Improved Topological Fiducial Tracking in the reacTIVision System

Ross Bencina, Martin Kaltenbrunner and Sergi Jordà
Music Technology Group, Audiovisual Institute
Universitat Pompeu Fabra, Barcelona, Spain
{rbencina,mkalten,sjorda}@iua.upf.es

Abstract

This paper describes reacTIVision: a camera based two dimensional fiducial (marker) tracking system developed for the reacTable, a table based tangible musical instrument. Key features of reacTIVision include: (1) the ability to track a large number of fiducials with faster than real-time performance and (2) fiducial size may be varied depending on the number of distinct fiducial identities required. We describe recent advances in our implementation of topology-based fiducial recognition, including a generalised method for accurately computing fiducial location and orientation. A method of graph naming known as left heavy depth sequences is applied to the identification of topologically distinct fiducials. Also discussed is our approach to generating fiducial images for the system, in which we employ evolutionary computation to produce compact fiducials with specific geometric properties.*

1. Introduction

ReacTIVision is an open source system for tracking the location and orientation of fiducials (markers) in a real-time video stream. The system was developed for the reacTable* tangible user interface [1, 2] after initial prototyping with Costanza and Robinson's d-touch system [3]. The decision to develop a new tracking system stemmed primarily from the relatively low frame rates achieved with d-touch. We sought to support frame rates which exceeded the capabilities of our 640×480 60 Hz image sensors, while maintaining the overall robustness and accuracy of d-touch. Later, additional requirements emerged such as reducing the size of the fiducials and increasing the number of uniquely identifiable fiducials. We were able to address these requirements during the development of the reacTIVision system.

This paper is structured as follows: We begin in section 2 by outlining the reacTable* project – the context in which reacTIVision was developed[1], followed in section 3 by a review of related work in table based musical instruments and optical marker tracking systems. In section 4 we explain the topological region adjacency based fiducial recognition approach, and the properties of the fiducials which we designed for the system. Section 5 gives an overview of the reacTIVision software architecture and describes important modules of the system in more detail. Section 6 presents data demonstrating improvements over the d-touch system.

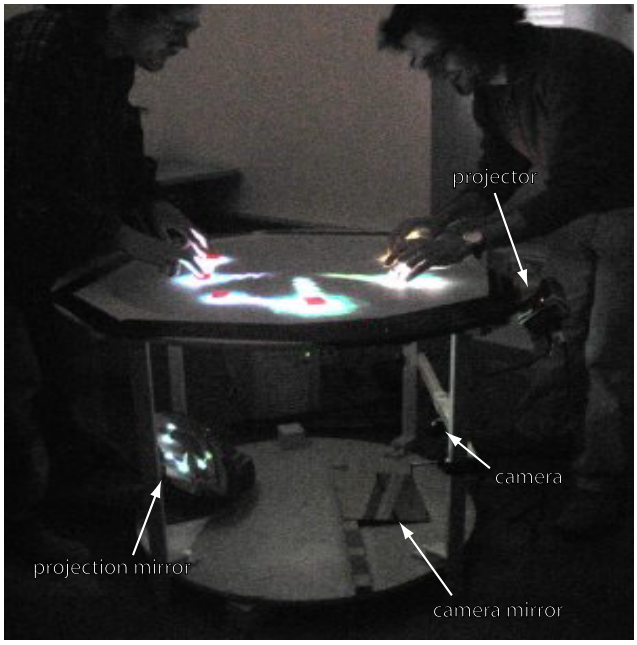

Figure 1: Musicians playing the reacTable*. Objects on the table's surface are tagged with fiducials which are tracked by a camera beneath the table.

2. Context

The reacTable* (shown in figure 1) is a tangible user interface where physical objects represent the components of a software sound synthesizer. Fiducials are attached to the underside of objects placed on a translucent table. A camera beneath the table captures images which are processed by reacTIVision to recognise the location, orientation and identity of the fiducials. This information is sent to other components of the reacTable* software via a network socket

[1] For further information about the reacTable* and its software components see http://www.iua.upf.es/mtg/reacTable/

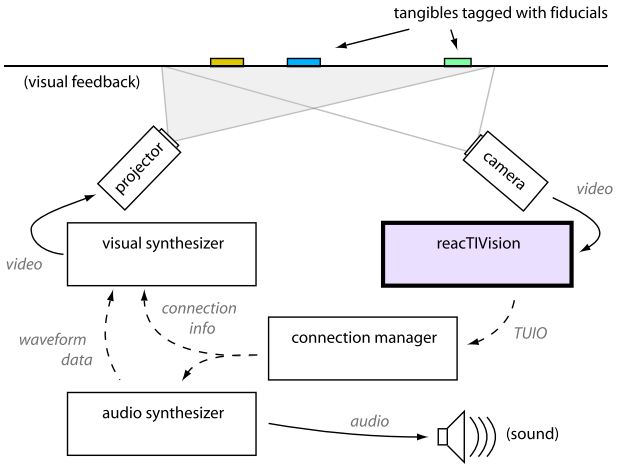

Figure 2: ReacTIVision's role in the reacTable* system. Solid lines indicate physical connections, dashed lines show Open Sound Control message paths.

using the TUIO protocol [4], a protocol layered on top of Open Sound Control [5] (see figure 2). Physically, the table is illuminated from below with infrared light. Optical filters isolate infrared image capture from projected visual feedback.

The reacTable* can be characterised as a synchronous real-time musical instrument. Manipulating objects on the table's surface provides a direct and immediate method of articulating nuanced musical gestures [2]. For example, rotating one class of objects effects their pitch, much like moving a finger placed on a violin string. This type of interaction requires accurate, low-latency tracking with sufficient temporal resolution to capture fast gestures.

Each tangible reacTable* object is associated with a specific behavior, such as making a particular sound, or transforming sounds in a certain way. This requires each object to be uniquely identifiable. At the time of writing the reacTable* used around forty unique fiducials, although some proposed scenarios suggest the use of thousands.

The reacTable* project imposes the following requirements on the system: (1) the system should be able to process all available video frames, since the video frame rate is typically lower than what is generally considered adequate for musical control; (2) that the system make efficient use of CPU resources so that other components of the reacTable* system such as the real-time sound synthesis engine can operate in parallel on the same machine; (3) that the system should maximise potential table area for a given camera resolution by simultaneously tracking many small fiducials; and (4) that the system should be scalable in the number of uniquely identifiable fiducials supported.

The physical setup of the reacTable* offers a number of opportunities for optimisation compared to free space 3D marker tracking systems such as the ARToolKit [6]. These include: (1) relatively controlled lighting conditions, (2) only 2D location and orientation information is required, and (3) because the fiducials are constrained to the plane of the table surface they are not subject to significant perspective distortion or variations in scale.

3. Related Work

In this section we consider related work in two separate areas: We give context to the reacTable* project by discussing table based tangible musical instruments, following which we review some other optical marker tracking systems.

3.1. Table-based tangible musical instruments

Most table-based tangible musical instruments use pre-existing optical marker tracking systems, while others utilise passive electromagnetic tags [7]. Unlike the reacTable*, which aims to be an expressive synchronous real-time musical instrument, the instruments surveyed below usually attempt to overcome speed and robustness limitations of existing visual tracking systems by employing an asynchronous musical scheme, such as a spatial representation of musical sequences scanned at a relatively low rate.

The Music Table [8] is a musical systems which utilises the full augmented reality capabilities of the ARToolKit [6]. The Music Table uses specially marked cards onto which it superimposes virtual 3D objects. Arranging the cards on the table surface allows the specification of simple melodic or rhythmic patterns. Other musical examples using the ARToolKit as a sensor component are the Music Blocks spatial sound installation [9], and the Augmented Groove dance music performance system [10].

Using their original d-touch system, which will be discussed in greater detail below, Costanza et al. created a series of three tangible musical toys called audio d-touch [11], which can all be classified as tangible sequencers. Audio d-touch consists of a standard USB web-cam mounted on a table lamp, which tracks fiducial-tagged wooden blocks from above. Each block represents a musical event whose properties are determined by its position. In one application melodies can be constructed by placing blocks on a sheet marked with a musical stave. In another, blocks are placed on a grid to control the timing and choice of sounds played by a drum machine.

Non-musical interactive surfaces incorporating optical marker systems include: The Magic Table, an augmented whiteboard which combines coloured marker based interaction techniques with projected video feedback [12]; and the ZOOMlab at the Vienna Childrens Museum which incorporates tangible interaction using the Assembly Table [13] interface based on the proprietary Marker XtracT system developed at Fraunhofer Institute.

3.2. Optical marker tracking systems

The present work stemmed from our experience with d-touch, a system developed specifically for tangible user interfaces [3]. D-touch makes use of a topological recognition approach which we have not encountered elsewhere. Below we review some other systems and approaches.

Within our field we observe that the ARToolKit [6] is the most commonly employed optical marker tracking system. ARToolKit markers can be arbitrary images framed in a black square. Markers are recognised with a simplified template matching algorithm. The use of arbitrary images offers a significant benefit to many applications as the markers can be made readable by users. In addition to marker tracking ARToolKit provides tools for overlaying aligned computer graphics on a real-time video stream. We note that the success of ARToolKit is not necessarily due to the quality or speed of its marker tracking subsystem. Recent projects such as ARTag [14] have specifically aimed to address shortcomings in ARToolKit's marker tracking.

Zhang et al. [15] evaluated the strengths and weaknesses of four publicly available square-shaped marker tracking systems including ARToolKit. They considered criteria of usability, efficiency, accuracy, and reliability. They also provide an insightful discussion of the diversity of requirements placed on such systems in different application areas.

A range of other marker recognition techniques have been explored in the literature, including: projective invariant line pencils [16], circular bar codes [17], Discrete Cosine Transform basis functions [18], and non-symmetric patterns of equally spaced dots [19].

4. Topology Based Fiducials

This section introduces the representations and data structures used by reacTIVision to recognise fiducials in binary images. First the topological region adjacency graph is introduced. Following subsections discuss the application of this representation in d-touch and reacTIVision. We then describe our approach to determining the location and orientation of fiducials. Next we outline the method used to generate the topology and geometry of individual reacTIVision fiducials. A final subsection introduces left heavy depth sequences, a method for computing canonical names for unordered trees which we use to recognise subgraphs corresponding to fiducials.

4.1. Topological Recognition

ReacTIVision employs the topological fiducial recognition approach introduced by Costanza and Robinson in the d-touch system [3]. In this approach, a region adjacency graph is derived from a binary image of the scene through the process of *segmentation*. The graph can be understood as a tree representing the containership hierarchy of the image, that is, which black regions are contained inside which white regions and vice-versa. Regions of the image containing no other regions (unbroken blobs for example) appear as leaf nodes in the region adjacency graph. Figure 3 illustrates some simple images and their corresponding region adjacency graphs. Observe that figures 3b and 3c have identical region adjacency representations even though their geometries are different.

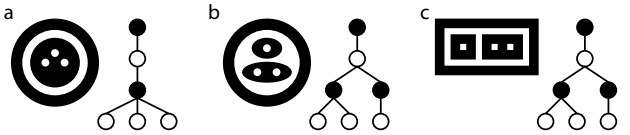

Figure 3: Some simple topologies and their corresponding region adjacency graphs.

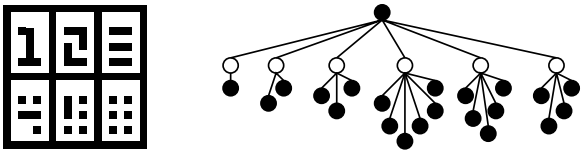

Figure 4: A d-touch fiducial and its region adjacency graph.

One important property of region adjacency graphs derived from binary images is that they belong to a class of graphs known as rooted trees [20]. Since the children of any node in the graph are not ordered the graphs are more specifically known as unordered rooted trees.

4.2. D-touch Fiducials

D-touch employs a single topology for all fiducials in a set. Figure 4 shows a d-touch fiducial and its region adjacency graph. The fiducial belongs to a set with a black grid containing six white regions each of which contains a different number of black regions from one to six. The set contains 120 unique fiducials which are differentiated using a permutational code expressed by the number of black leaf regions contained within the spatially ordered set of white regions. For example: reading clockwise, the code in Figure 4 is (1,2,3,6,5,4).

We identified a number of aspects of the d-touch approach which offered opportunities for improvement in our context: Firstly, d-touch binds geometric extraction of the permutational code to specific fiducial sets. Secondly, the original d-touch publication does not prescribe a specific method for computing location and orientation, leaving some steps to traditional computer vision techniques. Finally, the simple geometries of d-touch fiducials (which incidentally have the advantage of being easily drawable by hand) were not designed to minimise fiducial size.

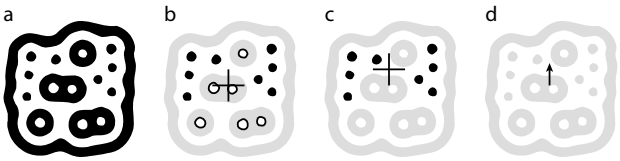

Figure 5: (a) a reacTIVision fiducial (b) black and white leafs and their average centroid (c) black leafs and their average centroid, and (d) the vector used to compute the orientation of the fiducial.

4.3. ReacTIVision Fiducials

Following our evaluation and re-implementation of a d-touch fiducial recogniser, we set out to produce smaller fiducials, and eventually arrived at a scalable technique which allows us to vary the size of the fiducials depending on the number of unique fiducials required. We also aimed to explore whether we could accurately compute the location and orientation of fiducials without resorting to additional image processing techniques such as corner or edge detection. Given that the region adjacency graph already existed and was relatively expensive to compute, it seemed wise to make maximum use of it.

ReacTIVision fiducials are identified purely by their topological structure. Each fiducial in a set has a unique topology. In addition to expressing the correct topology, the geometry of each fiducial is constrained by the method used to compute its location and orientation. In the next subsection we describe the method used to compute the location and orientation of fiducials; after that we discuss the generation of trees describing the topology of each fiducial; following which we give an overview of how we generate geometries which conform to the requirements of our location and orientation calculation method while minimising size. Note that the separation of this discussion is somewhat artificial as the methods were co-designed, with the requirements of each effecting the design of the others to some extent.

4.4. Fiducial Location and Orientation

Our method for computing fiducial location and orientation was influenced significantly by the design of our segmentation algorithm, which only retains axis aligned bounding boxes for each region. The center of a region's axis aligned bounding box provides a good approximation for the center of the region if the region is square, circular, and/or relatively small. We reasoned that as leaf regions will always be the smallest regions in a rendering of a tree, their centers are likely to be the most accurate spatial information we have about a fiducial. Consequently we choose to compute a fiducial's location and orientation as a combination of the bounding box centers of its leaf nodes.

As illustrated in figure 5, we compute the center point of the fiducial by taking a weighted average of all leaf centers. The vector from this centroid to a point given by the weighted average of all black (or white) leaf centers is used to compute the orientation of the fiducial. Each leaf is weighted by a function of its depth in the tree to account for the total area it consumes.

We selected this method because it can be applied to any fiducial with at least one black and one white leaf region. Thus allowing us to vary the topological structure of fiducials without changing the method used to track them. In practice we found that a minimum of four black and four white leaf nodes was necessary to achieve the tracking stability which we previously achieved with our d-touch fiducial tracker.

4.5. Fiducial Tree Generation

Before generating the geometry for a set of fiducials we generate a set of unique trees. Given a set size it is possible to calculate the number of tree nodes required to accommodate the set, however other constraints are also important such as ensuring a certain number of black and white leaf nodes. The number of nodes in a tree and its depth also impacts the minimum size of the geometry which can be generated for a tree. Additionally, smaller trees are less unique, leading to a greater potential for encountering false detections when trying to recognise them in a scene.

Rather than enumerating all possible trees (which is possible, see [21]), we randomly generate trees with the desired number of nodes and select those which fulfill criteria including maximum depth and number of black and white leaves. The generation process slows when many of the candidates have been found in the search space. By observing this slowing we experimentally reduce the number of nodes to the minimum required to generate a set fulfilling our criteria.

4.6. Fiducial Geometry Generation

Given a tree representing a fiducial's topology we create a compact geometry which conforms to the constraints implied by the location and orientation method described above: that the computed centroid of the fiducial is the same as, or lies very close to the real center of the fiducial and that the centroid of all black leaves is sufficiently distant from the centroid of all leaves to allow the fiducial's orientation to be computed with reasonable accuracy.

Each tree can be drawn in a huge number of ways making an exhaustive search for 'optimally' rendered fiducials impractical. We chose to employ a genetic algorithm to optimise parameters such as fiducial area, aspect ratio, symmetry and centroid locations for black and white leaves. We experimented with many combinations of these parameters before achieving satisfactory results.

The method used to draw a single fiducial involves aggregating circular leafs at angles relative to a fixed starting point while maintaining a fixed spacing between regions. A list of these angles forms the genotype for the genetic algorithm. Circles are easy to pack together at arbitrary angles, which we consider important to optimise the location of the centroids used to compute location and orientation.

We implemented a simple but effective distributed computing system which allows us to compute a number of different versions of a fiducial set within a reasonable period of time. When producing the currently used set of 128 fiducials with 19 tree nodes with a maximum depth of 3, we obtained usable results after 12 hours of computation time on a cluster of 11 dual processor 1Ghz Pentium 3 machines. In this time we computed 20 pools of 500 fiducials for each of 128 trees and selected the best result for each tree according to size and orientation vector length.

4.7. Left Heavy Depth Sequences

In order to recognise subgraphs representing fiducials in the region adjacency graph of the whole scene it is useful to be able to compute a canonical name for any subgraph. This name can then be compared against a dictionary of recognisable fiducial names[2]. One canonical form for unordered rooted trees is the left heavy depth sequence discussed by Nakano and Uno [21] and Asai et al. [22] which we describe below.

The *depth sequence* of a graph is defined as the sequence of node depth values encountered in a preorder (depth first) traversal, where the root has depth 0 and the depth of any other node is the number of edges between it and the root. Depth sequences can be used to uniquely describe ordered trees, however some unordered trees can be described by multiple depth sequences, depending on the order in which children are visited. For example, the depth sequences (0,1,2,3,3,2,3) and (0,1,2,3,2,3,3) are both possible for the tree in figure 3c. To resolve this ambiguity we can order traversal according to the 'weight' of the depth sequence of each child, where heavier sequences are those which compare as lexically greater. Following from this, the *left heavy depth sequence* can be understood recursively as the concatenation of the left heavy depth sequences of the children of a node in lexically decreasing order, where the depth sequence of a leaf is simply its depth.

Figure 6 shows a fiducial and its corresponding region adjacency graph annotated with the left heavy depth sequence of each node. As in the reacTIVision system, the outermost black region (the top node in the graph) serves only to provide a boundary for the white region it contains.

[2]In practice we first identify candidate subgraphs using simple metrics such as their depth and number of descendants. Only then is the canonical name of the subgraph considered.

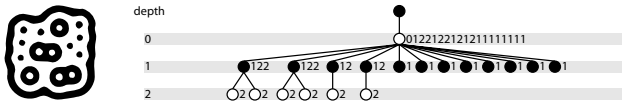

Figure 6: A fiducial and its left heavy depth sequence.

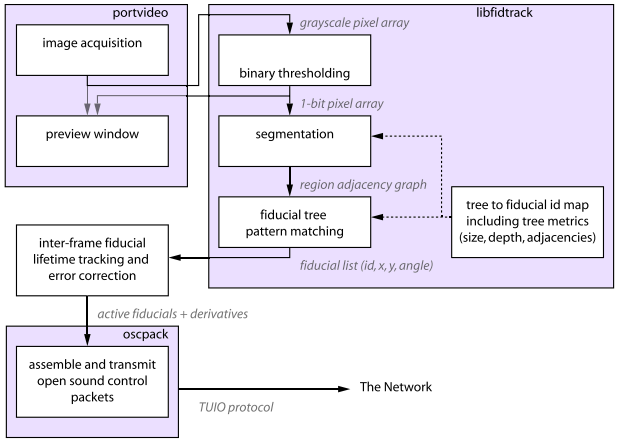

Figure 7: reacTIVision system diagram

5. Implementation

Figure 7 shows the module level design of the reacTIVision system, which aggregates three open source libraries: (1) PortVideo, a cross-platform real-time video acquisition and previewing library; (2) oscpack, a library for packing and unpacking Open Sound Control messages; and (3) libfidtrack, a library implementing the fiducial recognition algorithms described in this paper. The system communicates with clients using the TUIO protocol [4]. Glue code connects the libraries and handles fiducial object lifetime tracking, error correction and formatting of TUIO messages.

Figure 7 also illustrates the flow of data between modules: PortVideo acquires grayscale video frames from the operating system and passes them to libfidtrack. A thresholding algorithm produces a binary image which is fed to the segmentation module to construct a region adjacency graph. This graph is passed to the fiducial recognition module which recognises subgraphs whose topology matches those in a dictionary loaded from a fiducial tree description file. The recognised fiducial identifiers along with their location and orientation are then passed to a module which tracks fiducials as they become visible and disappear. This module also calculates second order information such as velocity and acceleration vectors. Finally this information is formatted into Open Sound Control packets conforming to the TUIO protocol and passed to clients via a network socket.

5.1. Thresholding

As noted by Pintaric [23], thresholding (binarisation) is an important and sometimes poorly handled step in optical marker tracking systems. Some earlier systems simply used a global threshold, while more recently, locally adaptive thresholding has been employed [23, 3]. Although the lighting conditions in the reacTable* are not as unpredictable as in some augmented reality applications, we found an adaptive scheme to be useful.

Unlike many applications where quality is of greatest importance, we were equally concerned with speed. One property of our system which allowed us to select an efficient algorithm was that correct thresholding is only required on image areas containing clean, well lit fiducial images within a known scale range. The segmentation process tends to take time proportional to the number of black and white regions in the image, so ideally our thresholder should output non-fiducial areas in a single colour. Having experimented with a variety of schemes based on locally computed statistics we currently employ a tile-based variation of Berensen's method [24]. This method computes the threshold for each tile (e.g. 32×32 pixels) as the mean of the minimum and maximum intensities in a larger surrounding tile. Areas of low dynamic range are clamped to black.

5.2. Segmentation

Segmentation constructs a region adjacency graph from the binary image produced by thresholding. This section presents some aspects of the technique used to build the graph. A key feature of our approach is that in general we construct an *incomplete* region adjacency graph.

The main data structure is the Region which represents a node in the graph. An axis aligned bounding box for each Region is stored for later use by the fiducial recogniser. During segmentation, the auxiliary RegionReference structure provides an additional level of indirection to support efficient region merging. Large pre-allocated arrays of Regions and RegionReferences are reused for each frame. Individual structures are allocated by taking the next free block from the end of the appropriate array. This mechanism is efficient and allows Regions to be enumerated by stepping through the array.

```
struct Region{
  int flags;
  short left, top, right, bottom;
  ...
  int adjacent_regions_count;
  Region *adjacent_regions[ MAX_ADJACENT ];
};

struct RegionReference{
    RegionReference *redirect;
    Region *region;
};
```

During segmentation, arrays of pointers to RegionReferences corresponding to pixels in the current and previous scan line are maintained. The algorithm proceeds through the source image line-by-line, extending existing Regions from the previous line and allocating new Regions when necessary. Adjacent Regions are linked together as discussed below. Some inputs require Regions to be merged. During merging the redirect pointer of one of the corresponding RegionReferences is set to point to the other RegionReference. This allows pointers to merged RegionReferences to be resolved lazily rather than requiring all pointers to be updated after every merge operation.

Bidirectional adjacency relations between Regions are represented using fixed size arrays of pointers to facilitate the allocation scheme mentioned earlier. This places an upper bound on the number of adjacency relations which can be maintained for a Region. When an adjacency array becomes full, new adjacencies are discarded and the participating Regions are marked as *saturated* (for the region whose array is full) or *fragmented* (for the other region) using the flags field. As a result of this mechanism, the segmentation algorithm may construct incomplete region adjacency graphs, however, note that subgraphs containing no saturated or fragmented nodes are complete. These complete subtrees are sufficient to perform fiducial recognition provided that the capacity of each region adjacency array is sufficient to represent the region adjacency graphs of the target fiducials.

5.3. Fiducial Recognition

Fiducial recognition proceeds in two steps: First, candidate subgraphs are identified in the incomplete region adjacency graph produced by the segmenter. Candidates are selected according to properties precomputed from the dictionary of recognisable fiducials, including total node count and maximum child depth. Second, the left heavy depth sequence of each candidate is computed according to the definition in section 4.7 and used as a key against a dictionary of depth sequences. If the sequence is found in the dictionary a fiducial has been identified and we compute its location and orientation according to the method described in section 4.4.

The incomplete region adjacency graph produced by the segmenter precludes top down traversal. As the graph is undirected, the parent of each node must be deduced during traversal. We employ a progressive bottom up traversal in which we iterate through each leaf attempting to traverse upwards. A counter in each parent is used to ensure that we only progress upwards once all of the children of the parent have been visited. Upwards traversal is terminated if a fragmented or saturated node is encountered, or if a criterion for candidate selection such as descendant count is exceeded.

Observe that in the worst case each node will be visited three times: once during the iteration to find leaf nodes, once during the upwards traversal to identify candidate sub-

Library	CPU Usage	Frame Rate
libfidtrack	18%	30 fps
libdtouch	82%	30 fps
dtouch_old	100%	10 fps

Table 1: CPU utilisation and frame rate of d-touch and reacTIVision fiducial trackers tracking 12 fiducials in a 640×480 frame.

trees, and (assuming each node can be in at most one candidate subtree) once during the recursive calculation of the left heavy depth sequence. Ignoring constant factors, and assuming that depth sequence dictionary look-up is a constant time operation, the time complexity for recognising all fiducials in a scene is linear with respect to the number of nodes in the region adjacency graph. This is the same time complexity offered by Costanza and Robinson's method [11]. While their method is constrained to three level trees, our method works with trees of any depth, allowing reacTIVision to recognise fiducials generated from a larger class of trees.

6. Results

This section summarises improvements achieved by reacTIVision over the d-touch topological fiducial tracker in three areas: performance, fiducial size, and scalability to different fiducial topologies and set sizes.

Table 1 compares the CPU utilisation and frame rates[3] obtained with reacTIVision (libfidtrack) and two versions of d-touch: a recent version available online (libdtouch) [25] and the original version obtained from the author (dtouch_old). The table indicates that the current reacTIVision implementation is over 4 times faster than the current d-touch system, and over 16 times faster than the version in use when development of reacTIVision began. These gains arise from the techniques described in the implementation section and are equally applicable to tracking d-touch fiducials. In fact, reacTIVision can track the type of d-touch fiducials shown in figure 4 with similar performance to its native fiducials. We note that these results are not comprehensive as other factors such as noise or non-fiducial inputs may also effect performance in the field. We also note that d-touch supports features not provided by reacTIVision, such as reporting the coordinates of the corners of rectangular fiducials.

The d-touch fiducial shown in figure 4 is representative of the set containing 120 unique fiducials. Decomposing the fiducial on a grid of pixel-like squares, it has a size of 21×21 units (including the required outer white boundary). Taking the diameter of the smallest circles on a reacTIVision fiducial (such as the one in figure 5) as an equivalent unit, the maximum size of the reacTIVision fiducials generated for a unique set of 128 is 15×15, an almost 50% reduction in area.

The generalised methods for recognising and tracking fiducials described earlier allow the reacTIVision system to support fiducial sets of varying sizes and topologies without any code changes. We consider this a significant improvement over d-touch in our context, where we wish to experiment with fewer smaller fiducials (at one extreme) and a larger number of uniquely identifiable fiducials (at the other).

[3]These results were obtained while tracking 12 fiducials filling a 640×480 frame. The system used was a 2.6 GHz Pentium IV with 512 Mb RAM running Fedora Linux 3, Kernel 2.6.11 with a fire-i camera in grayscale mode.

7. Future Work

The current system partitions leaf nodes in the region adjacency graph into two classes based on their colour. The left heavy depth sequence provides a partial ordering of nodes, which could be used to construct alternate partitionings of the graph. Alternate dual partitionings might create opportunities for more efficient spatial packing. Partitioning into more than two classes could facilitate an implementation of three dimensional tracking using a generalised method similar to the one we currently use to compute location and orientation.

Some recent marker tracking systems are able to recognise markers which are partially occluded, by a finger for example [16, 14]. ReacTIVision currently only recognises fiducials if their topology has not been modified by occlusion or thresholding errors. By using fiducials with sufficiently complex topology it may be possible to apply the method described by Lladós et al. [26] to implement occlusion or error tolerant tracking.

8 Conclusion

In this paper we have described reacTIVision, a fiducial tracking system for tangible user interfaces. We provided an explanation of the topological technique used to recognise fiducials, and presented advances beyond the earlier work of Costanza and Robinson in the d-touch system. Our enhancements include: the application of left heavy depth sequences for fiducial recognition and identification; a scheme for computing fiducial location and orientation which is independent of the fiducial's specific topology; and an incomplete region adjacency graph representation for efficient image segmentation. We also improved performance by employing an efficient locally adaptive thresholding scheme. In addition, we outlined our approach to designing fiducials using evolutionary computation.

Our implementation techniques yielded a four-fold increase in performance over the current d-touch system. Our

novel approach to geometry generation allowed us to generate a set of 128 unique fiducials each with an area almost 50% less than those in the set of 120 d-touch fiducials.

ReacTIVision is currently being used to track objects on a translucent table surface in the reacTable* tangible user interface. On our prototype platform, a dual processor AMD Athlon 2000, we are able to track many objects at 60 frames per second with a 640×480 pixel camera. Tracking is performed on one CPU and real-time audio is synthesised on the other, yielding a responsive musical instrument.

Acknowledgments

Thanks to Enrico Costanza for his insightful suggestions, to Bram de Jong for listening and for developing various fiducial to pdf schemes, to Pedro Cano and the AudioClas team for access to their computation cluster, and most of all to Gabi and Dare Remai for enduring the Fiducial Madness.

References

[1] S. Jordà, "Sonigraphical instruments: From FMOL to the reacTable*," in *Proc. of the 3rd Conf. on New Interfaces for Musical Expression (NIME)*, 2003, pp. 70–76.

[2] M. Kaltenbrunner, G. Geiger, and S. Jordà, "Dynamic patches for live musical performance," in *Proc. of the 4th Conf. on New Interfaces for Musical Expression (NIME)*, 2004, pp. 19–22.

[3] E. Costanza and J. A. Robinson, "A region adjacency tree approach to the detection and design of fiducials," in *Vision, Video and Graphics (VVG)*, 2003, pp. 63–70.

[4] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza, "TUIO: A protocol for table-top tangible user interfaces," in *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005 (to appear).

[5] M. Wright, A. Freed, and A. Momeni, "Open sound control: State of the art 2003," in *Proc. of the New Interfaces for Musical Expression Conf. (NIME)*, 2003, pp. 153–159.

[6] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. of the 2nd IEEE and ACM Int'l Workshop on Augmented Reality (IWAR)*, 1999, pp. 85–94.

[7] J. Patten, B. Recht, and H. Ishi, "Audiopad: A tag-based interface for musical performance," in *Proc. of the Conf. on New Interfaces for Musical Expression (NIME)*, 2002.

[8] R. Berry, M. Makino, N. Hikawa, and M. Suzuki, "The augmented composer project: The music table," in *The IEEE Int'l Symposium on Mixed and Augmented Reality 2003 (ISMAR)*, 2003, pp. 338–339.

[9] Audite. Audiocube installation description. [Online]. Available: http://www.audite.at/en/projects_audiocube.html

[10] I. Poupyrev, "Augmented groove: Tangible augmented reality instrument for electronic music," in *ACM SIGGRAPH 2000, Conf. Abstracts and Applications*, 2000, p. 77.

[11] E. Costanza, S. B. Shelley, and J. A. Robinson, "Introducing audio d-touch: A tangible user interface for music composition and performance," in *Proc. of the 6th Int'l Conf. on Digital Audio Effects (DAFX)*, 2003, pp. 63–70.

[12] F. Bérard, "The magic table: Computer-vision based augmentation of a whiteboard for creative meetings," in *IEEE Workshop on Projector Camera Systems (PROCAM)*, 2003.

[13] Fraunhofer. ZOOMlab installation description. [Online]. Available: http://www.fit.fraunhofer.de/projekte/kiwi/

[14] M. Fiala, "ARTag revision 1, a fiducial marker system using digital techniques," National Research Council of Canada, Tech. Rep. NRC 47419, 2004.

[15] X. Zhang, S. Fronz, and N. Navab, "Visual marker detection and decoding in AR systems: A comparative study," in *IEEE Int'l Symposium on Mixed and Augmented Reality 2002 (ISMAR)*, 2002, pp. 97–106.

[16] A. van Rhijn and J. D. Mulder, "Optical tracking using line pencil fiducials," in *Eurographics Symposium on Virtual Environments*, 2004, pp. 35–44.

[17] D. L. de Ipia, P. Mendona, and A. Hopper, "TRIP: a low-cost vision-based location system for ubiquitous computing," *Personal and Ubiquitous Computing journal*, vol. 6, no. 3, pp. 206–219, 2002.

[18] C. B. Owen, F. Xiao, and P. Middlin, "What is the best fiducial?" in *Proc. of the First IEEE Int'l Augmented Reality Toolkit Workshop (ART02)*, 2002.

[19] J. Molineros and R. Sharma, "Real-time tracking of multiple objects using fiducials for augmented reality," *Real-Time Imaging*, no. 7, pp. 495–506, 2001.

[20] E. Weisstein. Mathworld, entry for rooted trees. [Online]. Available: http://mathworld.wolfram.com/RootedTree.html

[21] S. Nakano and T. Uno, "Efficient generation of rooted trees," National Institute for Informatics (Japan), Tech. Rep. NII-2003-005E, 2003.

[22] T. Asai, H. Arimura, T. Uno, and S. ichi Nakano, "Discovering frequent substructures in large unordered trees," in *Proc. of the 6th Int'l Conf. on Discovery Science, DS'03*, vol. 2843. LNCS, 2003, pp. 47–61.

[23] T. Pintaric, "An adaptive thresholding algorithm for the augmented reality toolkit," in *Proc. of the Second IEEE Int'l Augmented Reality Toolkit Workshop (ART03)*, 2003.

[24] J. Bernsen, "Dynamic thresholding of grey-level images," in *Proc. of the 8th Int'l Conf. on Pattern Recognition (ICPR)*, 1986, pp. 1251–1255.

[25] E. Costanza. libdtouch (version of April 8, 2005). [Online]. Available: http://sourceforge.net/projects/libdtouch/

[26] J. Lladós, E. Martí, and J. J. Villanueva, "Symbol recognition by error-tolerant subgraph matching between region adjacency graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1137–1143, 2001.