Machine Vision and Applications
DOI 10.1007/s00138-011-0382-4

ORIGINAL PAPER

Rapidly constructed appearance models for tracking in augmented reality applications

Jeremiah Neubert · John Pretlove · Tom Drummond

Received: 17 August 2010 / Revised: 6 July 2011 / Accepted: 15 November 2011

Abstract This paper presents a method for rapidly generating crude, appearance-based edge models consisting of a set of planes. The appearance of each plane is modeled using a set of keyframes containing a list of edgels. These models are generated from a short video sequence with a few annotated frames indicating the location of the object of interest. The data requires 3–5 min to collect using a handheld device instrumented with a camera. The models produced are used with an existing edge tracking algorithm modified to select the appropriate edge keyframe and detect occlusion. A framestore is also created containing several views of the object represented as sets of point features. The framestore is used to provide an initial, rough pose estimate for initializing contour tracking. The presented system is used to create an augmented reality application to guide a user through a machine tool setup and a printer maintenance task. The models are shown to be an accurate representation of the object. Additionally, the performance of various aspects of the model making and tracking algorithms are evaluated.

Keywords Handheld augmented reality · Modeling · Visual tracking · Human–machine interaction

Thanks to ABB for their generous support of this work.

J. Neubert (✉)
Department of Mechanical Engineering, University of North Dakota, Grand Forks, USA
e-mail: jeremiah.neubert@und.edu

J. Pretlove
ABB, Billingstad, Norway

T. Drummond
Department on Engineering, Cambridge University, Cambridge, UK
e-mail: twd20@cam.ac.uk

1 Introduction

The deployment of augmented reality in large industrial plants has the potential to improve plant efficiency and safety. Technicians could get an augmented view of the plant using a networked handheld device instrumented with a camera. These augmentations would allow the plant to serve as a spacial index for accessing data and process controls currently only available on a server in the central control room. In order to fix augmentations to objects in the plant, a visual tracking system is needed. Several such systems have been proposed [1,5,14,21,24], but these methods require a 3D model, often augmented with image data. Unfortunately, acquiring the required 3D models is a daunting and expensive task. Large industrial plants are dynamic structures covering several square kilometers with more than 30,000 devices. The process of creating just one model can take hours and require several trips to the site to gather the measurements and image data. Even if 3D CAD models are available, due to the dynamic nature of the environment, they are usually out-of-date and do not yield the information needed for successful visual tracking.

Before augmented reality systems can be deployed in large industrial environments the modeling time must be greatly reduced. In this paper we propose a system for creating "crude" appearance-based models from information collected by a handheld computer instrumented with a cheap off-the-shelf camera. This will allow technicians to create models as they are needed. Despite the crude nature of the models, they contain the information required to robustly and efficiently track an object. The process of creating a model typically takes less than 4 min of user time and ten total minutes. Additionally, we propose a tracking algorithm based on the system presented in [19], which utilizes the appearance-based models.

Published online: 10 December 2011

1.1 Overview

The system presented in this work creates empirical, appearance-based models. The models represent objects as a set of planar regions. The appearance of each planar region is characterized using keyframes. The models contain all the information needed for robust, computationally efficient contour tracking and object detection. They are created from a video of the object and a small set of annotated frames indicating which planar regions are to be added.

Once the video sequence and annotated frames have been obtained, the remainder of the process is completely automated. The process begins with the reconstruction of the user-selected regions. For each of the regions, a series of keyframes—lists of edge features—are selected to model its appearance. In addition, a framestore—small set of object views represented with point features—is created for initialization. The strong descriptors associated with point features provide reliable correspondences for initial pose estimation and failure recovery. This pose estimate is used to initialize a contour tracking algorithm, based on [19], with modifications to accommodate for the "crude" representation of the object's structure. This algorithm includes occlusion detection and a mechanism for identifying the optimal edge keyframes for tracking.

The model creation and tracking system is evaluated with an augmented reality application. The application uses 3D graphics and text-based instructions to guide a user through a task. The evaluation shows that the created model is a reasonable approximation of the physical world. The experiments also demonstrate that the modeling process is robust to significant deviations from the assumed planar structure of the user-selected regions. In addition, the experiments show that the tracking algorithm is computationally efficient, capable of visual tracking at more than 40 Hz on a 1.2 GHz handheld device.

2 Background

Tracking using 3D contour models has been shown to be fast and accurate. The system presented in [5] matches the contour model to the nearest variation in image intensity. Thus it is sensitive to clutter and errors in the prior pose estimate. Reitmayr and Drummond [19] use a textured 3D CAD model rendered into a frame buffer to generate 1D feature descriptors for each edgel. The feature descriptors provide more robust localization and allow for failure detection. Alternatively, the algorithms presented in [21,24] augment contour tracking with point matching to obtain a better prior for improved robustness. The algorithm presented by Vacchetti et al. [24] uses keyframes of point features created offline, similar to the framestore presented here. These keyframes not only improve the robustness of the system, but are also used to initialize the contour tracker and recover from failure.

All the systems outlined above require a 3D model which is costly to generate. An alternative is to use a SLAM system, such as that outlined in [6], to build a model in real-time online. Typically, SLAM systems model the world as a cloud of 3D points and are computationally expensive. The system presented in [20] uses a handheld device to collect image information for a remote server running a SLAM algorithm. Augmentations are added to the point cloud by fitting planes to a small set of features in the map. Models consisting of point clouds contain no information about the world's structure so they are not capable of handling occlusions. In fact, features that become occluded are simply removed from the map. In [4,26] a point-based SLAM system is combined with a plane detection algorithm to extract structure from the point cloud. Another system presented in [18] creates a convex hull from the point cloud using Delaunay tetrahedralisation. Systems that extract structure from the 3D point cloud can only produce a model of the desired object if it is rich with features. Moreover, Ref. [18]'s requirement that the user rotate the object being modeled would be prohibitive in an industrial setting.

Alternative systems rely on human interaction to alleviate the need for a rich set of features. In [11] the user imposes structure on a 3D point cloud by annotating a video frame with a wireframe model of the desired object. The annotations must be precisely placed requiring considerable user interaction. In addition, manual image correspondence is needed if the 3D point cloud does not contain enough information to accurately reconstruct the wireframe model. The system presented in [2] allows interactive model creation from live video without the use of a SLAM system. It relies on the addition of a fiducial marker—sheet of paper—to the world to recover the camera trajectory. The model is created by manually placing points in the world using a novel mouse-like user interface instrumented with a camera. The points are used to delineate a polygon that is extruded to make a volume representative of the object. The result is a contour model for use with the tracking algorithm described in [5]. The system presented in the remainder of this paper allows the user to create a empirical object model without the need for precise annotations or manual image correspondence. Moreover, the system presented uses all available features in the selected region and the planar assumption to increase the robustness of the reconstruction process.

3 Model

The "crude" appearance-based models employed in this work are composed of a set of planar regions reconstructed up to a scale factor. The areas of the object corresponding to these

regions need only be roughly planar and deviations of several centimeters are acceptable. A set of edge keyframes is created for each planar region to capture changes in the plane's appearance due to unmodeled details in the object's 3D structure, variation in scale, and differences in viewing angle. The keyframes contain a list of edges to be used by the tracking algorithm. Each edgel is described by its 3D location, gradient direction, and 1D feature vector.

The appearance-based edge model is coupled with a framestore for initializing tracking and failure recovery similar to that outlined in [15]. The major difference is that the framestore images are selected autonomously to represent the minimal set required to initialize from nearly any pose contained in the image sequence. Each image in the framestore is represented by a list of point features. The list contains a 7×7 normalized image patch and 3D location for every selected feature.

4 Model creation

The objective of this work is to create the proposed models with minimal user interaction. Additionally, the process must result in a model of sufficient accuracy for stable, robust tracking. The basic outline of the process is as follows:

1. Gather image sequence
2. Annotate frames
3. Recover camera trajectory
4. For each planar region,
   (a) Reconstruct the plane
   (b) Identify keyframes
   (c) Determine neighboring frames
5. Create static framestore

Each of these steps will be described in greater detail in the remainder of this section. It is important to note that only steps 1 and 2 require user interaction. In these steps, the user collects an image sequence of the object to be modeled and annotates a small number of frames. The image sequence must contain views of the object from poses distributed throughout the desired workspace.

The user then views the image sequence on the handheld computer and pauses it when a surface to be added to the model comes into view. The user indicates to the system the location of the surface by annotating the frame as shown in Fig. 1. Adding a planar surface to the model requires an annotation in just one image. No additional user interaction is required.

The location can be specified in a couple of ways. One such method is having the user select a subset of all the pixels that represent the desired planar region $\mathbb{I}^*$. The set of selected

Fig. 1 A frame from the video used to model the lathe. The *line* shows annotations made by the user. Each *x* is the location of a mouse click. To complete the selection the user need only close the polygon by clicking on the *x* labeled with a *1*, the location of the first click

pixels $\mathbb{I}^s \subseteq \mathbb{I}^*$ could be as small as a single pixel from one mouse click to a set of pixels from a user-created polygon where $|\mathbb{I}^s| \approx |\mathbb{I}^*|$. This subset can then be grown to encompass the entire plane. The benefit of specifying the plane with a subset of $\mathbb{I}^*$ is that it can be assumed that all the pixels in $\mathbb{I}^s$ are on the plane. The problem with this method is that a significant number of edge features lie on or near the boundaries of a plane which may not be included in the grown region. The method used in this work requires the user to create a polygon around the plane using a series of mouse clicks as shown in Fig. 1. The polygon is constructed so that $\mathbb{I}^* \subseteq \mathbb{I}^s$, where $\|\mathbb{I}^s\| \gg \|\mathbb{I}^*\|$. Because $\mathbb{I}^s$ is a superset of $\mathbb{I}^*$ all the edge features needed for the model are contained in $\mathbb{I}^s$.

4.1 Plane reconstruction

The planar surfaces selected by the user are then reconstructed. The 3D location of the regions can be described with a four-vector $\mathbf{L} = [\mathbf{n}, d]^{\mathrm{T}}$, where $\mathbf{n}$ is the unit length three vector normal to the plane and d the distance from the origin to the plane. $\mathbf{L}$ is constructed such that if point $\mathbf{p} \in \mathcal{R}^3$ is on the plane described by $\mathbf{L}$ the following is true:

$$0 = \mathbf{n}^{\mathrm{T}}\,\mathbf{p} + d. \tag{1}$$

It is possible to use a direct approach to recover the plane equation similar to that of [12], but these methods are resource intensive. Moreover, due to the crude nature of the model the improved accuracy in unlikely to be of benefit.

We use a multi-step process to estimate the plane equation $\mathbf{L}$ of the user-selected regions. The process begins with the recovery of the camera's trajectory using a modern structure-from-motion technique, such as those outlined in [7,16]. While structure-from-motion techniques produce a 3D cloud of points, these points are not used in the reconstruction of the plane because there is no guarantee the planar region contains the points needed to obtain an accurate estimate

of **L**. The system could be modified to ensure the needed points are present in the map, but this results in degradation in the recovered camera trajectory. Instead, the system uses the camera trajectory to aid in feature matching and reconstruction so that **L** can be recovered. Additionally, the system is constructed to be robust to outliers, eliminating the need for precise placement of annotations.

4.1.1 Feature selection

To ensure an accurate estimate of **L**, an evenly distributed subset of unique features are selected from $\mathbb{I}^s$. The process begins with the detection of FAST features [22] in the user-selected region of frame r denoted $\mathbb{I}^s$. At the location of each feature a normalized image patch is extracted for use as a feature descriptor. The patch size was chosen through experimentation with image sizes of 640×480, 320×240, and 160×120. The optimal patch size in each case was found to be 7×7. Larger feature patches are more descriptive, but lead to false negatives. The chosen patch size is descriptive enough for robust matching, but also general enough to be tolerant to small rotations and changes in scale.

The next step is to ensure the descriptor is unique. A descriptor that appears multiple times in the image cannot be relocated accurately. This problem is prevalent in industrial environments which have a large number of manufactured objects containing repeating textures. One method of handling weak features is to calculate the covariance in localization from the distribution of possible matches and incorporate them into the solution. Unfortunately, an anecdotal investigation suggested that the contribution from such features, even when weighting the information by the inverse covariance, is often misleading and will skew the solution; thus it is better to neglect them. Weak features are identified by searching the surrounding pixels for a matching descriptor. If the descriptor is matched to a point more than two pixels from its original location, it is rejected. The remaining features, $\mathbb{F}$, are considered unique.

Another phenomenon that can lead to errors in reconstruction of the plane is weighting a region too heavily by selecting a disproportionate number of features from it. To avoid this a set of features, $\mathbb{F}_s$, is selected from all unique features, $\mathbb{F}$, so that

$$\forall f_i,\ f_j \in \mathbb{F}_s \ \mathrm{dist}(f_i,\ f_j) > \lambda, \tag{2}$$

where $i \neq j$ and $\mathrm{dist}(f_i,\ f_j)$ is the distance between f_i and f_j in pixels. The value of λ was determined experimentally—for this work $\lambda = 10$.

4.1.2 Feature reconstruction

It is assumed that a 2D feature, f_i, corresponds to a 3D point $\mathbf{p}_{f_i} = [x_{f_i}\ y_{f_i}\ z_{f_i}]^{\mathrm{T}}$. The vector $\mathbf{p}_{f_i} \in \mathcal{R}^3$ denotes the 3D position of feature i relative to the camera frame. By observing f_i in multiple frames, the value of $\mathbf{p}_{f_i}$ can be estimated. Unfortunately, the error in this reconstruction is not linear in depth and has a non-zero mean distribution. An unbiased estimate of $\mathbf{p}_{f_i}$ from image observations can be obtained using an unscented transform [13]. An alternative, and the solution employed here, is to form a linear solution using inverse depth coordinates as shown in [7]. This formulation has an approximately Gaussian error distribution with a mean of zero in all dimensions. A point, $\mathbf{p}_{f_i}$, can be expressed in inverse depth coordinates as

$$\mathbf{b}_{f_i} = [u_{f_i}\ v_{f_i}\ q_{f_i}]^{\mathrm{T}}, \tag{3}$$

where u_{f_i} and v_{f_i} are the normalized camera coordinates of f_i. The normalized camera coordinates $\mathbf{v}_{f_i} = [u_{f_i}\ v_{f_i}]^{\mathrm{T}}$ can also be described as the projection of the point onto the image plane

$$\mathbf{v}_{f_i} = \begin{bmatrix} \frac{x_{f_i}}{z_{f_i}} & \frac{y_{f_i}}{z_{f_i}} \end{bmatrix}^{\mathrm{T}} = \mathrm{proj}\left(\mathbf{p}_{f_i}\right) \ \text{ and } \ q_{f_i} = \frac{1}{z_{f_i}}. \tag{4}$$

If the relationship between $\mathbf{p}'_{f_i}$ and $\mathbf{p}_{f_i}$ can be described using a rigid transformation, it can be expressed as

$$\mathbf{p}_{f_i} = \mathbf{R}\mathbf{p}'_{f_i} + \mathbf{t}, \tag{5}$$

where the rotation $\mathbf{R} \in SO(3)$ and the translation $\mathbf{t} \in \mathcal{R}^3$. By multiplying (5) with q'_{f_i} it can be seen that a rigid transformation in inverse depth coordinates can be described as

$$\mathbf{b}_{f_i} = \mathbf{R} \begin{bmatrix} u'_{f_i} \\ v'_{f_i} \\ 1 \end{bmatrix} + q'_{f_i}\, \mathbf{t}. \tag{6}$$

Using this notation, a method to obtain an estimate of the inverse depth coordinates of a feature from observations made with a camera and its motion is presented. The process begins by obtaining an initial estimate from two observations. The feature was originally extracted in frame r, the user-annotated frame. Thus, one of the observations used in the initial estimate will be from frame r. The second observation can come from any frame that does not lead to a degenerate solution, but for simplicity of notation it is assumed that the second observation comes from the subsequent frame $r + 1$. The location of the point with respect to the camera location at frame r will be estimated from these observations.

The observations from frame r and $r + 1$ are obtained in pixel coordinates $\mathbf{w}_{f_i}(r)$ and $\mathbf{w}_{f_i}(r + 1)$. To obtain the normalized camera coordinates corresponding to the observations, the camera projection process needs to be inverted. In this work we assume a calibrated quintic camera model

$$\mathbf{w} = \mathrm{CamProj}(\mathbf{p}_{f_i}) = \begin{bmatrix} k_u & 0 & o_u \\ 0 & k_v & o_v \end{bmatrix} \begin{bmatrix} \gamma' u_{f_i} \\ \gamma' v_{f_i} \\ 1 \end{bmatrix}, \tag{7}$$

where $r = \sqrt{\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2}$ and $\gamma' = 1 + \alpha r^2 + \beta r^4$ provide the correction for radial lens distortion. The parameters k_u and k_v are the focal length of the camera and $[o_u, o_v]$ is the location of the projection of the optical center in the image. Using the inverted camera model the normalized camera coordinates $\mathbf{v}_{f_i}(r)$ and $\mathbf{v}_{f_i}(r+1)$ can be calculated from the observations. The covariance of these observations are $\mathbf{C}_{f_i}(r)$ and $\mathbf{C}_{f_i}(r+1)$, respectively, which are calculated by differentiating the inverse camera model and multiplying it with the expected measurement noise.

Given $\mathbf{v}_{f_i}(r)$, $\mathbf{v}_{f_i}(r+1)$ and their expected covariance, the optimum estimate of the feature's inverse depth coordinates minimizes

$$\mathbf{e}(r)^{\mathrm{T}}\, \mathbf{C}_{f_i}^{-1}(r)\, \mathbf{e}(r) \;+\; \mathbf{e}(r+1)^{\mathrm{T}}\, \mathbf{C}_{f_i}^{-1}(r+1)\, \mathbf{e}(r+1). \quad (8)$$

The residual error $\mathbf{e}$ is the difference between the observation $\mathbf{v}$ and the projection of the feature's inverse depth coordinates into normalized camera coordinates, given by

$$\mathbf{e} = \mathrm{ProjErr}\,(\mathbf{v}, \mathbf{b}, \mathbf{R}, \mathbf{t}) = \mathbf{v} - \mathrm{proj}\left(\mathbf{R}\begin{bmatrix} u_{f_i} \\ v_{f_i} \\ 1 \end{bmatrix} + q_{f_i}\mathbf{t}\right), \quad (9)$$

where $\mathbf{R}$ and $\mathbf{t}$ describes the camera pose at the time of the observation with respect to the camera at frame r. Using this equation, an iterative solution can be found to obtain the inverse depth coordinates, $\mathbf{b}_{f_i}$, that minimize (8). Additional observations of the feature are incorporated in the estimate using the extended Kalman filter framework. In our implementation the innovation is defined by (9), which is mapped to changes in $\mathbf{b}_{f_i}$ by differentiating (9) with respect to u, v, and q. The process for obtaining $\mathbf{b}_{f_i}$ is outlined in greater detail in [17].

4.1.3 Recovering the equation of the plane

Once a sufficient number of points have been reconstructed accurately, the equation of the plane can be estimated. To avoid having to convert the point locations from inverse depth to Cartesian coordinates, a solution for estimating $\mathbf{L}$ from inverse depth coordinates is found by manipulating (1). Noting that a point on the plane can be expressed as $\mathbf{p} = [x\ y\ z\ 1]^{\mathrm{T}}$ in projective three space $\mathcal{P}^3$, (1) becomes $\mathbf{L}^{\mathrm{T}}\,\mathbf{p} = 0$. Taking advantage of the fact that $a * \mathbf{p} \equiv \mathbf{p}$ in $\mathcal{P}^3$ and $a \in \mathcal{R}$, the following is true:

$$\mathbf{p} \equiv z^{-1}\ [x\ y\ z\ 1]^{\mathrm{T}} = [u\ v\ 1\ q]\,. \quad (10)$$

This allows the equation of the plane to be calculated directly from the inverse depth coordinates. The method presented in [9] is used to get an initial estimate of $\mathbf{L}$ and identify the inliers. The final estimate of $\mathbf{L}$ is obtained from the inliers using weighted least squares.

4.2 Edgel extraction

The user annotations are projected into other images in the sequence using the recovered plane equations so pixels belonging to the selected region can be identified. Images that contain a portion of the user-specified region viewed from the side annotated by the user in frame r are converted to monochrome images, blurred using a Gaussian kernel with a σ of 1.5 and processed with a standard edge detector such as Canny [3]. A list of detected edgels contained in the user-specified polygon are stored in what is referred to as an edge frame, E_i, where $E_i = \{e_{i1}, \ldots, e_{in}\}$ is the list of visible edge features found in frame i. The edge frames are compiled into the set $\mathbb{E}$.

Because the complete 3D structure of the object is unknown, there is no method to detect self-occlusion. Edgels generated by structure that are not on the plane are identified and removed by analyzing the edgel's motion. Edgels on or near the plane can be mapped to neighboring frames using a planar homography. Given that the motion of the camera between two images, i and j, is described by the homogeneous transformation $\mathbf{T}_{ji}$, the equation of the plane with respect to the camera at i can be calculated as

$$\mathbf{L}(i)^{\mathrm{T}} = \mathbf{L}(j)^{\mathrm{T}} * \mathbf{T}_{ji}. \quad (11)$$

Using the camera motion and the equation of the plane the planar homography is

$$\mathbf{H} = \mathbf{R}_{ji} - \mathbf{t}_{ji}\frac{\mathbf{n}_{L(i)}^{\mathrm{T}}}{d_{L(i)}}. \quad (12)$$

The homography is used to map edgels from E_i to edgels in the g frames before and after it, where g is a relatively small number. An edgel, $e_{il} \in E_i$, and its covariance are projected into the frames in the interval $[E_{i-g}, E_{i+g}]$ to locate matches. An edgel matches e_{il} if it is within 3σ of e_{il}'s predicted position and the difference in gradient direction is less than a user-defined threshold. If the number of failed match attempts for e_{il} indicates with more than 95% confidence that the edgel is not on the plane it is removed from E_i. Using $g = 3$, an edgel that is not in two or more sets has less than a 5% chance of being on the plane and is removed.

4.3 Edge keyframe creation

The set of edge frames, $\mathbb{E}$, contains every edgel that appears on the plane over the entire image sequence. A small subset of these frames are selected for use as keyframes. There is a large body of work on selecting a minimal set of keyframes to represent a larger image sequence. The methods vary significantly depending on the information being preserved by the keyframes as seen in [8,23]. Here the keyframes are selected to capture changes in the visible edgels and their 1D descrip-

tors as a function of camera pose. The selection process finds a minimal number of keyframes, $\mathbb{E}_{kf} \subset \mathbb{E}$, which provide a good approximation of $\mathbb{E}$. Simply stated, for any frame $E_i \in \mathbb{E}$ there should be a similar frame in $\mathbb{E}_{kf}$. This allows $\mathbb{E}_{kf}$ to serve the same function as the textured model did in [19], eliminating the computationally expensive rendering task and the need for a detailed 3D model.

The initial frame selected for addition to $\mathbb{E}_{kf}$ is the edge frame that satisfies the following

$$\arg\max_{E_i \in \mathbb{E}} (|E_i|) \tag{13}$$

to ensure that the frame with the maximal amount of information is in $\mathbb{E}_{kf}$. The remaining edge frames are evaluated using (14). If the amount of information shared between E_i and $\mathbb{E}_{kf}$ is below a specified threshold—0.7 in this work—the frame is added to $\mathbb{E}_{kf}$. Assuming E_j is the initial frame added to $\mathbb{E}_{kf}$, the next frame to be evaluated for addition to $\mathbb{E}_{kf}$ minimizes $||i - j||$. The frame with the lower index will be selected if two of the remaining frames are equidistant from E_j. The order of evaluation was created to leverage the fact that the appearance of the plane should vary smoothly over the image sequence; thus this will maximize the difference between keyframes, minimizing duplicate information in $\mathbb{E}_{kf}$.

The amount of information shared between a frame E_i and $\mathbb{E}_{kf}$ is defined as

$$best\_score(E_i) = \arg\max_{E_k \in \mathbb{E}_{kf}} (\text{sim\_score}(E_i, E_k)), \tag{14}$$

where the function

$$\text{sim\_score}(E_i, E_j) = \frac{|E_i \underset{\mathbf{H}}{\rightarrow} E_j| + |E_j \underset{\mathbf{H}}{\rightarrow} E_i|}{|E_i| + |E_j|}. \tag{15}$$

is the number of uniquely shared edgels normalized by the total number of edgels in the frames. The heuristic sim_score$(\cdot)$ reflects the information shared by the frames. It relies on an approximation to the intersection of the edge frames because when mapping pixels from one image to another multiple pixels in one frame may correspond to one in the corresponding image. Thus we define the intersection of E_i with E_j as $E_i \underset{\mathbf{H}}{\rightarrow} E_j$—the set of all the edgels in E_i that can be mapped via the planar homography $\mathbf{H}$ to a unique edgel in E_j where the angle between the gradient normals is less than a user-defined threshold.

The final step in the creation of $\mathbb{E}_{kf}$ is calculating the edgel's 3D locations required for tracking. The 3D locations $\forall e_{il} \in E_i$ can be obtained by determining the distance from the camera to the plane $\mathbf{L}(i)$ along the line defined by the optical center of the camera and the normalized camera coordinates of e_{il}. This distance corresponds to the point's depth. The function used to recover the depth of the point is

$$\text{depth}(\mathbf{v}_{e_{il}}) = \frac{-d_i}{u_{e_{il}} * n_{i1} + v_{e_{il}} * n_{i2} + n_{i3}}, \tag{16}$$

where $\mathbf{L}(i)$ can be obtained using (11) and its normal component is $\mathbf{n}_i = [n_{i1}\ n_{i2}\ n_{i3}]^{\mathrm{T}}$. Then given the definition of normalized camera coordinates in (4) the 3D position of e_{il} is calculated as $\mathbf{p}_{e_{il}} = \text{depth}(\mathbf{v}_{e_{il}}) * [u_{e_{il}}\ v_{e_{il}}\ 1.0]^{\mathrm{T}}$.

4.4 Neighbor selection

Once the construction of $\mathbb{E}_{kf}$ is completed, the neighbors of each keyframe are found so that the tracking system (Sect. 5.1) can traverse $\mathbb{E}_{kf}$ more efficiently. For each edge list, $E_i \in \mathbb{E}_{kf}$, the set of neighbors $\mathbb{E}_{n\_i} \subset \mathbb{E}_{kf}$ is defined as the set that maximizes

$$\sum_{\forall\ E_j\ \in\ \mathbb{E}_{n\_i}} \text{sim\_score}(E_i, E_j), \tag{17}$$

where $|\mathbb{E}_{n\_i}| = h$. Given the assumption that the appearance of the plane varies smoothly with the motion of the camera, it is likely that when the difference between the object in the current frame exceeds an acceptable level that the optimal keyframe to transition to will be in $\mathbb{E}_{n\_i}$ if h is sufficiently large.

Two examples of neighbor selection are shown in Fig. 2. The blue dashed lines show the sets of neighbors for edge keyframes 1 and 2. Keyframes near the edge of the graph have limited options for neighboring frames making dissimilar neighbors a possibility. Because of this the relationship is not bidirectional; E_j may be a neighbor of E_i, but E_i may not be a neighbor of E_j. The keyframe labeled 2 in the figure depicts how views of the object collected at poses close in Euclidean space with different viewing angles can contain significantly different images of the object.

4.5 Static framestore

Contour tracking algorithms require a prior pose estimate before tracking can begin. Previous systems relied on the user to begin tracking from a predetermined pose [5,19].

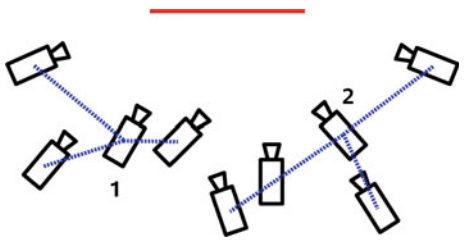


Fig. 2 The *solid horizontal line* represents a planar region. The camera pose for each keyframe is shown in the picture. The neighbors for two of the keyframes is indicated by the *dashed lines*

To eliminate the need for manual initialization and recovery, point keyframes, similar to those in [15], are used. The frames are constructed with unique point features that are tolerant to changes in scale and viewing angle. The point keyframes are only intended to provide a coarse estimate of object pose to initialize edge tracking. The requirement of only a coarse estimate allows feature extraction and matching with an image $\frac{1}{16}$th the original size. Down sampling the image will increase the uncertainty in the localization of the object, but will allow a larger portion of the image to be searched. The pose estimate, even with the increased uncertainty, provides the accuracy needed to initialize the tracker. The optimal amount to scale the image down is dependent on the application and the computational resources available.

The objective of selecting point keyframes for the static framestore $\mathbb{P}_{\text{sf}}$ is to find the minimal number of frames which guarantee that there exists a $P \in \mathbb{P}_{\text{sf}}$ such that P can be matched to the frame. The process of creating $\mathbb{P}_{\text{sf}}$ begins by extracting a list of point features, P_i, from each image i. The features are selected using the technique described in Sect. 4.1.1. The size of the point frame is further reduced by identifying and removing points not on the plane. To determine if a point feature in P_i is on the plane, it is reconstructed by projecting it onto the plane. The point's estimated 3D coordinates are used to locate it in the g frames before and after i. A feature is successfully located in a frame if there is a point within three standard deviations, 3σ, of its predicted location with a sufficiently high normalized cross correlation (NCC) score to be a match. As before, if the number of failures indicates that there is more than a 95% probability that the feature is not on the plane, it is removed.

The next step is to determine the set of frames, M_i, that the remaining features in P_i can be used to locate the object of interest. A frame is added to M_i if more than 60% of P_i's features can be located in it. The desired percentage of matches was determined experimentally to limit the number of false positives to an acceptable level. It is important to note that the image patch is not warped even though this would increase the likelihood of successful matching. This is because these features will only be used during initialization of the tracking system when no prior estimate of pose is available. Thus, lists composed of features that are the robust to changes of camera pose are desirable.

The lists of matching frames are used to identify the point frames that contain the most unrepresented information so that they can be added to $\mathbb{P}_{\text{sf}}$. This is the frame i that satisfies

$$\max_i |M_i|. \tag{18}$$

Those frames contained in M_i are now represented in $\mathbb{P}_{\text{sf}}$ so they are removed from the remaining lists of matched frames,

$$\forall j \; M_j = M_j \setminus M_i. \tag{19}$$

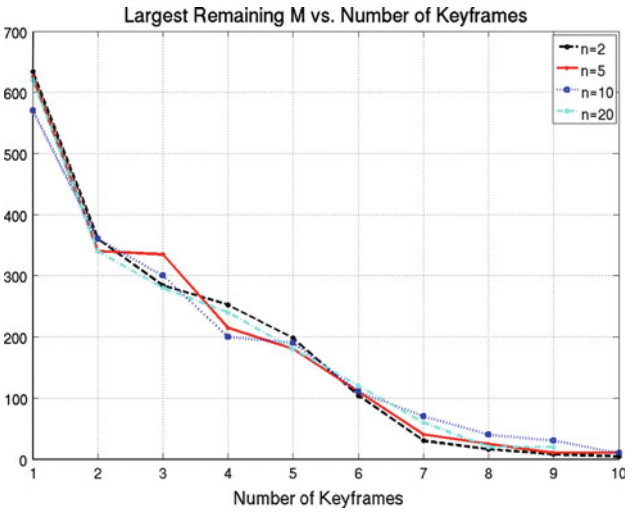


Fig. 3 The effect of n on the cardinality on $\mathbb{P}_{\text{sf}}$. The *values* on the y-axis represents the number of frames in the largest remaining M with the corresponding number of frames in $\mathbb{P}_{\text{sf}}$. The number of successfully matched frames is multiplied by n to maintain the same scale regardless of the number of frames used to find M

Removing frames that already are able to be matched to a frame in $\mathbb{P}_{\text{sf}}$ from the lists of matched frames limits the duplicate information in $\mathbb{P}_{\text{sf}}$. This process is repeated until

$$\max_i |M_i| < \eta. \tag{20}$$

The value of η needs to be large enough to prevent keyframes with little additional information from being added to $\mathbb{P}_{\text{sf}}$. A value of $\eta = 150$ is used in this work, but as shown in Fig. 3, values up to 300 will have little impact on $\mathbb{P}_{\text{sf}}$. The time required to create $\mathbb{P}_{\text{sf}}$ is greatly reduced by noting that the point frames change very little from one frame to the next. Testing with several models revealed that $\mathbb{P}_{\text{sf}}$ remained essentially unchanged when created using only every nth frame for values of $n \leq 20$ (see Fig. 3). Based on this observation, the process of selecting point frames was reduced from a couple of minutes to tens of seconds.

5 Initialization and recovery

As shown in the schematic in Fig. 4, the complete tracking system uses two framestores. The static framestore $\mathbb{P}_{\text{sf}}$, outlined above, contains the information needed to initialize tracking from nearly any pose. A small, dynamic framestore, $\mathbb{P}_{\text{df}}$, is used for quick recovery from transient failure, such as brief occlusion. $\mathbb{P}_{\text{df}}$ makes recovery from transient failure nearly unnoticeable to the user by limiting the matching process to a small number of the most recently successfully tracked frames, typically $|\mathbb{P}_{\text{df}}| \ll |\mathbb{P}_{\text{sf}}|$. This eliminates the performance loss that can occur if $\mathbb{P}_{\text{sf}}$ must be searched. A frame is added to $\mathbb{P}_{\text{df}}$ when the number of successfully

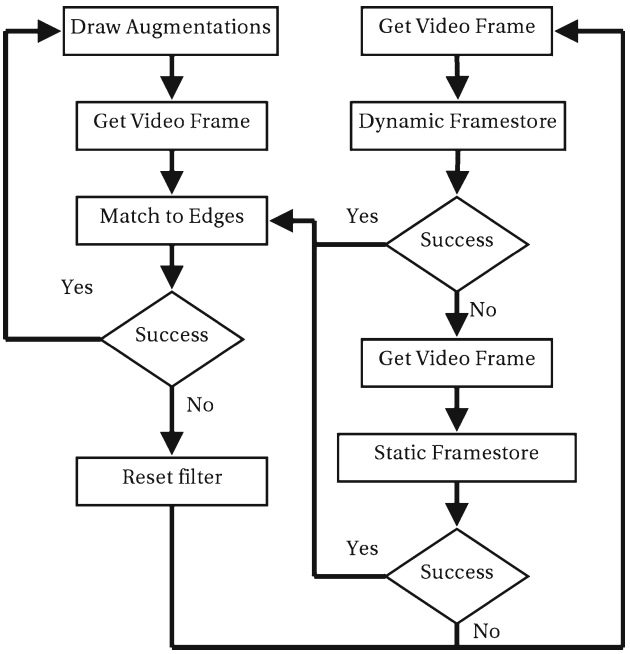


Fig. 4 A schematic of the complete tracking system. The *left side* of the schematic shows the edge tracking algorithm used when the object is visible and detected. The *right side* of the schematic is used to locate the object when edge tracking has failed

matched edgels is above a threshold and at least a second has passed since the last frame was added in a fashion similar to [19]. The point frames are composed of point features selected using the procedure presented in Sect. 4.1.1. If $\mathbb{P}_{\text{df}}$ now contains more than the predetermined number of frames, the oldest frame is removed.

To generate the most efficient method for locating the object in the image, reasonable assumptions were made about how the device is used. First, it is assumed that the rotation about the z-axis (principle axis) of the camera is near zero. This assumption seems reasonable given most industrial equipment is too heavy to be easily moved and must remain upright to function. If rotation of the handheld device about the principle axis of the camera is likely, an inertial measurement unit can be used to determine the orientation and make the appropriate accommodations. The second assumption is that a user will position an object of interest in the center of the image and not on an edge. Based on these assumptions, the set of point features is translated so that the center of their range is the center of the image. The system searches a 60 by 60 region (roughly 20% of the image) around a feature's location. A feature is successfully located in the image if a location is found where its NCC score is greater than a user-defined threshold.

The computation cost of attempting to match a frame from the framestore to the current image is significantly reduced by stopping when there is a 95% chance of failure. The probability that matching will fail is

$$P(\text{failure} \mid m, N) = \frac{P(\text{failure})\,P(m \mid \text{failure}, N)}{P(m \mid N)} \tag{21}$$

after N attempts with m successful matches. The value of $P(\text{failure} \mid m, N)$ is estimated from experimental data obtained from several image sequences. The required number of matches, m, for a given number of attempts, N, is calculated offline prior to tracking.

A frame is successfully matched to the image when $P(\text{success} \mid m, N) = 1 - P(\text{failure} \mid m, N)$ is greater than 0.5 after the matching process is complete. The 2D–3D correspondences are used to calculate the rigid transformation between the object in the current image and the matched frame using the RANSAC [9] framework. The process begins by selecting random sets of correspondences to generate possible transformations, calculating how many data points support that hypothesis, and stopping after a fixed number of iterations. The data points that support the best hypothesis are defined as the inliers. Using the inliers, a final estimate of the pose of the object is calculated for use as a prior with the edge tracking system. It is important to note that the pose will only be used if the edge tracker is able to successfully locate the object given the pose estimate.

5.1 Edge tracking

The edge tracking algorithm, developed for use with the appearance-based models described in Sect. 3, occupies the left side of the schematic shown in Fig. 4. It is an extension of the edge tracker presented in [19]. The algorithm required two modifications, outlined below, to address keyframe selection for each planar region and detecting occlusion. Detecting occlusion is particularly important because of the high probability of false edge correspondences and determining when to render augmentations associated with the plane. The measurements made by the edge tracking algorithm are incorporated into an extended Kalman filter which is used to estimate the pose of the object.

Locating edgels The first step in estimating an edgel's location requires an initial estimate of camera pose, $\mathbf{T}_{\text{cw}}^{-}$. The $-$ denotes a prior estimate. The pose estimate can be obtained from a framestore or the Kalman filter. The prior pose is used to determine which regions are visible in the image and viewed from the appropriate side. Note that $\mathbf{T}_{\text{cw}}^{-} \in SE(3)$ not only represents the camera pose but also the rigid transformation from the world to the current frame. The world frame refers to the camera location corresponding to the first video frame annotated. The initial estimate of the position of each edgel, e_i, contained in the current keyframe of visible surfaces is found by applying a rigid transformation to its 3D position calculated above using (5) to obtain $\mathbf{p}_{e_i}$. The edgel's location is found by searching along its gradient nor-

mal, $\mathbf{n}_{e_i}$, for the location that maximizes the NCC score with the extracted 1D feature descriptor. If the maximum score is below a minimum threshold, localization has failed and e_i is not used in pose estimation. A more detailed description can be found in [19].

Keyframe selection Defining a metric for selecting the keyframe that best reflects the current appearance of a planar region given an estimate of pose is difficult. Any metric proposed must reflect differences in both the viewing angle and scale [25], but it is not apparent how to create a metric that weighs the effects of viewing angle and scale appropriately. One metric proposed by Vacchetti et al. [25] attempted to capture the effects of changes in viewing angle and scale by selecting a keyframe based on the number of pixels occupied by each surface. The keyframe, k, that minimizes the sum of differences of the surface areas is defined to be the optimal frame for tracking. As pointed out in [17] and shown in Fig. 5, this method requires complicated models because images of a simple object, such as a rectangular plane, taken from significantly different viewing angles will have similar surface areas. Thus, the area metric is not suitable for determining the best keyframe on a plane-by-plane basis.

Because no apparent metric exists for selecting the appropriate keyframe using the prior estimate of pose, the quality of a keyframe's match to an image is used to determine when a transition is needed. If a preliminary matching of E_i reveals that the percentage of matched edgels is below a threshold the system searches $\mathbb{E}_{n\_i}$ for a better match. The best or optimal keyframe will successfully match the image (see Sect. 5.1) with the largest number of located edgels. This is an important departure from the method of [17] which used the keyframe with the highest percentage of matches. The problem with the previous method is that keyframe E_j, containing a foreshortened view of the region, may be constructed such that $E_j \subset E_k$. Simply selecting E_j will cause the tracking system to ignore the information offered by the additional edgels in E_k because a high percentage of E_j's edgels are located in the current frame. This phenomenon led to less than desirable tracking performance and instability at times.

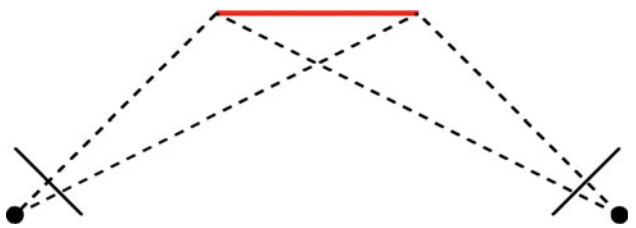


Fig. 5 The *solid horizontal line* depicts a 3D plane viewed from two positions represented as *dots*. The area of the image the plane occupies, *thin solid line* between the *dashed lines*, is the same for both camera poses. Using the area metric described in [25] the two images contain the same information and the appearance should be the same. Unfortunately, it is likely the plane's appearance is significantly different at these two poses

In addition to searching the neighbors, one randomly chosen keyframe is added to the search. The problem with limiting the search to $\mathbb{E}_{n\_i}$ is that if the optimal frame, E_o, is not in $\mathbb{E}_{n\_i}$ and E_i is a better match than any frame in $\mathbb{E}_{n\_i}$ the system will become stuck in a local minima. Adding a randomly selected frame from $\mathbb{E}_{kf} \setminus \mathbb{E}_{n\_i}$ to the search guarantees that the optimal keyframe will be found eventually.

Plane visibility and failure detection After the system has attempted to locate the edgels in the current image, the results are used to identify occluded planes based on the percentage of edgels successfully tracked. When the percentage of successfully tracked edgels in any particular planar region is below a threshold, in our system 60%, measurements obtained from that region are discarded because of an increased likelihood of false correspondences. The region is also marked as not visible and any augmentations associated with it are not rendered. Tracking has failed when none of the planar regions are visible.

A more "correct" method was proposed in [19] where the distribution of the NCC scores for successfully and unsuccessfully tracked objects are modeled. Tracking failure is detected by determining which distribution the NCC scores were most likely drawn from. Because the focus of our system is to create a method for quick, easy modeling of objects, estimating the needed distributions was rejected as too involved. The distributions change with the operating environment and the object being tracked. In the absence of this data, a threshold on the percentage of successfully tracked edgels is a reasonable approximation.

Updating camera pose Once the best keyframe is found for each visible surface, measurements are compiled for successfully located edgels. The measurement associated with e_i is the distance, d_i, from the prior estimate of e_i's location to the position along its normal, $\mathbf{n}_{e_i}$, that maximizes its NCC score. The actual position of the edgels differ from that of the prior estimate by a rigid transformation, $\Delta\mathbf{T} \in SE(3)$. Rigid transformations compose the Lie Group $SE(3)$; thus $\Delta\mathbf{T}$ can be decomposed into a six-vector, $\boldsymbol{\theta}$, using the exponential map. The goal of the system is to obtain an accurate posterior pose estimate by determining $\Delta\mathbf{T}$ in

$$\mathbf{T}_{\mathrm{cw}}^{+} = \Delta\mathbf{T}\,\mathbf{T}_{\mathrm{cw}}^{-}. \tag{22}$$

$\boldsymbol{\theta}$ can be estimated from the measurements d_i by calculating the Jacobian, which is the partial derivative of (7) with

respect to $\boldsymbol{\theta}$ multiplied with the edge normal $\mathbf{n}_{e_i}$ expressed as

$$\frac{\partial d_i}{\partial \boldsymbol{\theta}} = \mathbf{n}_{e_i}^{\mathrm{T}} \begin{bmatrix} \frac{\partial u_{e_i}}{\partial \theta_1} & \cdots & \frac{\partial u_{e_i}}{\partial \theta_6} \\ \frac{\partial v_{e_i}}{\partial \theta_1} & \cdots & \frac{\partial v_{e_i}}{\partial \theta_6} \end{bmatrix}. \tag{23}$$

The measurements d_i and their corresponding Jacobians can be stacked to form the N-vector $\mathbf{d}$ and the $N \times 6$ matrix $\mathbf{J}$ leading to the formulation

$$\mathbf{J}\boldsymbol{\theta} = \mathbf{d}, \tag{24}$$

which can be solved using an M-estimator.

6 Experimental results

The system was tested by creating the models for an office printer and a lathe. The models were used to guide a user through the process of removing a paper jam and setting up a lathe to machine a part. The instructions for each task were encoded in a simple set of augmentations with text-based instructions. Images of the augmentations used to instruct a user in the setup of the lather are shown in Fig. 6. The augmentations were added to the model by selecting a graphic and placing it on one of the planes. The arrows shown are parallel to the planes normal and can point toward or away from the plane.

6.1 Experimental setup

The data for the models presented here was collected using a VAIO UX280P with 1.2 GHz Core Solo Pentium processor instrumented with a Logitech QuickCam. The video sequences collected were 640×480 RGB images. Additionally, the tracking was done on the same system. The image sequence and annotations were transferred to a workstation with a 2.4 GHz Intel Core 2 Duo processor for the post processing needed to extract the model information. Because the device is networked, it seems reasonable to take advantage of remote computing in order to minimize overall model creation time.

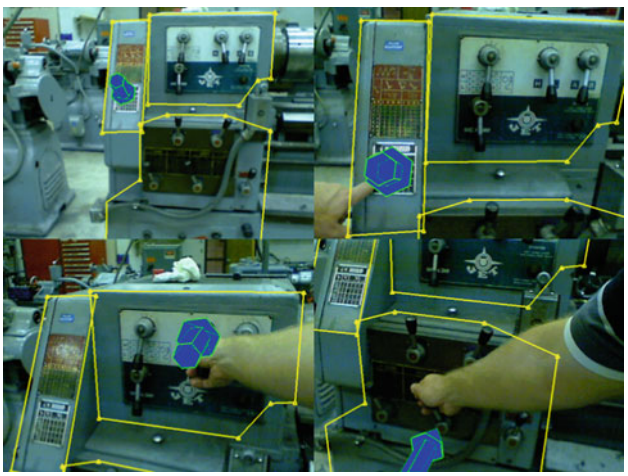


Fig. 6 These are frames from a sequence where the user is instructed to consult the table for lathe speed in the *top left*. The user is reading the table in the *top right*. The *bottom frames* depict the system directing the user to the levers that need to be adjusted to set the speed of the lathe

6.2 Modeling

One of the objectives of this work was to minimize the human interaction required to create the model. The average amount of user time required was about 2.5–3.5 min. This included the time required to create the video, typically 2–3 min. The remainder of the user time was spent selecting surfaces to add to the model. The lathe required two annotated frames and 21 mouse clicks to specify the three surfaces. The printer required three annotated frames and 18 mouse clicks.

The SLAM system outlined by [7] was used to recover the camera trajectory in this work, but any method that accurately provides the camera trajectory would be suitable. The method of [7] was selected because it was found to be robust and was made freely available to the authors. The accuracy of the camera trajectory was further improved by non-linear optimization over the entire sequence [10]. Typically, the optimization process requires less than 30 s for a video sequence of about 3 min. As long as the video sequence contained trackable features and no degenerate motions, trajectory recovery was successful. These problems can be easily avoided by adding features to the background and a small amount of user training.

The goal of the system is to facilitate the use of augmented reality in industrial environments. In order to demonstrate the ability of the system to accomplish the stated goal, an industrial lathe was modeled. The model of the lathe consisted of three planar regions and is shown in Fig. 7. The user annotations are shown in the figure as red lines with triangles. Each triangle represents the location of a mouse click. The region on the left contains a table of information required for using the machine. The other two regions contain control levers for operating the machine. It is important to note that the levers protrude from the plane more than 5 cm, but the surfaces are easily reconstructed and modeled using the current system. The creation process, including the post processing time and video collection, took 11–13 min. This is a significant reduction from the hours traditional methods require to create a detailed 3D model augmented with image information.

While the lathe demonstrates the ability of the system to model surfaces that are only approximately planar on an object similar to those found in an industrial environ-

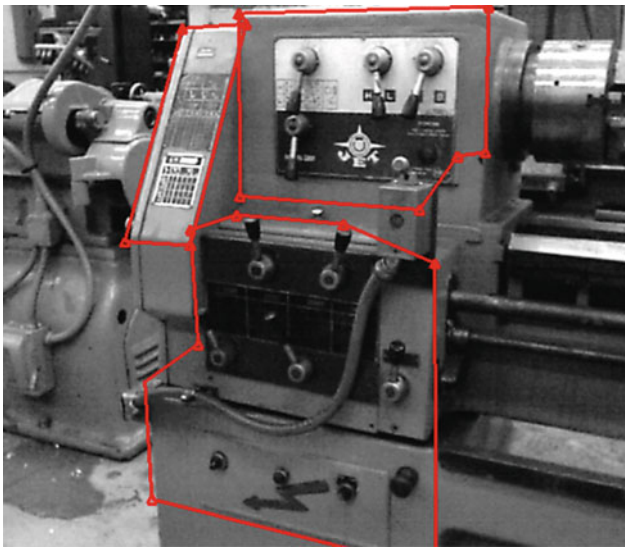


Fig. 7 The lathe model created using the semi-automated system

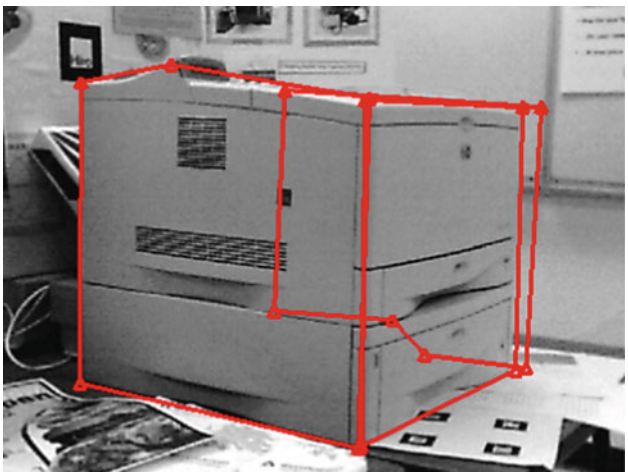


Fig. 8 The printer model created using the semi-automated system

ment, it is difficult to discern the 3D quality of the modeling. Determining the accuracy of the model is difficult because it is empirical and does not contain true boundaries of the planar regions. Moreover, the goal of this work is to construct a model for robust tracking so the importance of the dimensional accuracy of the reconstruction is not obvious. To demonstrate that the modeling technique produces a reasonable approximation of 3D structure an office printer was modeled. The cube shape offers planar surfaces with known angles between them which are not affected by the scale of the model. By evaluating the angles between the model's surfaces, the accuracy of the model can be quantified. Evaluating the model shown in Fig. 8, we found that the two parallel planes are within 0.5 ° of being normal to the center plane. Additionally, there is less than a one degree angle between the normals of the two parallel planes on the printer. This accuracy is sufficient for tracking and augmenting objects (Table 1).

Table 1 The neighbor relationships between the keyframes in Fig. 9

Keyframe	Neighbors		
42	987	958	879
183	987	879	790
673	760	879	790
760	673	790	958
790	879	760	958
879	958	790	987
958	879	987	790
987	879	42	958

6.3 Keyframe selection

The size of the set of neighboring keyframes was selected to reduce the computational load when transitioning to a new frame. Searching all keyframes for a better match is possible, but would reduce the framerate of the system. The size of the set of neighbors was selected so that only those keyframes that are likely to be transitioned to once the frame is no longer reliably matched are contained in the set of neighbors. The computational cost of increasing the set size grows linearly in time. Based on experimentation a set of three neighbors is sufficient. Evaluating the transitions during more than 12,000 tracked frames revealed that less than 5% of the transitions were to a random frame, indicating that the set size is sufficient. A small number of transitions to random frames is expected since during prolonged failure or quick motions the object pose will have changed significantly from the last successfully tracked frame.

The keyframes selected for the controls on the top of the lathe are shown in Fig. 9. They contain views from various angles around the object, ranging from the camera's leftmost pose in frame `#42` to its rightmost in frame `#673`. Given that the appearance of the planar region changes continuously as the camera moves around the object it is expected that a frame's neighbors contain views taken from a similar camera angle and distance. By evaluating the neighbors of frame `#760` it is evident that this is the case. Its neighbor `#673` represents the appearance of the plane as the camera moves to the right of its position in `#760`. Frame `#673` is the only frame taken from a farther right position than `#760`; thus the remaining two neighboring keyframes `#790` and `#958` represent how the appearance of the object changes as the camera moves to the left.

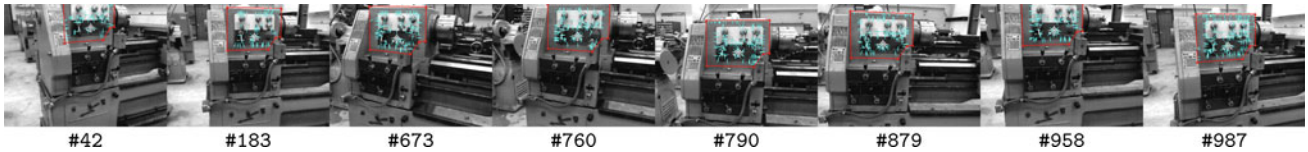


Fig. 9 The keyframes acquired to represent a planar region selected by the user in Fig. 7. The *numbers* below the images indicate the position of the frame in the original image sequence. Only a quarter of the total edgels, *dots*, found are shown. Their corresponding gradient normal is depicted with a *line*

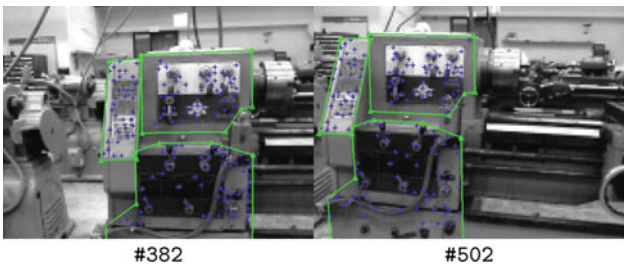


Fig. 10 The frames selected for the static framestore used in error recovery and initialization of the tracking system

6.4 Framestore

The process used to construct the static framestore attempted to find the minimal set of point keyframes which will match all the frames in the video sequence used to make the model. In the case of the sequence of the lathe, the two frames shown in Fig. 10 could be matched successfully to the 987 frames in the sequence. Frame `#382` views the object slightly from the left while `#502` contains a view from the right. From these two views the system was able to be initialized from almost any point. The static framestore was tested by taking the device to 20 random positions and starting the system. The object was successfully located in less than a second 80% of the time. The remaining 20% of the trials required a translation a few centimeters or a rotation of less than a degree—a slight jitter—to locate the object. This is likely due to quantization noise in the down sampled image.

A second test was conducted with the static and dynamic framestore to determine the system's ability to locate the object in each frame and the computational cost of using the framestore. The printer model was used in this portion of the investigation because its larger static framestore provides a more accurate view of the computational cost of initialization and recovery. The system was modified to use the framestores regardless of tracking success in the previous frame. During this test the object was successfully located more than 86.2% of the time. The dynamic framestore provided a 6.2% improvement in match rate and reduced recovery time during transient failure by as much as 60% for the printer.

Table 2 The time each process in the tracking system takes

Process	Time (ms)
Edge matching (all planes)	20.6
Searching neighbors (single plane)	3.6
No search (single plane)	1.6
Drawing augmentation	<1
Capturing and down sampling image	<1
Total time per frame	23.0
Point matching (five frames)	55.6

The times were gathered using the model of the lathe, with the exception of point matching which was done using the printer

6.5 Performance

The process of initializing or recovering using the static framestore took on average 55.6 ms (see Table 2). Before implementation of early stopping, the average was more than 100 ms depending on the number of features per point frame. The early failure detection scheme generated large variation in the computation per frame. In the rare case where a significant number of spurious matches are found, the system will attempt to match most of the points before failure is detected, taking up to 125 ms. This situation is only encountered as the object of interest comes into view and lasts less than a few frames.

The edge tracking system was able to use the crude appearance-based model to robustly track the object with minimal jitter. It is difficult without ground truth to characterize the accuracy of the tracking system, but evaluating the position of the user annotations, the yellow lines in Fig. 6, it can be seen that the annotations remained fixed relative to the object regardless of the pose. Additionally, the model was able to be used to track the object despite significant differences in viewing angle and distance from that contained in the original sequence. In Fig. 6, only the top left view looked similar to frames contained in the image sequence used to create the model. The remainder of the images shown contain views of the object at four times the scale of the views in the original sequence.

The efficiency of the tracking system has been drastically improved over the system presented [19] by eliminating the most costly step, model rendering. The time reported in

Table 2 for edgel matching reflects the average time it takes to complete two iterations of edgel matching and pose updating. This includes keyframe selection. The process of identifying a new edge keyframe for a planar surface required 3.6 ms, more than double the 1.6 ms when no search is required. This should be expected because of the increased number of edgels that are evaluated to identify a better keyframe. The current system during object tracking can achieve a frame rate of 40 Hz on the handheld device, but is limited to the framerate of the camera, 30 Hz.

7 Discussion

The objective of the project was to eliminate the costly process of creating models for visual tracking in large industrial environments. To accomplish this, a crude appearance-based model was presented. The model allows geometric details of the object to be omitted from the 3D model, opting instead to encode these details in a set of keyframes. These models required minimal mental effort and less than 4 minutes of user time to create. Most of the user time is devoted to collecting a video of the object. This is a significant improvement over traditional methods where a 3D CAD model is created and augmented with image data, a mentally demanding task requiring hours. Additionally, the model is created on-site using a handheld device with an off-the-shelf webcam. The limitation of the handheld device is that the images and annotations must be transmitted to a remote computer for post-processing. The post-processing could be done with the handheld device, but the limited memory and processor speed can lead to an unacceptable increase in post-processing time. More than a third of this time is spent selecting the edge keyframes. It is hoped that optimizing the edge keyframe selection process could result in a significant reduction in processing time.

Another important feature of the model is that it allows completely autonomous tracking. Most tracking algorithms require some form of initialization such as moving to a predetermined pose. The results demonstrate that the static framestore included in the model allows initialization from nearly any pose. The drawback to the initialization scheme was its computation cost. The impact of this was mitigated using a dynamic framestore so that transient failures can be corrected without loss in performance. If this system is to be deployed in a large industrial environments, it will have to locate and track multiple objects. The system can be made scalable by doing the initialization in a separate thread or on a remote computer, removing this computation from the tracking loop. This may result in an increase in the time required to locate an object initially, but a few seconds to detect an object would be unnoticed by most users. For recovery, the small dynamic framestore should not be removed from the tracking loop because the loss of an object's location for several seconds after a transient error would not be acceptable.

Acknowledgments The authors wish to thank ABB for their generous support of this work. Also we would like to thank Ethan Eade for supplying the SLAM software used to recover the camera trajectory.

References

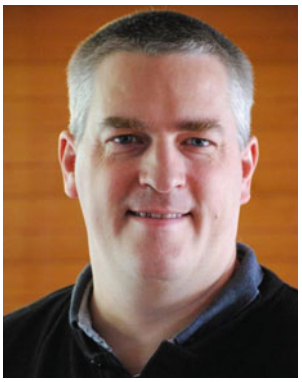
1. Bleser, G., Wuest, H., Stricker, D.: Online camera pose estimation in partially known and dynamic scenes. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality. IEEE, Santa Barbara, CA (2006)
2. Bunnun, P., Mayol-Cuevas, W.: Outlin AR: An assisted interactive model building system with reduced computational effort. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality, pp. 61–64. IEEE, Washington, DC, USA (2008). doi:10.1109/ISMAR.2008.4637325
3. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
4. Chekhlov, D., Gee, A., Calway, A., Mayol-Cuevas, W.: Ninja on a plane: automatic discovery of physical planes for augmented reality using visual SLAM. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality. IEEE, Nara, Japan (2007)
5. Drummond, T., Cipolla, R.: Real-time visual tracking of complex structures. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 932–946 (2002)
6. Eade, E., Drummond, T.: Edge landmarks in monocular slam. In: British Machine Vision Conference, vol. 1, pp. 7–16, Edinburgh (2006)
7. Eade, E., Drummond, T.: Scalable monocular slam. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 469–476. IEEE Computer Society, Washington, DC (2006)
8. Fauvet, B., Bouthemy, P., Gros, P., Spindler, F.: A geometrical key-frame selection method exploiting dominant motion estimation in video. In: International Conference on Image and Video Retrieval. Lecture Notes in Computer Science, vol. 3115, pp. 419–427. Dublin, Eire (2004)
9. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting applications to image analysis and automated cartography. In: Proceedings of the Image Understanding Workshop, pp. 71–88 (1980)
10. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Visions, 2nd edn. Cambridge University Press, Cambridge (2004)
11. Hengel, A., Dick, A., Thormahlen, T., Ward, B., Torr, P.: VideoTrace: Rapid interactive scene modelling from video. In: SIGGRAPH. ACM, New York, NY, USA (2007)
12. Jin, H., Favaro, P., Soatto, S.: A semi-direct approach to structure from motion. Vis. Comput. **19**, 377–394 (2003)
13. Julier, S., Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: International Symposium on Aerospace/Defense Sensing, Simulation and Controls (1997). http://citeseer.ist.psu.edu/julier97new.html
14. Klein, G., Murray, D.: Full-3d edge tracking with a particle filter. In: British Machine Vision Conference. BMVA, Edinburgh (2006)
15. Lepetit, V., Vacchetti, L., Thalmann, D., Fua, P.: Fully automated and stable registration for augmented reality applications. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality, pp. 93–102. IEEE, Washington, DC (2003)
16. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping. In: International Joint Conference on Artificial Intelligence. Acapulco, Mexico (2003)

17. Neubert, J., Pretlove, J., Drummond, T.: Automatic generation of appearance-based edge models from image sequences. In: IEEE Proc. of the International Symposium on Mixed and Augmented Reality, pp. 79–87. IEEE, Nara, Japan (2007). http://www.und.nodak.edu/instruct/jneubert/papers/neubert_ismar07.pdf
18. Pan, Q., Reitmayr, G., Drummond, T.: Interactive model reconstruction with user guidance. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality, pp. 209–210. IEEE Computer Society, Washington, DC (2009). doi:10.1109/ISMAR.2009.5336460
19. Reitmayr, G., Drummond, T.: Going out: Robust model-based tracking for outdoor augmented reality. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality. IEEE, Santa Barbara (2006)
20. Reitmayr, G., Eade, E., Drummond, T.: Semi-automatic annotations for remote collaboration. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality. IEEE, Nara, Japan (2007)
21. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: International Conference on Computer Vision, pp. 1508–1515. IEEE, Washington, DC (2005)
22. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Proceedings of the European Conference on Computer Vision, vol. 1, pp. 430–443 (2006). http://mi.eng.cam.ac.uk/~er258/work/rosten_2006_machine.pdf
23. Thormählen, T., Broszio, H., Weissenfeld, A.: Keyframe selection for camera motion and structure estimation from multiple views. In: Proceedings of the European Conference on Computer Vision, vol. 127, pp. 523–535 (2004). ftp://tnt.uni-hannover.de/pub/papers/2004/ECCV2004-TTHBAW.pdf
24. Vacchetti, L., Lepetit, V., Fua, P.: Combining edge and texture information for real-time accurate 3D camera tracking. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality, pp. 48–57. IEEE, Washington, DC, USA (2004)
25. Vacchetti, L., Lepetit, V., Ponder, M., Papagiannakis, G., Thalmann, D., Magnenat-Thalmann, N., Fua, P. : Stable Real-Time AR Framework for Training and Planning in Industrial Environments. In: Ong, S., Nee, A. (eds.) Virtual and Augmented Reality Applications in Manufacturing, pp. 129–146. Springer, Berlin (2004)
26. Ventura, J., Hollerer, T.: Online environment model estimation for augmented reality. In: IEEE Proceedings of the International Symposium on Mixed and Augmented Reality, pp. 103–106. IEEE (2009)

Author Biographies

Jeremiah Neubert is an Assistant Professor in the Department of Mechanical Engineering at the University of North Dakota and is the Director of the UND Robotics and Intelligent Systems Lab. Prior to joining UND, Dr. Neubert worked at Cambridge University as a Post Doctoral Researcher. Dr. Neubert earned his Ph.D from the University of Wisconsin-Madison (2005). He also received his M.S. degree in Computer Science (2003) and Mechanical Engineering (2001) from the University of Wisconsin. Dr. Neubert's research interests include machine vision, augmented reality and visually controlled robotics. He is a member of the IEEE and ASME.

John Pretlove is the Manager of Technology and Innovation and leader of Technical Product Managers in ABB's Oil and Gas Business. He is also a visiting Professor at Imperial College London and holds a visiting post at the University of Surrey, UK. Dr. Pretlove has worked for ABB for twelve years in various Management roles including R&D and innovation and has research interests in robotics, augmented reality, human-machine interaction and visualisation. He has a Ph.D in Mechatronics and Robotics and is a Fellow of the IMechE.



Tom Drummond studied mathematics at Cambridge University for his first degree before emigrating to Australia in 1990. He worked for CSIRO in Melbourne until 1994 when he went to Perth to do his Ph.D in Computer Science at Curtin University. He then returned to UK to undertake post-doctoral research in Computer Vision at Cambridge University and was appointed a permanent lectureship in 2002. In September 2010, he moved back to Melbourne and took up a professorship at Monash University. His research interests include real-time computer vision, visually guided robotics, augmented reality, robust methods and SLAM.