Kinectrack: Agile 6-DoF Tracking Using a Projected Dot Pattern

Paul McIlroy*
University of Cambridge
Microsoft Research

Shahram Izadi†
Microsoft Research

Andrew Fitzgibbon‡
Microsoft Research

ABSTRACT

We present Kinectrack, a new six degree-of-freedom (6-DoF) tracker which allows real-time and low-cost pose estimation using only commodity hardware. We decouple the dot pattern emitter and IR camera of the Kinect. Keeping the camera fixed and moving the IR emitter in the environment, we recover the 6-DoF pose of the emitter by matching the observed dot pattern in the field-of-view of the camera to a pre-captured reference image. We propose a novel matching technique to obtain dot pattern correspondences efficiently in wide- and adaptive-baseline scenarios. We also propose an auto-calibration method to obtain the camera intrinsics and dot pattern reference image. The performance of Kinectrack is evaluated and the rotational and translational accuracy of the system is measured relative to ground truth for both planar and multi-planar scene geometry. Our system can simultaneously recover the 6-DoF pose of the device and also recover piecewise planar 3D scene structure, and can be used as a low-cost method for tracking a device without any on-board computation, with small size and only simple electronics.

1 INTRODUCTION

6-DoF trackers are the backbone of augmented reality (AR), bridging the gap between real and virtual coordinate frames. A tremendous variety of approaches exist: some use hardware such as inertial measurement units, others are computer vision based. Some computer vision approaches "look in" on the object to be tracked whilst others "look out" to infer the object's pose from observations of its surroundings [6, 10, 21]. Each approach has its own strengths and limitations in terms of cost, accuracy, robustness, form factor and on-device power consumption. In this paper we present a low-cost solution using only commodity hardware that provides accurate and robust real-time tracking with minimal hardware on the object to be tracked. Like "look in" methods, the tracking device requires no on-board computation or communication with the host; like "look out" methods, it offers high rotational accuracy.

Figure 1 illustrates the essential components of our system. An infrared (IR) dot-pattern *emitter*, specifically the laser diffractive optical element (DOE) from a Kinect, casts a large number of pseudo-randomly arranged rays into an arbitrary 3D scene, and these rays are observed by an IR camera. Knowing the dot pattern (the *reference*) allows the rotation and translation of the emitter to be determined relative to the camera, even without knowing the 3D geometry of the scene, and even if that geometry is changing over time.

Our system is designed to be a cheaper alternative to fixed camera 6-DoF trackers (such as the Vicon motion capture system). The system is orders of magnitude lower cost than commercially available systems, can work with a single camera (although it scales to larger camera setups) and requires only a cheap and small form-factor emitter on each tracked object. Our contributions in this paper are twofold:

- A new 6-DoF tracking system for AR called Kinectrack, with advantages over existing systems as outlined above.
- A general-purpose matching algorithm for projected dot patterns: a form of "variable baseline" matching for dot projection systems.

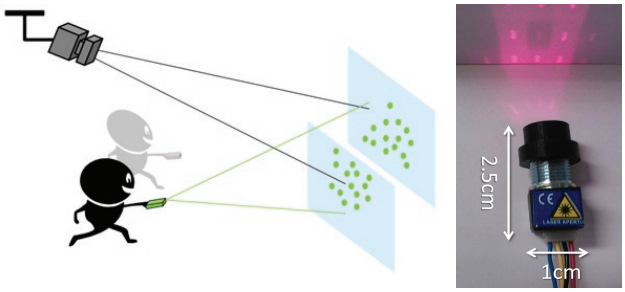

Figure 1: Tracking setup. (Left) A fixed camera, and free moving laser-pattern emitter provides the benefits of "look out" tracking, but the moving device is very simple—a laser, optics, and battery—and can be considerably smaller than "look in" remotes. (Right) The laser and optics of the remote unit. In practice, an infrared laser would be used; a visible-light unit is used here purely for illustration.

2 BACKGROUND

There has been a great deal of research that has explored 3D pose estimation of handheld devices for a variety of interactive scenarios including AR. Here we focus on vision-based trackers as most other technologies (e.g. inertial [29]) can be combined somewhat orthogonally with vision.

Perhaps some of the best known tracking techniques, exemplified by the commercial Vicon [25] and OptiTrack [17] systems, have used multiple fixed IR cameras and active diffuse IR illumination of the scene to localize retro-reflective markers placed on objects. These markers show up as a bright signal in the 2D IR camera images, and their 3D position can be derived through triangulation. Positioning multiple markers on an object enables 3D rotation to be computed. Whilst highly accurate in 3D translation, these systems are costly and require effort to set up and calibrate the infrastructure. To achieve high rotation accuracy for AR applications, the markers on the roaming unit need to be spaced sufficiently far apart, which can considerably increase the physical size of the roaming unit. Williams and Fitzgibbon [27] address this problem using a hologram to provide a large change in appearance for a small change in orientation, inspiring our current work.

The reverse of the "look in" setup is to keep the marker fixed in space and instead move a single camera to recover 6-DoF pose. These marker-based technologies [9, 7, 24, 23] have been adopted in a variety of AR applications, and can now even be supported on mobile phones. A variety of marker designs have been investigated including pseudo-random dot patterns [24]. However, these systems suffer from the need for careful placement of the marker

*e-mail: paulm@cantab.net
†e-mail: shahrami@microsoft.com
‡e-mail: awf@microsoft.com

before use, line-of-sight and occlusion issues, inaccuracies in tracking, and sensitivities to lighting. Systems have explored the use of projecting multiple fixed IR markers [4, 28] to resolve some of the issues of lighting and occlusions, the converse of our arrangement.

Removing the need for markers altogether, simultaneous localization and mapping (SLAM) systems [6, 10] obtain pose from markers naturally occurring in the scene, and, for applications involving video overlays, can use just one camera. Both SLAM and our method require a form of infrastructure: the scene. SLAM imposes the additional constraint that the scene is sufficiently textured. Our method only requires that the scene provide quasi-planar patches for matching and therefore also works for scenes dominated by textureless surfaces. SLAM is suitable for free AR as the sensing hardware moves with the user. Our method requires a fixed camera, which dictates the working volume, but in return offers drift-free, absolute 6-DoF pose per frame, even when the scene geometry is changing over time. Camera placement is flexible and setup is fast.

Several systems have been proposed which track laser pointers as interaction devices. In many instances, these are essentially 2-DoF trackers [3, 5, 1]. Higher degrees of freedom have been explored including roll [20] and the 3-DoF of a 1D projective transform [14]. Closely related to our work are 6-DoF trackers based on ray projectors. The "Sceptre" system of Wienss *et al.* [26] projects a 7-point pattern shaped like an "F". Our system differs in that the projected pattern has a much wider footprint, and we can match a small subset of the pattern while obtaining accurate pose. Using a wide-angle projector beam and requiring only a small fragment of the project pattern to be in view at each frame permits a much greater range of emitter poses and provides tolerance to occlusion, non-planar scenes, and varying surface albedo. A multiple camera implementation would provide full 360° pose if required. The matching problem with the 7-point system is simpler than ours, so allows matching of multiple overlapping patterns, which we have not demonstrated. Latoschik and Bomberg [13] describe a system which projects a regular grid onto a planar surface. Although this gives some improved accuracy, it still requires that the central ray of the grid is visible in every frame, making it intolerant to changing albedo or occlusion. Albitar *et al.* [2] propose a pattern design to facilitate matching by a structured light coding approach. Our matching method works with any pseudo-random dot pattern.

2.1 Wide-baseline dot pattern matching

The key to our approach is essentially the problem of establishing correspondence between two views of a large set of pseudo-randomly arranged 3D points. In the case where the correspondence is induced by a simple transformation such as a similarity transformation or homography, researchers have previously devised hashing schemes for point-pattern lookup.

Kolomenkin *et al.* [11] address the "lost in space" problem of matching an astronomical image into a reference star chart in order to recover the attitude of a spacecraft using a camera mounted on the spacecraft. The camera image in this case is related to the reference by a similarity transform, with unknown two-dimensional rotation, translation and scale. The proposed solution uses a geometric hashing approach with two stars used to form a canonical frame and the positions of a further two stars within this space providing a hash code. Lang *et al.* [12] solve for a similarity transform for earth-based cameras by computing invariants of four-point "quads" of points. As they require only a similarity invariant, they generate a four-valued vector for each quad, and index the vectors in a k-D tree, enabling precise lookup. Nakai *et al.* [16] use affine invariants of subsets of the nearest-neighbour graph to match document images in a large database. They look at the n nearest neighbours of each point, and compute invariants for each subset of size m to provide robustness to missing points.

Our work is very much inspired by these techniques, but we require our matching to work under more severe perspective distortions, in scenes where there may not be large planar areas, and rather than having a homography or other simple relation between the scenes, we must depend on the somewhat weaker constraints supplied by the essential matrix [8].

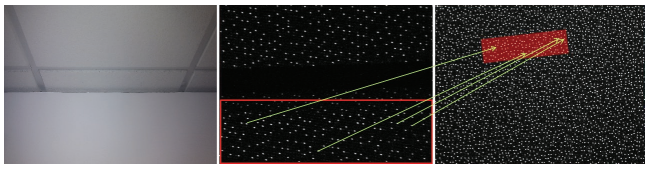

Figure 2: The matching process. A dense dot pattern projected onto a planar surface in the scene (left) is viewed through an IR filter (middle) and matched into the stored reference pattern (right). Correspondences from one or more planes allow the position of the roving emitter to be computed.

The image of a projected dot pattern on a plane in the scene is related to the emitter's reference pattern by a full projective transformation. Four points are therefore required to define a canonical frame that is invariant to plane orientation and emitter pose. Additional points are then required for hashing in the canonical frame. Each additional point required increases the likelihood that the hashing scheme will fail due to missing points resulting from detection failure. As the above works observe, there is a trade-off between hash code uniqueness and robustness to missing points.

We propose a technique inspired by this hashing approach that makes best use of all available points whilst remaining robust to missing data. In our method a voting scheme operates on the canonical frame which correctly models the effect of noise on both the points used to define the canonical frame and the points used for hashing. Figure 2 illustrates the matching problem we are solving. Here parts of the pattern projected by the handheld IR emitter are observed by the camera. The goal is to match this observed sequence with a the larger stored reference pattern.

3 Description

We now describe our matching and pose estimation algorithm. The system comprises two important components: a calibration phase, and a runtime phase.

The **calibration** phase computes the following:

- Intrinsic parameters of the fixed camera;
- Precise layout of the ray bundle in the emitter, called the *reference pattern*;
- A 2D lookup table (LUT) indexing 5-point subsets of the emitter points (called *kites*) by their projective invariants (2 scalars per kite).

The **runtime** phase operates on each captured frame, and proceeds as follows:

1. Detect points (subpixel centres of imaged dots);
2. Group points into kites;
3. Compute invariants for kites, index into the LUT, vote for correspondences;
4. Identify correct correspondences by RANSAC for the Essential matrix E;
5. Extract rotation and translation from E.
6. Compute "gold-standard" pose from inliers.

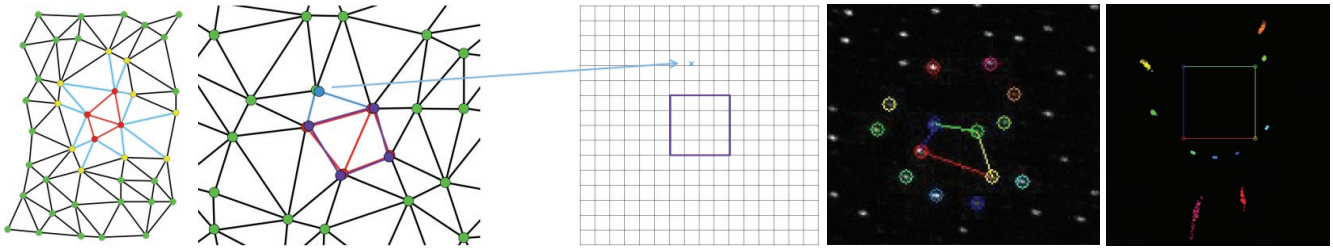

Figure 3: The core matching unit: a quad is generated from every pair of adjacent triangles, and the quad is paired with each of its neighbours to make kites. Kites have a perspective-invariant signature when transformed into the canonical frame in which the quad is the unit square. The signature is made robust to noise by jittering the reference points when building the lookup table, and to missed detections by using a voting scheme. The rightmost pair of images show an example of points extracted from a real image being mapped with noise into the canonical frame.

We will describe the run-time phase in detail, then the calibration phase, but to begin we shall introduce some notation. Let the reference pattern, which is projected from the roaming emitter, be denoted by the set of 2D points

$$R = \{\mathbf{r}_1, \ldots, \mathbf{r}_N\}. \tag{1}$$

At runtime, dots in an input image I are detected, giving the *test* set

$$P = \{\mathbf{p}_1, \ldots, \mathbf{p}_M\} \tag{2}$$

The goal of our matching procedure is to efficiently compute the correspondences $\{(m_k, n_k)\}_{k=1}^{K}$ between P and R, indicating the correspondence

$$\mathbf{p}_{m_k} \leftrightarrow \mathbf{r}_{n_k} \tag{3}$$

Given these correspondences, the transformation between the pattern emitter and camera can be recovered by estimating the essential matrix E, which, for noiseless points would satisfy

$$\mathbf{p}_{m_k}^{\top} E \mathbf{r}_{n_k} = 0 \quad \forall k = 1..K. \tag{4}$$

For noisy points, a two-view bundle adjustment is computationally cheap, and yields a high-quality position estimate.

4 Finding Correspondences

The core of our system is the rapid identification of corresponding points between the reference and detected dot patterns. We require a method which can deal with perspective transformation, missing dots, and positional error in the dot detection. This section describes the process, first at a high level, and then in detail.

It is clear that if each dot were accompanied by some unique identifier, such as a barcode or distinctive colour, correspondence would be trivial. Existing techniques to implement such coding, for example temporal coding or spectral modulation, would considerably increase the complexity and cost of the emitter, and introduce additional restrictions on speed of motion or scene colour. Furthermore no such techniques can reliably distinguish the several thousands of dots that we project (to distinguish 4096 dots with a temporal code would require a 12-frame coding window). Conversely, if available, such techniques can be used in tandem with our process to reduce ambiguity, for example in the presence of multiple emitters.

As an individual dot has no distinguishing characteristics, we group dots into small spatial groups—we chose to use 5-point groups, which we call "kites". A planar five-point group yields two real-valued perspective invariants, which are quantized and used as indices into a 2D lookup table to generate putative correspondences. Each input frame provides a large number of groups, even on scenes without many large planes, and after a number of pruning and verification stages, reliable correspondences are obtained, from which a RANSAC estimation of E provides accurate pose. These stages

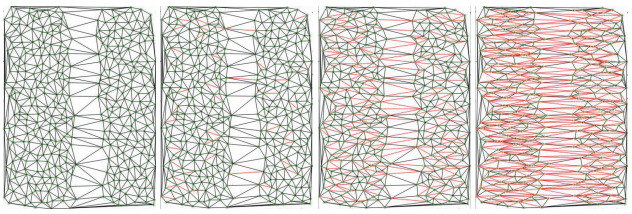

Figure 4: Stability of Delaunay triangulation. Triangulation was applied to a synthetic planar scene following a transformation (horizontal scaling) to simulate a change in viewing angle of 0° (frontal view), 30°, 60° and 80°. The views were then stretched back for comparison with the frontal view. Retained edges are shown in green, changed edges in red. Percentages in brackets show a large proportion of edges are retained up to 60° and even at 80° correct quads remain.

are now described in more detail. In some cases, the description is in terms of functions from the OpenCV library [19], in order to be precise about the computations involved.

4.1 Dot detection

Dot detection is achieved by contrast normalization followed by a level-set based connected components algorithm. Contrast is normalized by subtracting the mean in a 15×15 patch at every pixel. The zero level sets of this image are analyzed to identify closed curves, which are filtered for appropriate size and aspect ratio. The centres of mass of the remaining curves are used as point centres.

4.2 Generating quads and kites

For reliable tracking, we must extract kites from an arbitrary dot pattern such that the same kites are recovered under the nuisances described above. Delaunay triangulation is invariant to similarity transformations, and not to perspective, so at first glance appears a poor choice as a basis for a projectively invariant grouping primitive. However, as illustrated in figure 4, and as verified in our implementation, the Delaunay triangles are surprisingly stable even when the baseline between reference and test is quite severe. Thus we identify four-point groups which we call *quads* simply by grouping the points in every pair of adjacent Delaunay triangles. Then five-point groups are generated by enumerating all the points connected to a given quad point by one Delaunay edge (see figure 3). We call these neighbouring points "tail points". Notice that there are essentially two types of tail points: those connected to the quad by two edges (i.e. on adjacent triangles), and those connected by just one edge. Empirically we have found that the looser definition (i.e. including both types) provides more reliable correspondences overall.

4.3 Invariants of kites

A kite is represented by the points $\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \mathbf{q}_5\}$ where $\mathbf{q}_{1..4}$ are the quad points, and $\mathbf{q}_5$ is the tail point. To compute the two scalar invariants of the group, we determine the homography mapping the quad to the unit square, and apply that homography to $\mathbf{q}_5$, yielding $\mathbf{q}'_5$. Then the group invariants are simply the (x, y) coordinates of $\mathbf{q}'_5$. In terms of OpenCV functions, we compute

Function kiteInvariant($\{\mathbf{q}_{1..5}\}$)

$U = \{[00], [01], [11], [10]\}$;
$H =$ `getPerspectiveTransform`$(\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4\}, U)$;
$\mathbf{q}'_5 =$ `perspectiveTransform`$(\mathbf{q}_5, H)$;
return $\mathbf{q}'_5$;

It will be noticed that this calculation assumes that the order in which the quad vertices were listed is always the same, when in fact there is a fourfold symmetry. There are a number of strategies to overcome this. The simplest is to try all four rotations, and generate invariants for each. This will increase the overall number of correspondences by a factor of four, but will in general increase the number of correct correspondences, improving the RANSAC estimate, especially at highly oblique angles. This oversampling can be done at runtime or at calibration time, when the lookup table is built. Doing so at runtime increases the cost of this phase, but reduces ambiguity, whereas entering all four hypotheses into the LUT is effective only if the LUT is sparse. As we have just 2D invariants, and we must allow for noise on the values, the runtime approach is preferred.

4.4 Correspondence voting

At this stage of runtime, the M test points are represented by a set of quads, with each quad further generating a set of kites. At calibration time, the reference points have also been divided into quads and kites, so finding correspondences between the point sets is equivalent to finding correspondences between the quads. The LUT built at calibration time is a 2D array where each cell maps from a 2D kite invariant to a histogram of quad indices, stored as a list of (reference quad, weight) pairs, where *weight* represents the number of times the reference quad was added to that cell at training time. We defer the description of the LUT's construction to section 5.2, and describe here only how it is used.

Algorithm 1: Putative correspondence generation (runtime)

Initialize *Correspondences* $= \{\}$;
foreach Q *in test quads* **do**
  Initialize Votes$[Q'] = 0\ \forall Q'$ in reference quads;
  foreach *one-connected neighbour* $\mathbf{q}_5$ *of* Q **do**
    $\mathbf{q}'_5 =$ `kiteInvariant`$(Q \cup \{\mathbf{q}_5\})$;
    foreach *pair* (Q', w) *in LUT[Quantize*$(\mathbf{q}'_5)$*]* **do**
      Votes$[Q']$+=w;
  $Q' = \text{argmax}_q \text{Votes}[q]$;
  $H =$`getPerspectiveTransform`(Q, Q');
  $I =$ inlier (test, reference) correspondences to H;
  if $|I| > 50$ **then**
    Correspondences = Correspondences $\cup\ I$;

Algorithm 1 describes the process that generates the list of putative correspondences. Each quad accumulates votes from its kites to generate a most likely corresponding quad in the reference image. From this *quad* correspondence, a homography is computed, and then region growing on the points generates a list of *point* correspondences. If this list is sufficiently large (typically we require 50 correspondences), the correspondences are added to the global putative correspondence list.

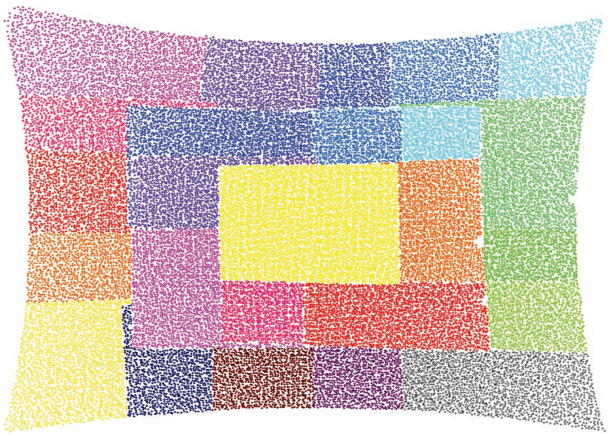

Figure 5: The emitter reference pattern. The calibration method recovers the emitter reference pattern from multiple views by a two-stage bundle adjustment process.

4.5 From correspondences to pose

In order to estimate the emitter pose relative to the camera (up to scale) for an unknown scene, we first estimate the essential matrix from the point correspondences using a RANSAC scheme and Nister's 5-point pose algorithm [18]. If the scene is planar (assumed known at calibration time), we instead estimate a homography, and compute the pose, although it is recommended that at least 2 planes are visible. In the case of a static scene, it is possible to coarsely identify at calibration time which regions in the (static) camera's field of view are planar, so the RANSAC objective can be modified to ensure that representatives from all planes are included.

The RANSAC estimate of the emitter pose is used to initialise a bundle adjustment (or "gold standard" estimator [8]) for the emitter pose and scene structure relative to a world coordinate frame centred on the camera. For two views, this is quite cheap, but the cost can be reduced further by running on only a subset of the correspondences. It remains worthwhile in terms of accuracy to include this step even when only a small number of correspondences are included.

5 Calibration

The calibration of our tracker occurs in two stages. The first stage determines the camera intrinsics, the emitter reference pattern and the lookup table used to index into the emitter reference. This phase is required when a device (camera or emitter) is first introduced. A second stage determines the camera pose relative to the scene, and optionally the positions of coarse planar regions in the camera field of view that will be static during tracking. In practice, both calibration stages are done simultaneously using a procedure akin to bundle adjustment, but it is worth emphasizing that the amount of extra work required to update the calibration if the camera is repositioned is smaller than the full calibration process.

5.1 Geometric calibration

In order to estimate the emitter pose relative to a camera from the observed emitter pattern it is necessary to calibrate both the camera intrinsics and the emitter reference pattern. Auto-calibration of the system is possible from multiple images of the emitter pattern projected onto a single plane. Images of overlapping segments of

the emitter reference pattern are matched using the technique described in section 2.1. The homography estimates from the matching process are used to register the images and initialise a bundle adjustment parameterized by homographies. There is one homography corresponding to each emitter pose which maps from emitter normalized coordinates to camera normalized coordinates. Let the emitter positions be given by 3×4 projection matrices P_j for $j = 1..J$ and let the (fixed) camera position be given by P_c. The fixed world plane is represented by a 4×3 matrix M of points on the plane, and we have the freedom to set coordinates so that it is aligned with the $Z = 0$ plane, giving

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

Given M, P_c, and P_j, the j^{th} homography H_j is given by

$$H_j = P_c M (P_j M)^{-1}. \tag{6}$$

Having computed correspondences $\mathbf{z}_{jn}$, we can now estimate the homographies and the reference points $\mathbf{r}_n$, by minimizing the bundle-like objective

$$E(H_1, ..., H_J, \mathbf{r}_1, ...\mathbf{r}_N, f, c_x, c_y, \kappa) = \sum_j \sum_n \left\| \mathbf{z}_{jn} - \pi \left(K \phi \left(f \left(\pi \left(H_j \phi \left(\mathbf{r}_n \right) \right) \right) \right) \right) \right\|, \tag{7}$$

where

$$\pi \begin{pmatrix} u \\ v \\ w \end{pmatrix} := \begin{pmatrix} u/w \\ v/w \end{pmatrix}, \tag{8}$$

$$\phi \begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{9}$$

$$K := \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{10}$$

and

$$f \begin{pmatrix} p \\ q \end{pmatrix} := \begin{pmatrix} p \\ q \end{pmatrix} \left(1 + \kappa \left(p^2 + q^2 \right) \right). \tag{11}$$

The initial estimates of H_j and $\mathbf{r}_n$ are obtained by using standard panoramic stitching, choosing the first emitter position to be roughly fronto-parallel to the plane. This does not affect the accuracy of the final converged positions, as the overall gauge is constrained by the camera intrinsics. The intrinsics are initialised with standard values $(f = 1, c_x = c_y = 0, \kappa = 0)$.

The refined homographies are then decomposed [22] to provide an initial rotation and translation (up to scale) to initialise a second bundle adjustment process parameterized by camera and emitter 6-DoF pose. A two stage bundle adjustment is adopted as the homography bundle stage is less strongly nonlinear than the Euclidean bundle that follows and therefore widens the basin of convergence. One rotation and translation is required for the fixed camera $P_c = [R_c, \mathbf{t}_c]$ and an additional rotation and translation for each emitter pose $P_j = [R_j, \mathbf{t}_j]$. The second bundle is summarized by the minimization

$$\min \sum_j \sum_n \left\| \mathbf{z}_{jn} - \pi \left(K \phi \left(f \left(\hat{x}^c_{jn} \right) \right) \right) \right\|, \tag{12}$$

where

$$\hat{x}^c_{jn} = \pi \left(P_c M \left(P_j M \right)^{-1} \phi \left(\mathbf{r}_n \right) \right). \tag{13}$$

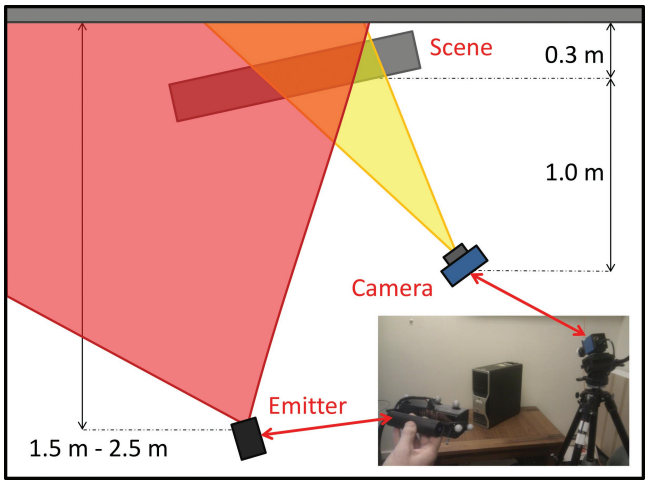

Figure 6: Experimental setup for the multi-planar scene. A fixed camera mounted on a tripod observes a multi-planar scene at a distance of 1.0 m. The scene depth is 0.3 m. The handheld emitter is in motion at a distance of 2.0 m from the scene. Only partial overlap is required between the field of view of the camera (in yellow) and the emitter beam (in red).

The emitter poses P_j are simply discarded, and system calibration parameters $(R_c, \mathbf{t}_c, c_x, c_y, f, \kappa)$ are retained along with the recovered emitter reference pattern, shown in figure 5.

In practice, a variety of engineering factors must be considered to allow this calibration to work. The camera should view the target plane obliquely, and at least five views should be generated of one section of the reference pattern, by varying the emitter pose, in order to constrain the camera intrinsics. Difficulties were caused by the repetition of the Kinect pattern, but they are too esoteric to include here—simply masking the repetitions would probably be the best way to deal with this.

Algorithm 2: Building the LUT at calibration time.

```
foreach Delaunay edge E do
    Create quad Q' from the two facets associated with E;
    repeat
        Perturb quad vertices by adding Gaussian noise;
        H = homography mapping perturbed quad vertices to
        the unit square;
        foreach one-connected neighbour P of the quad do
            Compute transformed coordinates for P' = H * P;
            LUT[quantize(P')].Add(Q);
    until #samples iterations;
```

5.2 Building the quad look-up table

At training time the neighbouring points vote for the current quad index in the appropriate bin of the quantized canonical frame. Algorithm 2 provides an outline of the method. In order to correctly model the effect of noise a number of samples are generated for each quad by adding Gaussian noise to the pixel positions of the points in image space before the canonical frame mapping takes place. The result of training is a two dimensional array in which each element contains a histogram over quad indices that voted for the position in the canonical frame corresponding to that element.

6 Experiments

The system as described above has been implemented, and we show two examples of its performance: a qualitative demonstration of its use as a 6-DoF input device, and quantitative comparisons with a Vicon tracker. Experiments run on an Intel Core i7 desktop with a 640x480 1/4" CCD monochrome camera, 6mm lens and an IR bandpass filter. The camera is mounted on a tripod and observes a planar or multi-planar scene at a distance of 1.0 m, with a plane separation of 0.3 m in the multi-planar case. The Kinect emitter is held by the user, operating at a distance of 1.5 m – 2.5 m from the scene. Figure 6 shows the experimental setup.

6.1 Emitter as user interface device

The supplementary material [15] shows a user moving the device to control a virtual light sabre (four frames are shown in figure 7). As can be observed, translational and rotational accuracy are comparable to other look-out systems, with a much smaller and simpler device. As shown in the video our system can robustly track the pointing device in real-time.

6.2 Comparison to Vicon

In figure 8 we show a 140-frame test sequence of Kinectrack compared to the Vicon tracking system, which is taken as ground truth. Translation accuracy was found to be 1.86 cm RMS error, with RMS error in rotation of 1.29°. This shows that at a greatly reduced cost, setup time, and with only a single camera, adequate tracking can be achieved when compared to a commercially available and expensive multi-camera system.

Figure 9 shows results for scene geometry restricted to a single plane. The accuracy of the pose estimate is reduced, particularly with respect to translation, but it remains sufficient for many applications involving 3D user interaction with planar scenes. The motion trajectories for the Vicon (plotted in red) when compared to the Kinectrack system (plotted in blue) are shown in figure 10 for both multi- and single-planar scene geometry.

7 Constraints and Limitations

The proposed method requires a camera to be fixed relative to the scene, which prevents the use of the system for free AR applications where the user is permitted to move anywhere. The advantage of a fixed camera is that it prevents the drift problem suffered by SLAM systems by providing absolute pose every frame, even when the scene geometry is not static. As outlined in section 5, the placement of the fixed camera is flexible and the emitter and camera intrinsics may be pre-calibrated, so that only the external camera pose remains to be determined once the camera has been repositioned.

The emitter beam must partially overlap at all times with some part of the scene in the camera's field of view. The emitter must be designed with a wide beam spread to provide sufficient range of movement (in the case of the Kinect sensor the diagonal field of view is 70°). The method is readily extended with either multiple fixed cameras or multiple emitters on the device to provide full 360° coverage. Full 360° is available in one axis using a single camera by suitable choice of emitter and camera placement. For example, projecting the dot pattern onto the floor or ceiling achieves full 360° freedom in terms of pan, with only tilt and roll restricted by the emitter beam spread. By using a dense dot pattern and only requiring a small part of that pattern to be visible in order to determine the emitter pose, the system is made robust to occlusion, non-planar scenes and varying surface albedo. The scene must contain some quasi-planar structure for the matching algorithm to succeed, but the surface need only be locally flat. Examples of sufficiently quasi-planar surfaces tested include a beach ball and a round teapot. Robustness to varying surface albedo is important as whilst the Kinetic dot pattern is clearly visible on most surfaces, some surfaces present more of a challenge, such as shiny or translucent plastics.

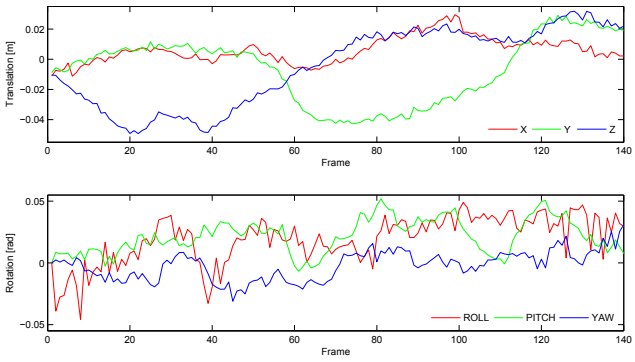

Figure 8: Comparison to Vicon with multi-planar scene geometry: Translation (top) and rotation (bottom) performance for a 140-frame sequence (RMS error is 1.86 cm in translation and 1.29° in rotation.)

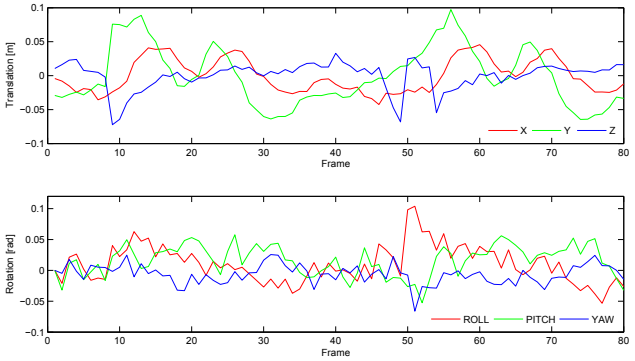

Figure 9: Comparison to Vicon with single-planar scene geometry: Translation (top) and rotation (bottom) errors for an 80-frame sequence (RMS error is 3.08 cm in translation and 1.53° in rotation.)

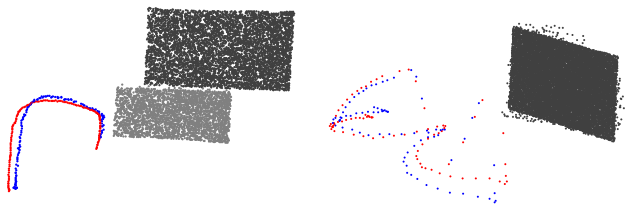

Figure 10: Qualitative comparison of Vicon (red) and Kinectrack (blue) trajectories with recovered multi-planar (left) and single-planar (right) scene geometry.

The distance of both the camera and emitter from the scene determines the density of the dot pattern in the image. If the emitter is too close or the camera too far away, the quads become too small and matching fails due to the increase in noise relative to the canonical frame. Conversely, if the emitter is too far or the camera is too close, matching fails due to insufficient dots on any one quasi-planar surface for matching. Emitter pose accuracy also degrades with distance relative to the scene as the emitter beam angle subtended by the camera image decreases. In practice, the emitter dot pattern could be chosen to achieve the optimal image dot density for the emitter and camera positions required by the application.

The use of active illumination also poses some restrictions. As

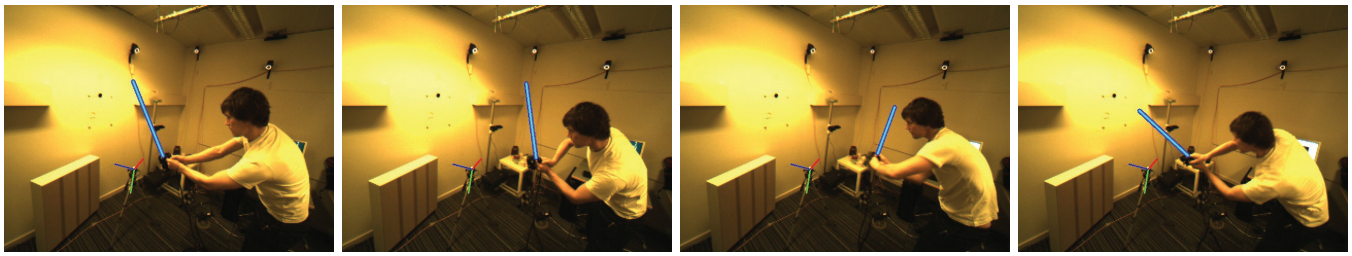

Figure 7: Example application. A user holding the emitter is augmented in an RGB camera view with a light sabre. The coordinate axes of the camera observing the projected IR dot pattern are also shown.

with other projector-based systems, outdoor daytime use is limited due to laser strength. In our implementation we use the Kinect device to generate the dot pattern which has an optimal emitter-to-scene operating range of 1.2 m to 3.0 m. Performance degrades outside this range as multiple overlapping dots appear in the image due to the laser DOE method used to form the dot pattern. An alternative dot pattern generator would enable use over other ranges.

8 Conclusions

We have introduced a new 6-DoF tracker and demonstrated a preliminary implementation, which shows that a computationally passive tracking device can provide some of the benefits of a "look out" system as exemplified by SLAM coupled with some of the benefits of "look in" systems such as Vicon. In particular, the system is low cost and low effort to deploy, is robust to different piecewise planar scene structures and works in texture-less environments and low-lighting. It provides absolute pose with good tracking accuracy and no drift over time.

Our system currently assumes a planar scene for its calibration stage, which simplifies the construction of the reference pattern, but probably reduces the calibration quality away from the target plane. It would be interesting to see how a more 3D scene affects the results. We also have the option to improve translational accuracy further in cases where the reference camera can see the device itself. Alternatively a second room camera can be used.

Overall we feel that Kinectrack provides an interesting new tracking technology for AR, it is cheap, robust to lighting, supports a variety of piecewise planar scene structures, recovers scene geometry and is easy to implement for real-time performance. We also feel that the variable baseline matching algorithm described here is general and can be used in other scenarios to find correspondence between two frames of a pseudo random dot image.

References

[1] B. Ahlborn, D. Thompson, O. Kreylos, B. Hamann, and O. G. Staadt. A practical system for laser pointer interaction on large displays. In *Proc. ACM Symp. Virtual Reality Software and Technology*, 2005.

[2] I. Albitar, P. Graebling, and C. Doignon. Robust structured light coding for 3d reconstruction. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–6. Ieee, 2007.

[3] D. Cavens, F. Vogt, S. Fels, and M. Meitner. Interacting with the big screen: pointers to ponder. *Human Performance*, page 678, 2002.

[4] L. Chan, H. Wu, H. Kao, J. Ko, H. Lin, M. Y. Chen, J. Hsu, and Y. Hung. Enabling Beyond-Surface Interactions for Interactive Surface with An Invisible Projection. *Glass*, pages 263–272, 2010.

[5] J. Davis and X. Chen. Lumipoint: multi-user laser-based interaction on large tiled displays. *Displays*, 23(5):205–211, 2002.

[6] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052, 2007.

[7] M. Fiala. ARTag, a Fiducial Marker System Using Digital Techniques. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2005.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[9] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. IEEE and ACM International Workshop on Augmented Reality*, 1999.

[10] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[11] M. Kolomenkin, S. Pollak, I. Shimshoni, and M. Lindenbaum. Geometric voting algorithm for star trackers. *IEEE Transactions On Aerospace And Electronic Systems*, 44(2):441–456, 2008.

[12] D. Lang, D. Hogg, K. Mierle, M. Blanton, and S. Roweis. Astrometry .net: Blind astrometric calibration of arbitrary astronomical images. *The Astronomical Journal*, 139:1782, 2010.

[13] M. E. Latoschik and E. Bomberg. Augmenting a Laser Pointer with a Diffraction Grating for Monoscopic 6DOF Detection. *Journal of Virtual Reality and Broadcasting*, 4(14):1860–2037, 2007.

[14] S. V. Matveyev and M. Göbel. The Optical Tweezers: Multiple-Point Interaction Technique. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, page 184, 2003.

[15] P. M. McIlroy, S. Izadi, and A. W. Fitzgibbon. Supplementary material. http://research.microsoft.com/kinectrack, 2012.

[16] T. Nakai, K. Kise, and M. Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. *Document Analysis Systems VII*, pages 541–522, 2006.

[17] NaturalPoint. Optitrack tracking. http://naturalpoint.com/optitrack.

[18] D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. IEEE Computer Vision and Pattern Recognition*, 2003.

[19] OpenCV.org. Opencv v2.4 documentation.

[20] Y. Qin, Y. Shi, and H. Jiang. Structured laser pointer: enabling wrist-rolling movements as a new interactive dimension. In *Proc. International Conference on Advanced Visual Interfaces*, 2010.

[21] T. Shiratori, H. Park, L. Sigal, Y. Sheikh, and J. Hodgins. Motion capture from body-mounted cameras. *ACM Transactions on Graphics*, 30(4):31, 2011.

[22] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. IEEE and ACM International Symposium on Augmented Reality*, 2000.

[23] H. Uchiyama and E. Marchand. Deformable random dot markers. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality*, 2011.

[24] H. Uchiyama and H. Saito. Random dot markers. In *IEEE Virtual Reality Conference*, 2011.

[25] Vicon.com. Vicon. http://www.vicon.com/.

[26] C. Wienss, I. Nikitin, G. Goebbels, and K. Troche. Sceptre: an infrared laser tracking system for virtual environments. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, 2006.

[27] O. Williams and A. Fitzgibbon. Optical pointing device. US Patent Application US2009/0237356 A1, Sep. 2009.

[28] K. Willis, I. Poupyrev, S. Hudson, and M. Mahler. SideBySide: ad-hoc multi-user interaction with handheld projectors. In *Proc. ACM Symposium on User Interface Software and Technology*, 2011.

[29] D. Wormell, E. Foxlin, and P. Katzman. Advanced inertial-optical tracking system for wide area mixed and augmented reality systems. In *Proc. Eurographics Workshop on Virtual Environments*, 2007.