Dynamic Adaptation of Projected Imperceptible Codes

Anselm Grundhöfer, Manja Seeger, Ferry Häntsch, and Oliver Bimber
Bauhaus-University Weimar*

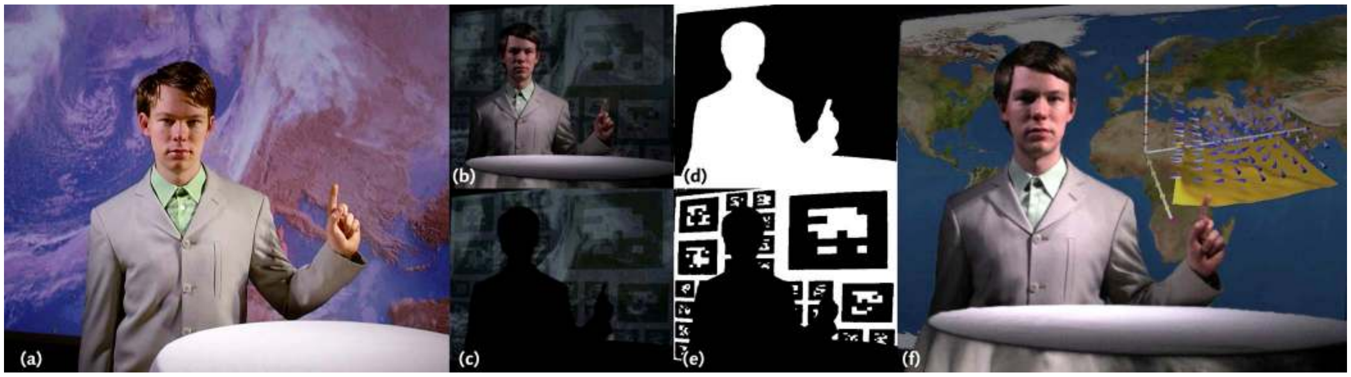


Figure 1: A TV studio mock-up with back projection encoding adaptive imperceptible patterns (a), images captured by a synchronized camera at 120 Hz (b, c), computed foreground matte from real-time flash keying (d), extracted multi-resolution marker pattern for in-shot camera pose estimation (e), and composite frame with virtual background and 3D augmentation (f).

Abstract

In this paper we present a novel adaptive imperceptible pattern projection technique that considers parameters of human visual perception. A coded image that is invisible for human observers is temporally integrated into the projected image, but can be reconstructed by a synchronized camera. The embedded code is dynamically adjusted on the fly to guarantee its non-perceivability and to adapt it to the current camera pose. Linked with real-time flash keying, for instance, this enables in-shot optical tracking using a dynamic multi-resolution marker technique. A sample prototype is realized that demonstrates the application of our method in the context of augmentations in television studios.

Index Terms: H.5.1 [INFORMATION INTERFACES AND PRESENTATION]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.4.8 [IMAGE PROCESSING AND COMPUTER VISION]: Scene Analysis—Tracking

1 Introduction and Motivation

Video projectors can modulate images spatially as well as temporally. Due to limited capabilities of the human visual system, code patterns can be integrated into projected pictorial content that remain invisible to the observers. Supported by a high frequent temporal image modulation, synchronized cameras, however, are able to reconstruct the code. The code patterns can then be used for a variety of applications, such as projector calibration, camera tracking, 3D scanning, and more. An integration via temporal coding requires the projection of each image twice: one containing the actual code information (e.g., by varying the image intensities locally by a certain amount (Δ) - depending on the code) and a second one that compensates the missing image information. The invisibility of a temporally embedded code, however, depends on a variety of parameters that vary dynamically depending on the projected content and on the integrated code pattern. Thus, using a constant contrast window of Δ for coding might lead to the perception of the code.

In this paper we present two main contributions: The first one is an *adaptive imperceptible pattern projection technique* based on a high frequent temporal image modulation that overcomes limitations of existing approaches. The second one is a *dynamic multi-resolution marker tracking method* that –based on our imperceptible pattern projection– ensures a continuous in-shot camera tracking despite possible occlusions and individual camera perspective.

Combined together, these techniques allow displaying an arbitrary projected content that is visible to observers in the familiar way. The poses of synchronized cameras, however, can be estimated with the extracted code patterns. We have also combined our methods with *real-time flash keying* using either a high-speed white-light LED illumination or the temporally coded projection itself.

Finally, we present a prototype system that integrates our techniques into the mock-up of a television studio. We demonstrate camera tracking, keying and the composition of computer generated backgrounds with the recorded image of moderators or actors in an arbitrary studio environment. With this, we want to emphasize the possible applicability of such techniques in real television studios [3] (e.g., hosting a live audience), rather than being limited to virtual sets only.

The remainder of this paper is organized as follows: After a discussion of relevant related work in chapter 2, an overview of our adaptive imperceptible coding technique is presented in chapter 3. We will explain the basic concept and provide details on the image analysis, coding, decoding and code blending approaches. Furthermore, we describe the results of an informal user evaluation that validates our results. Chapter 4 details our adaptive code placement technique in the example context of a TV studio prototype. In addition we show how real-time flash keying is used. All computation steps are summarized in chapter 5. Finally, chapter 6 concludes our work and points out potential areas for future enhancements.

*e-mail:{grundhoe,seeger,haentsch,bimber}@uni-weimar.de

2 Related and Previous Work

The related work discussion in this chapter is organized as follows: An overview over existing projector-based imperceptible coding techniques is presented first in section 2.1. Section 2.2 describes related work that is focused on special marker technologies for optical tracking, while section 2.3 summarizes existing keying technologies. Note, that only techniques are discussed that are most relevant to our approach. A full state-of-the-art review within the different areas is out of the scope of this paper.

2.1 Embedded Imperceptible Pattern Projection

High-frequent temporal modulation of projected images allows integrating coded patterns that are – due to limitations of the human visual system – not perceivable. Binary codes, for example, can be integrated into so-called *code images* by slightly increasing or decreasing pixel intensities by a certain amount Δ depending on the code values. *Compensation images* are computed that visually neutralize the effects of the embedded codes. If a code image and its compensation image are projected in a high speed, the code is not visible. Synchronized cameras, however, are able to reconstruct these codes. This principle has been described by Raskar et al. [20] and is referred to as *embedded imperceptible pattern projection*. Extracted code patterns allow, for instance, the simultaneous acquisition of the scenes' depth and texture for 3D video applications [26, 28].

An advanced technique was presented in [5], where a specific time slot of a DLP projection sequence is occupied exclusively for displaying a binary pattern within a single color channel. Multiple color channels are used in [6] to differentiate between several projection units. However, unless the DLP mirror flip sequences within the chosen time slot are not evenly distributed over all possible intensities (which is not the case in practice) this technique can result in a non-uniform fragmentation and a substantial reduction of the tonal values. Since the patterns are encoded independently of visual perception properties, local contrast reductions and flickering effects should be visible in unfavorable situations, such as low intensities and low spatial image frequencies, as well as during the temporal exchange of the code patterns. The possible intensity modification of each pixel's color also leads to slightly miscolored image regions. Artifacts are diffused in this case using a dithering technique.

In [30] an imperceptible calibration technique for radiometric compensation is presented which uses a high-frequent temporal modulation to apply geometric calibration and to acquire radiometric information of the projection surface. The techniques described in [30] are based on the fundametal coding scheme described in this paper.

Instead of increasing or decreasing the intensity of a coded pixel by a constant amount (such as in [20]) or by an amount that depends purely on technical parameters (such as mirror flip sequences [5]), our method considers the capabilities and limitations of human visual perception for embedding codes. It estimates the *Just Noticeable Difference (JND)* and adapts the code contrast (Δ) on the fly - based on regional properties of projected image and code, such as luminance and spatial frequencies. Thus, only the global image contrast is modified rather than local color values. This ensures an imperceptible coding while providing a maximum of intensity difference for decoding. Yet, it enforces only a small and linear contrast compression. As mentioned above, intensity coding can also be supported with our method, rather than being limited to pure binary patterns, as shown in [30]. Besides a projector linearization, additional calibration is not necessary. The integrated codes can be dynamically exchanged. Because of the fact that an abrupt code transition would be temporary visible for human observers, this is realized by a temporal code blending technique to seamlessly exchange different code patterns.

2.2 Projected, Invisible, and Multi-Resolution Markers

In chapter 4, we present an adaptive camera tracking technique that integrates imperceptible markers into background projections using our invisible coding approach. Thus, they are displayed directly within the field of view of the camera without being directly perceptible by human observers. In contrast to various marker-less tracking techniques [9, 25], marker-based approaches do not rely on the robust detection of natural scene features. If invisible, however, a marker-based tracking can share the advantages of a robust marker-less approach.

Visibly projected markers have been used earlier for geometric projector calibration on planar screens [8]. We propose a dynamic multi-resolution approach to ensure a continuous in-shot camera tracking rather than a projector calibration. In contrast to similar nested marker techniques [24] that embed several code scales in a single printed pattern, our multi-resolution markers are projected and can consequently be automatically exchanged and modified depending on the actual camera pose and possible occlusions of the background projection.

Another imperceptible marker tracking approach was presented by Park et al.[17]. They use a coaxial camera pair to capture the scene with and without infrared (IR)-filter. Thus physical marker codes that are painted with IR-reflecting ink remain invisible and can be detected when illuminating them by IR light.

Our goal is to support camera tracking with dynamically projected markers that are imperceptible for live observers and that are also not visible in the recorded video stream. One key requironment is the ablity to separate foreground from background.

2.3 Keying Techniques

To support a separation of foreground and background pixels a variety of different keying techniques can be applied. Conventional luma keying or chroma keying cannot be used in real environments due to the lack of a static or uniform background. Variations of difference keying, such as flash keying, however, are applicable. While some complex flash keying approaches, such as [23], do not support real-time rates and can only be applied off line, a technique that uses pulsed blue LEDs for real-time flash keying was described in [1]. In combination with our invisible code projection, we apply a difference keying technique that is based on an adapted version of real-time flash-keying [1] for separating foreground and background pixels. However, instead of blue LEDs that provide an additional chrominance masking, we apply pulsed white-light LEDs for creating a uniformly white illumination. We also discuss how the projectors themselves can be used for supporting flash keying instead of installing an additional LED illumination. This, however, assumes that the system setup is equipped with a projector-based illumination framework. While flash keying can be realized with a single camera, two coaxial aligned cameras can support a real-time depth-of-field based keying, similar to [22], and to overcome focus issues in cases of short focal depths. Although we have not implemented a depth-of-field based keying, we do use such a camera system to address different focal depths simultaneously.

3 Dynamic Δ-Coded Temporal Projection

Based on the initial suggestions of embedded imperceptible patterns using high-frequent temporal modulation [20] we have developed an enhanced method that projects and captures encoded images and their complements at a speed of 120 Hz. However, instead of increasing or decreasing the intensity of a coded pixel by a constant amount of the code contrast Δ, we compute the JND and adapt local Δ values on the fly - based on regional image intensity and spatial resolution of projected image and embedded code. This ensures an imperceptible coding while providing a maximum of intensity differences for decoding to avoid the destruction of the code through camera noise or low camera responses. The advantages of

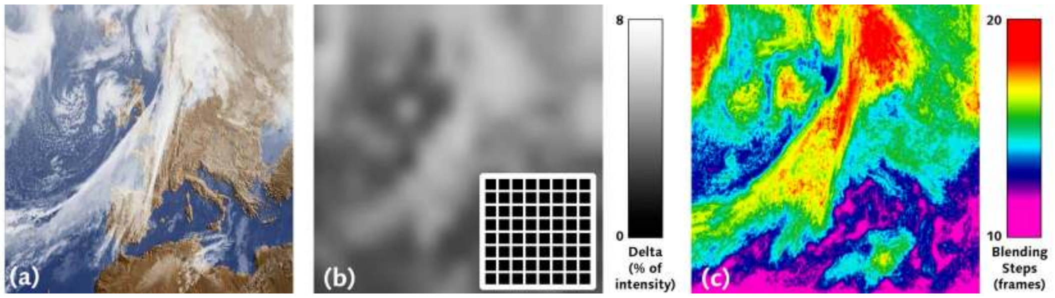


Figure 2: For each pixel of the projected image (a) the largest non-perceivable intensity variation is computed (b - contrast enhanced). Together with the code's local spatial frequency (b - inlay image) locally optimized Δ values are derived. From the same information, the smallest number of required blending steps (c) for dynamic code transitions are derived (c - color coded for each pixel)

this approach compared to previous methods have been discussed in chapter 2.1. In this chapter we explain how binary codes are embedded into arbitrary images based on parameters of human visual perception.

3.1 Static Codes

In the case that a static binary code image *C* is embedded into the displayed original image *O* we simply compute the projected image with *I=O-*Δ and its complement with *I'=O+*Δ. Projecting both images at a speed that is above the *critical flicker frequency* (CFF) [16], a human observer will perceive roughly *(I+I')/2* which approximates *O* (cf. background in figure 1a). Depending on the binary code bit in *C* (i.e., 0 or 1) we decide whether Δ is positive or negative on a per-pixel basis.

To avoid clipping at lower and higher intensity levels when subtracting or adding Δ, *O* has to be scaled. Theoretically a contrast reduction of 2Δ is sufficient. However, for our currently applied projector and camera hardware the brightness of the original image has to be increased by approximately 10% to reduce camera noise in dark regions. Practically speaking, this leads to a maximum contrast reduction of ∼10-20% at the moment. This can be reduced significantly by applying cameras and optics that are less sensitive to noise, or brighter projectors. Compared to other approaches, such as [5] (where large tonal shifts for lower intensities in individual color channels or maximum dynamic range reductions of up to 50% are reported), *O* is linearly scaled in our case. It does not lead to regional color or intensity artefacts.

Synchronizing the camera to the projection enables both images to be captured separately (cf. figures 1b+c). Dividing or subtracting them allows identifying the encoded state per camera pixel (cf. figures 1e): The ratio of both images are above or below one, while the difference of both images is above or below zero - depending on the integrated bit. It is essential that cameras and projectors are linearized to avoid an effect of their transfer or response functions. A gamma correction can be applied after linearization to ensure color consistency. Thus, projecting and capturing *I* and *I'* at 120Hz leads to a perception of *O* and to a reconstruction of *C* at a speed of 60Hz.

Despite the integration of binary codes, our approach allows to embed and reconstruct multiple code intensities at each pixel up to a certain extent. This gives the opportunity to encode more information at the same time. Based on the fundamental techniques described in this paper, an intensity coding for the imperceptible calibration of projector-camera systems has been shown in [30]. Note, that this is not possible with related approaches, such as [5, 20].

One problem with this temporal coding approach, however, is that for fast camera movements *I* and *I'* might be no longer geometrically aligned. Although the misalignment will not be larger than a few pixels (due to the high capturing rate), reconstructing the code bits on a per-pixel basis might still fail due to the geometric misregistration. By applying a 5x5 median filter kernel, however, these small misregistrations can be removed efficiently (cf. figure 3c,d). In our experiments, this simple technique proved to be applicable for a capturing rate of 120Hz - even for fast camera movements.

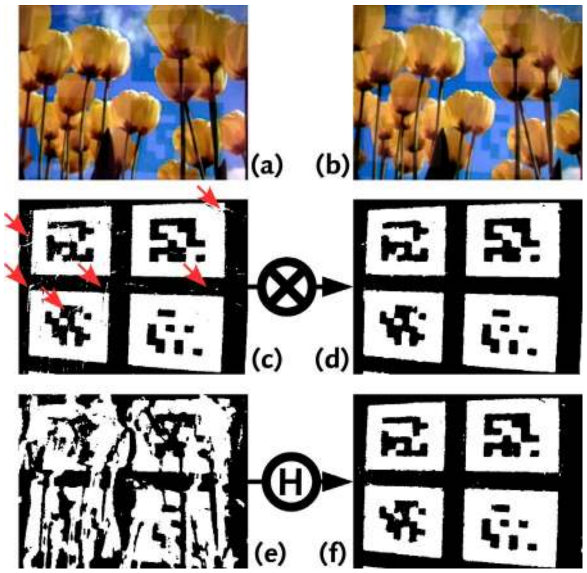


Figure 3: Camera movement leads to a misregistered image pair due to the temporal offset of 8.33 ms between code- and compensation image (a,b). The errors in the reconstructed code (c) (red arrows) can be minimized by applying a 5x5 median filter (d). Strong misregistrations that are due to slower capturing rates lead to a complete destruction of the embedded code (e). By calculating the optical flow between the image pair, the image can be re-registered (f).

Larger misregistrations that are due to slower capturing rates can be corrected by calculating the optical flow between both images. This is realized by applying a Canny edge detector to both images, compute the optical flow from the result, and filtering outlier flow vectors. With the remaining correspondences, a homography matrix is estimated that allows re-registering both images (cf. figure 3e,f). In this case, we assume that the images are projected onto a planar surface. We found that this technique becomes useful for capturing rates that are below 40Hz.

A more critical problem is that both images also appear misregistered on the human retina during fast eye movements which makes the embedded code well visible. In visual perception research a similar effect is known as the *phantom array* [10] – resulting from saccadic eye movements (cf. figure 4 for an illustration). A related

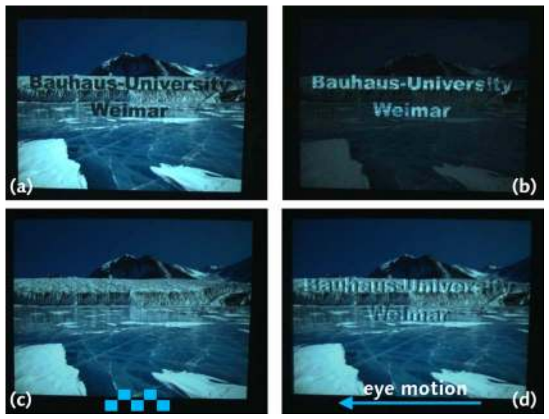


Figure 4: The Δ-coded image pair (a+b) is projected alternately which leads to a perceived image (c) for no or slow eye movements (the blue squares illustrate the sequence of projected code image and compensation image). During faster eye movements, however, the code becomes temporarily visible (d).

effect also appears during the temporal color modulation of DLP projectors, for which it is better known under the term *rainbow effect*.

The strength of the *phantom array* and consequently the perception of the integrated code during eye movements can be reduced and even eliminated by using only small amounts of code contrasts Δ. If too small, however, the reconstructed code bits are perished by camera noise.

Note that the perception of the *phantom array* is not a problem of related techniques that do not compensate the coded images temporarily [5]. For approaches that do perform a temporal compensation for avoiding contrast artifacts and tonal shifts, such as in our case, this effect can be overcome.

A result of our observations is that the JND of the *phantom array* and consequently the largest tolerable amount of Δ in a static image depends on several parameters: the regional brightness and spatial frequency of *O*, the spatial frequency of *C*, the temporal frequency of *I* and *I'*, and the speed of the eye movements. Knowing the relation between these parameters enables a dynamic and content dependent regional adaptation of Δ. Since we have not found any literature that reports on an exact function which correlates these parameters, we have carried out informal user tests to approximate this function. The user tests were carried out in two phases: First a user study was performed with a small numer of subjects to estimate the Δ function within a defined parameter range. Then this function was validated (also outside the defined range) during a subseqent user evaluation with a larger number of subjects. Since the speed of eye movements cannot be measured in the normal application case, we want to assume fast eye movements as the worst case for the following.

3.2 Δ-Function

For estimating the Δ-function, we asked four subjects (one female, three male) to carry out a user study. The participants were questioned to identify Δ at the *JND* point for different projected images with integrated codes. They were sitting at a distance of 95 cm in front of a 110 cm high and wide back projection screen - covering the entire foveal field of view of 60 degrees. The displayed images contained regular checkerboards representing a two dimensional box function with spatial frequencies (*Fb*) ranging from $1/120$ to $1/8$ cycles per degree (cpd) of visual field in both directions, and intensities (*Lb*) ranging from ∼4-27 candela per square meter (cd/m^2). The embedded codes where also represented by a box function with spatial frequencies (*Fm*) ranging from $1/32$ to $1/2$ cpd. Code patterns and image patterns were always phase shifted to avoid a cancellation.

Due to the fact that the human eye is most sensitive to green light, the Δ-value of the embedded code was lowered to $1/4$ of its intensity in the green channel - thus the perceptibility of the embedded code could be lowered while the code reconstruction did not degrade.

To guarantee equal conditions, the subjects were given time to adapt to different luminance levels first. Then they were asked to follow a target on the screen that moved up and down quickly at a constant speed to enforce fast eye movements. While changing Δ, the subjects were asked to indicate the point at which the code could just not be perceived anymore (i.e., the *JND* point). This process was repeated about eighty times per subject to cover a combination of five different image frequencies over five luminance levels, and four different code frequencies. The study took about 4-5 hours for each subject - limiting the total number of subjects. The results of all four subjects were averaged and are presented in figure 5a. They were later validated during a user evaluation with 28 participants (see chapter 3.5).

Due to their mainly linear behavior, the sample points were fitted to planes using multidimensional linear regression (figure 5b). The four parameters of each plane shown in figure 5b are plotted as small circles in figure 5c.

Applying the general plane equation Δ=-(***a****Lb*+***b****Fm*+***d***)/***c*** for parameterizing the fitted functions in figure 5b requires to find continuous functions that approximate the discrete plane parameters (*a,b,c,d*) over all image frequencies *Fb*. Figure 5c illustrates the result of a one-dimensional curve fitting:

$$a = 0.6108Fb + 0.0139 \quad (1)$$

$$b = 1.144/cosh(65(Fb - 0.031)) - 2 \quad (2)$$

$$c = -1 \quad (3)$$

$$d = -0.73914/(1 + exp((Fb - 0.04954)/0.01)) + 1 \quad (4)$$

While the parameters *a* and *b* correspond to the gradients of the planes in directions *Lb* and *Fm*, *d* and *c* represent a shift and a scaling of Δ. The scalar *c*=-1 is relative to our study with a temporal frequency of 120Hz. For other temporal frequencies, it has to be adapted (increased for higher frequencies, and decreased for lower frequencies).

Note that the basis functions were selected to fit the samples with a minimal error. We chose a straight line to fit *a*, a trigonometric function to approximate *b*, and an exponential function to represent *d*. With this, the average deviation of our analytical solution with respect to the experimentally acquired values is 0.266cd/m^2 (this equals 0.89% of the projected intensity levels, or ∼2 projected gray scales).

Besides comparing the analytical solution with the results of the user study, it was also exploited for values outside our discrete test samples. It was confirmed by the subjects participating in a subseqent user evaluation that the function approaches the *JND* point in these cases as well. Section 3.5 summarizes this.

3.3 Computation of Δ

Regionally adapting the Δ values for arbitrary animated or interactive content using the function derived in section 3.2 requires the real-time analysis of *O* and *C*.

For acquiring the spatial frequencies of particular image regions, we apply the Laplace-pyramid approach presented in [4]. In our case we found six levels of the Laplacian pyramid to be sufficient for the analysis. As described in [19] we use the absolute differences of each level of the Gaussian pyramid and normalize each of the resulting Laplacian pyramid levels. The results are the ratios of

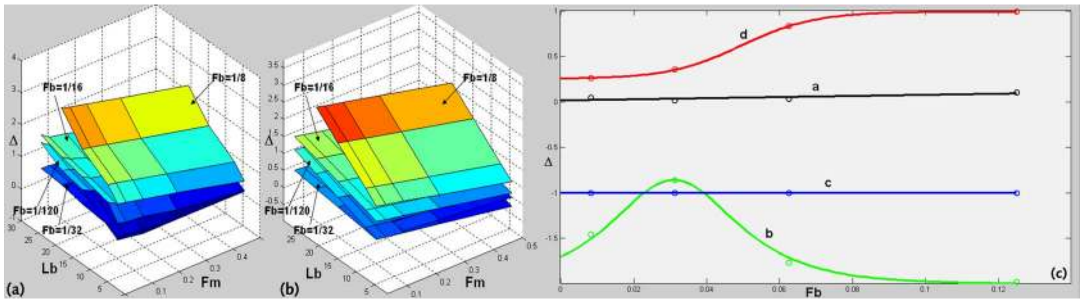


Figure 5: Average Δ responses at the *JND* point for a combination of four subjects, four discrete image frequencies (*Fb*), five luminance levels (*Lb*), and five code frequencies (*Fm*) (a). Plane function fitted to sample points (b) for each considered *Fb*. Approximated discrete plane parameters and fitted continuous functions (c).

spatial frequencies within each of the generated frequency bands. This is converted to units of cpd, that depend on the observers' distance to the image plane and the physical size of the projection. The input image is transformed into its physical luminance representation in cd/m^2 (the response function of the projector has been linearized and its luminous intensity has been measured with a photometer). With these parameters, we can apply our Δ-function to compute the largest non-visible Δ value for an arbitrary region within *O* and *C* (cf. figure 2b).

As mentioned in section 3.2 the visibility of the encoded patterns can be significantly decreased by reducing Δ in the green channel. Decreasing the Δ in the green channel down to a fourth of the intensities in the red and the blue channels did not lead to a noticeable quality reduction of the extracted patterns when the maximal difference of all three color channels was used for decoding. Note, that this does not result in a tonal shift of *O* since the embedded code (no matter how large Δ in different color channels is) is always compensated. In practice, for our setup, Δ ranging from 0.29 to 1.45 cd/m^2 (i.e., 1-5% of the projected intensity levels, or ∼2.5-13 projected gray scales) were computed.

3.4 Temporal Code Blending

Besides the perceived *phantom array* that is caused by fast eye movements, another visual effect can be observed that leads to the perception of the code patterns in cases when they are temporally exchanged. This is illustrated in figure 6.

For photopic vision, it can be assumed that the integration times of the human eyes are between 16ms and 50ms, depending on the perceived brightness (shorter for bright situations). If the projected image and its compensation contain a static code over a period of time, the subtraction or addition of Δ at each pixel of both images *I* and *I'* does not change. Figure 6a visualizes this situation. Plotting the relative amount of integrated light for all possible integration times between 16ms and 50ms, and for all possible phase shifts (in contrast to the camera, the integration process of the eyes is of course not in synchronization with the projection) leads to the presented green area. The average integration amount (dotted line) is zero in figure 6a (assuming no eye movements). Exchanging the code at a particular point in time (i.e., switching from a binary 0 to a binary 1) leads to the integration results shown in figure 6b. The average integration amount during code switching equals Δ, which leads to a visible flickering during this time.

To overcome flickering caused by code transitions, we do not switch between code states abruptly, but temporally transfer from one stage to another stage over multiple blending steps. As illustrated in figure 6c, the average integration amount reduces to Δ/2 for three blending steps. In general we can say that it reduces to Δ/*s* for *s*+1 blending steps if we continuously decrease Δ by Δ/*s* in each step until Δ=0, and then increase Δ by Δ/*s* until the original amount is reached. During the center stage (i.e., when Δ=0 and *I*=*I'*=*O*) the code switched.

The maximal average integration amount Δ/*s* that cannot be detected, and consequently the number of required blending steps, depends on the just noticeable luminance and contrast difference which can be derived from the *threshold-vs-intensity (TVI) function* and the *contrast sensitivity function* as explained in [15, 18]. They are functions of local spatial image frequency and luminance level. Consequently, the optimal number of blending steps for a particular region in *O* can be computed from *O*'s local spatial frequency and luminance level by using these functions.

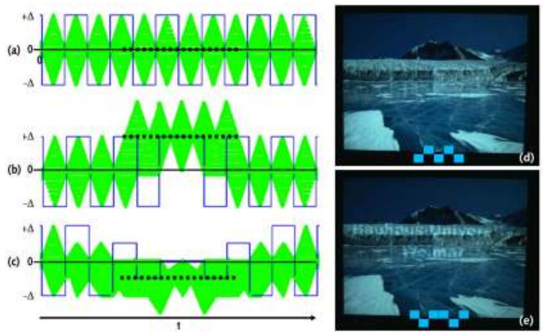


Figure 6: If the code is not exchanged (a,d) it remains invisible. If an abrupt transition occurs, the correct sequence of code image and compensation is not maintained (b,e) which results in a detectable flickering. If the codes are temporally blended (c) it also remains invisible during transitions. The blue line in (a,b,c) illustrate the intensities of the projected code and compensation images. The green area visualizes all possible amounts of perceived intensities that are due to the temporal light integration of the human eye. The dotted line represents te perceved average intensity.

We use the average spatial frequencies and luminance levels of image regions that are already computed for the estimation of Δ (see section 3.3). The TVI function and the contrast sensitivity function are applied for these areas and their results are used to compute the *threshold map* as described in [19] for computing the largest not-detectable luminance difference Δ/*s* (cf. figure 7). This leads to the smallest number of individually required blending steps *s* for each particular code region. If the content in *O* changes during a blending sequence (e.g., in case of videos or interactive content), then the values of Δ and *s* are adapted and the blending is continued until Δ first decreases to a value ≤ 0 (for switching the code) and then increases again until it reaches the new original Δ value. Varying Δ only by the maximum non-perceivable luminance difference ensures that the code cannot be detected during blending. In practice, 10-20 blending steps were derived (i.e., 3-6 code transitions per second can be supported at 120 Hz) (cf. figure 2c).

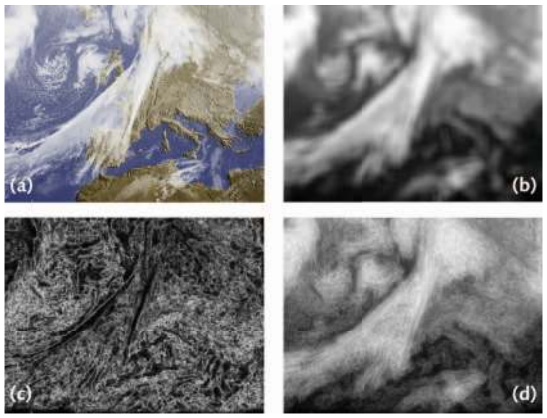


Figure 7: Temporal code blending is realized by calculating the TVI values (b) and the contrast sensitivity information (c) from the input image (a) as described in [19] . This information is used to generate the *threshold map* [19] (d) that is used to derive the number of blending steps for code transitions (b, c and d images are contrast enhanced).

3.5 Validation of the Δ-Function

To validate our experimentally derived function for the Δ-coding (especially outside the parameter ranges defined for the conducted user study), an additional user evaluation was carried out. Test subjects had to observe projected images and videos and judge if they perceive any irritating effects in the projection. Note that they were not informed about the integrated code patterns. They were asked to rate their impression in three scales: "nothing unpleasant in the projection", "weak unpleasant impression" and "strong unpleasant impression". Each image was projected seven times with varying code and different settings for Δ. While some images did contain an embedded code with the locally derived Δ-intensities, others did not contain any code pattern at all. Each participant had to rate 28 projections with varying Δ-intensities. Furthermore, temporal code blending was applied. Twenty-eight test subjects (8 female, 20 male) participated in the evaluation. Note that the presented image and video content contained parameters inside, but are also well outside the initially defined parameter ranges (cf. section 3.2).

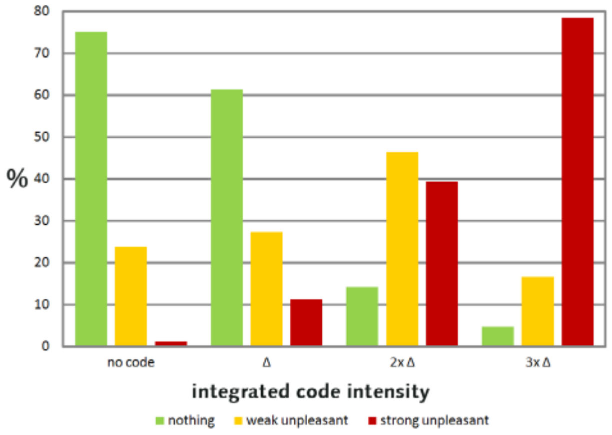


Figure 8: Average results for delta-coding tests under different scalings. The percentage of each given answer is plotted on the y-axis.

As shown in figure 8, the participants did not detect any substantial difference between the Δ-coded projection and the projection without integrated code. But the perception of the integrated code patterns increased significantly for overcontrolled Δ-values. Overcontrolling was enforced by scaling the Δ values manually to exceed the computed optimal values.

In addition, it could be shown that the temporal code blending was also completely undetectable to the subjects. As illustrated in figure 9, the participants did not perceive any difference between a projection without integrated codes and a projection that contained code patterns which were continuously blended. As for the Δ values, the local blending steps were individually computed based on the presented content.

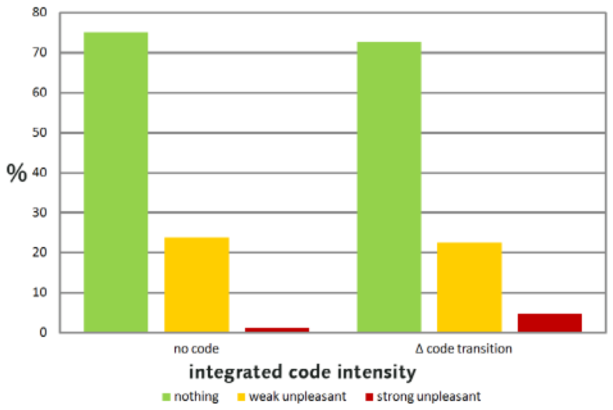


Figure 9: Average results of the temporal code blending experiments with and without continuously changing codes.

Another interesting effect that was observed during our user evaluation is the fact that the presentation of animated content leads to a weaker perception of the embedded code, compared to embedding the same code into a still image with the same delta and blending parameters. This appears to be a result of the observers attention to the animation in the presented video. Additional studies have to be carried out to investigate these effects with the goal to further increasing the Δ values for animated content. If this is understood, the presented video stream could be analyzed in addition for regions of visual attention as proposed in [11] and [29]. This, however, has not yet been done and belongs to our future work.

In addition to our informal user evaluation, our prototype (cf. section 4.1) was presented at a public trade show [1] for the duration of one week. It and was viewed by more than one thousand visitors. During the demonstration, no one of the questioned visitors perceived the embedded codes.

In the next chapters we present an application example of our coding technique that applies an adaptive code placement for marker-based tracking.

4 Adaptive Code Placement

As explained earlier, for supporting optical in-shot camera tracking, imperceptible two-dimensional markers of different scales are embedded into the projected images (cf. figure 1e). Thereby the Δ values and the number of blending steps are computed individually on the fly for each single marker region by averaging the corresponding image luminance and spatial frequency of the underlying area in O and the spatial frequency of the corresponding area in C. For spatial frequencies, the values located within the marker regions of each of the calculated six frequency bands are averaged. The peak

[1]Cebit 2007

frequency is then approximated by choosing the frequency band containing the largest average value.

For a proof-of-concept of the described algorithm we have realized a mock-up of a TV studio, in which a back-projection screen is used to display dynamic video content with integrated imperceptible codes in form of two-dimensional markers for optical camera tracking. A synchronized LED-based illumination system is used to lighten the foreground with high frequent flashes. This prototype enables the composition of real and virtual content with respect to the current camera position, and enables real-time keying with the aid of the synchronized illumination. Section 4.1 gives a detailed overview over the single components of our prototype.

To ensure a continuous tracking despite possible occlusions or individual camera perspectives, the code image C is dynamically regenerated, and marker positions as well as their sizes are adapted. Consequently, the foreground objects have to be keyed and geometrically related to the projected image for determining occlusions with the foreground. These techniques are described in sections 4.2 and 4.3.

4.1 System Overview

All of the presented techniques are combined in a sample prototype of an television studio mock-up that is illustrated in figure 10: A moderation desk and a rear-projection screen serve as backdrop. An off-the-shelf stereo-enabled DLP projector (InFocus DepthQ) displays an arbitrary and dynamic background at a speed of 120Hz. Our camera system consists of two optically aligned CCD cameras (a). For real-time flash keying, a dimmable 4800 Lumen LED illumination system (c) has been built.

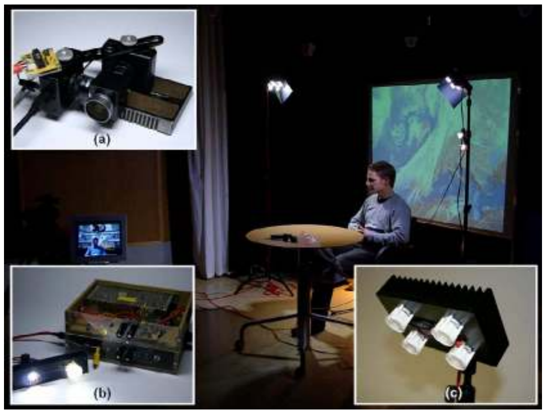


Figure 10: Overview over the components of the TV studio mock-up: moderation desk with rear-projection screen and LED illumination, coaxial camera system (a), synchronization units (b) and LED illumination module (c).

A customized synchronization electronics (b) receives the shutter signal that is generated by the graphics card that triggers the stereo projector. This signal is also being used for triggering the camera and the illumination system at 120Hz. The LED illumination can be switched to a flash mode (i.e., on-off sequences) or to a demodulated (i.e., rectified) constant lighting. Point Grey Dragonfly Express cameras deliver raw-format images in VGA resolution over Firewire 800. The protorype runs on a single PC[2].

4.2 Real-Time Flash Keying

To separate foreground (e.g., the moderator in the context of our TV studio application example) from background (the Δ-coded background projection), we apply *real-time flash keying*. This is necessary for the image composition and the adaptive code placement.

[2]Core2Duo 6300, 1.8 GHz / NVidia Quadro FX 1500 / 2GB Ram

By using high performance LED flash illumination we are able to lighten the scene 60 times per second by short flashes with a length of 8.33 ms. Thus, every other captured frame contains an illuminated foreground (cf. figure 1b), while the remaining frames contain an un-illuminated foreground (cf. figure 1c). This allows separating both using a variation of flash keying (cf. figure 1d). Due to their high frequency, the flashes are not detectable. In contrast to [1] we use white-light LEDs with a white point of 5600K for direct illumination, rather than applying blue LEDs for chrominance masking. This ensures a constant white illumination. Color filters can be used in addition for supporting the required illumination situation, if necessary.

The matting process in this case is straightforward: Due to the fact that one of the captured images is taken under full illumination and the other one under no illumination, we can easily extract the pixels belonging to the foreground by analyzing the intensity difference between corresponding camera pixels and comparing it with a predefined threshold. However, care has to be taken because the delta coded projection in the background also differs in its intensity. The difference threshold has to be set to a value that is larger than twice the largest encoded Δ value. To increase the robustness, we evaluate the maximum difference in the three color channels for thresholding instead of using the average difference of the gray values.

In case of the camera resolution being lower than the projector resolution, individual pixels might be misclassified at the transitions of marker boundaries. This results from the fact that an integration over several Δ-coded projector pixels (also during fast camera movements) can lead nearly to the same intensity in both images. These defects can be removed efficiently by applying a median filter to the generated matte image. In a final step the matte is smoothened by a 5x5 Gaussian filter kernel to soften the transitions between foreground and background. Of course, more advance keying techniques could be applied to generate a real alpha matte. This is part of future investigations.

Instead of applying an LED illumination, video projectors themselves can be used to support flash keying if installed in the recording environment. In contrast to simple LEDs, projector-based illumination [21, 2] supports generating a synthetic, spatially varying illumination of real objects on the fly. Thus, in addition to a temporal illumination coding, a virtual lighting situation can be defined, computed and physically approximated within the environment using projectors - without changing the physical light sources. Although we have realized flash-keying with projectors using a uniform illumination in our prototype, a combination with a true projector-based illumination technique belongs also to our future work.

No matter if projectors or LEDs are applied for illumination, flash keying is supported at a capturing speed of 60Hz for both images. One camera is normally sufficient for this. However, if the Δ-coded projection is out of focus (e.g., due to a short focal depth when focusing on the foreground) marker tracking might fail. As mentioned earlier, two coaxially aligned cameras (cf. figure 10a) are used for avoiding this problem: While one camera is focused on the background, the other camera is focused on the foreground. Registering both camera images and synchronizing the capturing process supports recording the focused foreground while processing the focused background. Furthermore, this would allow to evaluate relative defocus values of corresponding pixels in both images to enable a depth-of-field based keying, as in [22]. A real-time keying from defocus, however, has not been implemented yet.

4.3 Dynamic Multi-Resolution Markers

As mentioned before, we embed binary markers in our example for optical camera tracking. The stability of the optical tracking strongly depends on a constant visibility of a certain amount of markers with optimal sizes. Moving the camera further away from the screen requires displaying larger markers for avoiding a lower tracking quality due to the limited camera resolution. If the camera moves very close to the screen on the other hand, smaller markers are needed to ensure their full visibility in the camera image.

While tracking is not possible if the entire projection is occluded from the camera's point of view, an adaptive marker placement leads to a more robust tracking compared to static markers in the case of partial occlusions.

Hence we adjust the projected imperceptible markers within *C* in each frame by analyzing the visibility of the displayed pixels from the camera's perspective. To ensure the invisibility of the embedded markers during code transitions we apply the temporal blending techniques described in section 3.4.

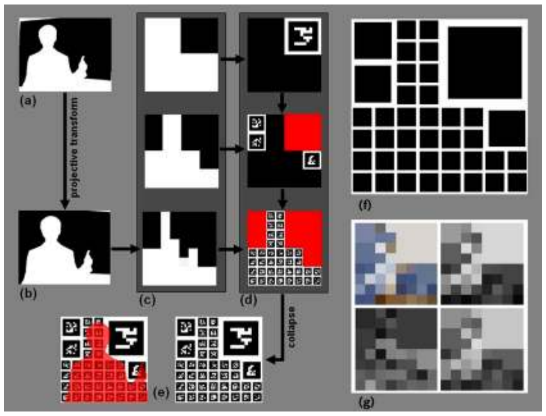


Figure 11: Adaptive generation of marker sets: projective transform of foreground matte from the tracked camera perspective (a) to the projection screen (b), construction of visibility tree (c) and labeling of marker tree (d), collapsing of the labeled marker tree (e). The temporal code blending parameters (g, compare with figure 7) are based on the average image intensities and frequencies within each marker region (f).

For optical tracking the ARTag library [7] is used which offers the possibility to generate arbitrary array sets from 1024 predefined markers. This feature is used to define a multi-resolution marker array containing different sized markers for the same spatial locations - all sharing the same coordinate system.

We pre-compute a quad-tree that contains multiple markers at different scales in each level. From a higher to the next lower level, the number of markers doubles while their size decreases by factor 2. We refer to this as the *marker tree*. Adaptive marker placement is implemented in several steps (cf. figure 11).

First, a full screen quad is rendered in projector resolution and a projective transform is computed that maps the generated foreground matte from the perspective of the camera (a) onto it. This is achieved by using the model-view matrix that results from tracking of the previously displayed frame. The result is a binary image containing the visibility of each projector pixel from the camera's view, which we want to refer to as *visibility map* (b). This technique is similar to conventional shadow mapping.

The initial visibility map is then used to analyze the smallest possible marker size that will be used by geometrically determining the number of projector pixels which are visible in the camera image from the previous perspective.

We sub-sample the visibility map into an image pyramid that covers the largest possible marker size in the highest level (e.g., by definition 2x2 markers in *C*) down to the determined smallest possible marker size in the lowest level (e.g., 16x16 pixels per marker in our case). This leads to a multi-resolution visibility map that we call *visibility tree* (c).

During runtime, the marker tree and the visibility tree are combined at corresponding levels (d): In a top-down direction, only entries that are neither occluded (i.e., marked as visible in the same visibility tree level) nor already occupied by markers of higher levels are processed. The remaining entries are then labeled as occupied within the current level of the marker tree. Regions which are not visible throughout all levels are labeled at the bottom level of the marker pyramid. If the bottom is reached, the labeled marker tree is collapsed and the non-overlapping entries that are occupied by different levels are combined. This results in a code image *C* that contains the set of optimally scaled and placed markers with respect to foreground occlusions and camera perspective (e). The same constellation from the perspective of the camera is illustrated in figure 1e.

As explained in section 3.4, local marker regions have to be temporally blended if a code transition within a particular area in *C* occurs to avoid the visibility of the embedded code within this time.

5 Implementation

Our software prototype is implemented in C++ and OpenGL. The complete image analysis (except the optional optical flow calculation that was realized with OpenCV) and the processing to generate the embedded code, the extraction, matting and the marker placement is implemented entirely on the GPU.

To guarantee a constant switching between both Δ-coded images at a fixed frame rate of 120 Hz, quad buffer rendering in combination with a stereo-enabled DLP projector is used. The response functions of the projector and cameras are linearized to realize a correct integration of the imperceptible code. By applying a gamma correction to the displayed image before embedding the code, the linear response does not reduce the quality of the presented content.

Instead of using the demosaicing functionality offered by the camera driver, we implemented a pixel grouping demosaicing algorithm [12] that is optimized for reducing color seams at intensity boundaries. This offers a good trade-off between quality and performance. The algorithm is implemented as fragment shaders on the GPU and delivers a better quality for this application at significantly higher frame rates compared to the driver's internal CPU based algorithms.

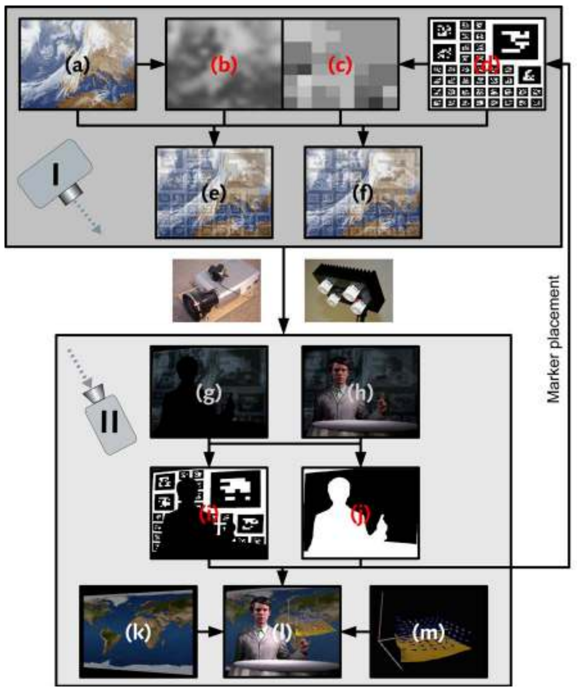
The LED system for flash keying was mounted in front of the projection screen and physical apertures were used to avoid direct illumination of the screen. Care has to be taken to illuminate the foreground completely. For avoiding misclassifications due to cast shadows, a low keying threshold is used to classify very dark pixels (below the black level of the projector) as belonging to the foreground.

Figure 12 summarizes all computation steps: The upper part presents the projection-dependent components while the lower part describes the camera-dependent processing steps.

The input image (a) as well as the current code image (d) is analyzed for its spatial frequencies and local average luminance values. From the results the Δ-values (b) and the local blending steps (c) are computed. Both are used to compute the Δ coded image pairs (e+f). These images are projected sequentially at a speed of 120 Hz while the foreground is flashed by the synchronized LED illumination system.

The imperceptible code (i) as well as the matte (j) are calculated from the image pair captured by the synchronized camera (g+h). The result is used to estimate the pose using ARTag and to compute the new optimized marker placement for the next frame as described in section 4.3. Knowing the actual camera pose as well as the foreground allows integrating a virtual background (k) and 3D objects (m) into the final image (l).

The tables in figure 12 summarize the durations of all relevant processing steps on our hardware (cf. footnote 2). Note that track-



Projector-related performance (milliseconds)	
Calculation of Blending Steps	3
Delta Calculation	35
Marker Embedding and Display	0.5

Camera-related performance (milliseconds)	
Demosaicing	9
ARTag Tracking	45
Adaptive Marker Placement	4
Matting/Composite Generation	8

Figure 12: Flow diagram of all processing steps: The projector-dependent steps are illustrated in the upper part (I) while the camera-dependent steps are summarized in the lower part (II). The timings for individual processing steps are listed on the right hand (cf. footnote 2 for an overview of the used hardware).

ing and the delta calculation can be processed in parallel which increases the overall frame rate.

Our current prototype supports one projector - an expansion to multiple projectors will be implemented in future, and can easily be realized due to the fact that only the calculated visibility tree has to be shared between the camera and the different projector modules.

6 Summary and Future Work

In this paper, we presented a novel imperceptible embedded code projection technique that, in contrast to previous work, considers parameters of human perception for optimal encoding and decoding of integrated patterns. It is based on a high-frequent temporal image modulation, and does not lead to a reduction or a non-uniform fragmentation of tonal values and intensities. Furthermore, it can be applied with unmodified projectors and does not require advanced calibrations other than a linearization and a gamma correction. An analytical function for computing the optimal code contrasts used for coding, decoding, and code transitions was derived from a user study, and validated through a subsequent user evaluation. This function is used to analyze the image content and the code patterns to adapt local Δ-values on the fly. Despite the integration of static code patterns, a way to efficiently exchange the code was described. The maximum number of locally required blending steps are computed automatically from image parameters of the projected content. We demonstrated a real-time flash keying approach in combination with our coded projection for foreground extraction. By combining both techniques in a proof-of-concept prototype of a TV studio mock-up, a dynamic multi-resolution marker method was introduced that ensures a continuous in-shot camera tracking, despite possible occlusions and individual camera perspectives.

While the coded projection and illumination, as well as the capturing process are synchronized at a speed of 120Hz, projected dynamic content was presented at a frame rate of 60Hz. The final image composition that includes tracking, keying, matting, and rendering of augmented content (i.e., foreground / background / 3D / composite) was carried out at 10-20 frames per second on our current hardware (depending on the number of detected markers and adjusted video resolution, the ARTag library required the corpus of 20ms-60ms for processing). This is clearly not acceptable for professional applications. Preliminary experiments showed that distributing the different steps to multiple PCs leads to a significant speed-up.

As explained before, our implemented flash keying technique can be combined with depth-of-field based keying, such as in [22], to support stable real-time matting. Furthermore, the tracking performance and quality needs to be improved significantly for professional applications (e.g. for TV productions). Since our approach is widely independent of the utilized marker tracking library, further investigations have to be carried out to experiment with alternative solutions like ARToolkit [13], ARToolkitPlus [27] or Bazar [14]. At the moment, our system is limited to the performance and the precision of the ARTag library (see [7] for details). Currently, we support online and offline augmentations. In the latter case, the captured images *I* and *I'* are only recorded to disk during run-time. During a post-production step, tracking, keying, matting and rendering can be carried out at a much higher quality level. Integrating more sophisticated techniques for high quality off-line post-processing is also part of our future work.

Although, our informal user study confirms the general validity of our approach and the experimentally derived coding principles, more complex user studies have to be carried out with respect to visual attention effects for animated content. This would possibly allow to increase the Δ-values in animations. All of the described techniques were implemented directly on the GPU to achieve interactive frame-rates.

In the long term, we envision the combination of projector-based and analog illumination in modern television studios [3]. Together with appropriate image correction techniques, such as geometric warping, radiometric compensation, and photometric calibration, this holds the potential to display imperceptible code patterns, such as the markers used for camera tracking, which are integrated into pictorial content or into the projected illumination spatially anywhere within a television studio. A temporally coded projector-based illumination would also support an ad-hoc synthetic re-illumination (as already shown in the small scale [2, 21]), and the extraction of depth-information (similar as explained in [28] and [26]).

A technical challenge in a TV studio context will also be to adapt current studio camera technology to support fast-capturing and synchronization. Today, such cameras are synchronized to external displays via the standard BlackBurst signal at a speed of 50Hz for PAL or 60Hz for NTSC. Thus, the capturing at a field rate 60Hz would decrease the extraction of the embedded code patterns to a maximum speed of 30Hz. The projection speed and consequently the perception characteristics, however, are not effected by slower cameras (e.g., if only every third of the 120 Hz projected frames is captured with 40 Hz). Future projectors will certainly provide even higher frame rates.

Acknowledgments

Special thanks to Daniel Kolster, Stefanie Zollmann, Hongcui Lu, and Steffen Hippeli for contributing to the user study, and to the 28 subjects who participated in the subsequent user evaluation. The investigation and realization of concepts and techniques outlined in this paper have been proposed to the Deutsche Forschungsgemeinschaft (DFG) under reference number BI 835/2-1.

References

[1] T. G. Alexander and R. R. Thomas. Flash-based keying. *European Patent Application EP1499117*, 2005.

[2] O. Bimber, A. Grundhöfer, G. Wetzstein, and S. Knödel. Consistent illumination within optical see-through augmented environments. In *ISMAR '03: Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 198–207, Washington, DC, USA, 2003. IEEE Computer Society.

[3] O. Bimber, A. Grundhöfer, S. Zollmann, and D. Kolster. Digital illumination for augmented studios. *Journal of Virtual Reality and Broadcasting*, 3(8), Dec. 2006.

[4] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983.

[5] D. Cotting, M. Näf, M. H. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 100–109, 2004.

[6] D. Cotting, R. Ziegler, M. H. Gross, and H. Fuchs. Adaptive instant displays: Continuously calibrated projections using per-pixel light control. In *Proceedings Eurographics 2005*, pages 705–714, 2005. Eurographics 2005, Dublin, Ireland, August 29 - September 2, 2005.

[7] M. Fiala. Artag, a fiducial marker system using digital techniques. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[8] M. Fiala. Automatic projector calibration using self-identifying patterns. In *Proceedings of the IEEE International Workshop on Projector-Camera Systems (Procams 2005)*, San Diego, USA, 2005.

[9] J.-M. Frahm, K. Koeser, D. Grest, and R. Koch. Markerless augmented reality with light source estimation for direct illumination. In *Conference on Visual Media Production CVMP, London, December 2005*, 2005.

[10] W. A. Hershberger. Saccadic eye movements and the perception of visual direction. In *Perception & Psychophysics*, 1987.

[11] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[12] C. kai Lin. Pixel grouping for color filter array demosaicing. http://web.cecs.pdx.edu/~cklin/demosaic/ (last viewed: 25/05/2007), 2006.

[13] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, Washington, DC, USA, 1999. IEEE Computer Society.

[14] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 775–781, Washington, DC, USA, 2005. IEEE Computer Society.

[15] J. Lubin. A visual discrimination model for imaging system design and evaluation. *Vision Models for target detection and recognition*, pages 245–283, 1995.

[16] R. A. Moses and W. M. Hart. The temporal responsiveness of vision. In *Adler's Physiology of the Eye*, 1987.

[17] H. Park and J.-I. Park. Invisible marker tracking for ar. In *ISMAR '04: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 272–273, Washington, DC, USA, 2004. IEEE Computer Society.

[18] S. N. Pattanaik, J. A. Ferwerda, M. D. Fairchild, and D. P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 287–298, New York, NY, USA, 1998. ACM Press.

[19] M. Ramasubramanian, S. N. Pattanaik, and D. P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 73–82, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[20] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Computer Graphics*, 32(Annual Conference Series):179–188, 1998.

[21] R. Raskar, G. Welch, K.-L. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques, London, UK, June 25-27, 2001*, pages 89–102, 2001.

[22] E. Reinhard and E. A. Khan. Depth-of-field-based alpha-matte extraction. In *APGV '05: Proceedings of the 2nd symposium on Applied perception in graphics and visualization*, pages 95–102, New York, NY, USA, 2005. ACM Press.

[23] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum. Flash matting. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 772–778, New York, NY, USA, 2006. ACM Press.

[24] K. Tateno, I. Kitahara, and Y. Ohta. A nested marker for augmented reality. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 152, New York, NY, USA, 2006. ACM Press.

[25] G. Thomas, J. Jin, and C. A. Urquhart. A versatile camera position measurement system for virtual realitytv production. In *International Broadcasting Convention*, pages 284–289, 1997.

[26] M. B. Vieira, L. Velho, A. Sa, and P. C. Carvalho. A camera-projector system for real-time 3d video. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 96, Washington, DC, USA, 2005. IEEE Computer Society.

[27] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, 2007.

[28] M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, and M. H. Gross. Scalable 3d video of dynamic scenes. *The Visual Computer*, 21(8-10):629–638, 2005.

[29] H. Yee, S. Pattanaik, and D. P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. In *ACM Transactions on Graphics*, pages 39–65. ACM Press, 2001.

[30] S. Zollmann and O. Bimber. Imperceptible calibration for radiometric compensation. In *Proceedings Eurographics 2007, Short Paper*, 2007.