Real-Time Video Annotations for Augmented Reality

Edward Rosten, Gerhard Reitmayr, and Tom Drummond

Department of Engineering, University of Cambridge, Cambridge CB1 2PZ, UK
{er258, gr281, twd20}@cam.ac.uk

Abstract. Augmented reality (AR) provides an intuitive user interface to present information in the context of the real world. A common application is to overlay screen-aligned annotations for real world objects to create in-situ information displays for users. While the referenced object's location is fixed in the view the annotating labels should be placed in such a way as to not interfere with other content of interest such as other labels or objects in the real world. We present a new approach to determine and track areas with less visual interest based on feature density and to automatically compute label layout from this information. The algorithm works in under 5ms per frame, which is fast enough that it can be used with existing AR systems. Moreover, it provides flexible constraints for controlling label placement behaviour to the application designer. The resulting overlays are demonstrated with a simple hand-held augmented reality system for information display in a lab environment.

1 Introduction

Augmented reality (AR) is an excellent user interface for mobile computing applications, because it supports an intuitive display of information. In an AR environment, the user's perception of the real world is enhanced by computer-generated entities such as 3D objects, 2D overlays and 3D spatialised audio [1]. Interaction with these entities occurs in real-time providing natural feedback to the user. A common application for augmented reality is information browsing. Annotations to real world objects are presented to the user directly within the view of the environment. Typical example are labels containing textual or pictorial information [2]. Such applications are especially interesting in the context of mobile augmented reality, where a user can roam a large area and subsequently request information on many objects.

Annotations can be general information displays such as heads-up displays that stay in fixed locations on the screen. Or they may be associated with objects in the view and appear close in the view. In the latter case they usually move with the objects or are connected to them by follower-lines which make the user aware of the relationship. However, placing annotations into the user's view is not trivial. To avoid distraction annotations should not move or jitter. They should also not overlap scene features or each other. Finally, readability of text annotations depends strongly on background colour and texture.

Automatically placing annotations is more difficult. Models of the environment may be limited to explicitly tracked objects and may not include a general description of the environment. Even where such a world model exists, it usually cannot capture fine details that determine the background clutter in images. Also, the application model may

G. Bebis et al. (Eds.): ISVC 2005, LNCS 3804, pp. 294–302, 2005.

(a) (b)

Fig. 1. A hand-held augmented reality system for information browsing using labels attached to markers and "screen-stabilized" labels. The system determines the label locations in real-time in such a way that overlap with interesting parts of the image is minimised.

not even contain all possible objects of interests or have means to track them: for example, a person walking into view can be considered interesting to the user and should therefore not be occluded. However, mobile AR systems typically have no possibility of tracking all persons or mobile objects they encounter. To create useable information overlays in all such situations, image based methods are required.

To enable automated annotations for resource-constrained systems and unmodelled environments, we present a new, fast method which finds visually uninteresting areas in live video and hence determines candidate regions for label placement. The basic idea is to describe the fitness of a certain label location in terms of the number of occluded image features were the label to be placed in that location. The fitness distribution over the entire image is computed from features detected in each frame and tracked in a non-parametric filter framework over time (see section 3). Application designers can further modify the fitness distribution for each label to enforce constraints (see section 4) such as proximity to a location in the image or placement in one of several distinct locations such as image corners. The whole algorithm requires 5ms per frame (on a 1GHz Pentium-M) and is therefore suitable for real-time operation and integration into more complex systems (such as mobile augmented reality applications) as demonstrated in section 5.

2 Related Work

Traditional label placement research has focused on the problem of arranging large numbers labels on static display without overlap and unlimited computational time. Moreover, complete information about the display and any constraints on the labels is assumed as well. Many approaches exist and the Map-Labeling Bibliography [3] hosts an extensive selection. A good overview of typical approaches is given by Christensen et al. [4]. Azuma et al. [5] evaluate such label placement strategies for augmented reality displays both statistically and with a user study. Our work supplements this work by introducing a method to derive constraints for labels based on the visual content of

the view. Therefore, we are not concerned with overlap between labels themselves, but rather with overlap of labels and interesting features of the environment. Once candidate locations of labels are known, global label placement strategies could be employed to reduce occlusion between labels.

Video annotations such as subtitles have been used for many years in the broadcasting industry. Typically such annotations are placed offline in a manual process and at fixed locations. In recent years, real time, world aligned annotations for sports casts have become possible such as the display of the first down line in American football [6] or live information for Nascar races [7]. These solutions rely on the accurate and expensive tracking of camera parameters and accurate models of the environment and pixel colours to achieve high precision. Moreover, manual fine-tuning by operators ensures robustness of the systems.

Information browsing is a common application for augmented reality systems. Feiner et. al [8] annotated persons and objects with application windows in an optical see-through setting. The NaviCam [2] demonstrated labels for objects tracked with optical fiducials. Simple rendering of annotations for multiple objects leads to clutter on the screen and overlap between labels themselves and other objects of interest. One approach is to reduce the number of labels by applying a filtering scheme [9] to select information based on properties of the tasks and the users as well as on proximity. Another promising approach is to directly manage the 2D display space and place labels only in regions that are not interesting. Bell et al. [10] describe a view management system that keeps track of free areas on the display. A 3D world model and tracking of real objects is required to accurately project the objects and environment back into the view and update the occupied areas as the user moves through the environment.

Other research investigates the properties of labels in augmented reality displays. Gabbard et al. [11] compare user performance in reading differently coloured text labels in optical see-through systems over various backgrounds. In contrast to that Leykin and Tuceryan [12] use a machine learning approach to automatically determine the readability of text over textured backgrounds. The information gained by classification can be incorporated in to our method by providing additional constraints.

Automatic placement without knowledge of the environment was attempted by Thanedar and Höllerer [13] to produce annotations for video sequences. Their approach selects candidate regions based on image properties such as absence of motion and uniformity of colour. The location is optimised over a series of frames, also taking future frames into account; it is therefore not suitable for real-time operation. In contrast to this approach, our method is causal and sufficiently optimised that it can be used as an additional component in a larger real-time system.

3 Tracking Candidate Locations

Our method finds uninteresting parts of the image in order to place labels without obscuring interesting parts of the image. To do this, a distribution of feature density is calculated in the current video frame. Areas with a high feature density are assumed to contain information interesting to the user. Labels are placed in the frame such that the integral of the density distribution over the label area is minimised. In principle any

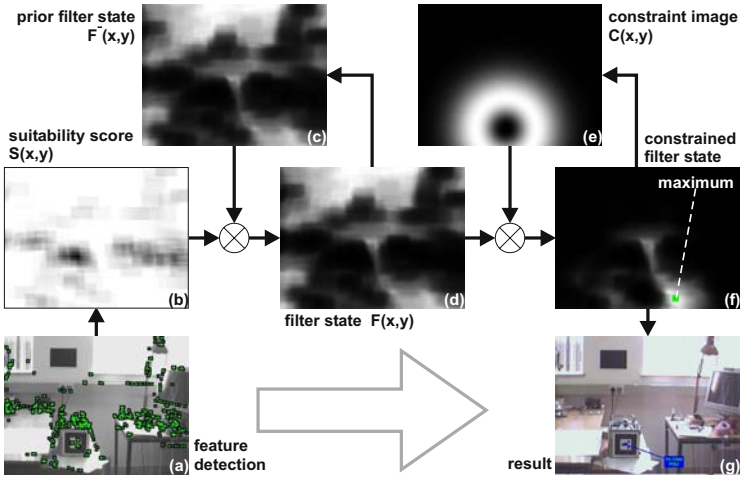

Fig. 2. Overview of the processing steps. (a) shows the detected features; the computed suitability score (b) updates the prior filter state (c) to arrive at the current filter state (d); a constraint image (e) is multiplied with it to compute the constrained filter state (f) and select the maximum location. (g) shows the resulting label placement. The maximum location is fed back to the constraint image as a location prior for the next frame.

feature detector could be used, for example the Harris [14] or SUSAN [15] detectors, or even edgel detectors. In our experience, corner detectors yield visually better results than edge detectors. Our method has a strong emphasis on low compute cost, so we use the very efficient FAST feature detector, presented in Appendix A.

For each label to be placed in the image a placement cost distribution $P(x, y)$ is calculated in the following manner. We define $P(x, y)$ to be the number of features occluded by the label if the label is placed at x, y. Our implementation uses an integral image [16] to quickly calculate this for each label. Locations with the lowest cost are the best locations for labels. However, the placement cost distribution is sensitive to noise. This can result local jitter of the optimal location or frequent large scale jumps if several locations are near optimal. Both effects are undesirable and can be avoided by filtering over time (see Fig.2 for an overview).

A non-parametric filter state $F_n(x, y)$ describes the suitability of a location for placing a given label within a frame n. The location of the maximum of $F_n(x, y)$, written as $(\overline{x}, \overline{y})$, describes the optimal location for the label. The first step is to update the last filter state for frame $n - 1$ with process noise by adding a constant c, yielding a prior $F_n^-(x, y)$. The prior is then updated with the suitability $S(x, y)$, defined as

$$S(x, y) = 1 - P(x, y)/\max_{x,y}(P(x, y)). \tag{1}$$

Then we multiply the suitability score with the prior state and normalise by a factor z. Therefore, the update step is

$$F_n(x, y) = F_n^-(x, y)S(x, y)/z. \tag{2}$$

The location $(\overline{x}, \overline{y})$ of the maximum of the filter state $F_n(x, y)$ at step n is then used as the optimal placement for a label, in the absence of any further information (see Section 4). The process noise parameter c controls the flexibility of the filter. Larger process noise results in a more responsive filter that quickly captures major changes in the distribution, but allows the label to jitter.

4 Placement Control

The filter state generated by the tracking process describes the overall fitness of locations for label placement. However, real applications require more control over the label location. Depending on the type of label different constraints need to be enforced on the label position. Thanedar and Höllerer [13] give a short taxonomy of possible labels in video overlays. Our method supports all of these label types.

To enforce a constraint on the location of a label, the application designer specifies a constraint image $C(x, y)$ that describes the suitability distribution of placing a label at a certain location. The constraint image is then multiplied with the filter state to yield a constrained filter state. The location of the maximum of the latter is the optimal placement for the associated label.

Such constraint images can either be static or changing over time depending on the type of label. For example, to place an annotation close to a tracked object an application multiplies a ring-like distribution around the object location with the filter state (see Fig.2). The resulting maximum and hence label will lie in an uninteresting part of the image near, but not too close, to the object being labelled. A directional constraint places the label in the approximate direction of the visible arrow on the marker (see Fig.3).

To place "screen-stabilised" (fixed) labels in one of a set of possible locations, a static constraint image is constructed that only contains support for the desired label locations. Again, the maximum of the resulting constrained suitability distribution is one of the chosen locations.

Even with these constraints, if large areas of the image are uninteresting, then there will be many equally suitable locations for the label, which can cause the label to jump around. To prevent this, a constraint is used which increases the value of the constrained filter at the previous location of the label. If this position is stabilised relative to the marker position, then this will bias the label towards moving smoothly around with the marker. If the absolute position of the label is used, the bias will be towards the label staying in the same place on the screen.

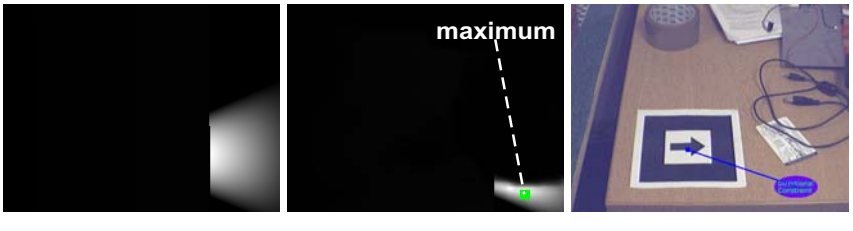

Fig. 3. Example of a directional contraint. The first column shows the constraint images, the second the constrained filter state, and the last the resulting location.

Avoiding overlap of multiple labels also becomes possible. A simple greedy algorithm can operate on a ranked list of labels: the first label is placed at the maximum location of its filter state (or a constrained filter state). The states for the remaining labels are then modified to discourage placement of subsequent labels in locations that could result in overlap with the first one. The modified state is then the input for placing the next label. The iterative modification of the filter states accumulates the information about all placed labels and constrains the placement of new ones to unused areas.

5 Results

To demonstrate the results of our automated label placement method, we created a small information browsing application for a hand-held AR system. A tablet PC with a 1GHz Pentium M processor and a Firewire camera mounted to its back becomes a lens-like tool to view annotations on real objects and locations in our lab (see Fig.1). To track objects and locations we employ the ARToolkit [17] library and a set of markers placed in the environment on interesting objects such as a printer. The system works at video frame rate of 30 fps recording the full 640×480 image but operating on quarter-sized images (320×240 pixels) for feature detection and filter representations. In this configuration, the system uses 17% of the available computing time.

When a marker becomes visible in the camera view, a corresponding label is placed next to it in an area free of any interesting features (see Fig.4 (a)). The distance to the marker is controlled by a circular constraint as described in section 4. The label is connected by a follower-line to the centre of the marker to disambiguate the association. As the tablet is moved about labels typically stay in the same location in the window to avoid unnecessary movement. However, if a label starts to occlude interesting image features it jumps to a different location with less overlap (see Fig.4 (b,c)). In general labels appear stable but can change their location instantly without any interpolation between locations, if required. The display also contains screen-stabilised information on the lab which is placed in one of the four screen corners.

Table 1 shows the average time spent calculating label placement for our demonstration system. All of the per-label costs are streaming operations on large quantities of pixel data, and so could be further optimised exploiting a streaming instruction set such as SSE2. Nevertheless, our portable C++ implementation exhibits good performance.

(a) (b) (c) (d)

Fig. 4. Example of the dynamic behaviour. (a) A label is placed in an uninteresting position. As a person moves into the view (b), the label jumps to another location (c). The weak position prior prevents the label from quickly jumping back to the previous position (d).

Table 1. Timings for various steps of the algorithm. A single precomputed constraint is used.

		Time (ms)
Per-frame cost	Feature detection	1.76
	Integral image	0.65
Per-label cost	Suitability measurement	1.03
	Filter measurements	0.63
	Insert constraint	0.51
Total cost	For one label	4.58
	For n labels	$2.41 + 2.17n$
Cost per frame	over 10 frames	0.46

For real-time operation the label placement does not need to be computed for each frame, since large changes in label position happen infrequently. Therefore, another performance improvement consists in computing label placements only for every n^{th} frame and distribution computation over the intermediary frames to reduce the per-frame computation cost to $1/n^{th}$ of the total.

6 Conclusions

Our method implements a straightforward, yet flexible approach to determining annotation locations for overlays of live video images. Processing time per frame is minimal making it appropriate for inclusion into mobile and hand-held augmented reality systems that are constrained in terms of computing power. The bulk of the processor time is still left for other tasks such as tracking or interaction management. The non-parametric tracking of candidate locations can be tuned to select between slow or fast reaction to image changes or to prefer past locations over new ones. The complete modelling of state via a non-parametric representation captures the problem's multi-modal properties and allows for instant jumps to different locations. The inclusion of constraints gives application designers a flexible way to define label behaviour.

The underlying assumption that features appear predominantly in visually interesting areas is the key to the simplicity of the method. The assumption breaks down for feature rich areas of no importance to the user or task at hand. However, as long as there are some image regions with few features the label will be placed in a good position.

Direct comparisions to other methods described in section 2 are difficult, because none exhibit the same properties. Either they use models that include all interesting objects that should not be occluded, or they do not operate causally in that they optimise label locations using frames yet to be shown. The later can avoid surprising jumps, but is not applicable in real-time.

The presented approach is also applicable to general video feeds, but our work focuses on mobile augmented reality applications. Here the real-time constraints, limited availability of tracking solutions for model based annotation placement and ubiquitous use of video see-through displays match the features of the proposed method very closely. However, the method could also be adapted to optical see-through displays through the use of a calibrated camera that captures images that match the user's view.

An example video of a walk through our lab with annotations is available in the supplementary videos, and the FAST corner detection code is available from `http://savannah.nongnu.org/projects/libcvd`.

A FAST Feature Detection

We present here the FAST (Features from Accelerated Segment Test) feature detector. This is sufficiently fast that it allows on-line operation of the label placement system. A test is performed for a feature at a pixel p by examining a circle of 16 pixels (a Bresenham circle of radius 3) surrounding p. A feature is detected at p if the intensities of at least 12 contiguous pixels are all above or all below the intensity of p by some threshold t. This is illustrated in Fig.5. The test for this condition can be optimised by examining pixels 1 and 9, then 5 and 13, to reject candidate pixels more quickly, since a feature can only exist if three of these test points are all above or below the intensity of p by the threshold. With this optimisation the algorithm examines on average 3.8 pixels per location on a sample video sequence.

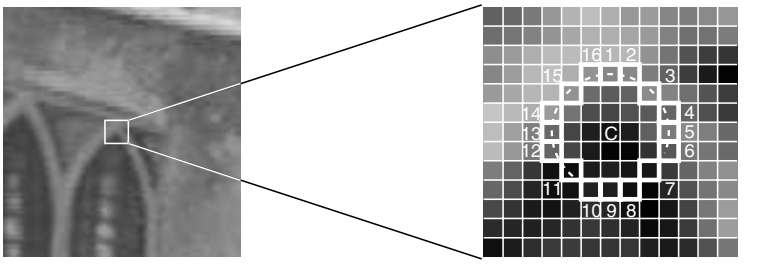

Fig. 5. FAST Feature detection in an image patch. The highlighted squares are the pixels used in the feature detection. The pixel at C is the centre of a detected corner: the dashed line passes through 12 contiguous pixels which are brighter than C by more than the threshold.

References

1. Azuma, R.T.: A survey of augmented reality. Presence - Teleoperators and Virtual Environments **6** (1997) 355–385
2. Rekimoto, J.: Navicam: A magnifying glass approach to augmented reality. PRESENCE - Teleoperators and Virtual Environments **6** (1997) 399–412
3. The Map-Labeling Bibliography. http://i11www.ira.uka.de/ awolff/map-labeling/bibliography/ (2005)
4. Christensen, J., Marks, J., Shieber, S.: An empirical study of algorithms for point-feature label placement. ACM Transactions on Graphics **14** (1995) 203–232
5. Azuma, R., Furmanski, C.: Evaluating label placement for augmented reality view management. In: Proc. ISMAR 2003, Tokyo, Japan, IEEE (2003) 66–75
6. Sportvision: 1st & Ten. http://www.sportvision.com/ (2005)
7. Sportvision: Race F/X. http://www.sportvision.com/ (2005)
8. Feiner, S., MacIntyre, B., Haupt, M., Solomon, E.: Windows on the world: 2D windows for 3D augmented reality. In: Proc. UIST'93, Atlanta, GA, USA (1993) 145–155

9. Julier, S., Lanzagorta, M., Baillot, Y., Rosenblum, L., Feiner, S., Höllerer, T.: Information filtering for mobile augmented reality. In: Proc. ISAR 2000, Munich, Germany, IEEE and ACM (2000) 3–11
10. Bell, B., Feiner, S., Höllerer, T.: View management for virtual and augmented reality. In: Proc. UIST'01, Orlando, Florida, USA, ACM (2001) 101–110
11. Gabbard, J.L., Swan II, J.E., Hix, D., Schulman, R.S., Lucas, J., Gupta, D.: An empirical user-based study of text drawing styles and outdoor background textures for augmented reality. In: Proc. IEEE VR 2005, Bonn, Germany, IEEE (2005) 11–18
12. Leykin, A., Tuceryan, M.: Automatic determination of text readability over textured backgrounds for augmented reality systems. In: Proc. ISMAR 2004, Arlington, VA, USA, IEEE (2004) 224–230
13. Thanedar, V., Höllerer, T.: Semi-automated placement of annotations on videos. Technical Report #2004-11, UC, Santa Barbara (2004)
14. Harris, C.J., Stephens, M.: A combined corner and edge detector. In: Proc. of the 4th Alvey Vision Conference. Number 4, Manchester, UK, Alvery Vision Conference (1988) 147–151
15. Smith, S., Brady, J.: SUSAN - a new approach to low level image processing. Int. Journal of Computer Vision **23** (1997) 45–78
16. Viola, P., Jones, M.: Robust real-time object detection. Int. Journal of Computer Vision (2002)
17. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferenencing system. In: Proc. IWAR'99, San Francisco, CA, USA, IEEE CS (1999) 85–94