PTrack: Introducing a Novel Iterative Geometric Pose Estimation for a Marker-based Single Camera Tracking System

Pedro Santos[1], **Andre Stork**[2]
Fraunhofer IGD
Darmstadt,Germany

Alexandre Buaes[3]
CETA SENAI-RS
PPGEE/UFRGS
Porto Alegre, Brazil

Joaquim Jorge[4]
INESC-ID/IST
Technical University of Lisbon,
Portugal

Abstract

Inside-out tracking for augmented reality applications continues to present a challenge in terms of performance, robustness and accuracy. In particular mobile augmented reality applications are in need of tracking systems which can be wearable and do not cause a high processing load. In this paper we introduce a novel iterative geometric method for pose estimation from four co-planar points and we present the current status of PTrack, a marker-based single camera tracking system benefiting from this approach. The system uses an IDS uEye [1] camera, equipped with infrared flash strobes and infrared pass filter, which acquires grayscale images and sends them to a computer where an image pre-processing algorithm identifies potential projections of retro-reflective markers. Our novel pose estimation algorithm identifies possible labels composed of markers in a 2D post-processing using a divide-and-conquer strategy to segment the camera's image space and attempts an iterative geometric 3D reconstruction of position and orientation in camera space. In the end successfully reconstructed labels are compared to a database for identification. Their 6 DoF data as well as their ID is made available to applications through OpenTracker [2] framework. To assess the performance of our approach we compared PTrack to ARToolKit [3] - which has been adapted to work with the same camera - and final results show that pose estimation is more accurate and precise, in both translation and rotation.

Keywords: Tracking System, Augmented Reality, Virtual Reality.

[1] pedro.santos@igd.fhg.de
[2] andre.stork@igd.fhg.de
[3] agreff@uol.com.br
[4] jaj@inesc-id.pt

IEEE Virtual Reality 2006
March 25 - 29, Alexandria, Virginia, USA
1-4244-0224-7/06/$20.00 ©2006 IEEE

1. Introduction

The presented work focuses on PTrack, an innovative marker-based, single camera tracking solution developed within two master theses, in cooperation projects among the authors' institutions.

The paper explains the implementation of the relevant algorithms and presents results of a direct comparison of PTrack against ARToolKit.

The target operating scenario for PTrack are mobile augmented reality applications, such as maintenance and installation systems such as the ones developed within the ARVIKA consortium [4]. The data format, in which the results are conveyed, is compatible and already integrated within the OpenTracker framework, an abstract unified tracker interface widely used in immersive environments.

This paper will focus in more detail on the innovative aspects of the system implementation, namely the reconstruction of the position and orientation of a set of labels (object pose) from single images, based on projected 2D marker coordinates of each label on the tracking camera's plane.

2. Related Work

Computing the position and orientation of a rigid body relative to the camera from a single view can be understood as the correspondence between a geometric feature in object space and its 2D projection in image space. This in turn can be expressed by a set of equations, whose unknowns are the extrinsic parameters. In general analytical and numerical methods can be distinguished. The first are represented by Fischler and Bolles [5] which recovered the camera location by computing distances between the focus of the camera and three object space points. Adding information about the distances and angles between those points leads to a system of eight-degree polynomial equations. For three correspondences, eight solutions may be found. Dhome [6] uses line correspondences and decomposes the object-to-image space transformation in two transformations by introducing an additional plane which is the interpretation plane of one of the line segments. Also here, the result is a system of eight degree polynomial equations. As can be seen, one of the drawbacks of these methods is the multiplicity of solutions and their sensitivity to noise.

The goal of numerical approaches is to minimize an error function and thus iteratively approach the correct pose. Such algorithms are less affected by noise than analytic ones. In addition their real-time performance is adjustable by controlling the number of iterations and therefore allowing for a tradeoff between speed and accuracy, which can be useful for certain applications. One of the most prominent and accurate numerical approaches to pose estimation is the least square minimization from Lowe [7]. The points measured in image space are compared to the results of the projection of the object space points to image space and the quadratic error between them is computed for each axis. To use Newton's method, for each parameter the partial derivative of the error function is calculated. The resulting system of equations is then used to iteratively compute the extrinsic parameters. The POSIT algorithm by DeMenthon [8] is also iterative as the methods of Lowe, but does not require an initial pose estimate and matrix inversions in its iteration loop. The method finds approximate image poses by assuming image points have been obtained by scaled orthographic projections, shifting the results to the lines of sight and repeating the step.

ARToolKit developed by Kato and Billinghurst [3] takes advantage of two opposing edges of tracked square targets being parallel to each other. In object space as well as in image space those edges represent lines from one corner of the label to the other. Using the perspective projection matrix, one can define two planes by the focal point and one of these lines respectively. The cross-product of the normal vectors of both planes results in the direction vector of both lines in object space corresponding to the rotation sub-matrix of the affine transformation from image space to object space. The remaining unknowns for the translation can then be calculated iteratively. The drawbacks of numerical approaches can be conversion issues. However they are generally outweighed by less sensitivity to noise and a good speed vs. accuracy relation.

PTrack, the presented single-camera system, inserts itself into the numerical approaches. However and similar to ARToolKit, PTrack takes advantage of geometrical conditions, namely that the corners of a square label in object space and the corresponding projection in image space must lie on the same projection lines commencing in the focal point and crossing the four corners of the label. In PTrack the projection of a square target is iteratively twisted into the correct orientation by the presented algorithm and then scaled to the correct location in object space.

Although there have been several approaches to estimate pose from a projected rectangle, the authors believe this algorithm to be well suited to address this problem. The evaluation at the end of the paper and the comparison to ARToolKit show that the algorithm is fast and accurate.

3. System Setup

The PTrack test environment is made up of a computer and an IDS uEye UI1210-C camera (Figure 1) with 640x480 resolution, configured to acquire grayscale images, equipped with infrared (940nm wavelength) flash strobes, controlled by a digital trigger output of the camera. At each frame the camera captures the image of labels made of retro-reflective markers and sends the uncompressed image to the PC through a USB 2.0 interface. A pre-processing software module - Camera module – extracts the 2D coordinates of the label image feature points projected on the plane of the camera's CMOS sensor. The image pre-processing also corrects lens radial distortions, based on pre-calibrated lens distortion coefficients. The image is then passed to the tracking algorithm on a frame-by-frame basis.

The camera is equipped with 8 modules of 15 infrared LEDs each, totalizing 120 infrared LEDs around the lens. A RG-850 (850nm) daylight blocking filter was built in to minimize interference from undesired optical sources in daylight spectral region.

The labels used by PTrack are composed of 6 markers each (Figure 2). Four markers represent the corners of a square label with fixed edge length. One marker is on the top edge, identifying the top orientation of the label, splitting it in 1/3 and 2/3. And the sixth marker is a coding marker which must lie somewhere within the square formed by the other markers.

The markers used are flat circular retro-reflective markers with a diameter of 10mm. The distance between corner markers on a label is 80mm. Therefore for encoding there is a grid of 60mmx60mm which is enough for 36 different encodings. If the label is made bigger or the inter-marker distance is decreased to 5mm then more encodings can be applied.

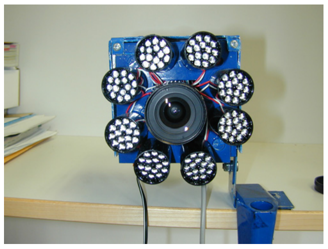

Figure 1: IDS uEye UI1210-C with IR flash strobes.

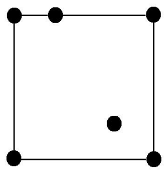

Figure 2: PTrack label with retro-reflective markers.

4. PTrack - System and Algorithm

PTrack is a static library, which can be regarded as a data transformation pipeline, receiving the image, extracting 2D image space feature points, identifying and reconstructing pose of possible labels in 3D and broadcasting this information to interested applications. Figure 3 shows PTrack's system architecture.

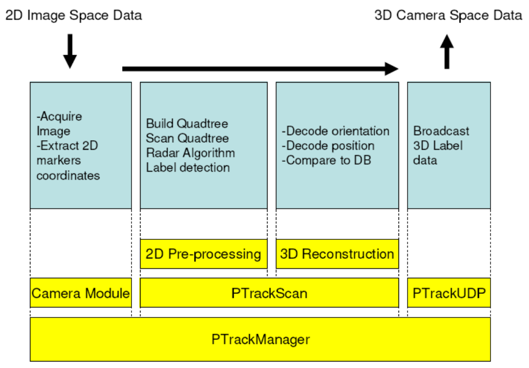

Figure 3: PTrack's system architecture.

The coordination of all activities in PTrack is done by PTrackManager, which controls data flow between the modules, which are described in more detail in the following sections.

4.1. Camera Module

The IDS Camera Module is one possible module supported by PTrack. Each 2D input system wishing to be supported by PTrack must provide its intrinsic camera parameters, a label configuration file and a continuous flow of 2D image space feature point data. The Camera Module runs in a thread of its own constantly controlling the camera drivers to grab images in the highest possible rate. Then, in a per frame basis, a global thresholding algorithm is applied, followed by an 8-connect operator to identify regions in the image. Following, geometric and size constraints are applied to those regions, so that only elliptical and round shaped regions are passed along. Then the 2D pixel coordinates of region centers are calculated with subpixel precision. After that, radial distortion is corrected and intrinsic parameters of the camera are applied, resulting in the 2D real coordinates of the markers on the camera's sensor plane. Intrinsic parameters as well as distortion coefficients are obtained by a simple calibration procedure using Intel's OpenCV library [9]. The 2D marker coordinates are then passed to pose estimation processing whenever PTrackManager polls the module. Thread-safe access is ensured through critical sections.

4.2. PTrackScan - 2D Pre-processing

Whenever a new data-set with 2D image space feature point coordinates is received from the Camera Module, PTrackManager sends it to PTrackScan which does pose estimation processing. PTrackScan in turn is split in 2D pre-processing and 3D pose estimation.

In 2D pre-processing due to computational efficiency a "divide-and-conquer" mechanism has been developed by the author, which segments the image space using a quad-tree approach. Within quad-tree nodes representing 2D regions, potential projections of 3D labels are attempted to be found which are then sent to the 3D pose estimation and after reconstruction compared to an object space database of registered labels for a match.

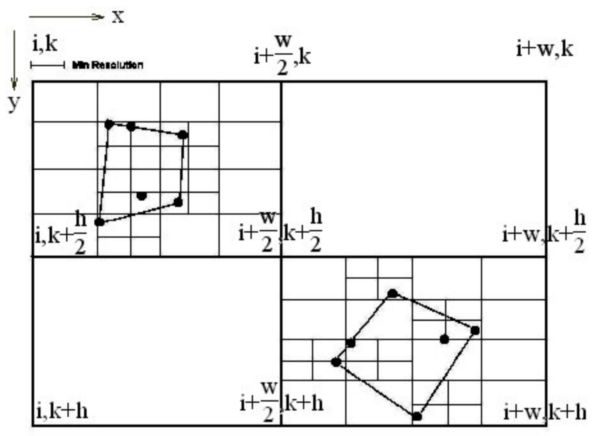

Figure 4: Quad-Tree segmentation of the image plane – quad-tree prototype

4.2.1. Build Quad-tree

A time efficient way to identify possible labels is to build a quad-tree segmenting the frame. Figure 4 shows an example where 2 labels are visible. In practice it means, that the initial image space is subdivided in four regions and each region is again subdivided in another four regions and so on. Subdivision only takes place if a region still contains image space points. Subdivision only goes down to a certain bounding box resolution which can be set previous to the use of the algorithm.

Let $\mathcal{R}_{l,n,i,j,w,h}$ be a region of the screen in image space. Let l be the level of the quad-tree and n the clockwise index of the region. i,j are the coordinates of the upper left corner. w and h are the width and height respectively. $\mathcal{R}$ is a set of two sets: the father region it is a sub-set of and the set of markers in the current region. The initial region (root-node) is the screen $\mathcal{R}$, which is not a sub-set of any other region, where Θ contains all N markers in image space: $\mathcal{R}_{0,0,0,width,height}=\{\{\},\Theta_{0,0,0,width,height}\}$. The quad-tree is recursively built as long as width and height of a region remain above the thresholds δ_w and δ_h for minimal width and height.

$$\mathrm{Quadtree}(\mathcal{R}_{l,i,j,w,h})=(w>\delta_w \wedge h>\delta_h)\Rightarrow$$

$$\mathrm{Quadtree}(\mathcal{R}_{l+1,i,j,i+\frac{w}{2},j+\frac{h}{2}} \leftarrow \{\mathcal{R}_{l,i,j,w,h},(\Theta_{l+1,i,j,i+\frac{w}{2},j+\frac{h}{2}})\}) \wedge$$

$$\mathrm{Quadtree}(\mathcal{R}_{l+1,i+\frac{w}{2},j,i+w,j+\frac{h}{2}} \leftarrow \{\mathcal{R}_{l,i,j,w,h},(\Theta_{l+1,i+\frac{w}{2},j,i+w,j+\frac{h}{2}})\}) \wedge$$

$$\mathrm{Quadtree}(\mathcal{R}_{l+1,i+\frac{w}{2},j+\frac{h}{2},i+w,j+h} \leftarrow \{\mathcal{R}_{l,i,j,w,h},(\Theta_{l+1,i+\frac{w}{2},j+\frac{h}{2},i+w,j+h})\}) \wedge$$

$$\mathrm{Quadtree}(\mathcal{R}_{l+1,i,j+\frac{h}{2},i+\frac{w}{2},j+h} \leftarrow \{\mathcal{R}_{l,i,j,w,h},(\Theta_{l+1,i,j+\frac{h}{2},i+\frac{w}{2},j+h})\})$$

where:

$$\Theta_{l+1,i,j,i+\frac{w}{2},j+\frac{h}{2}} \leftarrow \forall\, m_k \in \Theta_{l,i,j,w,h} \exists\, m_k \in \mathrm{BBox}(_{i,j,i+\frac{w}{2},j+\frac{h}{2}}), k \in [1,N]$$

$$\Theta_{l+1,i+\frac{w}{2},j,i+w,j+\frac{h}{2}} \leftarrow \forall\, m_k \in \Theta_{l,i,j,w,h} \exists\, m_k \in \mathrm{BBox}(_{i+\frac{w}{2},j,i+w,j+\frac{h}{2}}), k \in [1,N]$$

$$\Theta_{l+1,i+\frac{w}{2},j+\frac{h}{2},i+w,j+h} \leftarrow \forall\, m_k \in \Theta_{l,i,j,w,h} \exists\, m_k \in \mathrm{BBox}(_{i+\frac{w}{2},j+\frac{h}{2},i+w,j+h}), k \in [1,N]$$

$$\Theta_{l+1,i,j+\frac{h}{2},i+\frac{w}{2},j+h} \leftarrow \forall\, m_k \in \Theta_{l,i,j,w,h} \exists\, m_k \in \mathrm{BBox}(_{i,j+\frac{h}{2},i+\frac{w}{2},j+h}), k \in [1,N]$$

The main objective of using a quad-tree structure is to benefit from the fact that all markers belonging to a label are always close by. If two labels are visible in an image space it is very likely that using this approach, they will be in different quad-tree segments. Thus for recognition of a label in one segment no other markers belonging to another segment need to be considered, increasing processing speed.

The next sections describe how a quad-tree is traversed and potential labels are identified.

4.2.2. Scan Quad-tree

Depth First Search (DFS) is applied to the Quadtree. This means that the nodes are processed down to top of the tree from the finest granularity up to the root node. Let $\mathcal{R}_{l,i,j,w,h} = \{ \mathcal{R}_{l-1,p,q,r,s} , \Theta_{l,i,j,w,h}\}$ be a region during DFS at a given moment, then $\Theta_{l,i,j,w,h}$ is analyzed for possible labels. If labels are found, then the set of corresponding markers is $\Gamma_{\varphi,x}$.$\Gamma_{\varphi,x}$ is subtracted from the current region $\Theta_{l,i,j,w,h}$. If $\|\Theta_{l,i,j,w,h}\| \geq 6$ the region is analyzed for another label. If $\|\Theta_{l,i,j,w,h}\| < 6$ then all $\Gamma_{\varphi,x}$ found are subtracted from the father region and remaining markers are analyzed later during DFS:

$$\|\Theta_{l,i,j,w,h}\| < 6 \Rightarrow \Theta_{l-1,p,q,r,s} \leftarrow \Theta_{l-1,p,q,r,s} \setminus \Gamma.$$

4.2.3. Radar-Sweep Algorithm

$\Theta_{l,i,j,w,h}$ is analyzed using what we define as radar sweep algorithm. The centre of gravity of all markers in $\Theta_{l,i,j,w,h}$ is calculated. An imaginary line through the centre of gravity performs a clockwise rotation of 360° degrees. At each angle the algorithm looks for a possible top edge with three markers parallel to the radar sweep line. If a top edge is found the additional corner markers are identified and finally the coding marker is identified. This process is now explained in more detail:

$$\forall\, t\in[1,N].\vec{m}_t\in\Theta_{l,i,j,w,h},\ \vec{m}_t=\begin{pmatrix} m_t(x)\\ m_t(y)\end{pmatrix},\ \vec{m}_g=\frac{1}{N}\sum_{t=0}^{N}\vec{m}_t,\vec{m}_g=\begin{pmatrix} m_g(x)\\ m_g(y)\end{pmatrix}$$

The coordinate system is transformed to the current sweep line (Figure 5) for each $\varphi\in[0,2\pi[$.Then all marker coordinates are transformed.

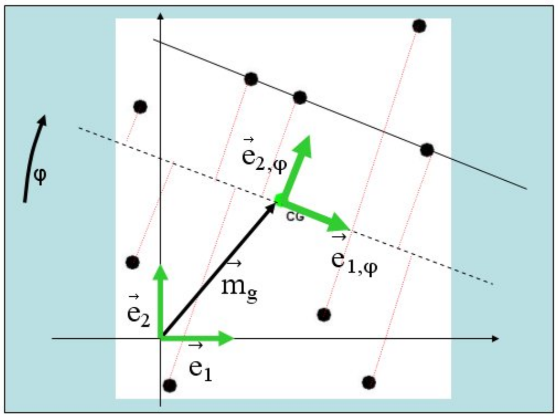

Figure 5: Coordinate Transformation

$$[\delta_\varphi]=\begin{pmatrix}\cos\varphi & -\sin\varphi & 0\\ \sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{pmatrix},\ T=\begin{pmatrix}1 & 0 & m_g(x)\\ 0 & 1 & m_g(y)\\ 0 & 0 & 1\end{pmatrix}$$

$$\forall\, t\in[1,N],\ \varphi\in[0,2\pi[,\ \vec{m}_t\in\Theta_{l,i,j,w,h}.\vec{m}_{t,\varphi}=[\delta_\varphi]T^{-1}\vec{m}_t,\ \vec{m}_{t,\varphi}\in\Theta_{l,i,j,w,h}$$

The sweep line is defined as $S_\varphi(\lambda)=\lambda\vec{e}_{1,\varphi},\lambda\in\mathbb{R}$. The set $\Theta_{l,i,j,w,h}$ is sorted in ascending y-order: $\Theta_{l,i,j,w,h} \leftarrow \mathrm{SORT}(\Theta_{l,i,j,w,h}, >y)$. To find the x-th. top line, groups $T_{\varphi,x}$ of three succeeding markers need to be identified which lie within a δ_y of each other:

$$T_{\varphi,x} \leftarrow \exists\, \vec{m}_{k,\varphi}\in \Theta_{l,i,j,w,h}\ ,\ k\in[t,t+2].\langle(\vec{m}_{k,\varphi}-a),\vec{e}_2\rangle < \delta_y,\ a=\frac{1}{3}\sum_{k=t}^{t+2}\vec{m}_{k,\varphi}$$

If found each of this groups is sorted in ascending x-order: $T_{\varphi,x} \leftarrow \mathrm{SORT}(T_{\varphi,x}, >x)$. The following is considered a valid top line:

$$\Gamma_{\varphi,x} \leftarrow \exists\, \vec{m}_{t,\varphi,x}\in T_{\varphi,x}.|\,\vec{m}_{t+1,\varphi,x} - \vec{m}_{t,\varphi,x}\,| < |\,\vec{m}_{t+2,\varphi,x} - \vec{m}_{t+1,\varphi,x}\,|$$

If $\Gamma_{\varphi,x}$ represents a valid topline, then only markers below this line need to be considered for a potential label:

$$\Theta_{l,i,j,w,h,\varphi,x} \leftarrow \exists\, \vec{m}_{t,\varphi,x}\in \Theta_{l,i,j,w,h,\varphi}\ .\ \vec{m}_{t,\varphi,x}(y)<\min_y\{P_{\varphi,x}\}$$

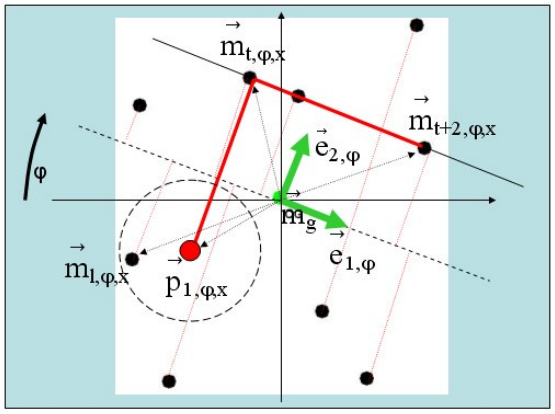

Figure 6: Finding remaining markers

To find the remaining corner markers that belong to the potential top line $P_{\varphi,x}$, we anticipate their possible location (Figure 6) by searching for them around temporary markers $p_{1,\varphi,x}$ and $p_{2,\varphi,x}$:

$$\exists\, \lambda_1\in\mathbb{R}.\vec{p}_{1,\varphi,x}=\vec{m}_{t,\varphi,x}+\lambda_1\vec{e}_{2,\varphi} \wedge |\,\vec{p}_{1,\varphi,x} - \vec{m}_{t,\varphi,x}\,| = |\,\vec{m}_{t+2,\varphi,x} - \vec{m}_{t,\varphi,x}\,|$$
$$\exists\, \lambda_2\in\mathbb{R}.\vec{p}_{2,\varphi,x}=\vec{m}_{t,\varphi,x}+\lambda_2\vec{e}_{2,\varphi} \wedge |\,\vec{p}_{2,\varphi,x} - \vec{m}_{t,\varphi,x}\,| = |\,\vec{m}_{t+2,\varphi,x} - \vec{m}_{t,\varphi,x}\,|$$

If they exist, the two possible corner markers are found as follows:

$$\vec{m}_{l,\varphi,x} \leftarrow \exists\, \vec{m}_{t,\varphi,x}\in \Theta_{l,i,j,w,h,\varphi,x}.\min\{|\,\vec{m}_{t,\varphi,x} - \vec{p}_{1,\varphi,x}\,|\}$$
$$\vec{m}_{r,\varphi,x} \leftarrow \exists\, \vec{m}_{t,\varphi,x}\in \Theta_{l,i,j,w,h,\varphi,x}.\min\{|\,\vec{m}_{t,\varphi,x} - \vec{p}_{2,\varphi,x}\,|\}$$

The five markers considered a potential solution are:

$$\Gamma_{\varphi,x} = \Gamma_{\varphi,x} \cup \{\, \vec{m}_{l,\varphi,x}\ ,\ \vec{m}_{r,\varphi,x}\,\}$$

The remaining markers considered to include the coding marker are:

$$\Theta_{l,i,j,w,h,\varphi,x} \leftarrow \Theta_{l,i,j,w,h,\varphi,x} \setminus \{\vec{m}_{l,\varphi,x}\ ,\ \vec{m}_{r,\varphi,x}\,\}$$

There must be one and only one coding marker in the bounding box of the potential label:

$$\vec{m}_{c,\varphi,x} \leftarrow \exists!\, \vec{m}_{t,\varphi,x}\in \Theta_{l,i,j,w,h,\varphi,x}.\ \vec{m}_{t,\varphi,x}\in \mathrm{BBox}(\Gamma_{\varphi,x})$$

The potential label and the remaining markers are:

$$\Gamma_{\varphi,x} = \Gamma_{\varphi,x} \cup \{\vec{m}_{c,\varphi,x}\}$$
$$\Theta_{l,i,j,w,h,\varphi,x} \leftarrow \Theta_{l,i,j,w,h,\varphi,x} \setminus \{\vec{m}_{c,\varphi,x}\}$$

If $\|\Theta_{l,i,j,w,h,\varphi,x}\| > 6$ another search for a label is done, else all labels found $\Gamma_{\varphi,x}$ are subtracted from the father region. Each possible solution is retransformed into the original coordinate system for further analysis:

$$\Gamma_{p,x} \leftarrow \forall\, \vec{m}_{\varphi,x} \in \Gamma_{\varphi,x}\ \exists\, \vec{m}_{p,x} \in \Gamma_{p,x}.\ \vec{m}_{p,x} = T\,\vec{m}_{\varphi,x}$$

4.3. PTrackScan - 3D Pose Estimation

After 2D pre-processing in image space, we attempt 3D pose estimation in camera space for each potential label identified in the 2D image space.

4.3.1. Iterating orientation

To know wether a certain $\Gamma_{p,x}$ represents a valid label, an iterative, geometric reconstruction is performed in camera space (Figure 7). Let the four corners of the potential label be:

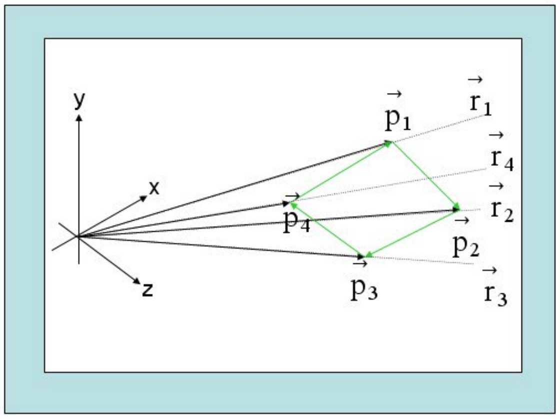

Figure 7: 3D - Pose Estimation

$\Omega = \{\vec{p}_1,\vec{p}_2,\vec{p}_3,\vec{p}_4\}$, $\vec{p}_1= \vec{m}_{t,x}$, $\vec{p}_2= \vec{m}_{t+2,x}$, $\vec{p}_1= \vec{m}_{r,x}$, $\vec{p}_1= \vec{m}_{l,x}$ in $\mathbb{R}^3$

The main idea of the algorithm is to process the corners in a round robin scheme. In camera space it rotates the initial projection of the label around the axis departing from the current corner in the direction where edge length differences in the label become minimal. After each rotation the corners of the edge opposing the axis are recalculated to continue to lie on r_i and the rotation angle is divided by two. By repeating this procedure iteratively the orientation of the label is found. Due to the reflective markers used and their inability to reflect at angles>45° an initial rotation angle of around 70° provides good results.

The function δ=maxdelta(Ω) calculates the biggest difference between edge lengths:

maxdelta(Ω) =
$\forall\ i,j\in[1..4]\ \exists\ \max\{\max\{\vec{p}_{i+1 \bmod 4}-\vec{p}_{i \bmod 4}\}-\min\{\vec{p}_{j+1 \bmod 4}-\vec{p}_{j \bmod 4}\}\}$

When a minimum difference of δ_{min} is reached, the iteration ends. The simplified main reconstruction algorithm follows:

Let initial angle $\varphi = \varphi_{init}$ and c=0 be the corner index.

```
$\delta$=maxdelta($\Omega$)
WHILE ($\delta>\delta_{min}$)
        $\delta_1, \Omega' \leftarrow$ rotation($\varphi$, c mod 4, $\Omega$)
        $\delta_2, \Omega'' \leftarrow$ rotation($-\varphi$, c mod 4, $\Omega$)
        if ($\delta_1<\delta_2$) then $\Omega=\Omega' \wedge \delta=\delta_1$ else $\Omega=\Omega'' \wedge \delta=\delta_2$
        $\varphi \leftarrow \varphi / 2$
        $c \leftarrow c + 1$
```

Each call to the rotation(..) function generates a new set of corner markers and the current edge difference. Please note that if the edge differences calculated are both higher than the previous edge difference, then the calculus is rolled back and the rotation(..) is called with $\varphi \leftarrow \varphi / 2$:

Let $\vec{n}=\vec{p}_{c+1 \bmod 4} - \vec{p}_c$ be the rotation axis.
Let $q_{\varphi,n} = [\cos(\varphi/2), \sin(\varphi/2)\,\vec{n}]$ be the rotation quaternion.
Let $\vec{m}=\vec{p}_{c+3 \bmod 4} - \vec{p}_c$ be rotated by $q_{\varphi,n}$: $\vec{m}'= q_{\varphi,n}\,\vec{m}\,q_{\varphi,n}^{-1}$

Then a surface is defined (Figure 8):

$$\langle(\vec{X} - \vec{p}_c) , (\vec{m}'^{\circ} \times \vec{n}^{\circ})\rangle = 0$$

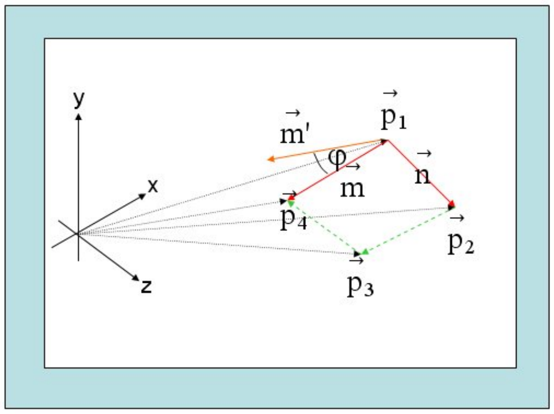

Figure 8: 3D - Iterative, geometric reconstruction

The two new corner markers of the edge opposing the rotation axis are calculated:

$$\begin{gathered}(\ \langle(\lambda\,\vec{p}^{\circ}_{c+3 \bmod 4}-\vec{p}_c),(\vec{m}'^{\circ} \times \vec{n}^{\circ})\rangle =0 \wedge \\ \langle(\mu\,\vec{p}^{\circ}_{c+2 \bmod 4}-\vec{p}_c),(\vec{m}'^{\circ} \times \vec{n}^{\circ})\rangle =0) \Rightarrow \\ \vec{p}'_{c+3 \bmod 4} = \lambda\,\vec{p}^{\circ}_{c+3 \bmod 4} \wedge \\ \vec{p}'_{c+2 \bmod 4} = \mu\,\vec{p}^{\circ}_{c+2 \bmod 4}\end{gathered}$$

The new resulting set of markers is:

$$\Omega = \{\vec{p}_c,\vec{p}_{c+1 \bmod 4},\vec{p}'_{c+2 \bmod 4},\vec{p}'_{c+3 \bmod 4}\}$$

The new maximal edge-length difference is:

$$\delta=\text{maxdelta}(\Omega)$$

When the maximal edge-length difference is lower than δ_{min} then the final orientation of the label is found.

4.3.2. Scaling position

In the last step the label is simply scaled to position in camera space, due to the fact that the actual edge-length of the label is known.

$$\forall\ \vec{p}_c\in\Omega,\ c\in[1..4]\ .\ \vec{p}'_c = \frac{|\vec{p}_c|}{|\vec{p}_{c+1 \bmod 4} - \vec{p}_c|}\,E\,\vec{p}^{\circ}_c$$

The position of the top and coding markers can easily be calculated and the label is compared to known labels in the database.

4.3.3. Compare to Database

After estimating the orientation and location of the projected potential label in camera space coordinates, a coordinate system transformation from camera space coordinates to object space coordinates of the label is applied. This allows the direct comparison of the label with registered labels in the database and its identification by its associated ID.

4.4. PTrackUDP

After computing the pose of all labels in camera space, the results are broadcasted to interested applications, using an output format which allows direct connection with Opentracker. PTrackManager receives the results and forwards them to PTrackUDP for broadcasting, which is done via UDP/IP.

5. PTrack – Evaluation

The evaluation test-bed built to test PTrack's accuracy should provide a means to compare pre-defined physical target paths (nominal condition) to computed pose estimation results (actual condition) by the tracking system.

A physical target path is the continuous change of position and orientation by a tracked target which physically moves from a starting point to an end point. The evaluation test-bed allows pre-definition of target paths to define the nominal condition for an experiment. It is possible to compare position and orientation of the tracked target to the results of pose estimation at any given moment in time.

The evaluation test-bed consists of two experiments. One experiment is used to measure positional accuracy, the other is used to measure rotational accuracy. This split is allowed because pose estimation of a tracked target (zero marker and normal vector of a target) in camera space are represented by an affine transformation, consisting of a rotation of the coordinate system, followed by a translation. The experiments have been built as accurately as possible, attempting to induce the least amount of error in the measured results. A data analyzer software was implemented to log measured samples and compute the following statistical results:

Accuracy: The accuracy is the error in the measurement of position of an object in camera space coordinates, calculated as the difference between measured values (actual condition) and the nominal values pre-defined by the target path in millimeter [mm].

Precision: is the degree of mutual agreement among a series of individual measurements, defined as the standard deviation in the measurement of position of a target during a series of experiments in millimeter [mm].

95% Confidence Intervals: establishes in which range the translation accuracy lies 95% the time in millimeter [mm].

Update rate: established how many times per unit of time the tracking system provides new position information [mm].

The difference between translational tests and rotational tests is that in the latter case, those values are separately computed for the three different angles of the normal vector on a label (Heading, Attitude, Bank).

5.1. Test-bed

The test-bed is composed of two experiments, inspired in already existing test configurations used by tracking systems' vendors.

The second experiment measuring rotational accuracy was in particular inspired by an experiment conducted by Malbezin et al. [10] to estimate ARToolKit's accuracy.

In the translation experiment (Figure 8), a label is placed on a car and moved in uniform motion along a track between two photocells. The length of the track is known and the data analyzer compares actual distance against nominal distance over time to computer the statistical results.

In the rotation experiment (Figure 9), a label is placed on a rotor with variable angle of attack and rotated 360 degrees in 48 steps of 7.5 degrees standing still for 4 seconds each step. The experiment is repeated for different angles of attack. This is similar to Malbezin et al. [10], were a label was placed on the floor and a camera was used to move around it on a circle under different angles and distances. However the here presented setup should be more stable to noise. In addition this experiment measures the accuracy of the normal vector, by measuring the accuracy of this vector's Heading, Attitude and Bank.

6. Results - PTrack vs. ARToolKit

The evaluation test-bed was used to compare translational and rotational accuracy of PTrack and ARToolKit. ARToolKit was chosen because it is one of the most widely used single-camera marker-based tracking systems. To compare both systems the authors have adapted ARToolKit's framegrabber to work with the IDS uEye camera, replacing the use of DirectShow drivers by the camera API to allow direct control over image grabbing and camera access. ARToolKit was calibrated before testing, using its calibration application. PTrack was calibrated using a 16-marker grid and the cvCalibrateCamera_64d routine developed in OpenCV [9].

Both PTrack and ARToolKit were tested on a P4 2.8GHz 1GB RAM. Tracking one label, PTrack achieved an average framerate of 29Hz, while ARToolkit reached 22Hz.

6.1. Translational Tests

The translational tests consist of a car carrying a label being pulled at uniform movement along a track between two photocells (Figure 9). The distance between both photocells is exactly 0.894m. The uEye camera was positioned 0.775m from the first photocell.

For each tracking system the test-run was repeated 3 times. According to the measured framerates, each test-run of PTrack yielded around 1508 samples for a nominal distance of 0.894 m, while for ARToolKit around 1015 samples were collected.The results under full illumination conditions (around 400 Lux) and for 3 test-runs are shown in Table 1.

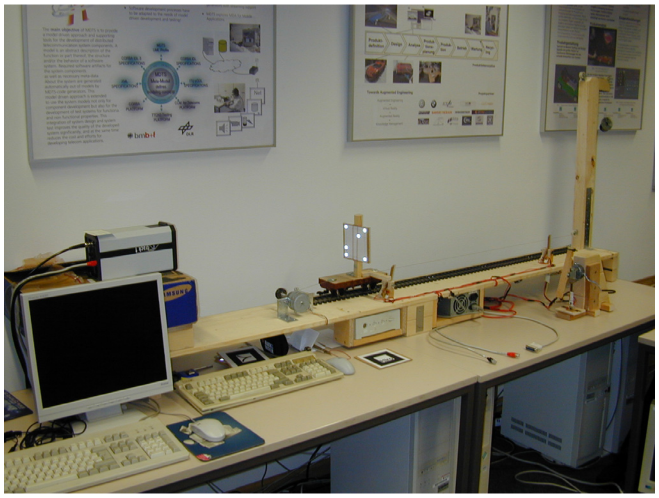

Figure 9: Translational accuracy tests

Table 1: Translational test results for ARToolKit and PTrack.

System	ARToolKit	PTrack
Average FPS [Hz]	22.0496	29.2016
Nominal distance [m]	0.8940	0.8940
Measured distance [m]	0.8404	0.9122
Avg. pos. error (**Accuracy**) [m]	0.0294	0.0065
St.dev. in pos. (**Precision**) [m]	0.0173	0.0046
Upper limit 95% Conf. Interval [m]	0.0305	0.0067
Lower limit 95% Conf. Interval [m]	0.0284	0.0063

6.2. Rotational Tests

The rotational tests consisted of placing a label on a rotor with adjustable angle-of-attack mounted on a stepper-motor (Figure 10).

Each test-run covered 47 rotation steps at 7.5° each totalling 352.5 degrees. Only data samples of the corresponding inner 60% of a 4s time-slot of each step were used for computation of the results. For each step the average result was computed and then the average of all averages per step was computed for a test-run.

The uEye camera was positioned 0,80m from the rotor, facing it as aligned as possible.

The angle of attack is defined as the angle at which the rotor is tilted from the position in which it is perpendicular to the axes of the stepper motor.

For each tracking system a test-run was made resulting in an average of 4000 samples for ARToolKit and 5200 samples for PTrack, per test-run, at each of the following angles of attack of the rotor: 0°, 5°, 10°,15°, 20°, 25°, 30°, 35°, 40°, 45°, 50°, 55°.

Due to space limitations Figure 10 just displays the average bank error for both systems. The average heading error is constant and around 3° for both systems at all angles-of-attack, and the attitude error values are very close to the bank error. values. However, for each angle of attack all statistical data was computed as for the translation tests.

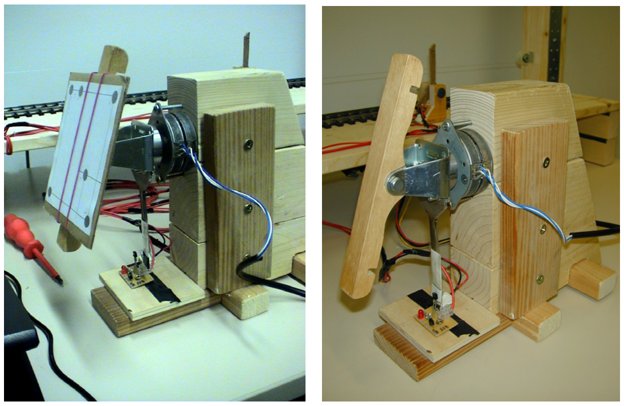

Figure 10: Rotational accuracy tests

A more detailed example for measurement of both systems at 30° angle of attack is given subsequently in tables 2 and 3, where values for the 3 angles are shown as well as a total value, which is merely the sum of the values for the 3 angles, being an attempt to show a global rotational error value for the system, just for comparison issues. Figures 11 and 12 show the normalized normal vector (the projection of all normal vectors on the camera image plane during a test-run) plotted for all angles using 30° angle of attack, for both systems.

Table 2: Rotation test results for 30° angle of attack – ARToolKit.

Angle of attack [deg]	30.438			
Angle [deg]	Head.	Attit.	Bank	Total
Avg. Error (**Accuracy**) [deg]	2.071	4.903	4.935	11.91
St. Dev. (**Precision**) [deg]	1.490	2.619	2.619	6.728
Upper Lim. 95% C.I. [deg]	2.497	5.652	5.684	-
Lower Lim. 95% C.I. [deg]	1.644	4.154	4.186	-

Table 3: Rotation test results for 30° angle of attack – Ptrack.

Angle of attack [deg]	40.321			
Angle [deg]	Head.	Attit.	Bank	Total
Avg. Error (**Accuracy**) [deg]	1.265	3.066	3.094	7.425
St. Dev. (**Precision**) [deg]	1.182	2.367	2.377	5.926
Upper Lim. 95% C.I. [deg]	1.603	3.743	3.774	-
Lower Lim. 95% C.I. [deg]	0.927	2.389	2.415	-

7. Conclusion

In this paper a new algorithm for iterative geometric pose estimation from four points was presented as well as an adequate 2D feature point pre-processing. The algorithms were embedded in a tracking system called PTrack.

A direct comparison of PTrack with ARToolKit, both using the same camera, yielded promising results: In translation PTrack achieved an average accuracy of 6.5mm and precision of 4.6mm, against 29mm and 17mm of ARToolKit, respectively; In rotation PTrack's accuracy was always under 10° for any angle of attack and under 5° for angles-of-attack equal or greater than 25° – ARToolKit's accuracy was always under 45° and under 10° for angles-of-attack equal or greater than 25°; for 40° or greater angles-of-attack, PTrack achieves 2° accuracy against 3.5° of ARToolKit. These values confirm results of Piekarski et al.[11], but show a higher accuracy and precision of ARToolKit in higher angles of attack.

One can observe that PTrack allows for stable pose estimation results over time, which is important for augmented reality applications. In addition PTrack is able to run at high update rates if needed and therefore provides small latency times for applications.

For the future it is planned to extend PTrack to eventually support regular video cameras and providing large area tracking by using multiple cameras.

Furthermore we plan to implement differential tracking taking advantage of knowing the orientation of a label in the previous frame to calculate the current frame, therefore increasing tracking performance

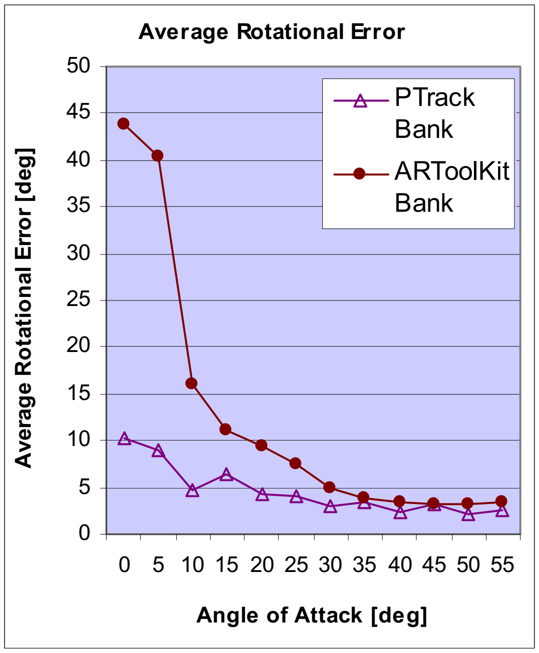

Figure 11: Average rotational error

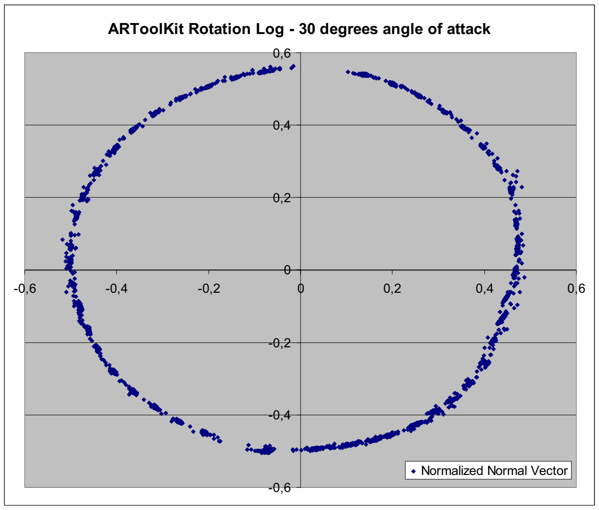

Figure 12: Normalized normal vector for all angles, for ARToolKit.

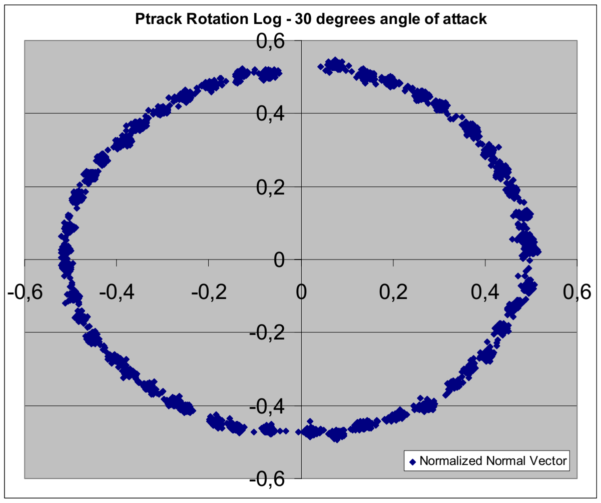

Figure 13: Normalized normal vector for all angles, for PTrack.

References

[1] uEye cameras by IDS Imaging GmbH, Website: http://www.ids-imaging.de

[2] OpenTracker, Unified Abstract Tracking Layer, Website: http://www.studierstube.org/opentracker

[3] Hirokazu Kato , Mark Billinghurst, Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System, Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, p.85, October 20-21, 1999

[4] ARVIKA, Augmented Reality in industrial applications, Website: http://www.arvika.de

[5] M.A.Fischler ad R.C.Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", Comm. ACM, vol. 24, no.6, pp.381-395, June 1981

[6] M.Dhome, M.Richetin, J.T.Lapresté, G.Rives: "Determination of the attitude of 3-D objects from single perspective view" – IEEE Trans. On pattern analysis and machine intelligence, vol.11, N°12, 1989, pp. 1256-1278

[7] Lowe, D.G., "Fitting Parameterized Three-Dimensional Models to Images", IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 13, pp.441-450, 1991

[8] Daniel DeMenthon, Larry S. Davis: Model-Based Object Pose in 25 Lines of Code. ECCV 1992: 335-343

[9] OpenCV - Intel - Open Source Computer Vision Library - Website: http://www.intel.com/technology/computing/opencv/index.htm - Last Visited: 04.11.2005
Website: http://www.intel.com/technology/computing/opencv/index.htm

[10] P.Malbezin, W.Piekarski, B.Thomas ,"Measuring ARToolKit Accuracy in Long Distance Tracking Experiments" .In ART02, 1st International Augmented Reality Toolkit Workshop September 29, 2002 - Darmstadt, Germany - Copyright (C) 2002 IEEE

[11] W.Piekarski, B.Thomas "Using ARToolKit for 3D Hand Position Tracking in Mobile Outdoor Environments". In ART02, 1st International Augmented Reality Toolkit Workshop September 29, 2002 - Darmstadt, Germany - Copyright (C)2002 IEEE