Using $k$-d Trees for Robust 3D Point Pattern Matching

Baihua Li    Horst Holstein
Computer Science Department
University of Wales, Aberystwyth, SY23 3DB, UK
bbl00@aber.ac.uk    hoh@aber.ac.uk

Abstract

We propose a new method for matching two 3D point sets of identical cardinality with global similarity but locally non-rigid distribution. This problem arises from marker-based optical motion capture systems. The point-sets are extracted from similar design poses of two subjects with underlying non-rigidity and possible distribution discrepancies, one being a model set (manually identified) and the other representing observation of another subject, to be matched to the model set. There exists neither a single global scale, nor an affine transformation between the point-sets. To establish the goal of a one-to-one identification, we introduce a $k$-dimensional tree based method, which is well adapted and robust to such data, typically with distribution errors due to underlying subject non-rigidity. First, we construct a $k$-d tree for the model set. Then a similarity $k$-d tree of the data set is constructed following the structure information embedded in the model tree. Matching sequences of the two point sets are generated by traversing the identically structured trees. Experimental results confirm that this method is applicable for robust spatial matching of sparse point sets under non-rigid distortion.

Keywords: non-rigid robust point pattern matching (PPM), $k$-dimensional tree, spatial data representation, point-set alignment.

1 Introduction

Point pattern matching (PPM) and related problems have been extensively studied within a rich literature covering both theoretical and practical issues relating to computer vision and pattern recognition [11, 9, 2, 20], astronautics [21, 20], computational biology and computational chemistry [22, 19].

We propose another application relating to PPM, arising from marker-based optical motion capture (MoCap). Such systems provide an impressive facility for recording 3D feature-point data. They are widely used in clinical gait analysis, sports studies, animation and computer games [1, 24, 14], emphasising human motion representation. During MoCap, passive reflective markers, acting as feature points, are attached on human subjects. A subject’s movement can be accurately recorded in “real-time”, represented by a sequence of 3D feature-point coordinates. Reconstruction of the subject’s movement requires an additional step of labelling the captured points. We do this by first setting up a correspondence between the known feature points on the human subject and one frame of captured but unidentified data from a design pose of the subject. The labelled feature point data in the design pose then serves as a model for feature-point identification in dynamic MoCap trials of the subject. Typically, the same design pose and marker protocol is used on many different subjects, yet manual identification is needed every time a new subject model is to be generated. This applies even when the same subject is used in different MoCap trials, if markers have become detached or displaced between trials. Model generation is labour intensive, highly non-productive and consequently costly in commercial situations. There is an urgent need to reduce this cost burden.

In this study, we focus on the model generation issue in MoCap systems. We aim to develop a method that can automatically identify a model point-set on human subjects, to reduce the manual effort. We formulate this problem in general as robust point-pattern matching: assume we have a manually identified model point-set $\mathcal{P} = \{p_i \in (\mathbb{R}^3, \text{label}_i)\}_{i=1}^M$ taken from one subject in a design pose. We require to match this model point-set to the corresponding observed point-set $\mathcal{Q} = \{q_j \in \mathbb{R}^3\}_{j=1}^M$ of another human subject in the same pose. Sets $\mathcal{P}$ and $\mathcal{Q}$ have overall spatial distribution similarity and the same cardinality. In general, there will be distribution errors between them, due to underlying non-rigidity structures and position displacement of feature points. Scaling is also not uniform.

Point pattern matching is commonly encountered in computer vision, computational geometry, image analysis and pattern recognition. It may concern points on rigid objects in Euclidean motions (translation and rotation) [18,
or additionally take uniform scaling and data noise into account [16], or approximate perspective transformations [9, 19]. The optimisation problem is usually expressed and solved in matrix algebra employing least squares regression [4, 12], Hausdorff-distances [18, 10, 16], or the well-known heuristic Iterative Closest Point (ICP) algorithm [8]. These methods are based on geometric invariance- or constraint-satisfaction embedded in affine transformations, to find approximative matches.

Other methods use relative position indexing analogous to “string matching” for the exact one-to-one PPM problem. In Griffin and Alexopoulos’s work [15], the two point-sets were first aligned in a common coordinate-system. Then points in each point-set were sorted lexicographically by polar angle and distance from the point-set centroid. String matching could then be carried out on the two ordered point lists. This simple method works well for sparse point-sets and is capable of handling translation, rotation, scaling and noisy position in 2D. In the later work by Murtagh [21], the basic idea is similar to [15]. He introduced “the world view vector of a point” for matching 2D star patterns. It is well adapted to astronomical problems with a limited number of points, and unaffected by translation, rotation, scaling, random perturbation and outliers, although it is considered as brute force. Alt et al. [3] used a concept similar to Griffin and Murtagh for exact point matching in three dimensions. They projected all points of a set $A$ onto a unit sphere around its centroid $c_A$, obtaining a set of points $A'$ on the sphere. By this means, points were not only ordered by distances from $c_A$, but were also labelled with an additional adjacency list pertaining to the convex hull of $A'$. The same process was executed on set $B$ for the matching task.

The above methods are based on geometric invariance under an affine transformations. They are not applicable for the robust PPM discussed here. We would appeal to the spatial index approach, but the interpretation must be robust to non-rigidly distributed data. It should tolerate distribution error and non-uniform scaling under approximate alignment. Benefiting from the well-studied $k$-dimensional tree ($k$-d tree in short) techniques, we propose a $k$-d tree based method to address the problem.

In the following section, we give a brief review on $k$-d tree related techniques. In section 3, we propose the new $k$-d tree based method for the robust PPM, applied to the problem presented above. Experimental results are given in section 4. Complexity and conclusions are stated in section 5 and 6.

2 Brief review on $k$-d trees

Data structures for performing geometric matching and searching in multidimensional spaces is crucial to a wide range of fields, such as computer vision and computational geometry [23], pattern recognition, spatial databases [13, 25], astronomy, geographic information system (GIS), information retrieval and natural language processing. Many data structures are instances of the general class of binary space partition (BSP) tree, such as the highly adaptable $k$-d tree. In many areas of research, the $k$-d tree and its variants have been one of the most efficient and versatile methods for accessing multidimensional spatial databases, searching nearest- or farthest- neighbours, and indexing structured data. We benefit from the well-established techniques of $k$-d trees and attempt to develop a new variant for our problem, which is to find a mapping between two 3D point-sets of global similarity but locally non-rigid distribution.

The binary $k$-d tree was first introduced by Bentley [5], improved to an adaptive $k$-d tree in [6]. An extensive study of an optimised semidynamic $k$-d tree was described in [7]. Another highly efficient method, well adapted to data distribution, is the LSD (Local Split Decision) tree [17]. The Quad-tree/Oct-tree with their variants are also close relatives of the $k$-d tree [25].

Although many different flavours of $k$-d trees have been devised for general or specific applications, their purpose is always to decompose data space hierarchically into cells such that each cell does not exceed a desired number of input data. There are two types of cell in a $k$-d tree, interior nodes and terminal nodes (or leaves). An interior node indicates the partition along a hyperplane and has two children placed to the left and right. Data on the hyperplane or with negative normal projection are stored in the left subtree, and data on another side are stored in the right subtree. A terminal node is a bucket containing a non-null data set. The main differences among $k$-d tree representations are selection of hyperplanes and the structure and meaning of interior nodes and leaves, for achieving efficient storage and search time.

2.1 $k$-d tree

The original $k$-d tree in [5] uses axi-orthogonal cutting hyperplanes through data points to partition recursively a point-set at each interior node into two subsets. The sequence of hyperplanes are chosen to be perpendicular in turn to the $k$ axes. With $k=2$ for example, splitting hyperplanes at even depths are perpendicular to the $x$-axis (the $x$-discriminator); at odd depths, they are perpendicular to $y$-axis (the $y$-discriminator); the root node has zero depth. Each splitting hyperplane contains at least one point. When a node has an $x$-discriminator, data with $x$-values not exceeding that of the discriminator are stored in the left subtree, otherwise they are stored in the right subtree. Fig.1 illustrates a $k$-d tree created using example data in 2D for ease of representation, with leaf size chosen to be one. It is designed as a dynamic tree and the tree shape depends on
the insertion order of the points.

Example data:

- p1(6,-4); p2(4,2); p3(-7,-7); p4(3,-1); p5(7,0);
- p6(2,-8); p7(5,-6); p8(-8,9); p9(8,8); p10(-3,-4).

2.2 Adaptive k-d tree

The adaptive k-d tree [6] is constructed in a way similar to the classical k-d tree [5] with, however, hyperplane directions chosen in a non-fixed order and not containing any of the data points. Data set are always split into balanced numbers on both side of hyperplane at each partitioning. The data are placed only in the leaves. At each interior node, a discriminator contains the orthogonal-axis dimension (e.g. \(x\), \(y\)) and coordinate of the corresponding split. Splitting proceeds hierarchically until a desired number of points is left in the leaves. An example adaptive k-d tree is shown in Fig.2. The adaptive k-d tree is well balanced by median partitioning if the data are known before tree building.

2.3 Data distribution dependent k-d trees

Taking tendency of data distribution into account, Bentley [7] introduced an optimised semidynamic k-d tree for a known point-set. To build such a k-d tree, it is necessary first to find the dimension of the data that has the largest spread in order to determine the orthogonal hyperplane axis. A quick-sort is then performed along that axis, and the mean-position of data extension in that dimension is calculated to locate hyperplane as discriminator. Data are placed into the left subtree if their values do not exceed the discriminator, otherwise they are placed on the right. Discrimination proceeds hierarchically to partition data at each node into equal subspaces, until a specified number of input data is achieved at the leaves. An example is shown in Fig.3. We can observe the semidynamic k-d partition globally presents the distribution control property and has good aspect-ratio in terms of distribution tendency. But locally, partitioning may choose a hyperplane to pass through or near some data points as an artifact of the partition strategy, making the strategy sensitive to distribution error.
The final split position is flexible than those described above. In order to accommodate non-uniform distributed data, the directory of the LSD tree, partitioning the data into disjoint subsets by axi-orthogonal hyperplanes. But the splitting strategy is more efficient and adaptable to non-uniform distributed data. The tree is organized as an adaptive k-d tree developed by Henrich et al. [17]. It is efficient to avoid distribution error ambiguities. In this study, we select that direction of the middle-position of the largest projected interval, perpendicular to an orthogonal-axis \( SP \). The orthogonal-axis \( SP \) of the splitting plane is determined from Eq.2 by the largest \( \psi \)-projected interval \( \Delta_{\psi,v} \) of the data coordinates,

\[
SP_v := \max_{\psi \in \{x,y,z\}} \Delta_{\psi,v} \tag{2}
\]

where \( \Delta_{\psi,v} = \max_{p_{k+j} \in P_v} \frac{\left| (p_{k+j})_\psi - (p_k)_\psi \right|}{\frac{1}{2} \left( \Delta_{\psi_0,v} + \Delta_{\psi_0,v} \right)} \), and \( v \) is an ordering index. If the maximum intervals on \( \{x,y,z\} \) are quite similar within a threshold 10%, as

\[
\min_{\psi_0,\psi_1 \in \{x,y,z\}} \frac{\left| \Delta_{\psi_0,v} - \Delta_{\psi_1,v} \right|}{\frac{1}{2} \left( \Delta_{\psi_0,v} + \Delta_{\psi_0,v} \right)} < 0.1 \tag{3}
\]

then we have a choice of candidates \( \Delta_{\psi,v} \), which can be used to maintain a favourable aspect ratio [17]. Thus, we select that \( \Delta_{\psi,v} \) for which the direction \( \psi \) gives the maximal projection extent of the nodal data,

\[
\psi := \max_k E_{\psi_k,v} \tag{4}
\]

where \( E_{\psi,v} = \max_{p_j \in P_v} \left| (p_i)_\psi - (p_j)_\psi \right| \) denotes the projected data extent in \( \psi \) direction.

Observably, instead of considering only the maximum distribution tendency as in semidynamic k-d trees [7], our partition strategy has a spatial low-density priority order. Splitting prioritised by sparse intervals provides good ability to avoid distribution error ambiguities. In this study, we

For all k-d trees described above, splitting planes may very possibly go through a dense portion of the data, or just be on some points. This implies the tree shapes are very sensitive to point position. They may produce completely different tree representations for two similar point-sets but with distribution error. Moreover, they are not capable of handling scaling problems existing in our robust PPM, since these k-d trees use absolute spatial coordinates to store partition information of tree's construction. A detailed review on multidimensional data structures related to k-d trees can be found in [13] and [25].

3 The new similarity k-d tree

For a one-to-one match of two pre-aligned 3D point-sets with similar spatial structure, we propose a similarity k-d tree which is capable of handling spatial distributions, with errors due to underlying structural non-rigidity. Like most k-d trees, we use axi-orthogonal hyperplanes to recursively partition a point-set into subsets, but the splitting strategy is adapted in a manner appropriate to scaling and distribution error. In the case of the 3-dimensional point-set \( P_v = \{p_i \in \mathbb{R}^3\} \) at an interior node \( v \), the splitting plane is located at the middle-position of the largest projected interval, perpendicular to an orthogonal-axis \( SP_v \).

Thus, we use an axi-orthogonal hyperplane to recursively partition a point-set into subsets, but the splitting strategy is adapted in a manner appropriate to scaling and distribution error.
emphasise tree consistency and robustness not only to distribution error, but also to similar distributed data with non-identical scaling. For this goal, the hierarchical partition and tree directory is constructed as follows. An intuitional illustration using the 2D example data is given in Fig.4. Each interior node contains the hyperplane orthogonal-axis (e.g. \(x\) or \(y\)) and the number of points split to the left subtree \(|P_l|\). The reason for including the number of left-child points rather than a partition coordinate (as in the semi-dynamic \(k\)-d tree) is that an absolute coordinate position can not indicate a consistent partitioning in a corresponding scaled point-set. Moreover, using the number of left-son points further increases the ability of handling distribution errors when we extract a consistent tree from observed point-set (Sec.4.2). Comparing partition map (a) in Fig.4 with those in Fig.1 to Fig.3, we observe that hyperplanes in the similarity \(k\)-d tree are located further away from individual data points, with better tolerance to data distribution error.

![Figure 4. A similarity \(k\)-d tree using the example data.](image)

We summarise the similarity \(k\)-d tree construction algorithm in Fig.5. In our algorithm, it is applied to the model data \(P\).

Input: a point-set \(P_c\)
1. \(\text{if } P_c\text{ contains only one point then}\)
2. \(\text{return a leaf storing this point.}\)
3. \(\text{else:}\)
4. Determine hyperplane orthogonal-axis \(SP_i\) by Eq.2.4;
5. Split \(P_c\) into left/right subset \(P_l\) and \(P_r\) with the hyperplane through the mean-position of the biggest gap perpendicular to the orthogonal-axis;
6. \(\text{return } V_i = \text{BuildSimilarityKDTree}(P_l));
\(V_r = \text{BuildSimilarityKDTree}(P_r);\)

![Figure 5. BuildSimilarityKDTree \((P_c)\)](image)

4 Experiments: using the similarity \(k\)-d tree for robust PPM

In this section, we demonstrate the ability of the proposed similarity \(k\)-d tree for solving a practical problem in marker-based MoCap systems described in the introduction (Sec.1). We illustrate the similarity \(k\)-d tree performance in a difficult situation of matching underlying non-rigid data from human subjects. All data we used are obtained from a commercial marker-based MoCap system - the Vicon 512. In our motion capture system, the world coordinate system has its origin on the ground, the \(xy\)-plane is parallel to the ground, and the \(z\)-axis is vertical.

The experiments requested a model subject to stand still in a design pose, such as in Fig.7. Markers have been attached on subjects as feature points. One frame of feature-points \(P\) of the model subject was captured and labelled manually, to define the model point-set \(P = \{p_i \in \mathbb{R}^3, label, \}, i = 1, ..., M\). In the same way, one frame of an observed point-set \(Q = \{q_j \in \mathbb{R}^3, j = 1, ..., M\} \) was acquired from another subject with the same marker protocol and the same model-pose. The latter, however, contains scaling and distribution disparities relative to the model, as would be expected from two different subjects adopting similar poses. The aim is to establish a one-to-one mapping between model \(P\) and data \(Q\). The procedure is discussed below.

4.1 Point-set alignment

The two point-sets \(P\) and \(Q\) are generally obtained in different coordinate systems with distinct location and orientation. They need to be aligned to a consistent coordinate system by centring and rotating. This is carried out in three steps.
Firstly, the centroids are calculated as in Eq.5.
\[
\bar{c}_P = \left[ \frac{\sum_{i=1}^{M} p_i}{M} \right], \quad \bar{c}_Q = \left[ \frac{\sum_{j=1}^{M} q_j}{M} \right]
\]
\[(5)\]

Secondly, the orientation vectors \( \vec{o} \) in the co-centered systems are determined from the first distribution moments in Eq.6.
\[
\vec{o}_P = \frac{1}{M} \sum_i (p_i - \bar{c}_p) / |p_i - \bar{c}_p|
\]
\[
\vec{o}_Q = \frac{1}{M} \sum_j (q_j - \bar{c}_q) / |q_j - \bar{c}_q|
\]
\[(6)\]

Finally for the alignment, each point in \( P \) and \( Q \) should be transformed with respect to their centroids \( \bar{c}_P \), \( \bar{c}_Q \), and the orientation of vectors \( \vec{o}_P \), \( \vec{o}_Q \), through suitable choices of corresponding translation vectors \( T \) and rotation operators \( R \) as in Eq.7.
\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = R \left( \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \right)
\]
\[(7)\]

For each case \( P \) and \( Q \), the displacement vector \( T \) is the negative centroid vector of Eq.5. In our experiments, human subjects were asked to stand straight, parallel to the vertical \( z \)-direction, as in Fig.7. Their orientations differed mostly in the \( xy \)-plane. In this case, it was only necessary to define \( R \) as a rotation matrix around the \( z \)-axis.

The \( xy \)-plane projections of example 3D point-set alignments are shown in Fig.6, in which the directed line indicates the orientation vector.

### 4.2 Point-set interpretation by similarity k-d tree

We apply the similarity k-d tree construction (Fig.5) to the pre-aligned model point-set \( P \). This generates an ordered point list on the leaves. Construction information is stored at interior nodes. Having completed tree building for the known model point-set \( P \), we proceed by extracting a consistent interpretation tree for the pre-aligned observed data \( Q \). The process may be demonstrated by a simple example in 2D. Suppose that Fig.4 represents the model tree of \( P \).

For constructing the tree for the corresponding set \( Q \), we consider the \((SPv, |P|)\) information at each interior node of the model tree, and partition the observed set \( Q \) accordingly. Thus, referring to the root information \((x, 3)\), we split the data \( Q \) with respect to a plane perpendicular to the \( x \)-axis, and take the points with the three smallest \( x \)-coordinates to the left child-node. The remaining points of \( Q \) are stored in the right child-node. Then, at the left child-node, split

---

Figure 6. 3D Point-set alignment in \( xy \)-plane and projections: original data (left) and aligned data (right).
point-pair matches. The label of each model point is assigned to its matching point in the observed data list to give the sought-after correspondence. Although the example is given for 2D data, the process of identification illustrates also the 3D situation, with the addition of the \( z \)-dimension taking part in the partitioning.

### 4.3 Results

The proposed PPM method has been tested on the data from a number of human subjects (with heights from 1.2m to 1.8m). Illustrative identification results from three representative marker attachments in MoCap systems are shown in Fig.7. Identification results of sets \( Q \) are noted by labels of their models, as shown in Fig.7(a) - a routine marker protocol in clinical gait analysis and Fig.7(b) - a typical usage for human character animation. The example in Fig.7(c) investigated a dense distribution with 51 markers, identifying labels being omitted due to space limitation. The identified points are also shown linked for visualisation, and demonstrate the non-rigid poses with respect to their models. In all three cases, correct identification was achieved.

Factors that support correct identification are close pose similarity and low point density. Generally, tree generation for dense point-sets is sensitive to small pose differences. In this case, pose similarity is a stricter requirement than in the case of matching sparse point-sets. For instance, to identify a 23 or 33 feature-point pattern as in Fig.7(a) and Fig.7(b), we always achieved a 100% correct identification rate, despite obvious differences in leg flexion, and in the case of Fig.7(b), the opening out and level of the arms. But an extremely dense and complex pattern, such as shown in Fig.7(c), may easily result in wrong matches for unideal poses, such as with substantial lowering or raising of the arms compared to the model.

### 5 Complexity

We created the model \( k \)-d tree for the set \( \mathcal{P} \) using full sorting of the points according to their \( x \), \( y \) and \( z \) coordinates. Sorting is carried out at each interior node, and is the dominant work contributor. Construction of a balanced binary tree of \( M \) leaves therefore requires \( O(M \log M)^2 \) operations. The average over all binary trees is similar. The worst case arises when only one point is split off at each partition. In that case the performance degenerates to \( O(M^2 \log M) \).

We carried out identification of the point set \( Q \) also by full sorting at each interior node. Sorting need only be for one coordinate as specified by the model \( k \)-d tree node information. Thus, the identification time is about one third of the model \( k \)-d tree construction time, but the overall complexity remains of the same order in the two cases.

![Figure 7. Point identification for non-rigid human subjects: left model set \( \mathcal{P} \) followed by identified observed data set \( Q \) from another subject on the right.](image-url)
It is to be noted that the model $k$-d tree need only be constructed once for a specific marker protocol. Moreover, partitioning the set $Q$ during identification requires only relative sorting between the sub-partitions. A possible saving from such partial sorting has not been investigated here.

6 Conclusion

We proposed a similarity $k$-d tree method for robust PPM, that matching two same numbered 3D point-sets of similar spatial distribution, but having non-identical scales. There may also exist some degree of distribution error due to underlying non-rigidity of the subjects. The proposed method has been found effective for solving a PPM problem in a real-world application of marker-based MoCap systems. Experimental results demonstrate its robustness in a difficult situation: point matching with underlying non-rigid structures obtained from human subjects.

In developing the similarity $k$-d tree, we emphasised the robustness and adaptability to data distribution by partitioning adaptively in a spatial low-density priority order, to guarantee split planes are located away from point data with the least uncertainty to distribution error. This research extends the usage of trees for PPM with underlying non-rigidity.

Acknowledgments

All 3D data in our experiments were obtained by a Vicon 512 MoCap system, installed at the Department of Computer Science, UWA.

References