Alternatives for optical tracking

W. Pasman, S. Zlatanova, S. Persa, J. Caarls

Internal Report. UbiCom, 30/5/1.

Introduction

One of the main goals for the UbiCom position tracking system is an accuracy in the centimeter range and a latency of 2 ms. We planned to use inertial tracking to reach the latency requirements, but inertial trackers have drift. The main job of the additional tracking system is to resolve this drift problem. We chose to use a camera-based tracker for this. Depending on the drift of the inertial trackers and the required precision and maximum jitter, we expect that the optical tracking system has to run at approximately 1 to 10 Hz.

An alternative for optical tracking would be realtime kinematic (RTK) GPS. RTK GPS systems can reach an accuracy of a 1 to 3 centimeters [Tiberius98, Ashtech98]. In wide open space, especially without nearby features, this will be a very useful approach. But for a cluttered urban environment and for indoors we expect that camera tracking is the best approach.

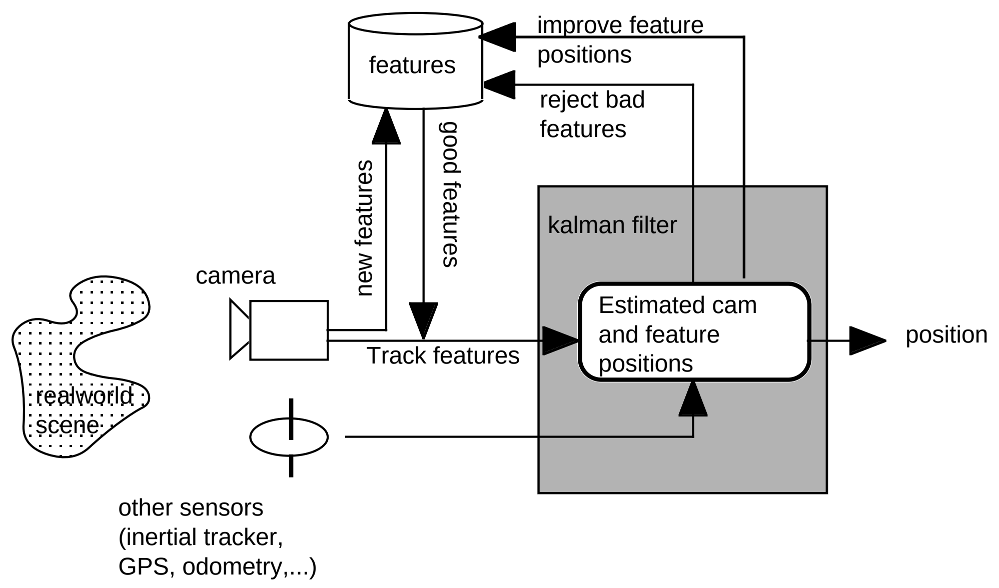
This report mainly aims at setting the basis for a discussion about the way our research should take. We explore the existing systems and alternatives, in order to find the proper configuration for the UbiCom tracking system and to find realistic research opportunities. We start with a sketch of the global system setup. Then we review existing systems, to get an impression of what has been achieved already. Then we estimate requirements for our system, and conclude with some suggestions for research opportunities.

Global system setup

Previous work [Davison99, Jiang00] has well established the global system setup (Figure 1) of an optical tracking system. Given this global setup, the choices left are the sensors to be used, and the features to be tracked in the camera image. This section briefly discusses the global setup, and then focuses on existing full-fledged systems and their performance.

The Kalman filter of the typical system takes all sensor data, including estimated noise on the measurements, and estimates the new system status. Kalman filters do a linear approximation, but this seems good enough in tracking practice. Kyger and Maybeck proposed to use an adaptive predictor to lower latencies. However, their gains are marginal: 59 ms latency for the best Kalman filter, and 50 ms latency with their system.

Various feature detectors have been used to accomplish camera position tracking. These will be compared below.


features
improve feature positions
reject bad features
new features
good features
kalman filter
camera
Estimated cam and feature positions
position
Track features
realworld scene
other sensors (inertial tracker, GPS, odometry,...)
Figure 1. Typical setup for camera tracking systems.

Feature performance in real-world setups

Various features can be tracked in the image, and these all have strong and weak points. This section discusses existing full-blown real-world tracking systems based on various features, and their performance.

Template tracking is popular for tracking the position based on real-world scenery [Davison99, Hammerhead00, Jiang00, Katani00, Shi94, Smith95]. Templates are small grayscale image patches corresponding to a point in space. In general a template can be used over a wide range of camera positions. Template tracking has been shown useful for real-time tracking [Jebara99]. Davison [Davison99] has shown that reliable templates can be extracted automatically, and their position can also be estimated automatically (Figure 2). To improve robustness of templates against translation and rotation of the camera, Ravela et al [Ravela95] converted the templates to polar space, but a general affine transformation is most popular [Triggs01, Kutulakos96, Schnörr95]. The developments mimick those in image-based rendering, where for speeding up rendering just a shift of previously rendered images was used [Regan94] and later affine warping of textures was used [Lengyel97], while we proposed a full perspective warp [Pasman99]. Template tracking has been shown to have accuracies better than 1 cm if nearby (1 m) features are visible. Such systems can work even when less than three features are in view [Jiang00].

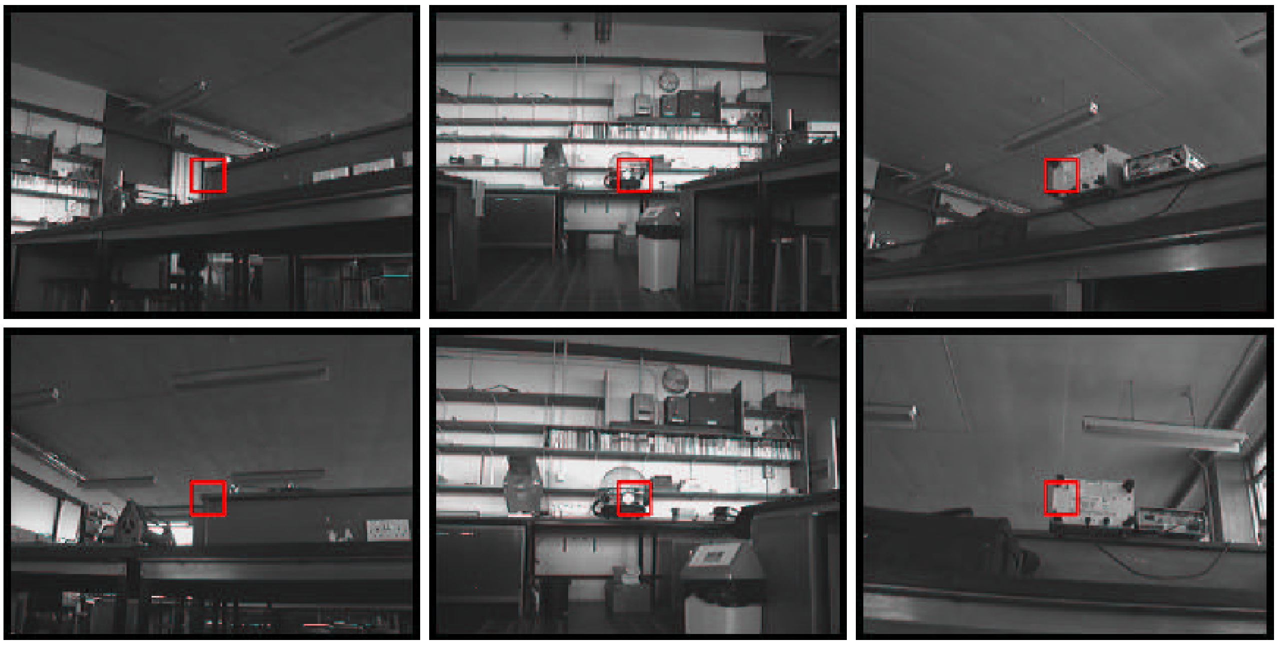


Figure 2. Automatically recognised and tracked templates (from [Davison99]). Red squares indicate template content and its real-world position.

Points on an edge with known orientation is very fast, probably the first real-world camera tracking algorithm that could run in realtime on a standard desktop computer [Harris92, Harris88] (Figure 3). Kosaka et al. [Kosaka95, Kosaka01] have improved the original algorithm on robustness for occlusion and on accuracy. The original Harris paper already indicated an accuracy of 0.5% using photos taken from an airplane, which is of the same order as the feature tracking system.

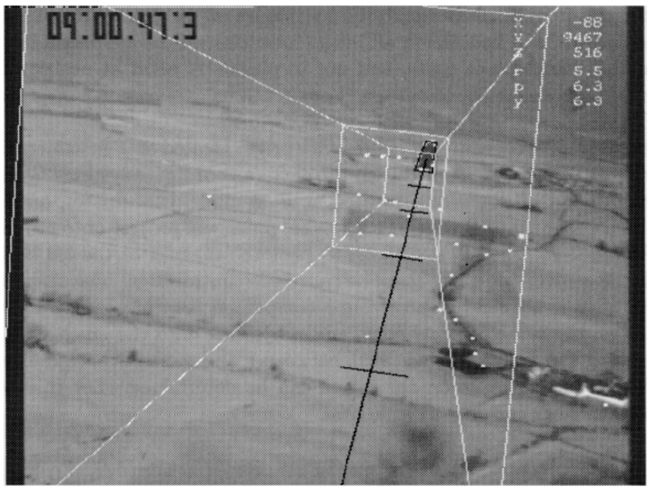


Figure 3. One of the first real-time position estimation algorithms, here tracking the position of an airplane. From [Harris92]

Various other types of corners can be used, for instance the crossings of lines, or based on gradient and curvature measurements. You et al [You99] corrected drift in their inertial tracking system using curvature corner detectors and matching those corners with a projection of stored geometry. Some care is required using these, as Deriche showed that common methods do not accurately estimate the corner position in the image [Deriche90].

Various algorithms using line matching have been presented. Kosaka et al [Kosaka95, Kosaka01] uses a database containing a 3D model of the environment and matches lines from this database using a hough transform of the image. They reach sub-centimeter accuracy and 1˚ orientation accuracy when the distance to the feature is about 20 cm, and they need about 2.5 seconds processing per frame, so this algorithm performs quite low as compared to the other alternatives. Schmid and Zisserman [Schmid97, Schmid00] have a similar approach but don't do a match in hough space (Figure 4). Usually match 97% to 100% of lines in pairs of natural images are matched properly, so theoretically it should be possible to make accurate position estimates, but they don't give accuracy estimations.

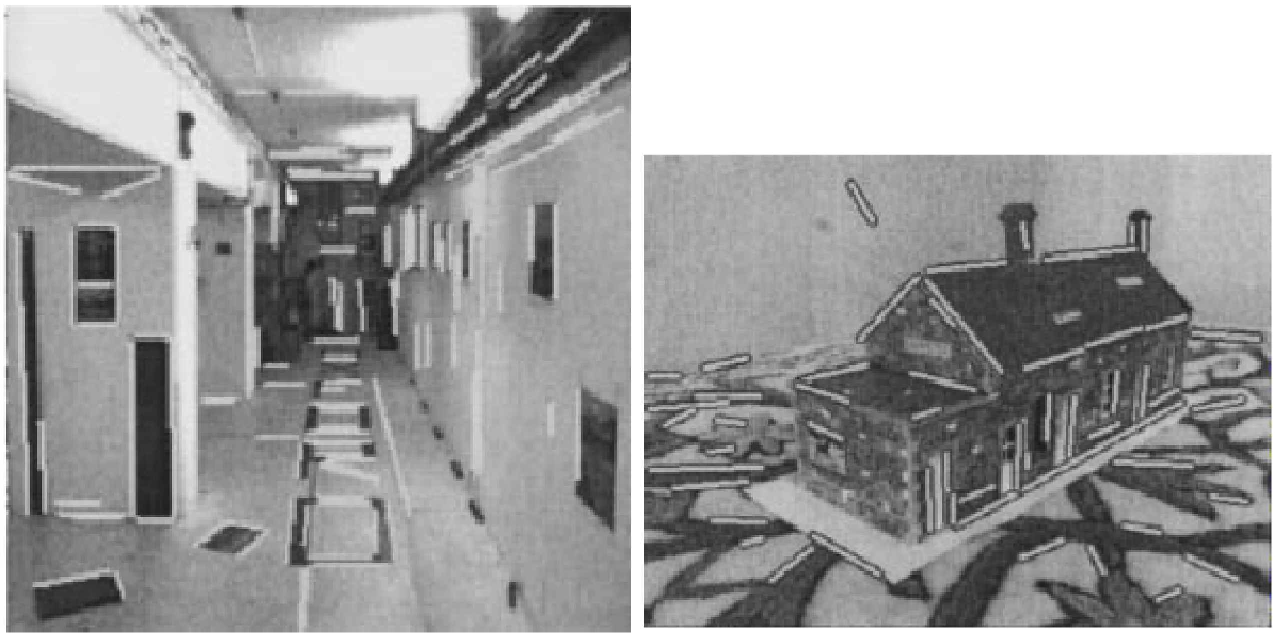


Figure 4. In [Schmid00] matching lines is shown to be quite robust.

Sundareswaran [Sundareswaran98] tracks a wireframe model of a computer case and a printer in real time with software running on a laptop (Figure 5), but again no accuracy estimations are given. Marchand can track a contour lines of a polygon model, for instance a nut or a printer connector plug, at 10 Hz on a 400MHz PC. He reaches an accuracy of 0.17˚ and 0.7mm when the object is at a distance of 40cm. Similar to [Schmid00], Berger et al. [Berger96] extends the lines to to curves. He can overlay the CAD model of a bridge over the real bridge (Figure 6), with an accuracy in the order of 3 pixels. His absolute error measures are not clear.

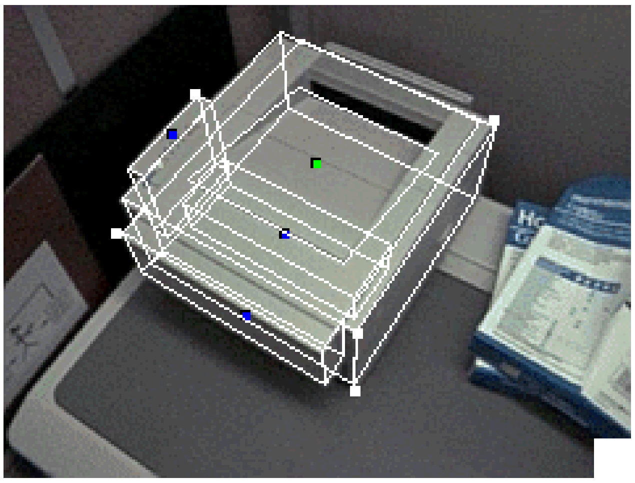


Figure 5. Tracking a printer. From [Sundareswaran98]

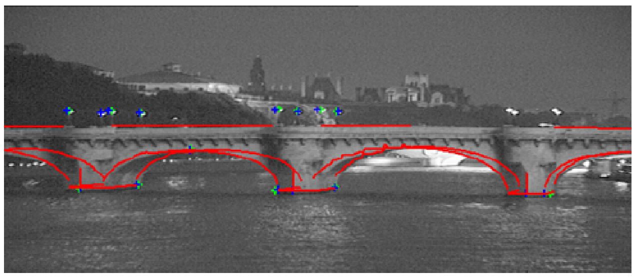


Figure 6. Tracking a distant bridge at night. From [Berger98].

Working backwards from the image to the position using lines is also possible. Montiel et al. [Montiel01] estimate the vanishing points from lines in the image, of which there are many in their indoor situation. With objects at a few meters, Their algorithm has a positional error of about 25mm and an angular error of about 0.35˚, where lines will be usually visible within a meter's distance. It is also possible to use vanishing points to reduce search space for position recovery [Weiss01].
Contours have not been used very often for positioning determination. Behringer uses contours of remote mountains [Behringer99], by matching it with height field data acquired by other means. but apparently the main goal is to estimate the orientation where the position is already known.
Color can be a powerful means to enhance tracking robustness. Birchfield showed that head tracking, which is notorious because the head has no rigid geometry, can be done very robust using only elliptical fit and color histograms [Birchfield99]. Apparently not very much has been done using color for tracking, probably because colors are highly invariant when the user moves around or lighting conditions change [BuluswarXX, Battle00, Finlayson95a,b].
Finally, various high-accuracy commercial packages are available, for instance a plug-in for Maya to track the camera position given a video track [SynaPix01]. These must reach a pretty high accuracy in order to get studio-quality movie effects, but exact data was not found. Also the used tracking methods are not clear, from the demos it seems that they use manually guided template tracking and path noise filtering.

Integrating various features

A few attempts have been done to integrate various image features into a single framework, in order to avoid specific weaknesses of individual features (Table 1). Toyama et al. [Toyama95] proposed to pick features depending on the task the tracking is being used for, and combined various feature trackers to avoid the weaknesses of individual feature trackers. In bad conditions the tracker reverts to imprecise but robust alternative features. Blaszka and Deriche [Blaszka94] compared accuracies and CPU power required for various feature trackers such as corners, edges and vertices, and conclude that 16x16 windows are sufficient to reach accuracies in the order of 0.05 pixel while being just acceptable with respect to required CPU power (~1 second per window). For our purposes this speed is too slow but current hardware is much faster.

Tracker Type	Input	Purpose	Difficulties
edge detection	pixel intensities of window	finds strong contrast edges	does not discriminate between detected edges
scale-adjusted edge detection	window sampling rates, pixel values taken at sampling rates	finds (non-predictable) distant edges	lacks precision with subsampling
intensity-based foreground correlation	pixel intensities of window, previous foreground intensities	biases edges with similar foreground as target	matches unreliably under changing illumination, does not discriminate between distraction and occlusion in some cases
hue-based foreground correlation	color values of pixels in window, previous foreground hues	biases edges with similar hue as target	does not discriminate between distraction and occlusion in some cases
endpoint search	pixel values for thin windows near endpoints	prevents aperture effect	relies on main edge detection
motion filter	past states, dynamic model	provides fallback during occlusion	expects of feature motion to conform to dynamic model
geometric model	states of other features, geometric model	integrates with other features	requires image-based confirmation

Table 1. Overview and weak points of various feature trackers. From [Toyama95].

Estimation of feature requirements

A few back-of-the-envelope estimations can be done to estimate what is required to reach centimeter accuracy based on optical tracking. We will estimate the distance to tracked features, discuss which features will typically be in view, and discuss which properties are important for automatic feature handling.

Distance to tracked features

A crucial aspect is the distance to the objects being tracked. Here, we consider features within 5 meters to be 'nearby', and features further away to be distant.
Tracking more distant features is advantageous in varous respects. First, a small error on the stored real-world position of that feature has less consequences on the calculated position, as in the projected image of the feature the error scales with the distance to the object. Second, the larger the distance the larger the choice becomes of known features in the database, and so it will be more likely that a good feature is available. Third, distant features will change less as the observer moves as compared to nearby features. Fourth, nearby features are nearby only a short amount of time, as the user is moving around, and therefore use of nearby features requires more often refresh of the features being used than distant features.
However, we highly doubt that distant features can be used for centimeter-accurate tracking of the user. Consider the following simple scenario, where a 1024x768 camera with a large-angle lens of 90° horizontal field of view is used. The average width of a pixel is now 90°/1024 = 0.09°, so the feature direction will be estimated 0.045° off on the average assuming we do a pixel-accurate tracking. We know the real-world location x and we want to know the distance to the object *d* (Figure 7).

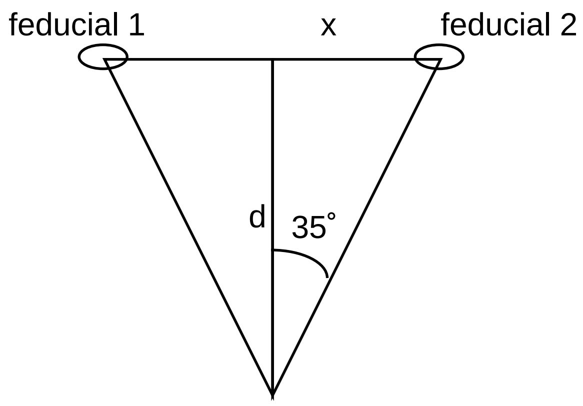


Figure 7. Two features within the camera's field of view. See text.

If we optimistically assume that we have two fedEucials in the corner of the camera's field of view, the angle between the two will be about 70°, and so d follows from $\tan 35° = x/d$. But we don't know this 35° exactly, it probably is 0.045° off, so we may be at another distance d' where $\tan 35.045° = x/d'$ Now we have for our uncertainty on the distance $d'/d = \tan 35°/\tan 35.045° \approx 0.99833$. So we have about 2 mm uncertainty per meter distance to the feducials. So if we want centimeter accuracy we have to have $d<5$ m. In less optimal cases, where the angle between the two is smaller, the maximum distance will be even less (Figure 8).

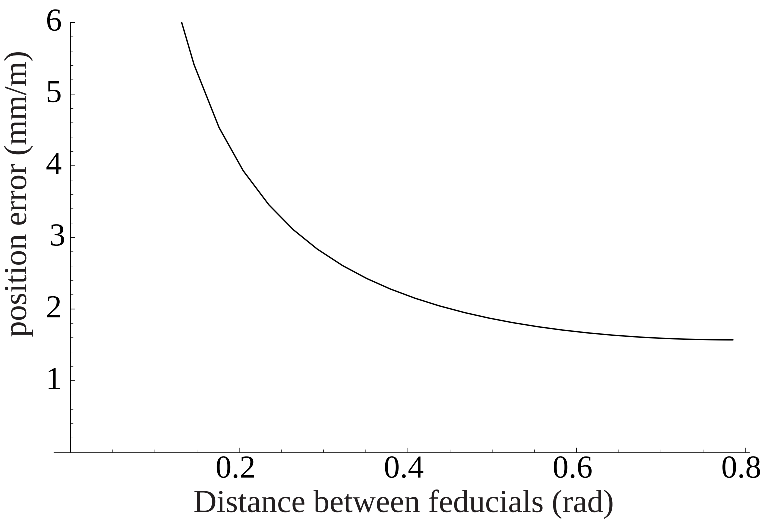


Figure 8. Uncertainty in the distance (mm of error per meter distance to the feducials) as a function of the angular distance between the feducials.

Subpixel accuracy is possible with most feducials, but the algorithms are CPU intensive [Blaszka94]. Furthermore, accurate lens distortion correction is required when subpixel accuracy is really required. Therefore weestimate that aiming for pixel-accuracy is a fair target.

Properties of nearby features

Indoors there may be enough corners of the room available at a small distance. However, often they will be outside of the field of view of the camera. Especially in relatively empty rooms, a camera with a normal viewing angle will often only see a blank wall, giving no tracking possibilities. We saw this problem already in the

creation of the pacman video [Pasman98] (Figure 9). Probably the only way to assure visibility of room features is to use an near-180˚ fish-eye lens. We probably should run some tests on this before deciding for some lens.
Outdoors, nearby features will often consist of tiles and marks on the ground, bricks and windows in the walls, and maybe even rubbish laying around (Figure 10). Trees, although not totally fixed in their position, may be usable near the ground. If there are no visible features nearby, for instance when the user walks on asphalt, tracking accuracy will drop as in Figure 8.

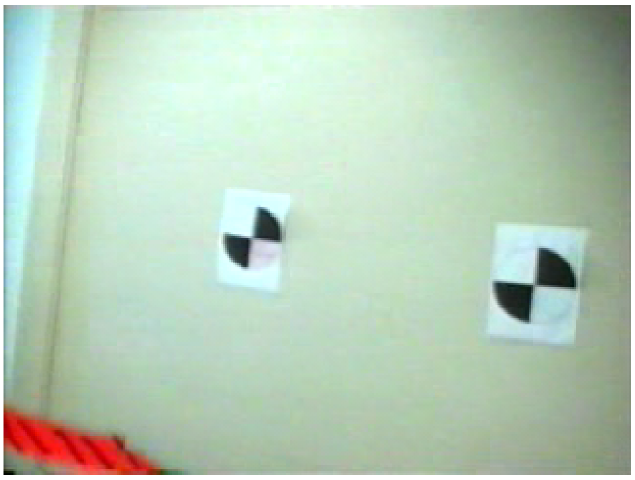


Figure 9. Shot from the pacman video, where extra features were added on blank walls to improve tracking.

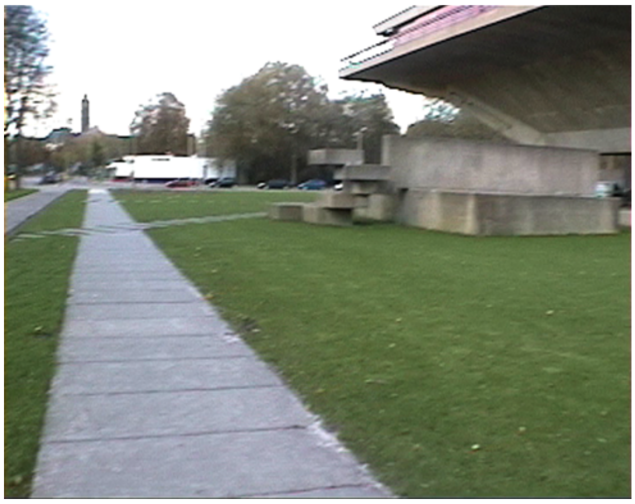


Figure 10. Typical real-world view (from Statue video).

Automatic feature handling
It is unlikely that enough of the above mentioned nearby features will be in the initial database. So to reach the required accuracy, automatic search, position estimation, and usage of new features is essential.

Features used for tracking have to fulfill a number of constraints:
- unoccluded most of the time
- stationary in the environment
- robustly trackable for large range of camera rotations and translations
- Possible to distinguish a feature from nearby features in the image

As we need automatic detection and using of new features, these aspects have to be checked automatically by the tracking system.
As many features become available in the database, massive clustering problems will occur for distant objects, because a huge amount of features will be available while the object itself covers only a small visual angle. A systematic approach is required to efficiently handle the administration concerning the required spatial validity of features. We are not aware of existing research in this area, but solutions may come from 3D level of detail rendering strategies.

Selecting proper frames
It seems important to select proper frames for analysis. Users can rotate their heads very fast, which would result in problems with excessive motion blur if a frame would

be analysed take at an unfortunate moment. Furthermore, it will be advantageous if good features are in sight, where good means that they should give a maximum increase in accuracy of the position and orientation estimation. It seems possible to have a small program run in the headset using the gyroscope information to estimate the optimal moment to grab a camera frame for analysis.

Absolute versus relative tracking

Two kinds of tracking can be distinguished. In the first system, called **absolute tracking**, the transformation of the user is determined as compared to an 'absolute' zero-position in the real world. In the second system, called **relative tracking**, the transformation is determined in a local reference frame local to the user.
An advantage of relative tracking is that tracking can be done relative to some object the application is interested in, for instance the engine of the car. If the user has a problem with his car's engine, the absolute position of the car in the real world is not very relevant. Absolute tracking would complicate matters, because the application would have to position the marker accurately in world-coordinates while it only tries to put a marker on a part of the engine. It seems typical for relative tracking that the range of the tracking is limited.
On the other hand, absolute tracking seems more convenient if large outdoor spaces are involved, for tourist information applications it is not very useful to know exactly where one is in the room but not where the room is in the city. Also when animated objects are involved, an environment-fixed coordinate system will make life easier for the application.
Of course, with an absolute tracking system one can do relative tracking as well. But the other way is difficult: a relative tracking system often does not give enough information to convert into absolute tracking.
VRML, which was chosen as the basis for the rendering system, seems flexible enough to use relative tracking, although we would have to solve problems when the user goes out of the tracker range.
Another problem with relative tracking is that multiple relative trackers might be required to run in parallel. For instance if the various parts of an engine each have their own customized relative tracker, and the user has various parts in his view. It is not clear how such a situation would be embedded within the VRML framework and how the application would handle this.

Research opportunities

Many solutions have been implemented already, and were shown to work. Some plausible open issues are:
• Make a thorough analysis of various possibilities and estimate the power usage. This can be done on theoretical grounds at least partially, and fits nicely into the UbiCom methods and approach. Hylke van Dijk already did such an analysis, and a preliminary report is on the BSCW [Dijk01].
• Integrate model-based and template-based tracking. Using a highly accurate, hierarchical initial model to arrange the templates hierarchically, to improve stability and to lower search times seems possible, and we have not yet seen such an integrated system. A disadavantage is that such a system will take a lot of implementation efford, mainly to implement already-existing components.

References

[Ashtech98] Ashtech (1998). Application notes and technical papers: Enhancing GPS with glonass. Available Internet: www.ashtech.com/Pages/techote10.html.

[Battle00] Battle, J., Casals, A., Freixenet, J., & Martí, J. (2000). A review on strategies for recognizing natural objects in colour images of outdoor scenes. Image and Vision Computing, 18, 515-530.

[Behringer99] Behringer, R. (1999). Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. Proc. IEEE Virtual Reality (Houston, March 13-17), 244 -251.

[Berger96] Berger, M. O., Chevrier, C., & Simon, G. (1996). Compositing Computer and Video Image Sequences: Robust Algorithms for the Reconstruction of the Camera Parameters. Computer Graphics Forum, 15(3), C-23. Available FTP: ftp.loria.fr/pub/loria/isa/Berger/eurographics96.pdf.gz.

[Birchfield99] Birchfield, S. (1999). Elliptical Head Tracking Using Intensity Gradients and Color Histograms. Available Internet: http://vision.stanford.edu/~birch/headtracker/.

[Blaszka94] Blaszka, T., & Deriche, R. (1994). Recovering and Characterizing Image Features using an Efficient Model Based Approach. INRIA Internal report No. 2422. Available Internet: http://graphics.cs.msu.su/library/publications/recov.pdf and http://graphics.cs.msu.su/library/stati/html/28.html.

[BuluswarXX] Buluswar, S. D., & Draper, B. A. (XX). Color machine vision for autonomous vehicles. Available Internet: http://citeseer.nj.nec.com/195189.html.

[Davison99] Davison, A. J. (1999). Mobile robot navigation using active vision. Ph.D. thesis, Robotics Research Group, University of Oxford. Available Internet: http://www.etl.go.jp/~davison/research.html.

[Deriche90] Deriche, R., & Giraudon, G. (1990). Accurate Corner Detection: An Analytical Study. 3rd Int IEEE Conf on Computer Vision (ICCV, Osaka, Japan, December 4-7), 66-70.

[Dijk01] Dijk, H. (2001). Resource Utilisation in an ARC framework. Internal Report, UbiCom Project, Delft University of Technology. Available Internet: http://bscw.ubicom.tudelft.nl/bscw/ bscw.cgi/d34622/ResourceUtilisation.pdf.

[Finlayson95a] Finlayson, G. D., Funt, B. V., & Barnard, K. (1995). Color Constancy Under Varying Illumination. Proc. Fifth IEEE International Conference on Computer Vision (ICCV), 720-725.

[Finlayson95b] Finlayson, G. D. (1995). Color constancy in diagonal chromaticity space. Proc. Fifth IEEE International Conference on Computer Vision (ICCV), 218-223.

[Hammerhead00] Hammerhead (2000). Ras_track. Hammerhead productions, Available Internet: http://www.hammerhead.com/ras_track/ras_track.html.

[Harris88] Harris, C.G., & Pike, J.M. (1988). 3D positional integration from image sequences. Image and Vision Computing, 6 (2), 87-90.

[Harris92] Harris, C. (1992). Tracking with Rigid Models. In A. Blake, Yuille (Eds), Active Vision, 59-74. MIT Press.

[Jebara99] Jebara, T., Azarbayejani, A., & Pentland, A. (1999). 3D structure from 2D motion. IEEE Signal processing magazine, 16 (3, may), 66-84. Available Internet: jebara.www.media.mit.edu/people/jebara/sfm/.

[Jiang00] Jiang, B., You, S., & Neumann, U. (2000). IEEE International Conference on Multimedia and Expo (III) 2000: 1637-1640.

[Katani00] Kanatani, K. (2000). Self-Calibration from Optical Flow and Its Reliability Evaluation. IAPR Workshop on Machine Vision Applications (MVA2000, November, Tokyo, Japan), 443-446.

[Kosaka01] Kosaka, A., Jones, A., Slivovsky, L., DeSouza, G., & Kak, A. (2001). Mobile robot navigation. Available Internet: http://rvl1.ecn.purdue.edu/v1/mobile-robot-nav/mobile-robot-nav.html.

[Kosaka95] Kosaka, A., & Nakazawa, G. (1995). Vision-based motion tracking of rigid objects using prediction of uncertainties. Proc. IEEE Int Conf on Robotics and Automation (Nagoya, Japan), Available Internet: http://rvl1.ecn.purdue.edu/~kosaka/.

[Kutulakos96] Kutulakos, K. N. and J. R. Vallino (1996). Affine Object Representations for Calibration-Free Augmented Reality. Proceedings of 1996 IEEE Virtual Reality Annual International Symposium, 25-36.

[Lengyel97] Lengyel, J., & Snyder, J. (1997). Rendering with coherent layers. Proceedings of the SIGGRAPH'97, 233-242.

[Montiel01] Montiel, J. M. M., & Zisserman, A. (2001). Automated Architectural Acquisition from a Camera undergoing Planar Motion. Accepted for VAA01 (Dublin, Ireland).

[Pasman98] Pasman, W. (1998). Pacman video. Delft University of Technology, Faculty of Information Systems and Technology.

[Pasman99] Pasman, W. (1999). Comparing 3D model descriptions for wireless AR. Internal report, November, Delft University of Technology, UbiCom.

[Ravela95] Ravela, S., B. Draper, et al. (1995). Adaptive Tracking and Model Registration Across Distinct Aspects. Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robotics And Systems (Pittsburgh, PA), 174-180.

[Regan94] Regan, M., & Pose, R. (1994). Priority rendering with a virtual address recalculation pipeline. Proceedings of the SIGGRAPH'94 (Orlando, FL, 24-29 July). In Computer Graphics, Annual conference series, 155-162.

[Schmid97] Schmid, C., & Zisserman, A. (1997). Automatic line matching across views. Conference on Computer visison and Pattern recognition (june 17-19, Puerto Rico), 666-671.

[Schmid00] Schmid, C., & Zisserman, A. (2000). The geometry and matching of lines and curves over multiple views. International Journal of Computer Vision, 40 (3), 199-234.

[Schnörr95] Schnörr, C., & Peckar, W. (1995). Motion-based identification of deformable templates. 6th Int. Conf. on Computer Analysis of Images and Patterns (CAIP '95), R. Sara, V. Hlavac (Eds.), Lect. Notes in Computer Science, Vol. 970, Springer Verlag. Available Internet: http://kogs-www.informatik.uni-hamburg.de/~peckar/publications.html.

[Shi94] Shi, J., & Tomasi, C. (1994). Good Features to Track. IEEE Conf on Computer Vision and Pattern Recognition (CVPR94, Seatle), 593-600. Available Internet: http://vision.stanford.edu/~birch/klt/.

[Smith95] Smith, S.M (1995). ALTRUISM: Interpretation of three-dimensional information for autonomous vehicle control. Engineering Applications of Artificial Intelligence, 8(3):271-280, 1995. Available Internet: http://www.fmrib.ox.ac.uk/~steve/altruism/.

[Sundareswaran98] Sundareswaran, V., & Behringer, R. "Visual Servoing-based Augmented Reality," in Procs. Intl. Workshop on Augmented Reality, San

Francisco, Nov 1, 1998. Available Internet: http://hci.rsc.rockwell.com/AR/3dReg.shtml.

[SynaPix01] Synapix (2001). SynaFlex: 3D Analysis, Choreography and Composition System. Available Internet: http://www.synapix.com/products/prod_frame_sf.html.

[Tiberius98] Tiberius, C. C. J. M. (1998). Recursive data processing for kinematic GPS surveying. PhD thesis, Delft University of Technology, faculty of Civil Engineering and Geosciences.

[Toyama95] Toyama, K., & Hager, G. D. (1995). Tracker fusion for robustness in visual feature tracking. SPIE Photonics East (Philadelphia, PA, October 23-26), 2589, 38-49. Available Internet: http://cs-www.cs.yale.edu/homes/toyama/.

[Triggs01] Triggs, B. (2001). Joint Feature Distributions for Image Correspondence. Submitted to ICCV'01. Available Internet: http://www.inrialpes.fr/movi/people/Triggs.

[Weiss01] Weiss, I., & Ray, M. (2001). Model-Based Recognition of 3D Objects from Single Images. IEEE Trans on Pattern Analysis and Machine Intelligence, 23 (2), 116-128.

[You99] You, S., Neumann, U., & Azuma, R. (1999). Hybrid inertial and vision tracking for augmented reality registration. Proc IEEE VR, 260-267.