occ}), we shall consider $\mathcal{G}$ to be the smaller of the two graphs, without loss of generality.

The notion of constructing graphs that remain globally rigid when nodes or edges are deleted is not new, and is commonly referred to as 'redundant rigidity' [21]. Such a notion is useful in robotics applications (for example), whereby a group of robots may maintain a rigid formation even if some are unable to communicate. However, the goal in such applications is typically to minimize the number of *edges* in the graph (i.e., the total amount of communication), whereas we wish to minimize the maximal clique size (so that inference is efficient). Fig. 4 (left) shows a graph that remains rigid after a single deletion, though it has maximal cliques of size 4. Instead, we are using a collection of graphs, each of which has maximal cliques of size 3. To continue the analogy with robotics, our approach corresponds to communication over several different channels—at least one of which will remain rigid even if some robots are unable to communicate.

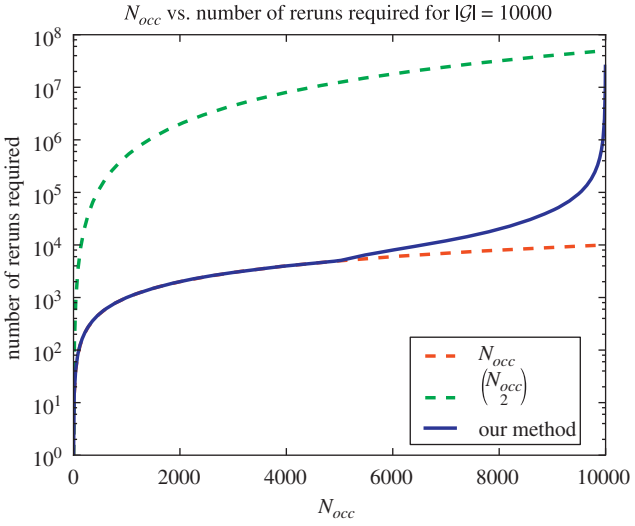

Fig. 3. The number of reruns required for different values of N_{occ} when $|\mathcal{G}| = 10{,}000$. The linear function $y = x$ and the quadratic function $y = \binom{N_{\text{occ}}}{2}$ bound the function below and above, and are shown in dotted lines (note the logarithmic scale).

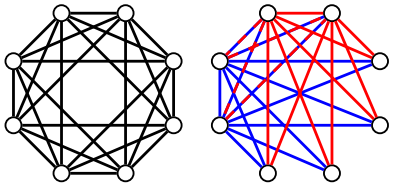

Fig. 4. The 'redundantly rigid' graph (at left) has large maximal cliques, rendering inference intractable. Our approach (at right) uses a collection of graphs (in this case, two graphs for $N_{\text{occ}} \leq 1$). The graphs are shown using two different colors, where the dotted edges are shared. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Experiments

4.1. Graph matching without occlusions

First, we compare the accuracy and running time of our method to those from [15–17,22].[4] For these experiments, we assume that all of the points in $\mathcal{G}$ have mappings in $\mathcal{G}'$, i.e., there are no occlusions.

[4] Note that we implemented the method of [22] in Octave, which is not directly comparable to the other methods, which were implemented in C++.

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008

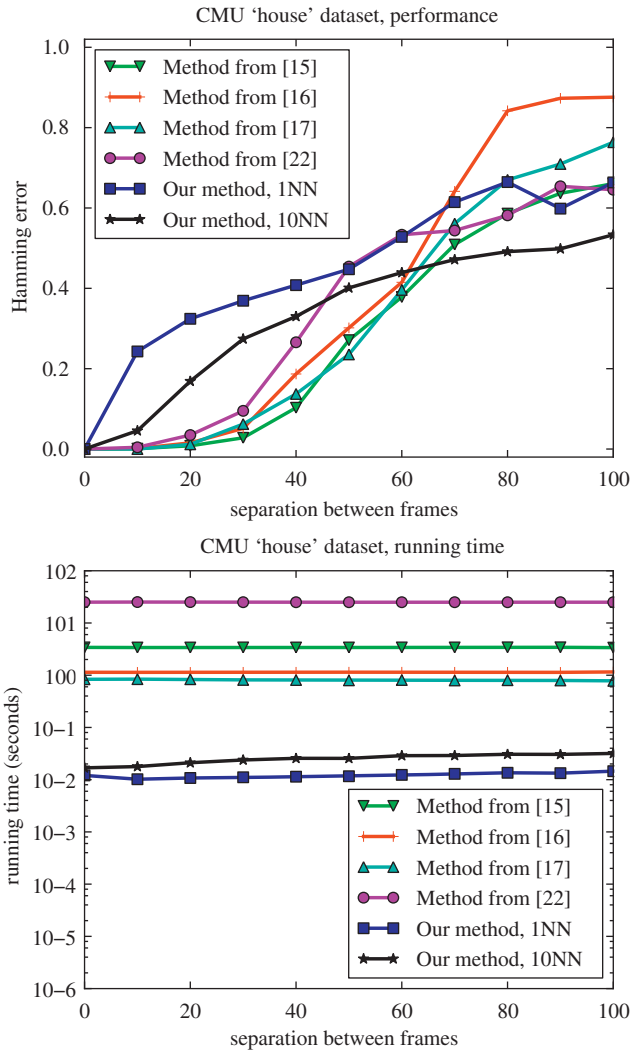

Fig. 5. Performance (top) and running times (bottom) on the CMU 'house' dataset. Note the logarithmic scale of the timing plot. All methods obtain similar performance, though ours is more than an order of magnitude faster.

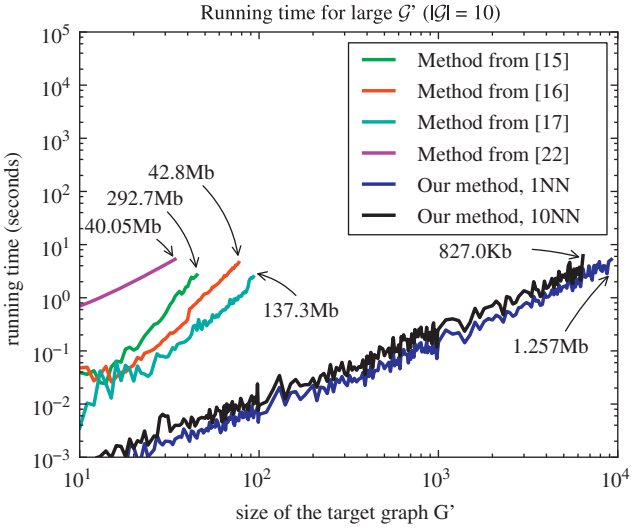

Fig. 6. Running time on large graphs (average of 10 repetitions). Note that *both* axes are shown on a logarithmic scale. Annotations show the memory footprints for the largest graph sizes tested.

4.1.1. House dataset

Fig. 5 shows the accuracy (top) and running time (bottom) of our method on the CMU 'house' dataset [23], which has been used in many computer vision papers [16,24–26]. This dataset consists of a series of 111 frames of a toy house, subject to increasing rotation in 3-D. The same 30 points have been identified in each frame so that the points in a pair of frames become $\mathcal{G}$ and $\mathcal{G}'$. The further apart these two frames appear in the video sequence, the more difficult the matching problem becomes. The error measurement is simply the 'Hamming error', i.e., the proportion of points incorrectly matched.

Fig. 5 (top) shows how the accuracy of our method varies as the separation between frames increases. When searching only for nearest-neighbors (1NN), our method is worse than its competitors, though when performing a 10-nearest-neighbor search (10NN), it achieves similar performance to the other methods (it is worse for nearby frames, but better for distant frames). The running time is shown in Fig. 5 (bottom); here we note that even the 10-nearest-neighbor version of our algorithm is *more than an order of magnitude* faster than its nearest competitor. Although our asymptotic bounds do not guarantee this level of improvement, our algorithm is able to achieve such results by terminating each search as soon as it has a higher cost than the best solution already found (see Algorithm 1, Line 10). By similar reasoning, the 10-nearest-neighbor version need not be significantly slower than the nearest-neighbor version, as it may find a low-cost solution earlier on. Consequently, the running time is not uniform across all baselines; it appears to become slightly slower when subject to high noise. Naturally, increasing the number of nearest-neighbors leads to better performance at the cost of increased running time, so in practice the number of nearest-neighbors should be tuned according to the running time allowance.

4.1.2. Exact matching in large graphs

Here we take a small graph $\mathcal{G}$ with $|\mathcal{G}| = 10$. Points in $\mathcal{G}$ are generated by uniform sampling, and randomly transformed to form $\mathcal{G}'$. Outliers are added to $\mathcal{G}'$, whose size is increased until each method takes over 5 s to perform a single match. There is no jitter in the point coordinates, meaning that each method reported obtains the same (correct) solution; thus only the running time varies. Fig. 6 shows the running time of each method as $|\mathcal{G}'|$ increases. Our method is able to run on graphs with $\sim$10,000 nodes in the same time as others are able to run on only $\sim$100. Our method exhibits a high variance in running time, which is again due to how quickly a 'good' solution is obtained.

Importantly, the other methods *could not* be run on larger graphs due to their memory footprints. The method from [15] has a memory footprint of $O(|\mathcal{G}||\mathcal{G}'|^4)$ meaning that it requires several 100 MB even for $|\mathcal{G}'| = 50$. Alternately, with memory requirements of only $O(|\mathcal{G}'|\log|\mathcal{G}'|)$, our method can in principle allow for graphs of several million nodes. Fig. 6 has been annotated to show the memory requirements of each method, for the largest graph on which it was run; the memory footprint of the proposed method is significantly smaller than its competitors, even for much larger graphs.

4.2. Graph matching with occlusions

In this experiment, we create a graph $\mathcal{G}$ by uniform sampling in the square $[0,1000)^2$, with $|\mathcal{G}| = 50$. The graph is randomly transformed to form $\mathcal{G}'$, though N_{occ} points are randomly replaced (again by uniform sampling). Random jitter from $[-5,5)^2$ is applied to all of the point coordinates. Here we set the occlusion cost κ to match the maximum amount of jitter, i.e., $\kappa = 10$. In practice κ would require manual tuning if the amount of noise is unknown.

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008

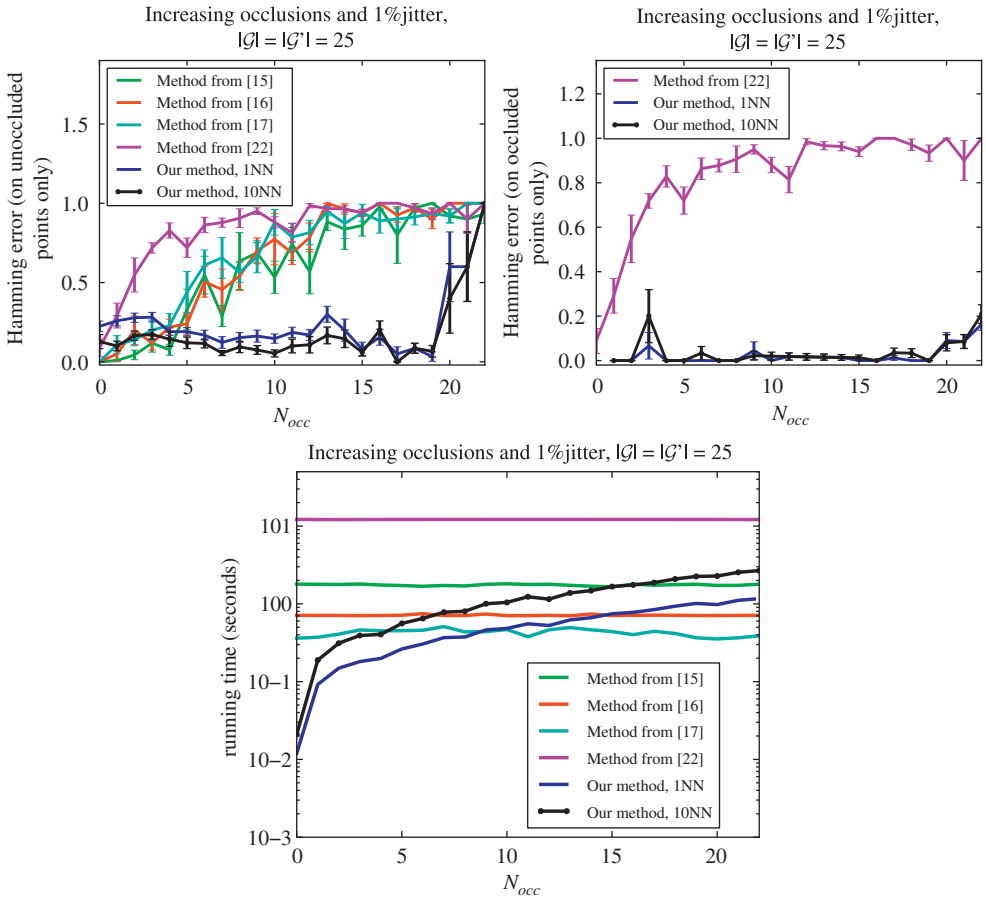

Fig. 7. Performance (top) and running times (bottom) as N_{occ} varies. Ten repetitions were performed, with error bars indicating standard error. The top left plot shows the error amongst only the unoccluded points, while the top right plot shows the error amongst only the occluded points (i.e., whether they were correctly identified as occlusions); only our method and that of [22] are shown in the top right plot, as the others are unable to identify occlusions.

Fig. 7 (top left) shows how each method performs on these graphs, for $0 \leq N_{\text{occ}} \leq |\mathcal{G}|-3$ (beyond which the 'correct' match is not unique). Naturally, only our method and that of [22] are able to *detect* occlusions, whereas the other methods generate arbitrary (incorrect) assignments for the occluded points. To allow for a meaningful comparison, we only measure how many of the *unoccluded* points are correctly matched by each method. Although the other models produce reasonable results for up to about $N_{\text{occ}} \simeq 10$, this graph clearly demonstrates that the other models are invalid for large numbers of occlusions (in fact, they are no better than random). Alternately, the performance of our method does not degrade as N_{occ} increases, until about $N_{\text{occ}} \simeq 40$; however, with so few unoccluded points, there may be spurious low-cost solutions, meaning that this result is to be expected.

The error on the remaining points (i.e., the proportion of occluded points incorrectly identified as *not* being occluded) is shown in Fig. 7 (top right). Although the method of [22] is in principle able to handle occlusions using our energy function, it quickly becomes inaccurate as the number of occlusions increases; the problem appears to be that the method of [22] requires (Eq. (2)) to be expressed as an 'argmax' problem with positive potentials, meaning that when the potentials are rescaled to be positive, the 'important' part of the potential (i.e., the part encoding Euclidean distances) is concentrated within a small range of values, and is subsequently ignored by the approximation.

Fig. 7 (bottom) shows the running time of each method. Naturally, for those models that do not encode occlusions, the running time is uniform as N_{occ} varies. Our method is faster up to about $N_{\text{occ}} \simeq |\mathcal{G}|/2$. However, at this point, the results obtained by the *other* methods are no better than random, as mentioned above. The running time of our method seems to closely follow the asymptotic prediction made in Fig. 3, though it is slightly better in the case of large graphs due to the optimizations discussed in Section 4.1.1.

It is only after about $N_{\text{occ}} \simeq 40$ that our algorithm is slower by a significant margin. However, as our method is the *only* one that is able to perform matching under these conditions, this is a promising result.

5. Conclusion

We have presented an algorithm for isometric graph matching that is significantly faster than its nearest competitors, while maintaining similar results in the case of noise. The asymptotic improvements in running time and memory requirements allow our algorithm to be run on graphs which are orders of magnitude larger than those allowed by previous methods. Furthermore, a simple modification to our model allows it to handle isometric matching with *occlusions*, meaning that we are able to solve what could be called the *maximal common isometric subgraph* problem, which has thus far not been possible for similar models based on global rigidity.

Acknowledgement

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008

ARTICLE IN PRESS

Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] P.F. Felzenszwalb, Representation and detection of deformable shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2) (2005) 208–220.

[2] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.

[3] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.

[4] P. van Wamelen, Z. Li, S. Iyengar, A fast expected time algorithm for the 2-D point pattern matching problem, Pattern Recognition 37 (8) (2004) 1699–1711.

[5] H. Chui, A. Rangarajan, A new algorithm for non-rigid point matching, in: CVPR, 2000, pp. 44–51.

[6] T. Akutsu, K. Kanaya, A. Ohyama, A. Fujiyama, Point matching under non-uniform distortions, Discrete Applied Mathematics 127 (1) (2003) 5–21.

[7] R. Nussinov, H.J. Wolfson, Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques, Proceedings of the National Academy of Sciences 88 (1991) 10495–10499.

[8] Y. Martin, M. Bures, E. Danaher, J. DeLazzer, I. Lico, A fast new approach to pharmacophore mapping and its application to dopaminergic and bezodiazepine agonists, Journal of Computer-Aided Molecular Design 7 (1993) 83–102.

[9] P.W. Finn, L.E. Lavraki, J.-C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, A. Yao, RAPID: Randomized pharmacophore indentification for drug design, in: Symposium on Computational Geometry.

[10] F. Murtagh, A new approach to point-pattern matching, Astronomical Society of the Pacific 104 (674) (1992) 301–307.

[11] H. Alt, L.J. Guibas, Discrete geometric shapes: matching, interpolation, and approximation: a survey, Technical Report, Handbook of Computational Geometry, 1996.

[12] P.J. de Rezende, D.T. Lee, Point set pattern matching in d-dimensions, Algorithmica 13 (1995) 387–404.

[13] H. Alt, K. Mehlhorn, H. Wagener, E. Welzl, Congruence, similarity and symmetries of geometric objects, Discrete Computational Geometry 3 (3) (1998) 237–256.

[14] M.T. Goodrich, J.S.B. Mitchell, Approximate geometric pattern matching under rigid motions, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (4) (1999) 371–379.

[15] T.S. Caetano, T. Caelli, D. Schuurmans, D.A.C. Barone, Graphical models and point pattern matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006) 1646–1663.

[16] J.J. McAuley, T.S. Caetano, M.S. Barbosa, Graph rigidity, cyclic belief propagation and point pattern matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (11) (2008) 2047–2054.

[17] T.S. Caetano, J.J. McAuley, Faster graphical models for point-pattern matching, Spatial Vision 22 (5) (2009) 443–453.

[18] K. Anstreicher, Recent advances in the solution of quadratic assignment problems, Mathematical Programming 97 (2003) 27–42.

[19] S.M. Aji, R.J. McEliece, The generalized distributive law, IEEE Transactions on Information Theory 46 (2) (2000) 325–343.

[20] J.L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[21] T. Jordán, Z. Szabadka, Operations preserving the global rigidity of graphs and frameworks in the plane, Computational Geometry 42 (6–7) (2009) 511–521.

[22] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, ICCV, vol. 2, 2005, pp. 1482–1489.

[23] CMU 'house' dataset. [link]. URL 〈http://vasc.ri.cmu.edu/idb/html/motion/house/〉.

[24] J. Yarkony, C. Fowlkes, A. Ihler, Covering trees and lower-bounds on quadratic assignment, in: CVPR, 2010, pp. 887–894.

[25] O. Duchenne, F. Bach, I. Kweon, J. Ponce, A tensor-based algorithm for high-order graph matching, in: CVPR, 2009, pp. 1980–1987.

[26] M. Leordeanu, M. Hebert, Unsupervised learning for graph matching, in: CVPR, 2009, pp. 864–871.

Julian J. McAuley received the BSc degree in mathematics and the BE degree in software engineering (with first-class honors and the university medal) from the University of New South Wales in 2007. He is currently undertaking a PhD at the Australian National University, under the supervision of Tibério Caetano.

Tibério S. Caetano received the BSc degree in electrical engineering (with research in physics) and the PhD degree in computer science (with highest distinction), from the Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. The research part of the PhD program was undertaken at the Computing Science Department at the University of Alberta, Canada. He held a postdoctoral research position at the Alberta Ingenuity Centre for Machine Learning and is currently a senior researcher with the Statistical Machine Learning Group at NICTA. He is also an adjunct senior fellow at the Research School of Computer Science, Australian National University. His research interests include pattern recognition, machine learning, and computer vision.

Please cite this article as: J.J. McAuley, T.S. Caetano, Fast matching of large point sets under occlusions, Pattern Recognition (2011), doi:10.1016/j.patcog.2011.05.008