Ninja on a Plane: Automatic Discovery of Physical Planes for Augmented Reality Using Visual SLAM

Denis Chekhlov* Andrew P. Gee Andrew Calway Walterio Mayol-Cuevas

Department of Computer Science, University of Bristol
Woodland Road BS8 1UB, UK

ABSTRACT

Most work in visual augmented reality (AR) employs predefined markers or models that simplify the algorithms needed for sensor positioning and augmentation but at the cost of imposing restrictions on the areas of operation and on interactivity. This paper presents a simple game in which an AR agent has to navigate using real planar surfaces on objects that are dynamically added to an unprepared environment. An extended Kalman filter (EKF) simultaneous localisation and mapping (SLAM) framework with automatic plane discovery is used to enable the player to interactively build a structured map of the game environment using a single, agile camera. By using SLAM, we are able to achieve real-time interactivity and maintain rigorous estimates of the system's uncertainty, which enables the effects of high quality estimates to be propagated to other features (points and planes) even if they are outside the camera's current field of view.

Index Terms: H.5.1 [Information Systems]: Multimedia Information Systems—Augmented Reality; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking, Surface Fitting

1 INTRODUCTION

For a variety of augmented reality (AR) applications, the aim is to be able to use previously unseen physical objects as the basis for the augmentations. This of course demands accurate, robust and interactive systems that can jointly position sensor and scene with little prior knowledge.

In the field of mobile robotics, the same problem has been attracting significant attention in the shape of simultaneous localisation and mapping (SLAM) [2, 3]. The need for SLAM appears when we want to concurrently estimate the surroundings while they are simultaneously being used to bootstrap the position of the mapping sensor. The key here is that algorithms for SLAM offer important advantages over other approaches such as conventional batch structure from motion. Specifically for AR, besides enabling real-time interactivity in previously unknown environments, SLAM algorithms offer a rigorous way to maintain estimates of the system's uncertainty.

The choice of sensor is also important for a truly ubiquitous AR system. While there are examples of reliance on external infrastructure, e.g. by using RF or ultrasound signals, an ideal system is capable of working everywhere. For this reason, in this research we employ a hand held camera as the sole 6-D sensor which is used to both map the environment and control an interactive game. Using a single camera is in line with future portability to the now common mobile platforms equipped with them, with the understanding that if needed, enhancing the system with other kinds of sensors is readily feasible.

*e-mail:{chekhlov,gee,andrew,wmayol}@cs.bris.ac.uk

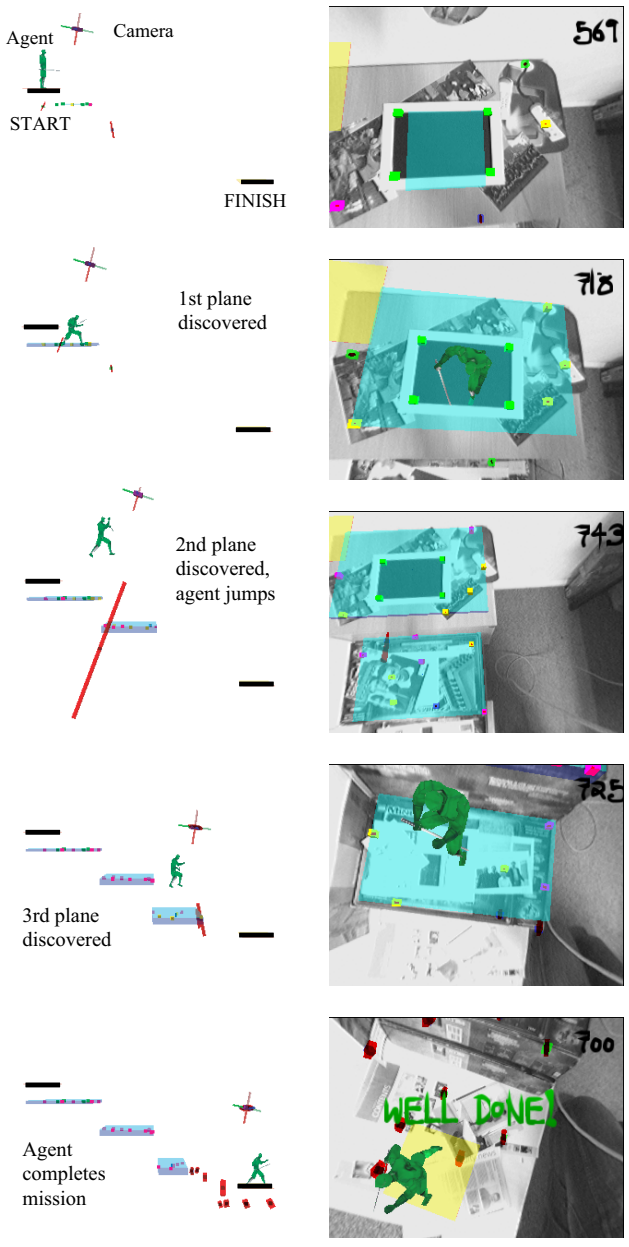

Figure 1: (left) view from the camera as it is used to build a map of previously unseen objects in real time and to discover planes on top of them. These planes are then used by an AR agent as it completes a game of jumping from object to object (right).

The rest of the paper starts by describing some related prior art, followed in section 3 by an overview of the plane discovery method using SLAM. Section 4 describes the interactive application we have developed to show the potential of the proposed system, before we finish with some discussion and conclusions.

2 To mark or not to mark

There are several recent examples of good AR systems that are accurate and robust enough to be part of an interactive system. An example is the collaborative interaction presented in [13] where ARToolkit markers are used to track objects of relevance from a mobile camera. In [5], the position of objects used to indirectly interact with a dynamic agent are known in advance so that the virtual agents can use them convincingly. These two examples have the common thread of benefiting from significant prior information. A different approach is taken in [16] where the motivation is to move away from known markers and to recover textured walls. In that work, homographies estimate a wall's orientation and position in a causal manner. This produces accurate and immediate planes for AR. However, as also noted there, SLAM is a more proper paradigm for ubiquitous AR, given that the map elements are fully correlated and thus refinement in the position of one feature leads to refinement in the position of other features.

A first attempt to exploit the benefits of vision-only SLAM for AR was presented in [9], where the emphasis was on remote collaboration and sharing annotations. That system used sparse 3-D points as the points to anchor simple static augmentations. In [6] known planar objects are recognised and added to the SLAM map to enable meaningful annotation of the environment. In this paper, we extend the use of visual SLAM to the automatic recovery of previously unknown higher level structures (planes) and, specifically, we emphasise the use of visual SLAM for interactive AR applications as demonstrated by an agent-based AR game.

3 Visual SLAM Framework

The visual SLAM framework integrates a plane discovery algorithm with a monocular EKF SLAM system in order to dynamically discover planar surfaces as they are added to a static scene, whilst maintaining full correlation of the SLAM map and camera pose. The technique is detailed in [12], where we provide thorough analysis of the consistency of the method, so we give here only a synopsis of the key elements of the system. In this paper we concentrate on the benefits of discovering planes within a SLAM framework for use in interactive AR.

3.1 EKF SLAM

The system is based on a Kalman filter, similar to that in [10, 7]. The system state, denoted $\vec{x} = [\vec{v}, \vec{m}]^T$, has two partitions: one for the camera pose, $\vec{v}$; and one for the structural components in the scene, $\vec{m}$, such as points or surfaces. A *process model* and an *observation model* encode the assumed dynamics and the relationship between the state and filter measurements, respectively [4]. We assume a constant position model for the camera motion, this gives the following process model:

$$f(\vec{x}, \vec{w}) = (\Delta\vec{q}(\vec{w}_\omega) \otimes \vec{q}, \vec{t} + \vec{w}_\tau, \vec{m}) \quad (1)$$

where $\vec{w} = (\vec{w}_\tau, \vec{w}_\omega)$ is a 6-D noise vector, assumed to be from $\vec{N}(0, Q)$, $\Delta\vec{q}(\vec{w}_\omega)$ is the incremental quaternion corresponding to the Euler angles defined by $\vec{w}_\omega$, and $\otimes$ denotes quaternion multiplication. The filter observations depend upon the structural components in the state. In the case of points, the observations are assumed to be corrupted 2-D positions of projected points.

The system begins by using only four known points on a calibration pattern, which provide an absolute scale for the constructed map. As the camera moves away from the pattern, new previously unseen feature points are initialised and added to the map, allowing tracking to continue as the camera explores new parts of the scene. Points are added using the inverse depth representation [14] and augmented to the state in a manner which maintains full correlation with the rest of the map [3]. Points are converted to a standard 3-D point representation once their uncertainty in depth becomes Gaussian [8].

For measurements, we use the FAST salient point detector [15] to detect potential features and SIFT-like descriptors with scale prediction [7] for repeatable matching of initialised features across frames.

3.2 Automatic Plane Discovery

The EKF SLAM system generates a sparse cloud of point features which is suitable for estimating the pose of the camera but does not provide high level grouping of features into structures. In order to build planes into our SLAM map, we infer planar structure within the point cloud using a RANSAC-based technique and then augment the SLAM map with parameters representing the planar surface and the points on that surface. This introduces knowledge of higher-level structure into the map and maintains correlations between the camera pose, point features and planes, which ensures consistency of the tracking and map estimate.

3.2.1 Discovering New Planes

During each frame of operation of the SLAM system, the RANSAC algorithm [11] is used to search for planes in the point cloud of the SLAM map. Plane hypotheses are generated from minimal sets of points randomly sampled from the subset of point features with variance $\sigma^2_{max} < \sigma^2_T$, where σ^2_{max} is the maximum of the variances along each dimension and σ_T is a suitably chosen threshold value, e.g. $\sigma_T = 3.0cm$. Each hypothesis is tested for consensus with the rest of the set. A point is deemed to be in consensus with the hypothesis if its perpendicular distance from the plane, d, is less than a suitably chosen threshold d_T, e.g. $d_T = 0.5cm$, and its Euclidean distance from the plane origin is less than d_{max}. The test for d_{max} ensures that we only initialise planes with strong local support. This is important because it is known that the relative positions of nearby points in a SLAM system are typically very accurate, even if the uncertainty in the global position of the point features is large, so introducing a plane through a set of local points is relatively safe. However, we can make no such assumptions about the relative positions of widely dispersed points in the SLAM map. The d_{max} threshold also helps to avoid initialisation of planes which do not correspond to physical planes in the scene.

The best-fit plane is determined from the inlying points from the plane hypothesis with most consensus. The origin is set to the mean and the orientation parameters are determined from the principal components. Specifically, if $\vec{M}$ is the $l \times 3$ matrix containing the l inlying points for the hypothesised plane, then the eigenvector of $\vec{M}^T\vec{M}$ corresponding to the smallest eigenvalue gives the normal to the plane and the other two eigenvectors give the basis vectors within the plane. The smallest eigenvalue, λ_{min}, is the variance of the inliers in the normal direction and provides a convenient measure of the quality of the fit. In order to avoid adding poor estimates of planes to the SLAM map, the best-fit plane generated by the RANSAC process is only initialised in the SLAM system if $l > l_T$ and $\lambda_{min} < \lambda_T$, where typically $l_T = 5$ points and $\lambda_T = d^2_T$.

3.2.2 Augmenting the SLAM Map with a New Plane

When a planar surface is discovered, a new 'planar component' is augmented to the SLAM map state (see Fig. 2). This new component has the form $\vec{m}_{n+1} = (\vec{p}_0, \theta_1, \phi_1, \theta_2, \phi_2)$, where $\vec{p}_0$ is the plane origin and the orientation is defined by two basis vectors, $\vec{c}(\theta_1, \phi_1)$ and $\vec{c}(\theta_2, \phi_2)$, which lie in the plane, i.e.

$$\vec{c}(\theta_i, \phi_i) = [cos\phi_i\ sin\theta_i, -sin\phi_i, cos\phi_i\ cos\theta_i]^T \quad (2)$$

The 3-D points in the map which lie on the plane and whose feature vectors $\vec{m}_i$ define their 3-D position, can then be transformed into 2-D planar points using

$$\vec{m}_i^{new} = \begin{bmatrix} (\vec{m}_i - \vec{p}_o) \cdot \vec{c}(\theta_1, \phi_1) \\ (\vec{m}_i - \vec{p}_o) \cdot \vec{c}(\theta_2, \phi_2) \end{bmatrix} \quad (3)$$

where $\cdot$ denotes the dot product. This augmentation of the SLAM map includes proper adjustment of the state covariance [12], which maintains full correlation amongst the existing and new state parameters. This process also results in a net decrease in state size if more than seven points are added to the plane.

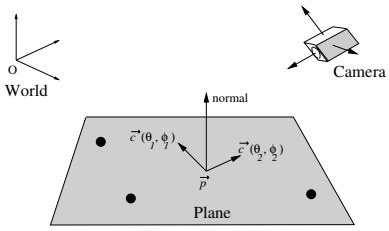

Figure 2: Coordinate systems.

It is not possible to update the SLAM system by making a direct observation of the plane in the image. However, each observed feature on the plane imposes a constraint between the camera pose, the position and orientation of the plane and the position of the observed 2-D point on the plane. The measurement model for the 2-D point on the plane is very similar to that of a standard 3-D point and simply requires an additional preliminary step to convert the 2-D point in the plane to a standard 3-D point in the world frame of reference.

The representation of the plane does not include any notion of the plane boundaries, and we don't attempt to estimate the physical plane boundaries from the image. Instead, we simply fit a bounding box around the points in the plane to define the physical extent of the plane in the augmented reality environment.

3.2.3 Adding Points to an Existing Plane

As well as looking for new planes each frame, the system also considers new 3-D points added to the SLAM map as candidates for addition to existing planes. A 3-D point is added to an existing plane if $d < d_T$ and $\sigma^2_{max} < \sigma^2_T$, where d is the distance of the 3-D point from the plane and σ^2_{max} is the maximum variance of the point.

A 3-D point which is identified to lie on a plane has its state transformed to a 2-D representation in the plane and the SLAM system covariance is updated accordingly. Since the new representation of the point in the plane is dependent on the parameters of the plane, this process introduces correlations with the other 2-D points on the plane and the plane itself.

When a new point is added to a plane, the bounding box of the plane points is recalculated, so the physical dimensions of the plane can increase and decrease as points are added and removed.

4 The "Ninja on a Plane" Game

The plane discovery algorithm described in the previous sections will be of particular interest for AR applications. In such applications, virtual objects are expected to interact with the physical objects in a scene in a realistic manner. Knowledge of the location of planar surfaces is essential for tasks such as collision detection and occlusion.

A simple game, "Ninja on a Plane", has been developed to illustrate how the technique of automatic plane discovery might be applied to an AR game. The goal of the game is to construct a sequence of descending platforms which allow the ninja agent to navigate from a starting platform, located near to the calibration pattern, down to a goal platform at a location placed somewhere below the starting position (see Fig. 1).

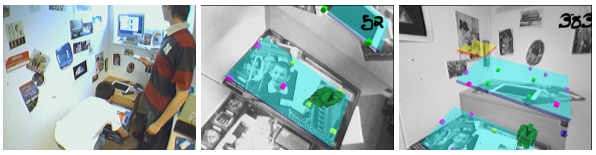

Figure 3: (left) The game allows addition of previously unseen objects (boxes to create a stair) that the user can discover interactively by moving the camera over them, allowing them to be added to the map as planes. (middle) An example of good tracking and registration during gameplay. (right) An example of poor registration when tracking is temporarily lost due to failure to match features at an acute viewpoint.

The player is responsible for constructing the platforms which the ninja will use to reach safety. The ninja is only capable of jumping between platforms which are close together, as it has a maximum horizontal distance which it can jump and a maximum vertical distance which it can safely fall. The ninja will automatically jump down to lower platforms when it is safe to do so and tries to reach the 'goal' platform.

In order to increase the excitement of the game, a time limit is set for the ninja agent to move between platforms. If the ninja remains on a platform too long, then the ninja dies if there are no other platforms which it can safely move onto within jumping distance. The ninja is also not allowed to return to a platform which it has previously visited.

4.1 Implementation

The current system is implemented on a 2.8 GHz Pentium IV desktop PC with a GeForce 4 MX graphics card. A Unibrain Fire-i digital firewire camera with a wide-angle lens with 81° horizontal field of view is used to view the scene. OGRE (Object-Oriented Graphics Rendering Engine) [1] is used for 3D graphics, and the ninja model and animations are part of this library.

The game is played in the corner of an office which has a good selection of planar surfaces to act as platforms. A selection of textured boxes are also available to be added to the scene by the player to construct new platforms. Of course, since the system does not rely on any pre-defined models, apart from the initial calibration pattern, the game could equally well be played in any environment with a sufficient number of textured surfaces to maintain tracking. Fig. 3 (left) shows a user playing the game with the described system. The player uses a graphical user interface (GUI) on the desktop PC to view the augmented scene and to perform a limited number of actions, such as starting the game.

The major mode of interaction with the game is the construction of new platforms when the player adds real, planar objects into the scene. Once a new object is added to the scene, the player is expected to move the camera around the scene and map a set of features on the object surface. This set of features allows the surface to be discovered as a new plane by the SLAM system and it is then recognised as a newly constructed platform by the game.

4.2 Results and Discussion

Although the gameplay is relatively simple, the dynamic construction of surfaces in the gameworld by physically adding unprepared objects to an unprepared augmented reality scene provides an interesting new form of interaction, which is difficult to achieve with other tracking systems. In particular, planes dynamically grow as

more features are discovered on the planar surface, so the player's control of the camera motion also provides some measure of control over the construction of the scene once new objects have been added. The game demonstrates that it is possible to add new objects to the scene at runtime and discover them, however, the current implementation assumes that the already mapped objects remain stationary. This is something that can be improved in future.

4.2.1 Performance

The game, including SLAM system updates, plane discovery, game logic and rendering, runs at a stable framerate of 15-20 fps for SLAM maps with up to 50 features. This is fast enough for the user to have real-time interaction with the game. The framerate begins to fall as the number of features grows beyond 50, due to the computational cost of updating the SLAM system each frame, which grows as $O(n^2)$, where n is the state size of the SLAM map. However, 50 features is enough to add several real planes to the scene and allows a reasonable gameplay experience. Performance improves on systems with better graphics hardware and there is also scope for running the SLAM system in a separate processing thread on a multi-core processor to obtain performance improvements.

4.2.2 Accuracy

The absolute accuracy of the camera pose has not been measured against ground truth. In general, the registration of the virtual graphics with the real scene appears good (see Fig. 3 (middle) and also companion video material[1]). Registration is poor when the camera is moved too fast, as the system may temporarily lose tracking. Loss of tracking may also occur if the camera is moved to a pose where only a few features are visible. This may occur if the camera is pointed towards a region of the scene which has not yet been mapped, or if the camera observes mapped features from a viewpoint which is very dissimilar to one from which the features were initially observed, as this makes matching of the features in the image difficult (see Fig. 3 (right)).

The normal directions and relative heights of the planar surfaces can be obtained from the SLAM map and compared with true measurements of the scene taken by hand using a tape measure. In a single test case, the position of planes close to the calibration target were found to be accurate within $\pm 3.0cm$ of their true position and their orientation was accurate to $\pm 2.0^\circ$. Planes that are far away from the calibration target will have greater uncertainty unless features in those planes are simultaneously visible with features on less uncertain planes.

Since we are working within a well-known gameplay framework, the inference of planes can be improved using some expert knowledge. For example, in the current game, we know that we are only interested in discovering horizontal planes. Therefore, we can ignore any other planes which are found and reduce the chances of adding potentially incorrect planar surfaces into the SLAM map. This is helpful because the RANSAC procedure used for plane discovery is not ideal. Typically, only a small subset of points in the map will belong to any one plane, so the data set contains a high proportion of noise and it is possible to fit planes which do not correspond to real, physical planes in the scene.

4.2.3 Recovery

The system is capable of recovery from loss of tracking. The SIFT-like descriptors [7] used for feature matching are distinctive enough to be matched after loss of tracking due to shake or occlusion, provided that the camera has not moved too far from its last known position. However, mismatches are still possible and in some cases these can cause tracking to be lost permanently if the camera pose is estimated incorrectly. One simple solution is to have a mechanism to reset the game following complete loss of tracking.

[1] See this publication's entry at www.cs.bris.ac.uk/Publications

5 Conclusions

This paper presented a method for the discovery of planar structures on previously unseen objects and described its application in an augmented reality game where objects are dynamically added to the game environment. A simultaneous localisation and mapping framework enables the system to achieve real-time interactivity and keep estimates of camera and scene uncertainty, which allows the effects of measurements to be propagated to all mapped features (points and planes) even if they are outside the camera field of view.

The automatic discovery of higher level structures, such as planes, in unprepared environments is an important step towards enabling more complex interactions between real and virtual objects in ad-hoc interactive augmented reality applications. Future work includes the possibility of being more flexible in terms of objects and structures that can be mapped and how they can be manipulated.

Acknowledgements

This work funded by ORSAS UK and the EPSRC Equator IRC.

References

[1] OGRE graphics engine, www.ogre3d.org.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I - The essential algorithms. *IEEE Robotics and Automation Magazine*, 13(2), June 2006.

[3] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II - State of the art. *IEEE Robotics and Automation Magazine*, 13(3), Sept. 2006.

[4] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li. *Estimation with Applications to Tracking and Navigation*. 2002.

[5] I. Barakonyi, M. Weilguny, T. Psik, and D. Schmalstieg. MonkeyBridge: autonomous agents in augmented reality games. In *ACM SIGCHI Int. Conf. on Advances in Computer Entertainment Technology*, 2005.

[6] R. Castle, D. Gawley, G. Klein, and D. Murray. Video-rate recognition and localization for wearable cameras. In *British Machine Vision Conf.*, 2007.

[7] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular slam using predictive multi-resolution descriptors. In *Int. Symp. on Visual Computing*, 2006.

[8] J. Civera, A. Davison, and J. Montiel. Inverse depth to depth conversion for monocular SLAM. In *Int. Conf. Robotics and Automation*, 2007.

[9] A. Davison, W. Mayol, and D. Murray. Real-time localisation and mapping with wearable active vision. In *Int. Symp. on Mixed and Augmented Reality*, 2003.

[10] A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. PAMI*, 29(6), 2007.

[11] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, 1981.

[12] A. Gee, D. Chekhlov, W. Mayol, and A. Calway. Discovering planes and collapsing the state space in visual slam. In *British Machine Vision Conf.*, 2007.

[13] A. Henrysson, M. Billinghurst, and M. Ollila. Face to face collaborative AR on mobile phones. In *Int. Symp. on Mixed and Augmented Reality*, 2005.

[14] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems Conf.*, 2006.

[15] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conf on Computer Vision*, 2006.

[16] G. Simon. Automatic online walls detection for immediate use in AR tasks. In *Int. Symp. on Mixed and Augmented Reality*, 2006.