

Stable Real-Time Interaction Between Virtual Humans And Real Scenes

L. Vacchetti¹, V. Lepetit¹, G. Papagiannakis², M. Ponder³

P. Fua¹, D. Thalmann³, N. Magnenat-Thalmann²

¹ CVlab, EPFL, Lausanne Switzerland ² University of Geneva Switzerland ³ Vrlab, EPFL, Lausanne Switzerland

Abstract

We present an Augmented Reality system that relies on purely passive techniques to solve the real-time registration problem. It can run on a portable PC and does not require engineering of the environment, for example by adding markers.

To achieve this result, we have integrated robust Computer Vision techniques into a powerful VR framework. The resulting AR system allows us to produce complex rendering and animation of virtual human characters, and to blend them into the real world. The system tracks the 3D camera position by means of a natural features tracker, which, given a rough CAD model, can deal with complex 3D objects. The tracking method can handle both large camera displacements and aspect changes.

We will show that our system works in the cluttered environment of a real industrial facility and can, therefore, be used to enhance manufacturing and industrial processes. The video sequences can be seen at <http://cvlab.epfl.ch/~vacchetti/research.html>

1. Introduction

Virtual and Augmented Reality are becoming inextricably integrated strands of the new emerging digital visualization fabric. With the advent of the recent powerful, low-cost consumer graphics, it becomes possible to build highly realistic real-time AR and VR simulations. At the same time, recent developments in Human Animation have led to the integration of Virtual Humans into synthetic environments. As the demand for Augmented Reality systems grows, so will the need to allow these virtual humans to coexist and interact with real objects and scenes.

This is especially true for industrial and manufacturing applications because AR systems are highly suitable for training, rapid evaluation of prototypes before manufacture and ergonomics. A novice user can be trained to use complex machines by the virtual teacher showing him how to correctly operate the machines. Similarly, AR is an ideal approach for designing objects by having a virtual human

interactively performing evaluation tests on an object composed of real and virtual components. Including real machinery and surroundings into the interactive simulation increases realism and decreases time that would be required to build complex virtual environments and the computation cost involved in rendering them. Fig. 1 illustrates this approach: In the top row a virtual worker demonstrates the use of a machine in a complex industrial environment. The user of the AR system can change the point of view at any time, and since the camera position is correctly registered to the real scene, the virtual worker will always be correctly blended into the streaming video. Similarly, in the bottom row of Fig. 1, the virtual human guides the user through an unknown building.

To achieve these results, we have integrated robust Computer Vision techniques into a powerful VR framework called VHD++. The resulting AR system allows us to produce complex rendering and animation of virtual human characters, and to blend them into the real world. The system tracks the 3D camera position by means of a natural features tracker, which, given a rough CAD model, can deal with complex 3D objects. The tracking method can handle large camera displacements, aspect changes and partial occlusions.

We view this as an important contribution because, while the offline camera registration problem can be considered as essentially solved, robust online tracking without markers remains an open issue: Many of the real-time algorithms described in the literature still lack robustness, tend to drift, can lose a partially occluded target object, and are prone to jitter that makes them unsuitable for applications such as Augmented Reality. To compute the motion in a given frame, we use a robust approach to 3-D feature matching that can handle wide-baseline matching to combine the information from preceding frames in traditional recursive tracking fashion with that provided by a very limited number of keyframes. This combination results in a system that does not suffer from any of the above difficulties and can deal with complex aspect changes such as those shown in Figure 1. We believe these results to be beyond the current state-of-the-art.



Figure 1. A virtual human demonstrates the use of a real machine in the top row and guides a visitor through an unfamiliar corridor in the bottom row.

In the remainder of this paper, we first discuss earlier tracking work and introduce the VHD++ VR/AR system. We then discuss our tracking approach and, finally, we present our experimental results.

2. Previous Work

2.1. Tracking

Markers-based trackers [7] have the important advantage to allow automated initialization, but at the cost of engineering the environment. They can also suffer from jittering effects.

Model-based approaches that look for 3-D poses that correctly re-projects natural features such as edges, line segments, or points, of either planar or fully 3-D model into a 2-D image [12, 9, 3, 8] are more dependable. They typically rely on finding the local minimum in an objective function. Most of these approaches derive the camera position by concatenating transformations between adjacent frames the tracking is accurate and there is no jitter because feature matching is done with respect to very close frames. Unfortunately, for long sequences these methods suffer from the error accumulation problem. Furthermore, the optimization procedure may get trapped wrong local minima, in particular in the presence of aspect changes or even when two edges of the same object become very close to each other. As a result, the tracker’s behavior can become unpredictable.

A number of methods [10, 4] track natural feature points,

for which the 3D positions have been recovered during a pre-processing stage. This approach is relatively similar to the keyframes only tracking presented Section 3.2, and should also suffer from jittering. One solution to smooth the results can be the use of Kalman filtering, for example. Nevertheless such filtering is known to be unsuitable for Augmented Reality applications.

To overcome this problem, we have developed a robust 3-D feature-matching technique that allows us to exploit both short and long baseline stereo information and, thus, at any given time, to use both preceding frames and keyframes that may have been seen from relatively different viewpoints. By fusing information from these two kinds of frames, our method prevents both jitter and drift. Our method can also handle arbitrarily complex scenes, with no limit to the camera displacement.

2.2. VHD++ for VR and AR Applications

AR systems rely heavily on the interplay of heterogeneous technologies. Because of its interdisciplinary nature, the AR domain can be viewed as a melting pot of various technologies, which although complementary are non-trivial to put together [1]. For instance, the results shown in Fig. 1 have required the integration of many technologies, such as real-time 3D rendering, skeleton and skin animation and behavioral control. In this specific example, we relied on keyframe animation. For generic motions, such as walking, pointing, grabbing, we could have used inverse kinematics animation instead.

Therefore, to produce such results, we require a real-time extensible audiovisual framework with dedicated support for VR/AR real-time virtual characters, which is exactly what the VHD++ framework provides. It is a software development framework that supports the production of high performance, real-time, interactive, audio-visual applications. It includes not only 3-D graphics, advanced synthetic character simulation and sound generation techniques but also networking, data base, artificial intelligence, content creation, and diagnostics tools. New applications can be quickly created by reusing and customizing existing, fully operational design and code.

Fig. 2 depicts VHD++ at a high level of abstraction. Its semantic backbone is the result of many years of design and development of interactive VR simulations that feature strong concurrency, network distribution, handling of various input and output devices and, of course, real-time performance. Most of them rely on relatively similar sets of low level components for data loading, containment and serialization, sharing of resources, multitasking, synchronization, time scheduling, event handling, networking, and brokering. These applications drew from the continuously growing spectrum of heterogeneous and complementary simulation technologies embedded into VHD++, which were frequently reconfigured to handle particular requirements such as camera tracking, 3-D rendering, 3-D sound, collision detection, physics, skeleton animation, skin deformation, face animation, cloth simulation, crowd control, behavioral control, multi-modal interaction, among others.

In short, VHD++ is a modular component-based framework, which makes it a natural choice for integrating our tracker as a plug-in and creating a powerful AR system.

3. Stable Real-Time Camera Tracking

Our tracking method is suitable for any kind of 3D textured object that can be described either as a wire frame model or a triangulated mesh. It starts from 2D matching of interest points, and then it exploits them to infer the 3D position of the points on the object surface. Once the 3D points on the object are tracked it is possible to retrieve the camera displacement in the object coordinate system using robust estimation. In the following paragraphs we will describe in detail our tracking algorithm. First we present a simpler version that tracks the camera displacement frame by frame. This method works well but suffers from error accumulation for long sequences. We show how to prevent this problem by adding information from key-frames (defined offline or online). This method is more complex but allows us to consider sequences without duration restriction. An early version of this work has already been presented [13], but we develop here a better solution to fuse the different kinds of information.

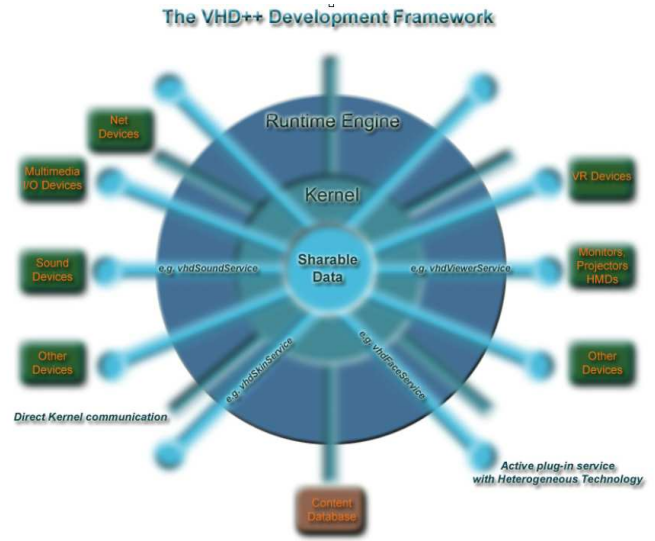


Figure 2. VHD++ Development Framework: The spheres at the end of the thin spikes coming out of the kernel represent the different technologies embedded in VHD++ and the rectangles the corresponding device abstractions.

3.1. Simple Recursive Tracking

In this section, we introduce our approach to tracking the camera-object displacement frame by frame by means of natural features, and without assumption on the object geometry.

Initialization We use a calibration grid to compute the matrix A of intrinsic parameters, offline. The algorithm starts when the user moves the camera or the object close to a known position that may be shown on the screen. This does not need to be done precisely, an approximate position is sufficient. The matching algorithm receives as input the incoming image and a “bootstrap” reference frame; if the frames are close enough, the point matching number increases above a given threshold and the tracking starts.

Robust Pose Estimation Through Point Matching First, we detect the strongest interest points in the current source image using the Harris corner detector [5]. Let the interest points detected at time t be:

$$m_t = \{m_t^0 \dots m_t^{n_t}\}.$$

They are matched with the points in m_{t-1} , the set of points detected in the previous frame, considering a appearance based measure to establish correspondence between close points. The 3-D position M_{t-1}^i of each point m_{t-1}^i are assumed to be known, and expressed in the world coordinates

system. They are propagated to the matched points in the frame at time t , and from the 2D-3D correspondences, the camera rotation R_t and translation T_t for this frame can then be estimated, for example using the POSIT algorithm proposed in [2] and the robust estimator RANSAC to discard outlier matches [6], or using a numerical minimization with a M-estimator [11] when an initial guess is available.

We also take into account the new parts of the object that have appeared in the new frame. The unmatched interest points in this frame are back-projected in order to find their 3D coordinates, keeping only the ones that are on the object surface. To do so efficiently, we first use the OpenGL accelerated rendering buffer to retrieve the facet on which the point lies, then, the intersection with the found facet and the line passing through the camera centre of projection and the 2D point in the image plane is computed.

3.2. Keyframe Based Tracking

In short, the simple method presented in the previous part works with good precision. However, the recursive approach is too weak from the point of view of error accumulation. In this section, we show how to consider some offline defined keyframes instead of previous frames and avoid this accumulation.

In next subsections, we explain how to build the keyframe offline set, and the criterion we defined to choose the best keyframe for the match.

Offline Keyframe Creation Stage During the offline stage, a small set of images representing the scene from different points of view has to be chosen and calibrated. Usually it is enough to take few snapshots all around the object. While tracking the sequence presented in this paper we only used 7 keyframes. They can be created using commercial post-production tools, such as the ones of RealViz^(tm) or 2D3^(tm). For the sequences presented in this paper, we developed an application in which the camera is calibrated by choosing very few points on the 3D model and matching them with the corresponding 2D points in the image. In this way creating a keyframe set is just a matter of minutes.

When the projection matrix is known for every keyframe, the system performs interest point detection and back-projects the points that lie on the object surface. In short, building a keyframe means collecting the following data for each frame: The bitmap image of the scene, the two sets of 2D and 3D points, the corresponding surface normals \vec{n} , used for wide baseline matching, and the projection matrix.

Keyframe choice The keyframe’s aspect must be as close as possible to the current frame. In order to find the most appropriate keyframe and to deal with object non convexities and self occlusions, we use an appearance-based method.

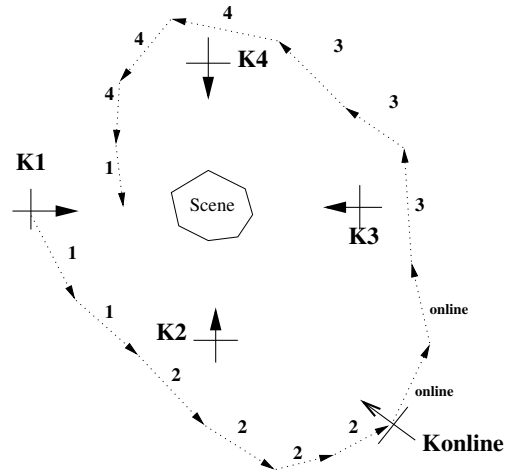


Figure 3. Using Keyframes. Tracked camera displacement with four offline keyframes and one online keyframe. The dotted arrows represent the camera displacement from one frame to the next, and the number shows which keyframe is being used. K1 to K4 are the camera positions of the offline keyframes. When the current camera position gets too far from any known offline keyframe, a new online keyframe denoted Konline is generated.

We use the following criteria:

$$\sum_{f \in Model} (\text{Area}(A[R_{t-1}|T_{t-1}], f) - \text{Area}(A_K[R_K|T_K]))^2,$$

where $\text{Area}(P, f)$ is the 2D area of the facet f after projection by P , and $A_K[R_K|T_K]$ and $A[R_{t-1}|T_{t-1}]$ the projection matrices of the key frame and the previous frame. Once more, the area are efficiently computed using an OpenGL rendering of the object model where every facet is rendered in a different color, representing the facet index. The area of every single facet is estimated by summing the number of occurrences of the facet’s pixels. This method has constant complexity, and requires only a single read of the image.

3.3. Wide Baseline Matching for tracking

This section presents our method to handle the perspective distortion on the correlation window. Conventional methods make use of a square bi-dimensional correlation window. This technique gives good points matching under the assumption of very small perspective distortion between two frames. However, to effectively use keyframes, the ability to match distant frames in a fast way becomes essential. Consequently we specify a point matching algorithm between a square 2D window in the current frame and a perspective distorted window in the keyframe image, that

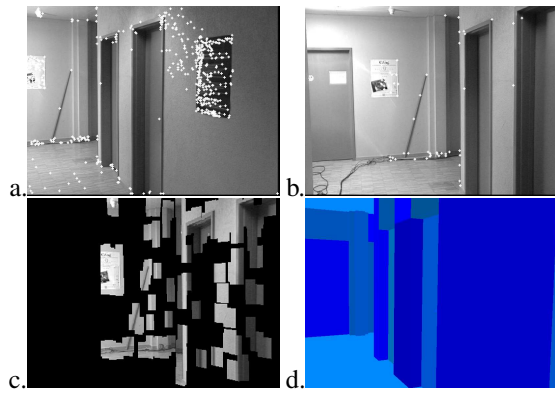


Figure 4. a. A keyframe; b. the current frame, c. the re-rendered key frame with respect to the previous camera position estimate; d. an example of facets index rendering used for back-projection and keyframe choice.

we call the “re-rendered” image. We skew the 30×30 pixel patches around each interest point from the keyframe image in order to bring them to a position close to the current one. Each patch in the keyframe is related to the corresponding image points in the re-rendered image by a planar homography. Given the patch corresponding plane π having coordinates $\pi = (\vec{n}, d)$ so that for points on the plane $n^\top X + d = 0$, the expression for the homography induced by the plane can be easily deduced from [6]. We get:

$$H = A_K(\delta R - \delta T \cdot \vec{n}'^\top / d')A^{-1} \quad (1)$$

with

$$\begin{aligned} \delta R &= R_{t-1}R_K^\top; \delta T = -R_{t-1}R_K^\top T_K + T_{t-1}; \\ \vec{n}' &= R_K \vec{n}; d' = d - T_K^\top(R_K \vec{n}). \end{aligned}$$

The resulting image is a re-rendering of the interest points’ neighborhood in a more convenient position as shown in Figure 4. This method allows us to effectively match views even where there is as much as 60 degrees of rotation. An alternative solution to the homography would have been to re-render an image of the object using an OpenGL textured 3D object, but we choose the other way to have a more precise result around the points.

3.4. Merging Offline and Inline Information

One of the key issues of our method is effectively combining keyframe and previous frames information. As we previously discussed, a pose estimation based exclusively on keyframes would have poor accuracy when few points are matched or when they are not regularly spread on the scene and in the image. This is also the typical behavior of markers-based trackers.

Formally, the keyframe-based tracking minimizes the residual sum:

$$r_t = \sum_i \rho_{LS} \left(\phi_t(M_K^{\mu(i)}, m_t^i) \right)$$

over R_t and T_t , where

- $\phi_t(M, m)$ is the reprojection error in the frame acquired at time t (with respect to R_t and T_t) of 3-D point M matched with 2-D point m ;
- the interest point m_t^i is matched with the 3-D point $M_K^{\mu(i)}$ of the keyframe chosen at time t ;
- ρ_{LS} is the least-square estimator.

To coherently combine the two different sources of information, we add to r_t the term

$$v_t = \sum_i \rho_{TUK} \left(\phi_t(M_{t-1}^{\nu(i)}, m_t^i) \right)$$

the sum of the reprojection error of the 3-D points M_{t-1} estimated in the previous image and matched with points detected at time t (ρ_{TUK} is the Tukey M-estimator and used for reducing the influence of wrong matchings). So the viewpoint at time t is estimated as:

$$\operatorname{argmin}_{R_t, T_t} r_t + v_t. \quad (2)$$

Before this non-linear minimization we first estimate the viewpoint from the points M_K with a simple combination of POSIT and RANSAC. This provides a good initial guess, and allow to remove the outliers from the set of matches with the keyframe. That is why no M-estimator is required for the expression of r_t .

3.5. Offline and Online Keyframes

Assuming we already have a consistent set of keyframes, we show in this subsection how to employ them to track a sequence. As shown in Figure 3, while the camera moves around the scene, the system switches from one keyframe to the other, always choosing the one that is closest to the images currently being seen. When the current camera position gets too far from any known offline keyframe, a new online keyframe denoted K_{online} is generated. It will be added to the keyframe set and treated like the other ones. The criterion we use for deciding to generate an online keyframe is a test on the matched point number and the robust pose discarded points number. After some time the camera will again pass close to a known position, re-using the keyframes that have been generated online. If the sequence is difficult, the system needs more offline keyframes. An interesting characteristic of this method is

that when some error has been accumulated over a part of the sequence, it will be reset to zero when an offline frame is used. The online frames can be considered as a kind of “second chance” method used to recover when there are no offline keyframes, and it has only to guarantee no complete divergence before the camera gets close to an offline frame.

4. Experiments and Results

To validate our results, we created a dedicated VHD++ application whose aim is to mix virtual humans and objects with images of real industrial environments for training and industrial maintenance purposes. The virtual characters act as instructors while the real images lend realism to the display, as shown in Fig. 1.

The tracking module has access to geometric and photometric descriptions of the target objects. Its output is made available to the rendering module and used for synthesis. The whole system runs at near real-time. VHD++ produces animations in real-time on a 2.3 GHz PC. The non optimized version of the tracking algorithm runs at about 4 frames per second for 768×576 images and about 11 frames per second for 384×288 images.

To check the tracking methods robustness, we made it work first in conditions that are closer to the more conventional approaches, than using the complete algorithm. We used our feature matching approach to track a scene three different times:

1. using only chained transformations, like a recursive tracker,
2. using only keyframes, like keyframe-based trackers,
3. combining both using our proposed method.

Figure 5 depicts the evolution of one of the camera center coordinates with respect to the frame index. The first plot compares our method to the keyframes method. The keyframe method suffers from jitter and fails at frame 295 because the camera is too far from all the keyframes. The second plot compares the recursive method to ours. Error accumulation in the recursive method does not corrupt the tracking immediately, but eventually also provokes tracking failure around frame 550.

The tracked object can assume any position with respect to the camera. For instance, there is no problem if the camera is inside the object to be tracked, and we can handle compound objects, or self-occluding parts of the scene, as long as they do not move with respect to each other. It is demonstrated Figure 6, where the camera is moved at a cross of two corridors. Despite of the huge aspect changes along the sequence, the camera trajectory is correctly recovered using only four key frames. When watching the video,

the reader can see that the reprojection of the model is particularly stable.

5. Conclusion

In this paper we presented a novel framework for building real-time VR/AR applications. Our system is not only an experimental setup able to work in a safe environment, such as a laboratory, but a robust framework designed to be operate in a real environment, such as a factory or an industrial environment. The focus of this work is to build applications for training and planning in industrial environment, but it can be used as well for many other tasks, such as ergonomics, repair, prototyping, emergency situations training, and all the cases where a dangerous — or too expensive to create — situation is present and it cannot be reproduced by a real scene.

The tracker embedded in the framework is based on natural feature points, and it can be used with a large set of scenes. The model information is exploited to track every aspect of a given target object even when occluded or only partially visible, or when the camera turns around the scene. This results in a practical system that we have been able to test in a real factory environment.

References

- [1] R. Azuma. A survey of augmented reality. In *In Computer Graphics (SIGGRAPH'95 Proceedings)*, pages 1–8, August 1995.
- [2] D. DeMenthon and L. S. Davis. Model-based object pose in 25 lines of code. In *European Conference on Computer Vision*, pages 335–343, 1992.
- [3] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In *ECCV (2)*, pages 20–36, 2000.
- [4] Y. Genc, S. Riedel, F. Souvannavong, and N. Navab. Markerless tracking for augmented reality: A learning-based approach. In *Proc. International Symposium on Mixed and Augmented Reality*, 2002.
- [5] C.G. Harris and M.J. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference, Manchester*, 1988.
- [6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
- [7] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, October 1999.
- [8] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time Visual Tracking Using a 2D-3D Model-Based Approach. In *International Conference on Computer Vision*, pages 262–268, Corfu, Greece, September 1999.

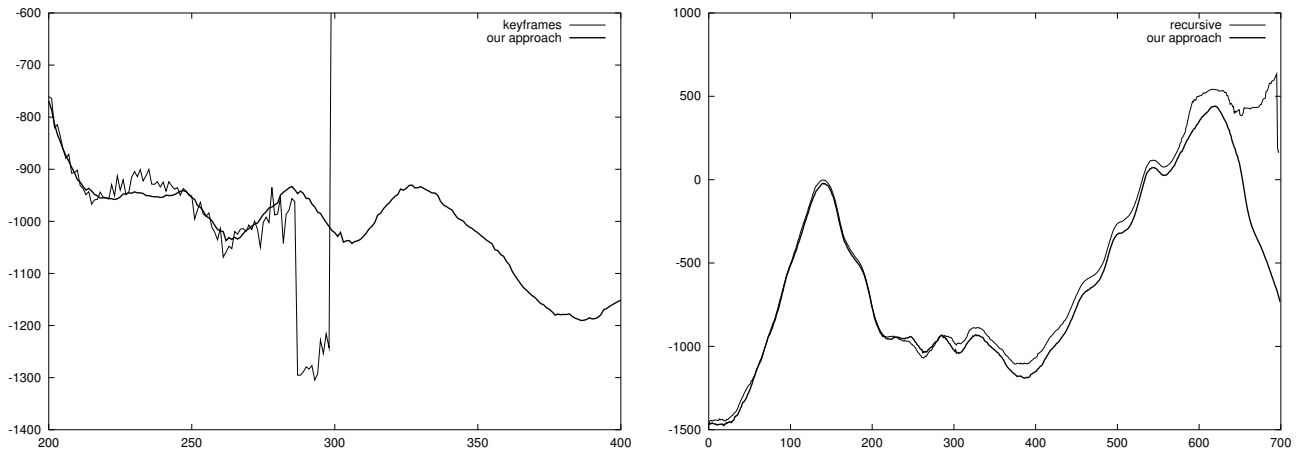


Figure 5. Plots showing a 700 frame sequence tracked using three different methods. The first plot shows the jitter of keyframe method compared to our method, the second one shows the error accumulation of the recursive method compared to ours. In both plots, the bold line corresponds to our results. We use our result as a ground truth being visually correct when we reproject the model.



Figure 6. Corridor results. Only four keyframes have been used.

- [9] U. Neumann and S. You. Natural feature tracking for augmented reality. *IEEE Transactions on Multimedia*, 1(1):53–64, 1999.
- [10] S. Ravela, B. Draper, J. Lim, and R. Weiss. Adaptive tracking and model registration across distinct aspects. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 174–180, 1995.
- [11] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.
- [12] G. Simon and M.-O. Berger. Real time registration of known or recovered multi-planar structures: application to ar. In *Proc. In British Machine Vision Conference, Cardiff (UK)*, 2002.
- [13] L. Vacchetti, V. Lepetit, and P. Fua. Fusing Online and Offline Information for Stable 3D Tracking in Real-Time (to appear). In *Conference on Computer Vision and Pattern Recognition*, 2003.