Low-Cost, Wide-Area Tracking for Virtual Environments

Low-Cost, Wide-Area Tracking for Virtual Environments

Steven Maesen*, Philippe Bekaert†
Hasselt University - Expertise Centre for Digital Media
transnationale Universiteit Limburg - School of Information Technology
Wetenschapspark 2, 3590 Diepenbeek, Belgium

ABSTRACT

In virtual, augmented and mixed reality applications, the position and orientation of the participating people or objects is often needed. In most cases this problem is solved by using expensive - and not always that accurate - special-purpose hardware. We propose an alternative position and orientation tracking system that only uses cheap off-the-shelf components like cameras and LED ropes. The construction of the scene doesn't require any off-line calibration or absolute positioning, which makes it easy to build and indefinitely scalable. There is also no restriction on the number of cameras participating.
We will show that we can calculate the orientation independent from the translation or position by using vanishing points of parallel lines. We also propose a new parameterization of the Hough transformation to make the calculations of the line detector sufficiently fast.
The proposed algorithms have been implemented and tested in a virtual set-up. The first results from our tracker are promising and can compete with many (expensive) commercial trackers. The construction of a room-sized lab set-up is in progress.

Keywords: Wide-area tracking, vanishing points, optical tracking system, computer vision.

Index Terms: I.4.1 [Computing Methodologies]: Digitization and Image Capture—Camera calibration I.4.8 [Computing Methodologies]: Segmentation—Edge and feature detection I.4.8 [Computing Methodologies]: Scene Analysis—Tracking

1 INTRODUCTION

A major step towards the immersive feeling in virtual reality is the ability to walk through the virtual environment instead of pushing buttons to move. This requires a wide-area tracking system. But many commercial systems (acoustic, mechanic, magnetic,...) don't support this kind of scalability.
The optical tracking system HiBall [13] is designed for this task and provides great speed and accuracy. The HiBall tracker uses a special-purpose optical sensor and active infra-red LEDs. Their use of specially designed hardware probably explains why the system is so expensive.
Our main goal is to build an optical wide-area tracking system at a low cost using only off-the-shelf components. We also don't expect the position of each LED to be known or calibrated, which makes the construction of the set-up fast and easy. By using only passive markers, we can support an indefinite number of cameras to be tracked because there is no synchronization required between them and each camera is a self-tracker [2]. It also makes it very easy to expand the working volume indefinitely provided you have sufficient ceiling space.

*e-mail: steven.maesen@uhasselt.be
†e-mail: philippe.bekaert@uhasselt.be

This paper is organized as follows: Section 2 categorizes other tracking systems and related work. Section 3 gives an overview of the tracking system. In the next sections the major steps of the system are explained in detail, namely the detection of the LED ropes (§4), calculating orientation from vanishing points (§5) and calculating position with known orientation (§5). In section 7, we describe our implementation of the virtual set-up and discuss the measured performance. Finally, in section 8 we will discuss our tracking system and future directions to improve the performance of our system.

2 RELATED WORK

Tracking of participating persons has been a fundamental problem in virtual immersive reality from the very beginning [10]. In most cases, special-purpose hardware trackers were developed with usually a small (accurate) working area. Most trackers are used to track the head of a person wearing a head mounted display (HMD) to generate the virtual world from their point of view. Many different technologies have been used to track HMDs: mechanical, magnetic, acoustic, inertial, optical, ... and all kinds of hybrid combinations.
The first HMD by Ivan Sutherland [10] used a mechanical linkage to measure the head position. Mechanical trackers are very fast and accurate, but suffer from a limited range because the user is physically attached to a fixed point.
Magnetic-based systems on the other hand don't have a physical linkage with the magnetic source, in fact they don't even need a line-of-sight between its source and receiver. But they suffer from a limited range (as do all source-receiver systems that use only 1 source) and are not very scalable. Also metal or other electromagnetic fields cause distortions in the pose measurements. Thus not really ideal for our studio with a metal cage to attach cameras.
Acoustic tracking systems use ultrasonic sounds to triangulate its position. This system does require a line-of-sight between source and receiver and also suffers from a limited range. The accuracy of the system also depends on the ambient air conditions.
A relatively new system uses inertia to sense position and orientation changes by measuring acceleration and torque. This system doesn't require any source or markings in the environment and therefore has an unlimited range. However this also means that there is no absolute position given and the measured position quickly drifts from the exact position. Inertial self-trackers are often combined with vision systems (optical trackers) to counteract the weak points of each other. An example of such a hybrid tracker is the VIS-Tracker [6].
With the rapid rise of CPU speeds and advances in affordable camera systems, computer vision based tracking systems can operate in real-time. But the best vision based systems, like the HiBall tracker [13], still are very expensive and require special hardware. Optical systems need line-of-sight to detect feature points in the world, but can have a great accuracy and update rate with hardly any latency as shown by the commercially available HiBall tracker.

More information about all these techniques can be found in the course "Tracking: Beyond 15 Minutes of Thought" by Gary Bishop, Greg Welch and B. Danette Allen [1].

Our goal is to make a wide-area tracking system that allows the user to walk around in a building. Most systems discussed above only have a limited range and therefore aren't really suited for this task. Only the HiBall tracker was specially designed for the same goal. In fact their set-up shows similarities in that way that we also chose an inside-looking-out system with markers on the ceiling.
The HiBall system uses a specially designed sensor with multiple photo diodes that measures the position of each sequentially flashed infra-red LED in the specially designed ceiling. The system uses the 2D-3D correspondences of each LED to accurately estimate the position and orientation of the HiBall as discussed by Wang [11] and Ward [12]. The final version of the HiBall tracker uses a single-constraint-at-a-time approach or SCAAT tracking [14].
We take a different approach to estimate rotation and position. We calculate the orientation, separate from the position, from the vanishing points of the constructed lines parallel to the X- and Z-directions. Camera calibration from vanishing points isn't a new technique. Caprile [3] used vanishing points for off-line calibration of a stereo pair of cameras. Cipolla [4] used a similar technique to calibrate images of architectural scenes for reconstruction purposes.

3 Overview Tracking System

3.1 The ceiling

In our test set-up, we have constructed a grid of LED ropes to identify the parallel lines in both the X and Z direction on the ceiling. We consider the distance between LEDs in a rope known which is needed for position tracking. This assumption is realistic because the LEDs inside the ropes are placed with a machine at predetermined distances with small deviances. Figure 1 shows a segmented input image of our virtual test set-up.
The person or object that needs to be tracked will have a camera placed on top of it pointing upwards. We will consider the intrinsic parameters of the camera known. Those values are constant if we assume that the camera does not have a variable zoom or focus.
We choose to use LED ropes instead of ordinary markers because it makes the construction and detection in a real lab set-up easier. Choosing a LED rope saves a lot of time because we don't need to attach each LED separately and minimal extra wiring is required. Mass production of LED ropes also reduces production costs, which makes them relatively cheap. By using light sources instead of markers, we can decrease the shutter time of our cameras. This means we can have a higher camera frame rate, less background noise and motion blur in our images. This increases the performance and robustness of our tracking system.

3.2 Overview Algorithm

Our tracking algorithm gets the images of the camera as input. The extrinsic parameters will be estimated in the following steps:

- Detection of the LED ropes
- Calculating orientation from vanishing points
- Calculating position with known orientation

These steps will be explained in detail in the following sections.

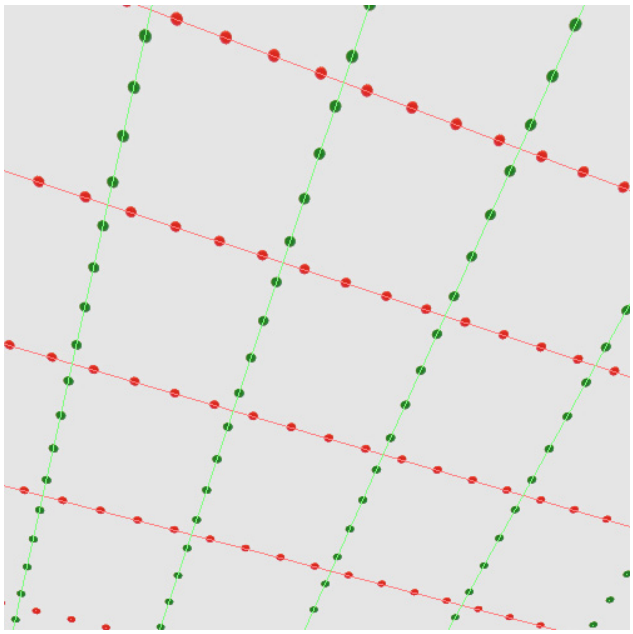

Figure 1: Segmented input image of our virtual test set-up. The lighter dot and line overlays indicate the detected markers and line patterns respectively.

4 Detection of the LED Ropes

Our first task is the detection of our constructed grid in the input image. We first segment the individual LEDs by evaluating the hue value of each pixel. Then we use a simple 'flood fill' algorithm to cluster the pixels corresponding to a LED and retain only its center. That way we speed up the line detection considerably and eliminate a lot of random noise.
Secondly we will estimate LED image correspondences with the previous frame. We use the temporal correlation between successive frames and the image velocities in the previous frames to make a fast estimate of the optical flow. These correspondences are needed to determine the translation between frames, but we also use them to speed up line detection. Because each LED corresponding to a line in the previous frame, will most likely still be part of the same line in the next frame.
Last step in the detection of the LED ropes is the line pattern recognition in the collection of detected LEDs. A mature technique for line pattern recognition is the patented Hough Transform [8].

4.1 Hough Transformation

In general the Hough transformation is a mapping of the input points to a curve in a dual parameter space. The parameterization of the pattern (in this case a line) determines the used parameter space and the shape of the dual curves. The most common used parameterization maps an input point (x_i, y_i) to a sinusoidal curve in the $\rho\theta$-plane with equation:

$$x_i \cos\theta + y_i \sin\theta = \rho \tag{1}$$

The geometrical interpretation of the parameters (θ, ρ) is illustrated in figure 2.
The Hough algorithm [7] attains its computational attractiveness $(O(N))$ by subdividing the parameter space into so-called accumulator cells (Figure 2). Each input point generates votes for every accumulator cell corresponding to its mapped sinusoid in parameter space. Finally the accumulator cells with the highest amount of

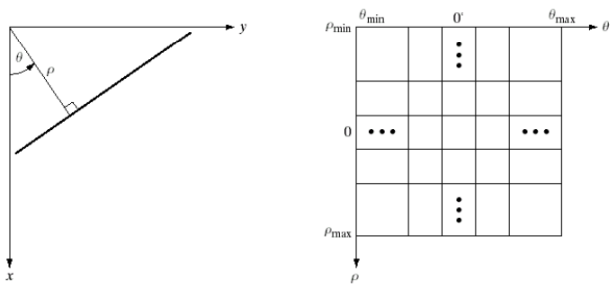

Figure 2: Left: Geometrical interpretation of the (θ,ρ) parameterization of lines. Right: Subdividing the parameter space into accumulator cells. (Image courtesy: [7])

votes (most intersections) represent the line patterns in the input space.
Although the Hough algorithm has a linear time complexity, it has a high constant cost of 'drawing' and searching for 'highlights'. Therefore most Hough transformations cannot be performed in real-time. Although we only have a small number of input points, the high constant cost weighed heavily on the trackers speed. Therefore we propose an other line parameterization to speed up the calculations with a relatively small amount of input points.

4.2 Line parameterization with circles

To increase the speed and accuracy of the line detection, we propose to calculate the intersections in the Hough parameter space analytical. But this isn't trivial with sinusoidal curves (eq. 1). Therefore we propose a new parameterization that maps each input point (x_i,y_i) to a circle in the XY-plane with equation:

$$\left(x-\frac{x_i+C_x}{2}\right)^2+\left(y-\frac{y_i+C_y}{2}\right)^2 = \left(\frac{\sqrt{(x_i-C_x)^2+(y_i-C_y)^2}}{2}\right)^2 \quad (2)$$

with (C_x,C_y) a fixed point in the image, like $(0,0)$. This means that for every point (x_i,y_i) we will construct a circle with the midpoint between (x_i,y_i) and (C_x,C_y) as center and radius equal to half the distance between these two points. The geometrical interpretation of this parameterization is shown in Figure 3.
The analytic intersection point of 2 circles is much easier to calculate than with 2 sinusoids. Given 2 circles with centers C_0 and C_1 and radiuses r_0 and r_1 and the fixed point C, we can calculate the second intersection point P (first intersection point is C) as follows:

$$P=(C_0+C1)+\frac{r_0^2-r_1^2}{d^2}(C_1-C_0)-C$$

with d the distance between C_0 and C_1.
Noise and measurement errors cause the intersection points to slightly vary their position. This means that we must define a dynamic error bin around possible intersection groups. The bins with the most intersection points in it will represent the line patterns. By choosing the initial position of the bins well, like the intersection of two circles whose points were closest together or from previous frames, the algorithm has an average complexity of $O(N^2)$. In applications with a small set of input points -like our tracker- our Hough algorithm greatly outperforms the standard algorithm.

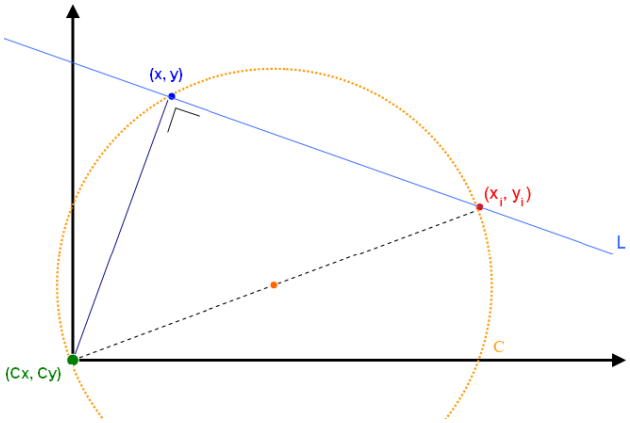

Figure 3: Geometrical interpretation of the circle parameterization of lines. Each line L through input point (x_i,y_i) is characterized by a point $(x,y)\in C$

5 Calculating Orientation from Vanishing Points

The lines detected in the previous step are the projections of the constructed parallel lines. Therefore we know that each set of lines corresponding to one axis (one color of LEDs) intersects in a single point, the vanishing point. In our system we calculate the best fit intersection point with the numerical algorithm suggested by Collins [5].
The vanishing point corresponds with a point at infinity (intersection of parallel lines) and therefore is unaffected by translation. This means that we can calculate the rotation independent from the translation [3, 4]. We can see this clearly in the projection equation of the vanishing point $V_i=(u_i,v_i,w_i)$, the projection of the point at infinity $D_i=(x_{D,i},y_{D,i},z_{D,i},0)$ (direction of the parallel lines):

$$\lambda_i\begin{bmatrix}u_i\\v_i\\w_i\end{bmatrix}=K[R|T]\begin{bmatrix}x_{D,i}\\y_{D,i}\\z_{D,i}\\0\end{bmatrix} \quad (3)$$

with λ_i a scale factor, K the calibration matrix with the intrinsic parameters, R the rotation matrix and T the translation vector. Which gives us:

$$\lambda_i K^{-1}V_i=RD_i \quad (4)$$

The only unknown factor λ_i can be calculated using the fact that the inverse of a rotation matrix is its transpose (so $R^T R=I$) [9] and D_i is a normalized direction vector (so $D_i^T D_i=1$):

$$\lambda_i=\pm\frac{1}{|K^{-1}V_i|} \quad (5)$$

By using the coordinates of the found vanishing points with their corresponding directions (X-,Y-,Z-axis), the rotation matrix can be calculated from Equation 4. Because of the uncertainties of the sign of λ_i, the orientation is ambiguous but this can be solved by looking at previous frames or adding an extra marker.
The calculation of the 9 unknowns of the 3x3 rotation matrix seems to require 3 vanishing points. But knowing that the 3 rows of the matrix form an orthogonal base [9], we only need 2 correspondences and therefore only 2 axis must be visible at all times (in our case the X- and Z-axis, the ceiling).

6 Calculating Position with Known Orientation

Given the rotation matrix and point correspondences between frames, the direction of the translation can be recovered [3]. The length of the translation is impossible to determine without a reference distance in the input image. In our system we choose to use the known distances between neighboring LEDs.
Caprile [3] demonstrates that by using 2 image points (with camera directions $\vec{D_1}$ and $\vec{D_2}$) and known length γ and orientation $\vec{D}$ between the world coordinates of LEDs P_1 and P_2, we can calculate the distances to both LEDs (α and β) by triangulation (see Figure 4). Thus we have the following system of linear equations with unknowns α and β:

$$\gamma\vec{D} = \beta\vec{D_2} - \alpha\vec{D_1} \tag{6}$$

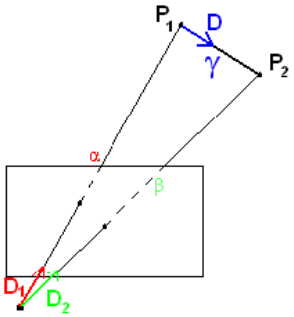

Figure 4: Depth estimation (α,β) of two adjacent LEDs P_1 and P_2 can be done by a simple triangulation given the distance γ and spatial orientation $\vec{D}$ of the LEDs from construction.

We now know the distance to each visible LED, but we don't expect the world coordinates of each LED to be known. If we use the algorithm proposed by Caprile [3], calculating the translation between successive frames, the measuring errors quickly accumulate and a lot of drift can be expected.
Instead we will estimate and remember the relative position of each LED to the start position. If we assume that we start from position P ($(0,0,0)$ or an other user defined or calibrated position), we calculate and remember the position of LED L_i as follows:

$$L_i = P + R^T(\lambda_i\vec{D_i}) \tag{7}$$

with λ_i the depth of LED i (calculated by triangulation) and $\vec{D_i}$ the corresponding camera direction of the image coordinates ($\vec{D_i} = K^{-1}(u_i, v_i, 1)^T$).
The positions of the LEDs are then located onto a grid at known distances from each other. This step reduces position errors and jitter because it enforces the relation of the LEDs to each other and tries to average out the deviances from the grid model. All positions are kept in memory until the corresponding LED isn't visible anymore.
Every visible LED that already has a relative position assigned to it in a previous frame, can vote on the current camera position by estimating P in Eq. 7 with L_i the remembered position. A median filter is than applied to the result to decrease the influence of outliers. Because the translation isn't calculated relative to the previous frame but relative to the first time the LED was visible, the drift and jitter of the system is greatly decreased.

7 Results

The proposed algorithms have been implemented and tested in a virtual set-up (see Figure 5). The virtual scene consists of a 8 by

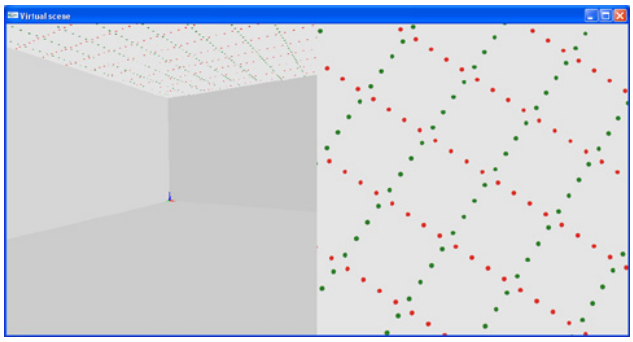

Figure 5: Implementation of the virtual set-up. Left: viewpoint of the user. Right: viewpoint of the tracker camera and input of the tracking system.

10 meters room and about 2.5 meters in height. The ceiling of this room consists of a 2D grid of markers (LEDs) in 2 colors: red LEDs to indicate the lines parallel to the X-axis and green to indicate those parallel to the Z-axis. The distance between 2 adjacent LEDs is 10 cm and the parallel lines are constructed 40 cm from each other.
A virtual test set-up has the advantage of knowing the exact position and orientation of the user. We use this data to evaluate our tracking system. The test consists of a path trough the virtual environment while looking around. The run has been performed on a standard home pc (2 GHz CPU, 1 GB RAM, NVidea Geforce4 Ti4400) and consists of about 720 frames at a resolution of 500x500. The measurements of the tracking system are not (yet) smoothed out, so no delay (except processing time) has been introduced.
Figure 6 shows the results of the measured orientation compared to the actual orientation. We immediately see that the yaw component is more accurate than the pitch or roll. This makes sense because yaw measures the rotation parallel to the ceiling. In Table 1 we even see that the average error value of the yaw component is about 0.03 degrees. This kind of absolute accuracy is highly accurate, but even the pitch and roll reach an accuracy of about 0.14 degrees. However there are some outliers that are wrong by more than 1 degree.

	Yaw	Pitch	Roll
Mean	0.0319	0.1322	0.1412
Max	1.1530	4.4951	4.1242

	X	Y	Z
Mean	0.0377	0.0461	0.0202
Max	0.1428	0.1603	0.1409

Table 1: Average absolute error values of the orientation (yaw, pitch, roll in degrees) and position (X, Y, Z in meters) tracking.

Due to the limited assumptions about the scene, position can not be absolutely recovered unlike the orientation. The calculated position will be estimated relative to the start position and drift can be expected if we move farther away. We can also see this in the results of the position tracker as shown by Figure 7. There is no real difference in the results of the X, Y and Z components. However one would expect the Y component (height) to experience less drift (because of absolute depth estimation to plane), but less accurate results because of the perpendicular triangulation. But this

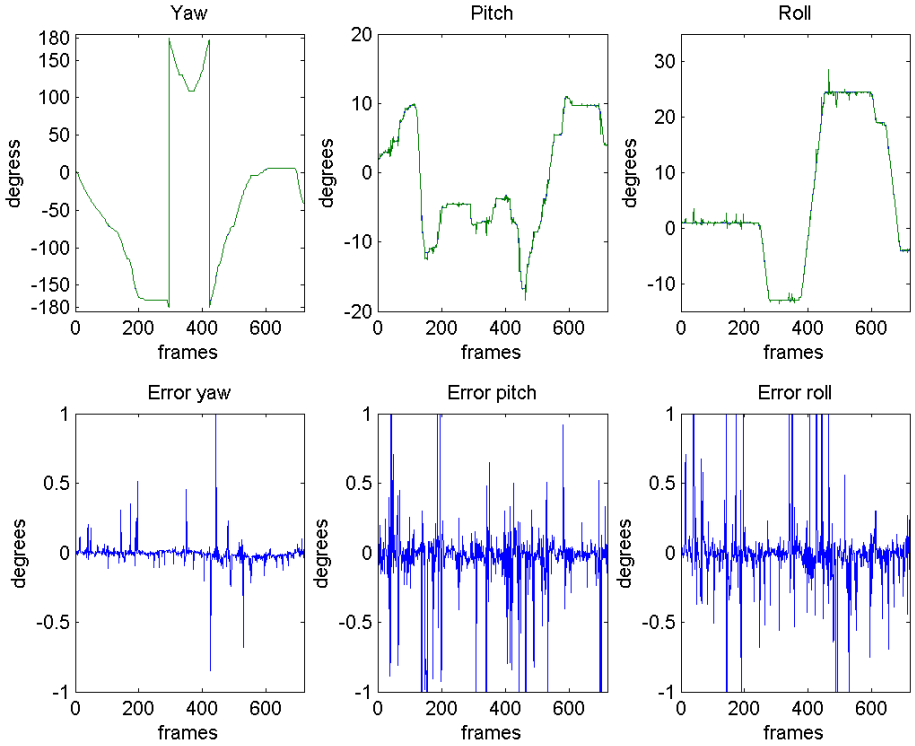

Figure 6: Orientation tracking results. Top row: Measured rotation subdivided into its Yaw, Pitch and Roll components. Bottom row: Error value of the measured orientation compared to the actual orientation.

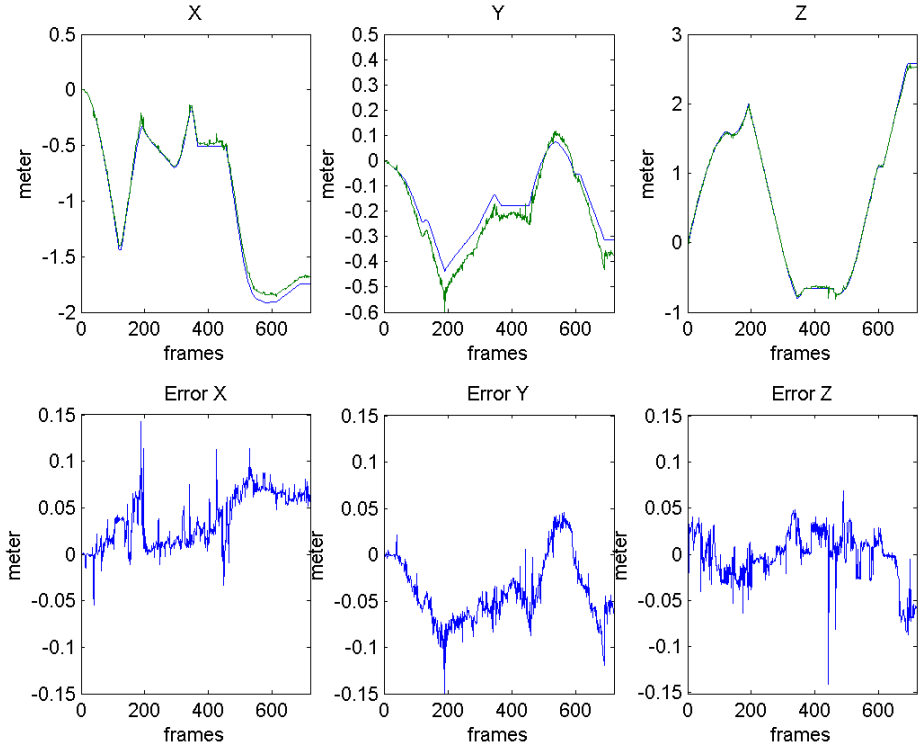

Figure 7: Position tracking results. Top row: Measured position subdivided into its X, Y and Z components. Bottom row: Error value of the measured position compared to the actual position.

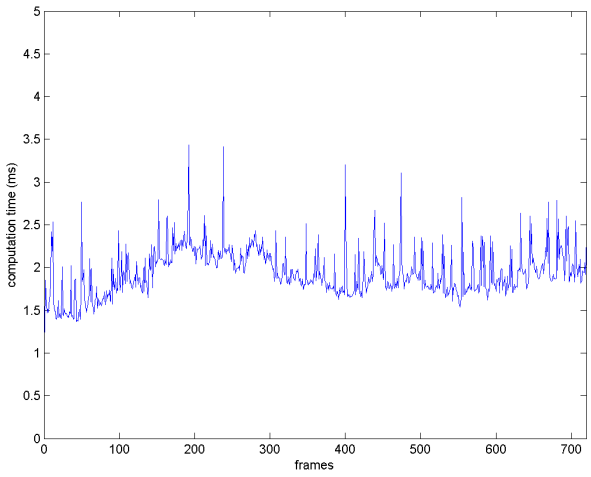

Figure 8: Processing time required to process each frame of the tracking system (in milliseconds).

isn't clear in the absolute error values shown in Table 1. Each component shows an absolute accuracy of about 3-4 cm and a drift that in this short run could reach about 14 cm.
Finally we also need to take a look at the speed of our proposed tracking system. In contrast to other vision tracking systems who barely reach interactive speeds of 20-25 fps, our system only requires on average 1.9379 milliseconds to process 1 frame. This does not include the time needed to grab a frame from the camera. Figure 8 also shows us that no more than 3.5 milliseconds was required to process any frame in this test run. The speed of the system depends for the most part on the amount of pixels to process and the number of LEDs visible in the image. If we assume that we get images from the camera at a rate of 30 fps and every frame takes no more than 3.5 seconds to process, the CPU will be idle for more than 90% of the time. That means that no dedicated tracking computer is required and other 'more useful' tasks can be done using the estimated pose data.

8 Conclusion and Future Work

In this paper we have proposed our low-cost wide-area optical tracking system using regular cameras and LED ropes. Our proposed real-time orientation tracking algorithm using vanishing points has been shown to be very accurate and very fast. This could be accomplished using a new parameterization of the Hough transform for detecting line patterns.
The position tracking addition has been shown to work very fast and with relative good accuracy. However more work is needed to increase accuracy from centimeters to millimeters, so jitter seen when inspecting an object at close range should be reduced significantly. A possible approach to accomplish this is to use an iterative algorithm.
We are currently working on a real room-sized lab set-up to test our tracking system processing real camera images. An other issue we will look into is the difficulty of detection separate LEDs if the distance to the LEDs becomes to large. A possible way to solve this is to detect the line patterns directly from the detected pixels and use the intersection of lines as feature points instead of the separate LEDs. To solve certain ambiguities and occasionally correct the relative position to an absolute position, extra LEDs or markings could be introduced, for example a point of origin.
The first results from our tracking system are very promising for a build-it-yourself wide-area tracker and can compete with many expensive commercially available tracking systems on the market.

9 Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development (ERDF), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT), which are kindly funding part of the research reported in this paper. Furthermore we would like to thank our colleagues for their help and inspiration.

References

[1] B. A. G. Bishop and G. Welch. Tracking: Beyond 15 minutes of thought: Siggraph 2001 course 11, 2001.
[2] T. G. Bishop. *Self-tracker: a smart optical sensor on silicon (vlsi, graphics)*. PhD thesis, 1984.
[3] B. Caprile and V. Torre. Using vanishing points for camera calibration. *Int. J. Comput. Vision*, 4(2):127–140, 1990.
[4] R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes, 1999.
[5] R. Collins and R. Weiss. Vanishing point calculation as a statistical inference on the unit sphere. In *International Conference on Computer Vision*, pages 400–403, December 1990.
[6] E. Foxlin and L. Naimark. Vis-tracker: a wearable vision-inertial self-tracker. pages 199–206, 2003.
[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., 2001.
[8] P. V. C. Hough. Method and means for recognizing complex patterns, December 18 1962. United States Patent 3069654.
[9] S. K. F. James D. Foley, Andries van Dam and J. F. Hughes. *Computer graphics: principles and practice (2nd ed. in C)*. Addison-Wesley Longman Publishing Co., 1996.
[10] I. E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the 1968 Fall Joint Computer Conference, AFIPS Conference Proceedings*, 1968.
[11] J. Wang, R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. Tracking a head-mounted display in a room-sized environment with head-mounted cameras, 1990.
[12] M. Ward, R. Azuma, R. Bennett, S. Gottschalk, and H. Fuchs. A demonstrated optical tracker with scalable work area for head-mounted display systems. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 43–52, New York, NY, USA, 1992. ACM Press.
[13] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. Highperformance wide-area optical tracking: The hiball tracking system, 2001.
[14] G. F. Welch. *SCAAT: incremental tracking with incomplete information*. PhD thesis, Chapel Hill, NC, USA, 1997.