

Context-Controlled Flow Visualization in Augmented Reality

Mike Eissele*

Visualization and Interactive Systems Group
University of Stuttgart

Matthias Kreiser†

Student
University of Stuttgart

Thomas Ertl‡

Visualization and Interactive Systems Group
University of Stuttgart

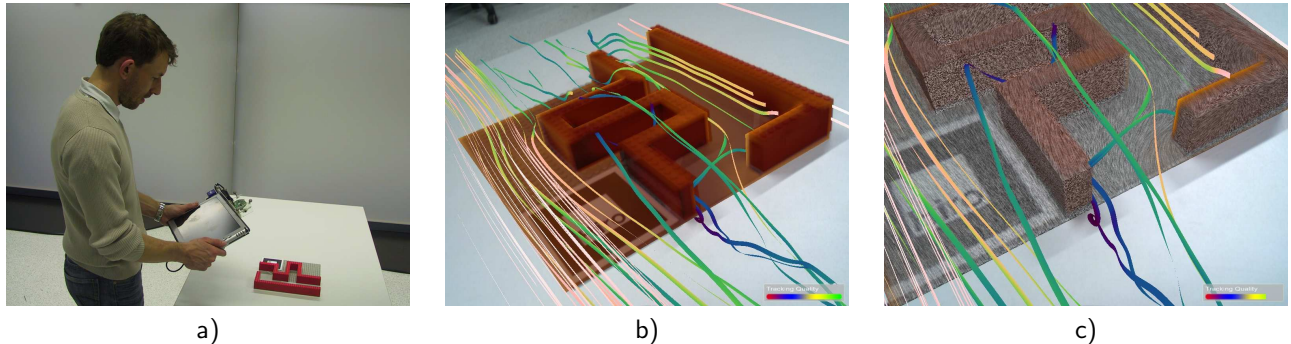


Figure 1: Flow visualization using Augmented Reality. Dependent on the current tracking quality the mobile device overlays different flow visualization techniques. Figure a) shows the client hardware used as an AR window; b) and c) depict different visualization methods of the air flow inside the test scenario.

Abstract

A major challenge of novel scientific visualization using Augmented Reality is the accuracy of the user/camera position tracking. Many alternative techniques have been proposed, but still there is no general solution.

Therefore, this paper presents a system that copes with different conditions and makes use of context information, e.g. available tracking quality, to select adequate Augmented Reality visualization methods. This way, users will automatically benefit from high-quality visualizations if the system can estimate the pose of the real-world camera accurately enough. Otherwise, specially-designed alternative visualization techniques which require a less accurate positioning are used for the augmentation of real-world views. The proposed system makes use of multiple tracking systems and a simple estimation of the currently available overall accuracy of the pose estimation, used as context information to control the resulting visualization. Results of a prototypical implementation for visualization of 3D scientific flow data are presented to show the practicality.

Keywords: Flow visualization, Augmented Reality, context awareness, mobile computer graphics

Index Terms: I.3.7 [Computer Graphics]—Three-Dimensional Graphics and Realism; I.3.6 [Computer Graphics]—Methodology and Techniques

1 Introduction

Highly accurate pose estimation techniques, demanded by most Augmented Reality (AR) applications, is still an active field of research. Many different technologies—e.g. infrared beacons, ultrasonic transmitters, cameras with and without markers, etc.—have

been evaluated in order to fulfill the requirements. Still, none of the known systems can provide a general solution for scenarios where, e.g., highly accurate/wide range/real-time pose estimations are required. Therefore, current augmented reality systems are rather limited or are forced to utilize multiple positioning systems to compensate for the lack of a single system [11].

In addition to this complexity, when different technologies are applied applications have to deal with different qualities of the pose estimation, provided by the various systems. For some scenarios further aspects are relevant in addition to accuracy like latency, performance, client energy consumption, etc. In contrast, most highly sophisticated AR applications rely on accurate tracking data to provide an overlaid real-world view where virtual objects and data are aligned with their real-world counter parts, thereby ignoring the afore mentioned facts. Even for systems that make use of multiple fused tracking systems or image-space correction approaches like [3] to enhance the tracking quality, such accuracy cannot be guaranteed for all situations.

In this paper we therefore present a technique to cope with different qualities of position and orientation information, available *after* optional correction and enhancement techniques have been applied. The system even makes use of, e.g., quality information to select an adequate visualization technique. In a more general sense, position and orientation can be seen as context information and together with other additional aspects about the current scenario, any available information can be used to influence the visualization. The presented work is focused on flow visualization in AR, but the general approach can also be applied to other visualization tasks.

Augmented Reality applications overlay real-world views by rendering virtual objects that are aligned to the real-world camera view. The use of virtual objects is not restricted and often text, pointers, or entire geometry models are embedded into real-world images. Dependent on the field of application, the augmentation of real-world views using additional virtual geometry have to be minimized in order to maintain the relationship of real and virtual objects and prevent occlusion of the user's view by virtual objects. For AR applications that visualize flow data, this is especially important since the real-world geometry directly affects the flow be-

*e-mail: mike.eissele@vis.uni-stuttgart.de

†e-mail: matthias.kreiser@gmx.de

‡e-mail: thomas.ertl@vis.uni-stuttgart.de

havior, occluding important details of it might prevent users from understanding the flow structures. To overcome this discrepancy, AR methods try to convey information directly on real-world surfaces [8], but this however requires a highly accurate tracking and alignment of virtual and real-world objects which might not be feasible.

Therefore, the herewith presented hybrid approach tries to find the most optimal visualization for the current context/situation. For example, fine details of the flow with minimal occlusion of the real-world object are presented using dense flow visualization techniques directly on real-world surfaces, if accurate tracking is available (Figure 1c). For situations where only tracking at moderate accuracy is possible, specifically adapted visualization methods—in the following call alignment-tolerant visualization techniques—are used to show the flow field in a way that misalignments of the augmented virtual objects do not distract users.

For the prototypical implementation we equipped a tablet PC with a fast USB 2.0 camera and infrared reflective markers so that the tablet position and orientation is recognized by a) the AR-Toolkit [6] and b) the A.R.T. tracking system [1]. The hand-held device, as can be seen in Figure 1a, is used as an AR window to look at augmented real-world views of the camera. First, the adaptive real-time visualization techniques are discussed. Afterwards, the quality approximation of utilized tracking systems is described and its application to control the visualization techniques.

2 Previous Work

An Augmented Reality system that is aware of uncertain position information is presented by Coelho et al. [2]. The presented general approach makes use of a modified scene graph to represent uncertain transformations. Based on the accumulated uncertainty, the final renderings of virtual objects are adapted. A reliable and robust characterization of the pose-estimation error—adequate for real AR applications—is achieved by querying the covariance information of each tracking system at run time to dynamically estimate the error. We, in contrast, only try to roughly approximate a pose-estimation error with a much simpler and very limited approach that is only valid for the described scenario and not generally applicable. It is expected that the proposed method to adapt visualizations based on pose errors would also benefit from the system described in [2]. An other approach to deal with inaccurate tracking data is presented by DiVerdi and Höllerer [3]. They utilize graphics hardware to search edges in the real-world images and modify the virtual objects so that edges of virtual objects match real-world edges. This way, a misalignment of virtual objects that share edges with real-world geometry can be hidden from users. A similar approach presented by Klein and Drummond matches edges in the camera image with a 3D model of the real world, used to calculate the occlusion of virtual objects that augment the scene [7]. A different approach to compensate for inaccuracies and noticeable errors when virtual objects are embedded into a real-world camera image is presented by Fischer et al. [5]. Stylized non-photorealistic rendering is used to modify the resulting images, this way captured camera images cannot be distinguished from computer-generated objects and, therefore, the user's immersion is enhanced.

Augmented Reality applications combine multiple areas of research and, therefore, related work can also be found in the field of visualization and in research on tracking systems. Lang and Wössner present a visualization system that can be used for AR setups [8]. Common flow visualizations like streamlines or color-coded surfaces, mapping attributes of the flow to color, are integrated. However, the presented system cannot consider specific aspects of an Augmented Reality setup where application have to work with inaccurate tracking or different visualization techniques. Schlemmer et al. present a flow visualization technique which controls the density of streamlines based on arbitrary context at-

tributes [14]. Thereby, context attributes are mainly properties of the data set, i.e. pressure or velocity. A more general approach is presented by Mattausch et al. [9] where streamline attributes—e.g. color, width, density, or opacity—are controlled by arbitrary aspects of the examined data. A concept of magic lenses is introduced by Mendez et al. and make use of context information to influence the rendering of objects within a scene graph [10]. Possible applications are shown using an AR system where objects are partially clipped away or rendered differently based on the user-defined context data. Within this paper we focus on the visualization of flow data, a good introduction into state-of-the-art techniques can be found in the flow visualization tutorial held at the Visualization Conference 2006 [4]. Besides the commonly known flow visualization techniques—like streamlines or streamribbons—we also make use of a dense representation on surfaces using a technique presented by Weiskopf and Ertl [15].

Combination of multiple tracking systems using an intelligent fusion algorithm is also a promising concept to overcome the limited accuracy of single systems. Newman et al. present a mobile hybrid tracking setup that makes use of inertial-, electromagnetic-, and vision-based tracking systems to calculate a fused pose estimation [11]. Although the setup is promising for some applications, an accurate enough tracking to overlay real-world surfaces without noticeable alignment errors cannot be guaranteed.

3 Scientific Data Visualization in Augmented Reality

Visualization of scientific data, aligned with their corresponding real-world scenario is a promising application for AR technology. Especially the visualization of flow data as fluid-mechanical engineers are interested in many aspects, e.g. pressure, speed, curl, vorticity, etc. Using a technology like augmented reality, users can more easily explore the complex nature of such data and therewith get a better understanding, required to further optimize the flow to its desired behavior. In contrast to common desktop-based flow visualization techniques, AR provides the ability to examine the flow data in context of its real-world scenario and therewith provides a much better spatial relationship. Using desktop-based VR techniques, the real-world context is only an approximated 3D model of the scenario. However, the current implementation of the prototype supports only a limited number of visualization techniques and does not provide a sophisticated user interface to fine-tune parameters. Therefore, a general evaluation of the proposed Augmented Reality visualization system versus a state-of-the-art desktop visualization tool is not reasonable.

Existing techniques for flow visualization using Augmented Reality—as mentioned in the previous section—are mostly designed for a specific tracking hardware setup with a constant accuracy and performance [3, 5, 8, 10]. As stated in the introductory section, an optimal tracking system, fulfilling all requirements for high quality AR systems, is not available up today. Therefore, high quality AR visualizations are seldom applied in practical setups. When using Augmented Reality specifically for flow visualization, some aspects have to be considered so that resulting visualizations are most beneficial to end users:

Minimize use of additional virtual objects. Augmenting the real-world view with additional virtual objects to represent features of the flow field should not hinder or occlude the real-world view of the scenario.

Direct overlay real-world surfaces. Showing the flow aspects as an aligned overlay on existing real-world surfaces. Therefore, no additional virtual objects are required, however, this requires a highly accurate tracking system.

Virtual and real-world view must be aligned. This is the basic principal of AR, but the accuracy of alignment has a direct

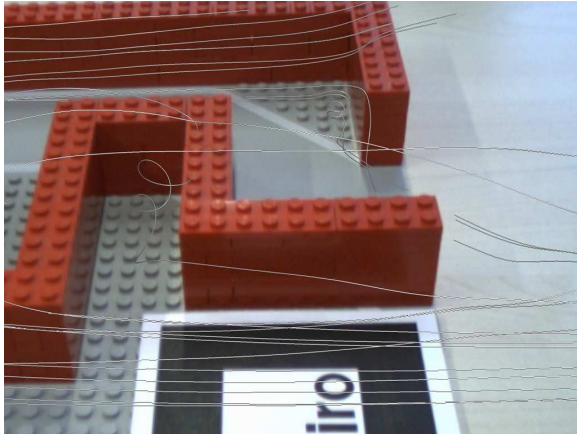


Figure 2: Flow visualization with automatically seeded streamlines. The lines are rendered with halos to enhance the contrast to the underlying real-world image.

affect on the visualization techniques applicable: Some techniques can tolerate misalignment to a certain degree, some require 100% accurate alignment to achieve acceptable results.

Real-time system response. To meet the requirement of alignment, applied visualization methods have to be real-time capable, updating the image in-sync with the real-world camera movement.

Fulfilling all requirements with a single tracking hardware and a single visualization technique seems not possible. Therefore, we present several visualization techniques for various cases and show how these are adequately integrated into the proposed system.

3.1 Alignment-Tolerant Flow Visualization Techniques

The basic idea of alignment-tolerant flow visualization techniques is to use pure virtual objects for the visualization. This conflicts with the aforementioned criteria to minimize the number of added virtual objects, but is an effective technique to hide alignment errors. For these virtual objects a loose coupling with real objects is sufficient, because a deviation from a highly accurate alignment is less noticeable to the user. Streamlines and streamribbons are two such alignment-tolerant techniques, which both visualize the flow direction and give the user a good overview of the flow. Additional scalar information, e.g. magnitude of the velocity or pressure, can be visualized by color coding the streamline geometry. Beside this information streamribbons visualize the curl of the flow. However, especially two problems have to be addressed regarding streamlines and streamribbons used in an Augmented Reality environment: The number of streamlines or streamribbons used for visualization must be chosen well and the limited depth impression of streamlines has to be improved.

The number of virtual objects used for flow visualization is a crucial aspect. On one hand the number has to be high enough to give users a good overview of the flow. On the other hand the number has to be limited to avoid an occlusion of the real-world view of the scene. Therefore, a tradeoff between these two aspects has to be applied. Another aspect is that the virtual objects should be preferably placed in interesting regions. In case of streamlines and streamribbons it is often up to the user to define seed points. For example by defining a region in which a certain number of seed points are evenly distributed. In an Augmented Reality environment, however, the possibilities for advanced user interfaces are often very limited. Users interact with real-world manipulation—e.g. changing the camera orientation—thus AR applications should

not require complicated user interfaces. Therefore an automatic placement of seed points is preferable. To achieve this we use a statistical method for seed point placement described by Zöckler et al. [16]. A scalar value p_i describing the relevance of the flow at a position i is used for the decision where to place a seed point. This could be e.g. the magnitude of the velocity or any other context attribute that characterizes interesting regions of the flow. The seed points are placed randomly and the probability that a seed point is placed at a position i is proportional to the scalar value p_i . To avoid too many seed points far away from real objects the scalar values p at positions within the bounding box of real objects can be multiplied with a constant. With this approach seed points for streamline integration are primarily placed in regions near real objects with high magnitudes of the velocity. Without this adaption it could happen that too many seed points are placed far away from real objects, because there might be higher velocities due to missing obstacles. Figure 2 shows an example visualization using streamlines that are automatically seeded.

To achieve a seamless integration of virtual objects into real-world camera images, a virtual 3D model of real-world objects is necessary. In our video-see-through scenario the real-world objects are displayed via a video stream, which is rendered as video background. The missing depth values of the real-world objects are obtained by rendering the virtual model of the real-world scenario with disabled color buffer. The correct placement of the virtual model is achieved with the information of a tracking system. Therefore, occlusions of virtual geometry with real-world objects are possible.

A misalignment is most noticeable at streamlines that are partially occluded by the modeled real-world geometry, as misalignment will cause incorrect occlusion. We therefore sort the generated streamlines into bins dependent on the distance to the real-world geometry. A simple sample-and-reject algorithm tries to fill the bins according to a given distribution function. If a streamline is generated during preprocessing and the corresponding distance bin is already filled, the line is rejected and another seed point for a new line is calculated. The maximum number of retries is limited by a user-defined threshold to ensure termination of the algorithm. The prototype makes use of a linear distribution function for the streamlines so that the largest bin holds the lines with largest distances and the bin with lines that have only minimal distance hold only few lines. As the probability that a streamline is (partially) occluded by real-world geometry increases if the distance of the streamline to the real-world geometry is decreased, occlusions are reduced if drawn streamlines preserve a certain distance to the geometry. Therefore, the visualization method chooses the bins of streamlines that should be rendered based on the accuracy of the position tracking to accomplish that misalignments are even less noticeable.

The second problem which has to be addressed is that streamlines are 1D objects and cannot provide a good depth impression, which is very important for spatial perception. To overcome this problem we illuminate our streamlines according to a method described by Zöckler et al. [16]. This helps to get a better depth impression and makes it easier for users to interpret the flow data. A further improvement is achieved by rendering the streamlines with halos, which results in a better depth impression and enhances the contrast to the underlying video background in AR setups. This way, users can more easily decide which streamlines are in front of other streamlines. Streamribbons are 2D objects and therefore provide inherently a better depth impression than streamlines.

3.2 Flow Visualization by Overlaying Real-World Surfaces

Important flow behavior can often be observed when the flow interacts with objects. In contrast to the previously described techniques, visualizing flow attributes directly on real-world surfaces

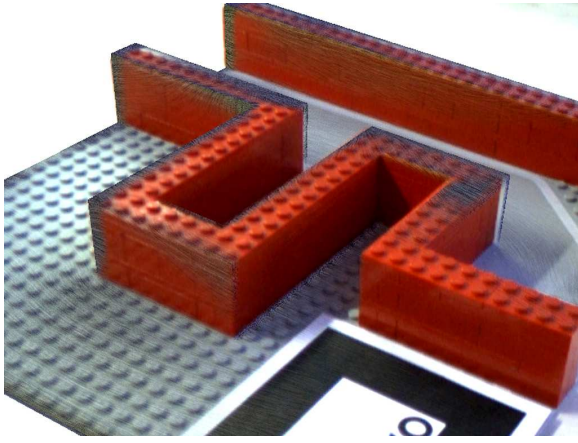


Figure 3: A high-quality flow visualization with context-aware blending of augmentation and underlying camera image, controlled via magnitude of velocity.

can convey these detail aspects, but require the use of a highly accurate tracking system. The basic idea is to overlay or partially replace real-world surfaces with virtual objects. Therefore, a 100% accurate alignment of the virtual and real objects are necessary to give users an impression that the information is directly *painted* onto the real-world object. Any misalignment would be strongly noticeable and distract the users.

Such techniques also make use of the virtual 3D model, to evaluate the coordinates of real-world surface points within the scenario. The simplest kind of surface visualization is to color code the surface according to scalar flow attributes like, e.g., pressure or temperature. But much more interesting are visualizations that are able to convey the movement of the flow. Therefore, a dense flow visualization method was integrated, as proposed by Weiskopf and Ertl [15]. They describe a LIC technique for curved surfaces that is also suitable for arbitrary geometries as it is the case in our setup. This dense texture-based visualization overcomes the finding of initial seed points, which is necessary for the techniques described in the previous section, and gives a detailed impression of the flow movement in every point on the surface. The main idea of their proposed method is to combine an image-space method with some aspects of object-space approaches. This hybrid method can efficiently be implemented on graphics processing units (GPU), which results in highly interactive rendering performance, a strong requirement for AR applications. As the LIC technique is able to present the flow movement by exploiting the contrast of a noise texture, further attributes of the flow data can be mapped to color.

For both described on-surface visualization techniques a context-aware blending was integrated. The blending linearly interpolated between the captured real-world video image and the overlaid virtual information, based on a scalar α value [0..1]. This way, context attribute—given per-surface—can control the to which amount the camera image or the virtual information should be shown. For example the overlay can be shown only at regions with high velocity or no curl. Figure 3 shows the virtual information only at interesting surface points, controlled by the magnitude of velocity. We distinguish between three overlay modes that define different blending of the visualization and the real-world camera image:

Opaque. In this mode only the surface visualization is shown and nothing of the augmented real-world object. This way, the visualization can be interpreted well, but the real object is hidden; therefore this mode is not suited for Augmented Reality.

Semi-transparent. This mode overlays the visualization and the

real object by using a fixed blending ratio. A visualization of the flow behavior on the entire surface with a good impression of the underlying real-world geometry can be achieved with this mode.

Adaptive. This mode is preferred for Augmented Reality applications. The visualization is only displayed in interesting regions, which are, e.g., characterized by a scalar flow attribute like pressure, temperature, or magnitude of the velocity. An example image is shown in Figure 3. This is realized by adapting the blending value α according to a scalar context attribute s as follows:

$$\alpha = \begin{cases} 1.0 & \text{for } s > T_{opaque} \\ \frac{s - T_{semi}}{T_{opaque} - T_{semi}} & \text{for } T_{opaque} > s > T_{semi} \\ 0.0 & \text{else} \end{cases} \quad (1)$$

T_{opaque} and T_{semi} are two thresholds which characterize the transition between opaque and semi-transparent rendering modes; respectively the transition between semi-transparent and invisible rendering. Both should be chosen according to the values $s \in [S_{min}, S_{max}]$ on the object surface which control the adaptive blending. Between T_{opaque} and T_{semi} the blending value decreases linearly from 1.0 to 0.0.

4 Visualization Control via Tracking-Quality Context

For engineers different aspects of flow data might be important: An overall impression of the global flow behavior, a detailed view of specific parts of the field, or a dense representation of the flow-surface interaction. Visualization systems that offer techniques to emphasize these aspects exist, but have to be configured by users to achieve the desired result. In addition, if these techniques are applied to AR setups most visualization methods require a highly accurate alignment of real-world and virtual objects.

An automatic technique to support the user in the selection of the most appropriate visualization technique can be achieved, if enough information of the current context, i.e. situation, is available. For the presentation of flow data using AR techniques, the utilized context information is based on the currently available quality of the camera tracking and the distance of the flow field from the viewer. Only if high-quality pose estimations are available visualizations techniques can be used that directly interact with real-world geometry. In contrast, if the examined flow field is relatively far away from the viewer, it could be assumed that the user wants to get an overview and is not interested in fine details which therefore do not have to be shown. With this exemplary subset of context information the presented prototype performs a reasoning to find the most adequate visualization.

Two independent tracking technologies were used, but the system can easily be extended to support more. To serve a moderately sized area of operation with a free field of view, i.e. the tracking of the user's view is independent of features captured by the camera, the system makes use of the A.R.T. Tracking system [1]. It is based on infrared reflective markers and a stationary array of four cameras, serving an area of approximately $7m^2$. The fixed stationary cameras view the scenario so that infrared reflective markers—attached to the mobile viewer/camera—are recognized. The A.R.T. system can already provide pose estimations at good accuracy, however the angular and positional error still tend to be too high to allow an accurate direct overlay of real-world geometry.

In contrast, optical-marker-based tracking using the common AR-Toolkit [6], based on computer-vision techniques, provide highly accurate positioning, as long as the entire marker is captured by the camera at a moderate size. A disadvantage of the AR-Toolkit is that relatively large markers (approx. $8cm^2$) have to be placed within examined scenario, thereby disturbing the real-world view.

4.1 Estimation of Tracking Quality and Information Fusion

In order to control the visualization based on the present tracking quality Q_f ; all utilized positioning systems have to provide a quality measurement Q_x . In the following we assume quality values Q_x and Q_f are already normalized to a range of $[0.0..1.0]$ to represent no/incorrect pose estimation up to highly accurate pose estimation. The pose estimations T_x calculated by various tracking systems might differ (e.g. due to measurement errors) and therefore have to be fused, based on their quality Q_x . Especially between transition where different systems are available—e.g. the AR-Toolkit marker can no more be detected in the captured image—special considerations have to be taken to guarantee a smooth transition, thereby avoiding sudden jumps in the orientation and position of the augmented virtual information. Modeling the quality of tracking is strongly dependent on characteristics of the utilized systems. A robust general solution to estimate and handle errors of pose estimations is presented by Coelho et al. in [2]. In contrast, we used a much simpler approach to approximate an error estimation of the utilized tracking systems as we focused on the adaptation to pose-estimation errors, not on the calculation of the error estimation itself.

The AR-Toolkit [6] library provides a scalar confidence value for each reported marker detection that reflects the probability that the marker detection is correct. Thereby, a confidence value of 0.0 describes a very low probability that the marker detection is correct. With the assumption that a reported marker detection with a high probability of correctness also allows a more accurate pose estimation than marker detections with a low probability, the confidence value can be used as a rough approximation of the pose-estimation error. Please note that this is only a simple approximation and might not be applicable in general. Furthermore, in several practical tests it has turned out that this confidence value cannot directly be used as quality measurement, as calculated orientations where inaccurate although the reported confidence is high, if the marker in the captured camera image was too small. Therefore, a linear fall-off for situation where the detected marker was further away than about 1.40m was added to the confidence value which could than be interpreted as quality value. AR-Toolkit seems to calculate the pose estimation for each frame individually; therefore for captured frames with awkward lighting conditions or marker occlusions an erroneous orientation is reported. To overcome this fact, approximated quality of the AR-Toolkit was reduced for 0.2s (about the first five frames) always after the first recognition of the marker, if the marker could not be detected within all camera image during this phase the time was reset to 0.2s.

For the infrared optical A.R.T. tracking system [1] a slightly different model was applied. Also A.R.T. foresees a quality value provided by the tracking system for the currently calculated pose estimation within its data structures. However, the utilized software version of A.R.T. (v1.19.2) provides only a *binary* quality of 1.0 for detected markers and 0.0 for not detected markers. Therefore, the utilized A.R.T. setup did not provide any value to estimate the current quality of the pose estimation. During tests it has been observed that the most accurate pose estimations A.R.T. can provide is still below the AR-Toolkit accuracy, thus we clamped the quality to a max. of 0.8 to adjust it to the AR-Toolkit approximated quality estimation. Further, the quality was reduced during the first 0.2s after the marker detection happened to stabilize the approximation. In contrast to AR-Toolkit, A.R.T. is synchronized to the AR-Toolkit system—described in the following paragraph—as the AR-Toolkit provides a higher quality pose estimations. After synchronization, the result of A.R.T. matches the result of the AR-Toolkit with a max. quality of 1.0 which will degenerate with subsequent changes in orientation and position, relative to the pose at the synchronization. We modeled this behavior with a timer that increases the qual-

ity of A.R.T. to 1.0 for 3.5s after the synchronization then the quality is lowered again to the defined standard quality of 0.8 for the A.R.T. system.

Information fusion and synchronization of the tracking systems helps to maximize the overall tracking quality. A synchronization, i.e. adjustment of a tracking system to a *reference tracking system*, is possible and beneficial if it can be guaranteed that a highly accurate pose estimation is known and concurrently the system that has to be synchronized also delivers good quality estimations, but with the to-be-corrected orientation offset. The difference between both pose estimations is measured and stored to correct following measurements, in the prototypical application the A.R.T. system is synchronized to the AR-Toolkit by calculating and storing the pose difference whenever the quality of both system is adequate. This way, high-quality pose estimations are available even if the AR-Toolkit markers are not recognized in the camera image. By sorting the tracking systems based on the max. possible quality in descending order and an optional per-tracking system scalar parameter a_x to fine-tune the quality (defaulting to $a_x = 1.0$) the final overall quality Q_f is calculated with the following equation:

$$Q_f = Q_1 \cdot a_1 + (1 - Q_1) \cdot [Q_2 \cdot a_2 + (1 - Q_2) \cdot [\dots [Q_{n-1} \cdot a_{n-1} + (1 - Q_{n-1}) \cdot a_n] \dots]] \quad (2)$$

Thereby the tracking system 1 with quality Q_1 is the system with the best quality on average.

In a similar way, the merged final pose estimation T_f can be calculated. When multiple tracking systems provide reliable information, a smooth fusion of the—most likely—different pose-estimation results is used. This ensures a frame-to-frame coherent positioning of the augmented visualizations to suppress noticeable translational or rotational shifts that might occur whenever the system selects tracking data from a different system than in the previous frame. The system therefore makes use of per-tracking system quality Q_x to interpolate between tracking systems transformations T_x to find the final result T_f : The orientational part of a pose estimation is represented as a quaternion and a spherical-linear interpolation is applied, for the translational part a simple linear interpolation is used. Both interpolations are controlled by the quality of the corresponding tracking system (Q_x). The following equation describes the computation, where T_x is written as a substitution for translational and orientational part of a pose estimation of the corresponding tracking system x .

$$T_f = Q_1 \cdot T_1 + (1 - Q_1) \cdot [Q_2 \cdot T_2 + (1 - Q_2) \cdot [\dots [Q_{n-1} \cdot T_{n-1} + (1 - Q_{n-1}) \cdot T_n] \dots]] \quad (3)$$

Both variables Q_f and T_f can then be used to control and select an adequate visualization technique.

4.2 Context-Based Visualization Selection and Adjustment

Context information composed of the two criteria Q_f and T_f is used to build a very simple reasoning system to evaluate the most appropriate technique applied for data visualization. Using context data to address the challenge of an automatic visualization selection is especially beneficial within Augmented Reality systems since this way, the human-computer interaction is minimized leading to a natural way of interaction within the real world, like changing the point of view by repositioning a real-world camera.

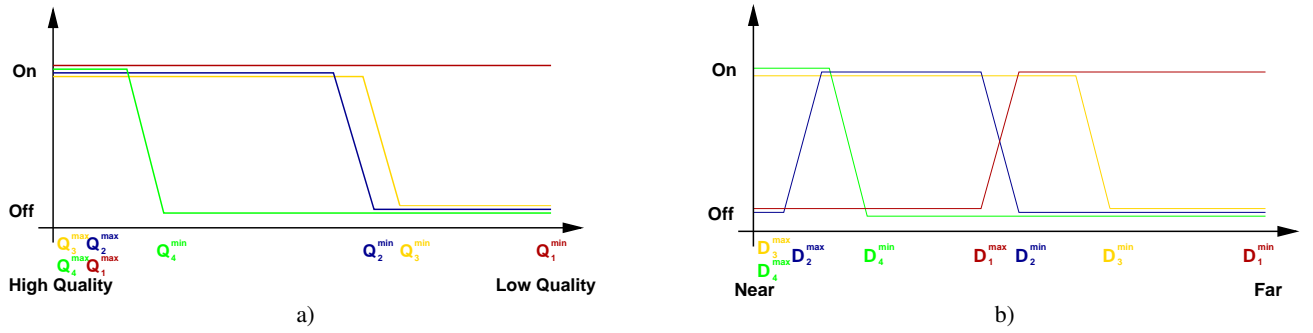


Figure 4: Exemplary threshold values for the selection visualization techniques; a) based on tracking quality, b) based on the distance.

The final transformation T_f is transformed to a distance measurement D of the camera relative to the center of interest, in the prototype we used the center of the flow field. To control the selection of different techniques, quality thresholds Q_v^{min} , Q_v^{max} and distance thresholds D_v^{min} , D_v^{max} are defined for each available visualization technique v . For the presented scenario we focus on flow visualization techniques described in Section 3. Techniques presented in Subsection 3.2 require highly accurate positioning, therefore the min. quality threshold Q_v^{min} for these visualizations have to be above the standard quality 0.8 of the A.R.T. system (i.e. not the enhanced quality after synchronization). In contrast, if the pose estimation is very inaccurate, i.e. below the low-quality threshold Q_v^{min} the system reverts to an alignment-tolerant technique, described in Subsection 3.1. This way, users are not confronted with noticeable misaligned information augmentation. In addition, a fine-grained control of the visualization is achieved with the context information of distance D . If the entire flow field is viewed from a distance of, e.g., 1.5m and greater, visualizations that present fine structures of the flow on the surface—like LIC—are inappropriate since it can be assumed that the user tries to get an impression of the overall behavior of the flow. Simpler techniques like streamlines or streamribbons can nicely present the overall flow, but fail to show detailed structures.

The described general approach also supports the use of multiple visualization techniques concurrently, simply via overlapping quality/distance ranges for the different techniques. A behavior where streamlines are used to visualize an overview of the flow field for distant views, and additional streamribbons are blended in if the view gets closer to reveal details of the flow can easily be modeled. The diagrams in Figure 4 show an example setup for the control of multiple visualization techniques; the left illustration shows the dependency on the experienced tracking quality while the right diagram depicts the camera-distance dependency. Another benefit of combined visualizations is that users are not distracted by suddenly changing illustrations, transitions are experienced smoother if at least one component of the entire visualization is maintained. Further, blending helps to avoid strong popping artifacts for added/removed visualization aspects. The blending cannot be controlled by the tracking quality as the quality might stay constant during a technique transitions, therefore a timer-based blending function is used. To avoid frequent technique changes, a hysteresis loop is automatically generated for the thresholds to abandon frequent technique changes whenever the values Q_f or D are similar to their respective thresholds.

Some alignment-tolerant techniques can additionally be controlled based on the quality of the pose estimation. As mentioned in Subsection 3.1 multiple sets of streamlines/ribbons are generated in a pre-processing step. With increasing tracking quality Q_f , additional line sets with less distance to the modeled real-world geometry can be activated. This is similar to level-of-detail techniques

where a set of alternative representations is pre-computed and selected during the visualization, e.g. based on distance to the camera.

The here presented scenario can only show a small subset of the possible functionality. If available, further context attributes—e.g. most important aspects of the flow, speed of movement of the mobile user, or client device aspects—can be integrated to enhance the selection and configuration of visualizations.

5 Results

The prototype implementation is based on OpenSceneGraph [12]. The visualization techniques are fully integrated as independent scene-graph nodes, this way existing techniques for OSG can easily be integrated and geometry models for real-world scenarios can be read via available OSG file importers. As mentioned before, for the implementation of the system a tablet PC has been equipped with markers and a camera (Figure 1a). The camera supports a resolution of 640x480 at about 30 frames per second (fps). The display of the tablet is configured for 1024x768 pixels and the prototype application is always executed in full-screen mode. For the scenario a simple flow channel was build physically and also modeled as 3D geometry model. We used a single simulation step of a flow computation that was executed for the described scenario for evaluation. The current implementation of the prototype supports the following techniques to visualize the flow and its attributes:

- **Streamlines.** Illuminated streamlines with halos to enhance the contrast relative to the video background.
- **Streamribbons.** Ribbons to convey curl of the flow with magnitude of velocity mapped to color.
- **Surface-Color Mapping.** Real-world surfaces are overlaid with, e.g., color-coded magnitude of velocity.
- **Line Integral Convolution.** LIC executed on the modeled real-world geometry of the scenario.

In addition, methods that render directly onto real-world surfaces can be blended with the underlying video background based on arbitrary context data. The prototype makes use of this feature and renders surface-color mapping only if the velocity is higher than a user-defined threshold. The configuration used to control the context-aware usage of the techniques is show in Figure 4; a) shows the dependency on the tracking quality and b) shows the distance-based behavior.

The utilized visualization techniques are designed to maximize GPU usage to free CPU resources used by the AR-Toolkit library to compute pose estimations. Although the first three techniques execute at moderate frame rates on the prototype hardware the fourth technique cannot be used due to limited OpenGL support of the integrated graphics hardware. Therefore, we used a simple remote

rendering setup to transmit the images—generated on a notebook computer with advanced OpenGL hardware support—to the mobile prototype hardware. Note that we chose to use a notebook as render server on purpose to show the practical feasibility for up-to-date mobile clients.

The benefit of alignment-tolerant visualization techniques is depicted in Figure 5. The left image shows a dense flow representation directly on a real-world surface. This way, the misalignment can easily be seen and will obviously distract users while examine flow fields. In contrast, the right image presents an alternative alignment-tolerant technique that generates real-world independent geometry to present the flow. Still, the error in the alignment is present but much less noticeable and, therefore, does not disturb the user’s immersion.

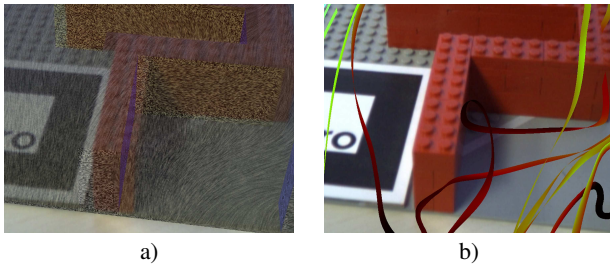


Figure 5: AR flow visualization with misaligned tracking: a) Dense flow representation using Line Integral Convolution on surfaces, hence users are strongly distracted by the alignment error; b) shows an alignment-tolerant visualization of the same flow with same tracking data using streamribbons and, therefore, the alignment error is not noticeable.

Hiding the misalignment might present false spatial relation of flow-simulation data and real-world geometry without feedback to the user. In flow simulation, often the applied discretization grid is at a much coarser resolution than the perceived alignment error. In the examples we used $256 \times 32 \times 256$ cells for the simulation, therefore a misalignment seen in Figure 5 is still within 1-2 simulation cells. Especially, if streamlines are used to present an overview of the entire flow, a coarse simulation resolution is applicable as fine details cannot be efficiently presented with streamlines. Nevertheless, if the pose estimation is very inaccurate and therewith the alignment error, e.g., is higher than 4-6 simulation cells (approx. one Lego peg) the misalignment should no longer be hidden as users might interpret the data incorrectly. In such conditions the system should inform users that the perceived visualization is based on a misaligned tracking. Optionally, graphical context information could be rendered to give clues about the assumed alignment as presented by Robertson and MacIntyre [13].

The upper row of illustrations in Figure 6 show snapshots of the system’s behavior when the camera approaches the flow field (images from left to right). Note that some intermediate states are left out, however the main aspects are captured by the illustrations. The images show that streamlines are used to present an overview of the flow for far away views. Views at mid-distance present the flow using streamribbons and surface-overlaid color coding. In the nearest visualization a LIC technique is added to convey the flow direction in fine details.

For comparison, the bottom row in Figure 6 also shows the behavior of an approaching viewer but with a low level of tracking accuracy. For in-between configurations—i.e. at moderate tracking quality—the visualization techniques are setup differently so that, e.g., additional streamlines and streamribbons with less distance to the geometry are rendered and the visualization is combined with color-coded surface augmentations. The first shown image is similar to the high-quality case; the second image shows streamribbons

without color mapped surfaces, since the tracking is not accurate enough. The final image shows a view at near distance also using streamribbons, since LIC requires a much higher accuracy.

During first preliminary tests, non-expert users were surprised by the simple, natural way of flow-field exploration. The focus on natural interaction, i.e. orienting and moving the tablet PC freely, helped users greatly to automatically understand and make use of all the features provided by the system. The use of further context information—besides position and orientation—to control the visualization was accepted by the candidates and used to examine different aspects. However, the relation of available tracking quality and utilized visualization technique was not obvious to the users which is an indication that the misalignments cannot easily be seen when the proposed alignment-tolerant methods are used.

For AR applications highly interactive frame rates are required in order to keep the real-world view and interaction—i.e. movement of the prototype—consistent with the augmentation. The overall performance of the system depends on the currently activated visualization method(s) (GPU load) and the utilized tracking system(s) (CPU load). Table 1 shows an average performance overview for different setups executed on two different machines. A notebook equipped with an Intel Core2Duo CPU running at 2.2GHz and a Nvidia GeForce 8600M GPU and a desktop machine with an Intel Pentium 4 CPU running at 3.4GHz and a Nvidia GeForce 7800GTX GPU were used as test machines. The timings given in Table 1 are measured in frames per second at a resolution of 1024×768 pixels.

Visualization Technique	Notebook	Desktop
Streamlines	(90)	(90)
Streamribbons	(90)	(90)
Surface-Color Mapping	(90)	(90)
LIC on surfaces	45	42
LIC and Streamribbons	43	39

Table 1: Performance of the AR prototype using various visualization techniques running on different machines. The measurements show average frames per second at a view port of 1024×768 .

For the measurements the image capturing was disabled as otherwise the frame rates are limited to 30fps of the camera-image updates. In addition, at 90fps there is another limitation based on tracking-system threads and threads executed by the utilized scene graph, therefore the measured 90fps for streamlines, streamribbons, and surface-color mapping are expected to be higher.

6 Conclusion

A technique to cope with the still existing major problem of highly accurate tracking, required by most Augmented Reality applications, is shown in this paper. We focus on flow visualization and utilize existing methods that are modified to generate alignment-tolerant visualization techniques. These can be used to augment real-world views already at a medium-accurate tracking. The presented system makes use of available context information to automatically select and configure the most adequate visualization methods for different situations. This way, the distraction of users because of misaligned or inappropriate augmentations is minimized. A fusion and synchronization of multiple tracking systems is used to increase the overall tracking accuracy to allow the usage of more sophisticated state-of-the-art visualization techniques.

Many aspects of the presented setup help to achieve a high acceptance by users: The utilized tablet PC is used as an AR window and does not deter users, like e.g. head-mounted displays. The AR typical requirement of real-time response is achieved via GPU supported visualization methods. For large-scale AR setups a

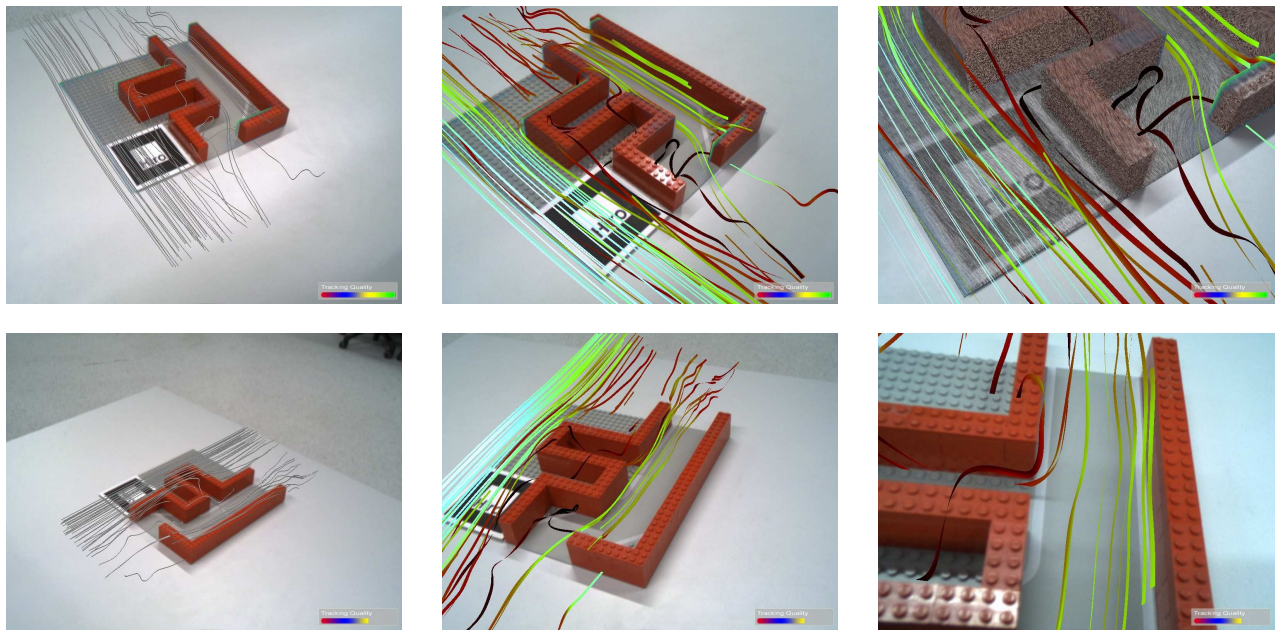


Figure 6: Snapshots of visualizations for an approaching viewer (from left to right). The upper row of illustrations show renderings if highly accurate pose estimations are available, the bottom row depicts visualizations for non-accurate tracking conditions.

major advantage is the ability to make use of multiple positioning systems. Systems with different accuracies can be used to cover a large area with dedicated spots of highly accurate tracking, where the context-aware system will automatically select the most appropriate visualization.

In future we will focus on further context aspects that can be used to enhance the control of visualizations. In practical setups a challenging problem is often the environmental lighting condition, especially in outdoor scenarios. Not only computer-vision techniques are problematic (e.g. camera-image capture for AR-Toolkit), but also display devices cannot provide enough intensity/contrast to render high-quality images. We will therefore try to adapt the visualization and rendering technique to the environmental lighting conditions, provided as additional context information.

References

- [1] A.R.T. Advanced Realtime Tracking GmbH. <http://www.ar-tracking.de/>, 2007.
- [2] Enylton Machado Coelho, Blair MacIntyre, and Simon J. Julier. OS-GAR: A scene Graph with Uncertain Transformations. In *Proceedings of the 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 6–15. IEEE Computer Society, 2004.
- [3] Stephen DiVerdi and Tobias Höllerer. Image-space Correction of AR Registration Errors Using Graphics Hardware. In *Proceedings of the IEEE Virtual Reality Conference (VR 2006)*, pages 241–244. IEEE Computer Society, 2006.
- [4] Gordon Erlebacher, Christoph Garth, Robert S. Laramée, Holger Theisel, Xavier Tricoche, Tino Weinkauff, and Daniel Weiskopf. IEEE Visualization 2006, Tutorial on Texture and Feature-Based Flow Visualization Methodology and Applications. <http://www.hagen.informatik.uni-kl.de/vis06-tutorial/>, 2006.
- [5] J. Fischer, D. Bartz, and W. Straßer. Artistic Reality: Fast Brush Stroke Stylization for Augmented Reality. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 155–158, 2005.
- [6] H. Kato and M. Billinghurst. The AR-Toolkit. <http://www.hitl.washington.edu/artoolkit/>, 2007.
- [7] Georg Klein and Tom Drummond. Sensor Fusion and Occlusion Refinement for Tablet-Based AR. In *Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 38–47. IEEE Computer Society, 2004.
- [8] U. Lang and U. Wössner. Virtual and Augmented Reality Developments for Engineering Applications. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS*, 2004.
- [9] Oliver Mattausch, Thomas Theußl, Helwig Hauser, and Eduard Gröller. Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics (SCCG '03)*, pages 213–222. ACM Press, 2003.
- [10] Erick Méndez, Denis Kalkofen, and Dieter Schmalstieg. Interactive context-driven visualization tools for augmented reality. In *In proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*, pages 209–218. IEEE Computer Society, 2006.
- [11] Joseph Newman, István Barakonyi, Andreas Stürzinger, and Dieter Schmalstieg. Wide Area Tracking Tools for Augmented Reality. In *Proceedings of Advances in Pervasive Computing 2006*, pages 143–146, 2006.
- [12] Robert Osfield, Don Burns, and Others. Open scene graph. <http://www.openscenegraph.org>, 2007.
- [13] C. M. Robertson and B. MacIntyre. An Evaluation of Graphical Context as a Means for Ameliorating the Effects of Registration Error. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, 2007.
- [14] M. Schlemmer, I. Hotz, B. Hamann, F. Morr, and H. Hagen. Priority streamlines: a context-based visualization of flow fields. In *Proceedings of EuroVis 2007, Data Visualization*, 2007.
- [15] D. Weiskopf and T. Ertl. A hybrid physical/device-space approach for spatio-temporally coherent interactive texture advection on curved surfaces. In *Proceedings of Graphics Interface*, pages 263–270. IEEE Computer Society, 2004.
- [16] Malte Zöckler, Detlev Stalling, and Hans-Christian Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proceedings of the 7th conference on Visualization '96 (VIS 96)*, pages 107–ff. IEEE Computer Society Press, 1996.