

Poor Man's Virtual Camera: Real-Time Simultaneous Matting and Camera Pose Estimation

István Szentandrás, Markéta Dubská,
Michal Zachariáš, and Adam Herout

Brno University of Technology

Abstract—Today's film and advertisement production heavily uses computer graphics combined with living actors by chroma keying. The matchmoving process typically takes a considerable manual effort. Semiautomatic matchmoving tools exist as well, but they still work offline and require manual check-up and correction. In this paper, we propose an instant matchmoving solution for green screen. It uses a recent technique of planar uniform marker fields. Our technique can be used in indie and professional filmmaking as a cheap and ultramobile virtual camera, and for shot prototyping and storyboard creation. The matchmoving technique based on marker fields of shades of green is very computationally efficient: we developed and present in this paper a mobile application running at 33 FPS. Our technique is thus available to anyone with a smartphone at low cost and with an easy setup, opening space for new levels of filmmakers' creative expression.

■ **SINCE THE EARLY** years of filmmaking, artists have wanted to create artificial worlds to bring

their ideas onto the film screen. Modern digital virtual production is still a new technology that requires several steps from different research areas: matting or background subtraction and visual tracking from computer vision, alternatively hardware-based tracking from robotics and realistic rendering from computer graphics. Each of

Digital Object Identifier 10.1109/MCG.2016.39

Date of publication 18 March 2016; date of current version 1 November 2019.

these steps is computationally extremely expensive. In this paper, we focus on computer vision based algorithms enabling a real-time mobile solution for virtual production for preview purposes and as a fast, simple, and cheap solution for low-quality production.

One of the early methods to mix the artificial and virtual worlds was *matting*—an effect for composition of several images into one frame. Several approaches for matting were developed in time: e.g., glass paintings, double exposure or using an optical printer.¹ Nowadays, the most commonly used method is *chroma keying*, where a specially selected color (often green or blue) is set to transparent and substituted.² Computer vision developed various methods for removal of background, whether with constant color,³ as used in filmography, or with an arbitrary image.^{4,5}

During rendering of a virtual world to replace the blue or green canvas, the exact movements of the camera must be determined. This information can be obtained by camera rigs programmed to follow a predefined track [e.g., Cyclops or Milo (<http://www.mrmoco.com>), TechnoDolly (<http://www.supertechno.com/product/technodolly.html>)], using sensors mounted to the camera [e.g., Insight VCS (<http://www.naturalpoint.com/optitrack/products/insight-vcs/>)], by tracking simple artificial markers placed on the canvas⁶ or natural image features in the captured real world.⁷

Recent advances in mixed reality filmmaking technology include the concept of *virtual camera*—the position of the real camera is detected and used to simulate the virtual camera in the digital world.^{3,4} Coupled with a classic camera (also known under a commercial name *Simul-Cam*), it forms a tool for superposition of the real and digital world and captures the scene in one moment. These concepts represent the best available technology today; however, they require high financial and technical resources.

Real-time replacement of a green screen with another scene (synthetic or real) can be seen as a kind of an augmented reality system,⁸ for example, the live TV production.⁹ The problem of camera pose estimation can be solved by using markers,^{10,11} natural features as points or edges,¹² textures,¹³ or by using other sensors such as GPS and a gyroscope on a handheld camera.¹⁴ Commercial virtual cameras are mostly based on motion capture,^{15,16} e.g., Insight VCS³ or ILM virtual camera (<http://www.ilm.com/>).

The previously mentioned solutions require complex and costly setup of infrared cameras, additional tracking extensions for the main cameras and external servers. They provide real-time visualization only for the virtual scene and not the augmented result. In the field of augmented reality, there has been prevalent research to create an integrated solution with the help of commodity hardware.¹⁷ Effort has also been put into taking advantage of the growing computational power of handheld devices for ultramobile augmented reality systems.¹⁸ However, Ventura *et al.*¹⁹ have shown that systems based on feature-points and point-clouds are still realizable only with the aid of additional server-side computations.

In this paper, we took advantage of Halide language and a compiler proposed by Ragan-Kelley *et al.*^{20,21} The main idea behind Halide is to separate the scheduling of the performed operations and the algorithm itself. The main advantage over other domain-specific languages is its transparent support for data-parallel computational units such as AVX, SSE, OpenCL, and—most importantly for our work—NEON (<http://www.arm.com/products/processors/technologies/neon.php>). Finding the best schedule for a given algorithm, however, is not trivial and requires deep knowledge of the underlying hardware architecture.

In this paper, we build upon our previous short paper²² introducing only the concept, and we propose a method based on camera pose estimation using a fiduciary marker field. The advantages of the used marker field compared to other marker-based solutions^{10,11} in a virtual production setting are mainly robustness against occlusion and low contrast. The marker field covers the matting canvas (as a whole or a selected fraction). Then, during the shooting, the camera position is established and a preview of the mixed scene is rendered in real time. Our algorithm is computationally extremely efficient. In contrast with state-of-the-art camera grids, our solution can thus run on common mobile devices, providing almost unlimited freedom in camera movement and a very low-cost virtual camera solution.

This solution is unprecedentedly cheap—it is available for a wide range of filmmakers, including amateurs. In the text, we show sample applications that are enabled by this technology. In general, this relatively simple technique unleashes

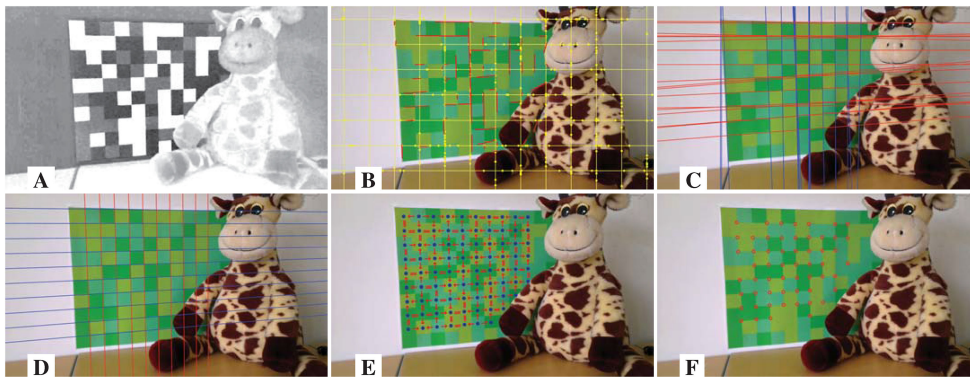


Figure 1. Detection of the grid of squares composed of suitable shades of green. (A) The YC_bC_r image is mapped to grayscale for the detection algorithm. (B) The grayscale image is processed in very sparse scanlines (for better visualization we use the source image). On each scanline, edges are detected (yellow points) and extended to edgels (red lines). (C) The edgels are grouped into two dominant groups using RANSAC; two vanishing points are computed by hyperplane fitting. (D) Based on the vanishing points, the optimal grid is fitted to the set of the edgels. (E) Edges between the modules are classified. (F) The annotated corner points are used for tracking and computing the three-dimensional (3-D) camera pose.

new creative and innovative approaches, so far unseen, especially in indie filmmaking. For simplicity, in the text we are using the term “greenscreen,” but the same applies to any other color (blue, etc.).

The main contributions of this paper are the following:

- Proposed usage of low-contrast marker field for simultaneous matting and camera pose estimation.
- New approach to visualization for storyboard prototyping on ultramobile devices.
- Proposed color mapping for AR use and improved color selection for the UMF greenscreen.
- Real-time performance even on mobile platforms using multiplatform optimization (Halide²⁰).
- The implementation of our solution is made open source under free license for research evaluation and practical use.

The underlying concepts (points 1 and 2) of our method, previously published,²² have been extended and reassessed for real-world applications.

FIVE SHADES OF GRAY—THE MARKER FIELD

In a chroma keying setting, the proposed algorithm first computes the chroma keying

mask to segment out the background containing a fiducial marker. For high-quality calibrated cameras, the raw data are sufficient to detect the difference between shades of green used by the marker. For videos acquired through low quality or smartphone cameras, the algorithm also maps the different shades of green into a single grayscale image to achieve higher contrast [see Figure 1(A)]. The mask and the mapping computations are described in Section “Selection of Proper Shades of Green.”

The fiducials used in our work are based on the easily identifiable marker field,²³ the Uniform Marker Fields. They were first introduced as a checkerboard structure, where the black and white modules are mixed up so that every n^2 tile for a given n is unique in the field in every rotation. The synthesis of such fields is highly time consuming and its size is limited by the size of the uniquely identifiable subwindows (n^2). (For $n = 4$, the theoretical upper bound of the field dimensions is 127×127 .²³)

An advanced version of the Uniform Marker Field detection algorithm, its evaluation, and comparison with other alternative camera localization markers was described by Herout *et al.*²⁴ The limitation of using only black and white colors was relaxed, so that different colors or shades of gray could be used to form the marker field. In this case, instead of the intensity of each field, the gradient between them was used to

determine the unique location inside a marker. A crucial improvement was that the contrasts between the individual modules of the markers can be fairly low and the marker field is still reliably detectable. These modifications not only helped the detection performance and the robustness of the detection algorithm against occlusion, but also much larger maps could be generated with smaller n . Since the marker field design and detection algorithm have been improved and adapted to a chroma keying setting since published by Herout *et al.*,²⁴ we give here a concise description of both.

The detection algorithm assumes that the grid of squares is planar and only distorted by a perspective projection. Thanks to this assumption, the algorithm is very efficient: the fraction of visited pixels (the algorithm's "pixel footprint") within an average input image is very small ($\sim 5\%$,²⁴).

Grid Detection: The algorithm first detects the grid. The grid detection algorithm performs the following three main steps (see Figure 1).

1) *Extraction of edgels (edge elements):* Typically, the algorithm extracts around 100 straight edge fragments in the whole image. The chroma keying mask is used to filter out foreground lines. The image is processed only in sparse horizontal and vertical scanlines [see Figure 1(B)] and edges detected along these scanlines are used as seeds for the edgel detection.

2) *Determining two dominant vanishing points* among the edgels [see Figure 1(C)]. Lines in one dominant direction are supposed to be coincident with the vanishing point. In homogeneous coordinates, this can be expressed by

$$\forall i : \mathbf{v} \cdot \mathbf{l}_i = 0 \quad (1)$$

for the vanishing point \mathbf{v} and the pencil of lines \mathbf{l}_i . Lines \mathbf{l}_i exactly coincident with a vanishing point in the projected space in homogeneous coordinates, lie on a hyperplane passing through the origin. The hyperplane's normal vector in this case corresponds to the vanishing point \mathbf{v} in the projective space.

3) *Finding the grid of marker field edges* as two groups (*pencils*) of regularly repeated lines coincident with each vanishing point. Two vanishing points $\mathbf{v}_1, \mathbf{v}_2$ define the horizon ($\mathbf{h} = \mathbf{v}_1 \times \mathbf{v}_2$). Marker edges of one direction can be computed

using the horizon as ($\hat{\mathbf{x}}$ denotes normalized vector)

$$\mathbf{l}_i = \hat{\mathbf{l}}_{\text{base}} + (ki + q)\hat{\mathbf{h}} \quad (2)$$

where \mathbf{l}_{base} is an arbitrarily chosen base line through the vanishing point, different from the horizon.²⁵ Parameter k controls the line density and q determines the position of the first line. These parameters are found by clustering and a subsequent linear regression (see Figure 1(D), blue and red lines).

Steps 2 and 3 are also applied in subwindows if the detection algorithm fails for the whole image. This improves the robustness against outliers and also allows for multiple markers in one scene.

Location extraction: As proposed,²⁴ the location information is encoded into the edges between the fields of the marker. In order to correctly classify an edge given the locations of the neighboring marker field modules, our algorithm samples pixels from the edge's vicinity. If an edge cannot be confirmed, the location between the modules is treated as a place without an edge (see the top of Figure 2).

The directions of the horizontal (e_{ij}^{\rightarrow}) and vertical (e_{ij}^{\downarrow}) edges inside unique n^2 subwindows are used for construction of a *decision tree* (bottom of Figure 2). The maximal depth of the decision tree is given by the unique subwindow size n^2 , hence the decision is made in constant time for the arbitrarily sized marker field with the given unique subwindow size.

When a compact piece of the marker field is detected in the input image, the edges are classified and used for traversing the tree. The edges in the subwindows are selected in a predefined order (see Figure 2). By using a larger number of deciding edges, the tree can also be constructed *fault-tolerant*—the tree nodes can tolerate one or more falsely classified edges.

Camera pose estimation: When the location is successfully found, high contrast corners in the marker provide the 2-D–3-D correspondences for the camera pose estimation. To improve precision, we employ a subpixel precise corner search. After a successful detection, the corner points in the grid are tracked by using the Kanade–Lucas–Tomasi tracker. For camera pose estimation, we

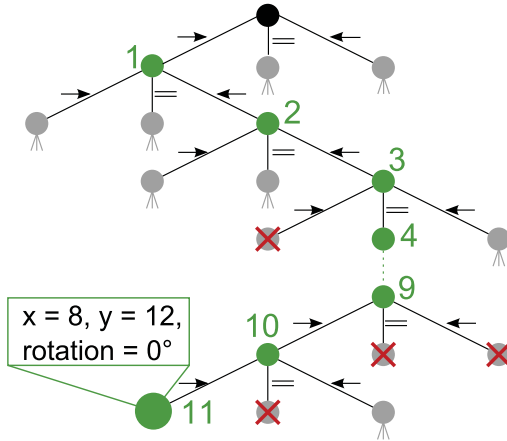


Figure 2. Localization within the Marker Field. The RGB decision tree is simplified so that the decisions are made based on a single value for each edge, not on all RGB components. Top: The order in which the edges are visited. Bottom: The decision tree. Leaves are either invalid or contain the location and orientation within the marker field.

use LHM²⁶ initialized by EPnP.²⁷ The pose ambiguity for planar targets is solved by the method described by Schweighofer and Pinz.²⁸

MATCHMOVING WITH SHADES OF GREEN

Most existing fiduciary marker designs use different marker features (typically edges) for localization of the marker and for recognition of its identity. Therefore, in order to assure reliable detection, the edges must exhibit a large contrast and none of them can be missed. On the contrary, marker fields use the same edges for detection of the marker and for its localization/recognition. That allows for the edges to be of low contrast, because even when a high fraction of the edges is missed, the marker field can still

be detected and recognized. Low contrast edges are necessary for a marker to amalgamate into a sufficiently homogeneous green (or blue or other) surface for the background subtraction.

Selection of Proper Shades of Green

The marker field modules' color must be a compromise between usage of as-similar-as-possible colors for the chroma keying and colors different enough to detect the edges. The selection also depends on the selected chroma keying algorithm. Using, for example, the keying equation of first choice $A = G - \max(B, R)$,²⁹ it seems suitable to vary R and B color components so that $\max(B, R)$ stays the same (i.e., value A is constant for the whole marker field). However, more advanced algorithms are able to also deal with changing of the intensity of the green color (with constant hue). For example, a method described by Jack³⁰ uses the YC_bC_r color model to achieve high-quality matting robust against changing intensities (shadows). This is very important due to the low dynamic range of contemporary smartphone cameras.

Contemporary smartphone and tablet cameras provide raw data in YC_bC_r color space (or a variant of it). Initially, this means no information loss due to conversion and saves computation time. Encoding the marker into C_bC_r channels provides more robustness against intensity changes (shadows) and white balance. This selection appears beneficial although the matting algorithms typically better tolerate variance in Y channel and prefer constant hue. However, with a proper marker color selection, the matting can still be performed correctly.

Mapping: For the detection algorithm, we encoded the edge direction between modules of the UMF into the C_bC_r channels. A good mapping is

$$I_m(x, y) = \text{atan2} \frac{X(x, y)}{Z(x, y)} \quad (3)$$

where I_m is the mapped image and X, Z are the rotated C_b, C_r channels, respectively, by the ϕ angle of the average key color in the C_bC_r space (see Figure 3). As an example for the choice of colors, in our experiments, we used three hues with identical Y channel with 20° difference on the C_bC_r plane (yellow dots in Figure 3). Such a

choice of the mapping is robust against changing intensities and to some extent also against image saturation levels on different smartphones. The detection algorithm then uses the resulting I_m mapped image as a grayscale UMF in further processing. Given the chosen mapping function, motion blur would only cause blurring in the mapped grayscale image. The UMF detection algorithm has proven to be highly robust against such distortion.²⁴

Matting for visualization: For camera pose reconstruction, a low-quality mask creation is sufficient to guide the detection algorithm to discard foreground pixels. For the user interface, a high-quality matting is done on the GPU, freeing resources for image processing on the CPU. Due to white balancing and other automatic image capture controls (generally present in commodity smartphones), having a predefined set of key colors is insufficient. We propose to progressively optimize the exact key colors (by using GMM³¹ in the experiments) once the marker has been successfully detected in the image. Its layout can be used to sample image regions belonging to different shades in the UMF to predict the exact shades of green.

Practical Setup: Projection

Classic green, blue, or white canvas together with a projector can be used as a marker field canvas. Two options for organizing the setup exist: front or back projections (see Figure 4). The main advantage of this approach is the opportunity to change the intensities/colors of the markers depending on the camera, lights, and other conditions. Because of the additional light produced by the projector, this approach is not suitable for high-fidelity shots. However, it is fine for prototyping, simple scene shooting, and instant tuning of the scene/setup.

Practical Setup: Special Canvas

An alternative to the projection of the marker field is using a special canvas with the marker field printed on it. It is possible to synthesize marker fields of arbitrary dimensions by selecting suitable kernel shape/size and a proper number of colors (Section “Realistic Dimensions of the Marker Field”). The marker field does not need to cover the whole matting plate; instead,

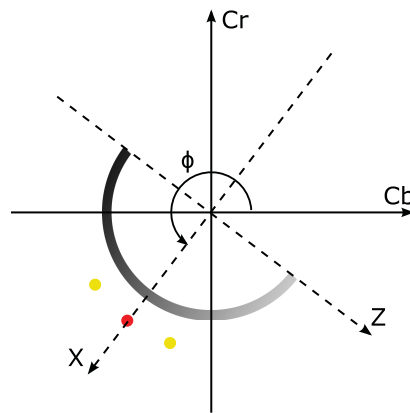


Figure 3. Normalized $C_b C_r$ mapping to XZ space. The red dot represents the main keying color. The red and yellow dots are used for the UMF fields. The half arc demonstrates the mapping of the XZ space to grayscale for the detector.

only a stripe or other shape can be covered by the identifiable marker field (see Figure 8, upper left). Using several sheets requires mutual and world calibration which can be done completely automatically by quickly scanning over the scene with a camera.

EXPERIMENTAL EVALUATION

A thorough side-by-side comparison of the shades-of-gray marker field with alternative markers is given in our previous work.²⁴ In Section “Unity 3D Plug-In,” we demonstrate our

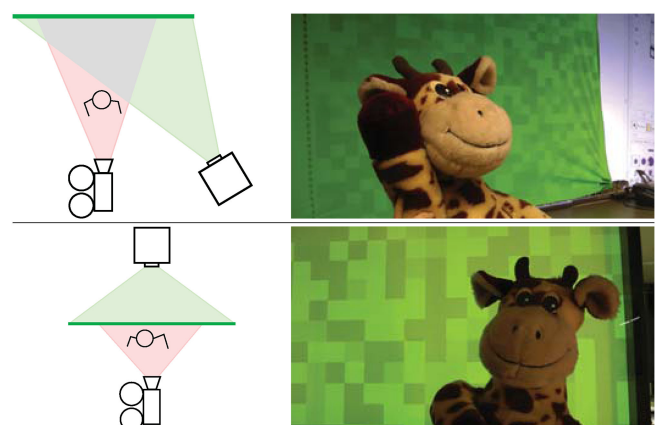


Figure 4. Projection of the marker field on green (or white or other) canvas. Top: Front projection—the actor’s movement is limited because he must not interfere with the projection rays. Bottom: Back projection—requires more space behind the greenscreen.

proof-of-concept implementation integrated into the Unity 3D cross-platform game engine. The rest of the sections evaluate different aspects of our solution.

Unity 3D Plug-In

We target live streaming applications using a webcam for PC or an integrated camera on smartphones. We created a plugin for Unity 3D, which integrates with our proof-of-concept implementation. Unity 3D (<http://unity3d.com>) is a free and cross-platform 3-D game engine, which supports all our targeted platforms.

For a real-world solution, even real-time performance of the detection algorithm causes significant lag due to other parts of the pipeline (frame acquisition, OS overhead, rendering overhead, texture copy between CPU and GPU). We propose to process the frames asynchronously, which provides smooth video stream, but out-of-sync rendering placement. On the ARM platform, the camera stream is acquired directly from the operating system simultaneously for rendering (as texture) and detection (as bitmap). On the PC, the transfer speed between GPU and CPU is sufficient for real-time performance.

To further free up CPU processing time, the costly high-quality chroma keying is done on the GPU using shaders. The detection algorithm uses the simplified method described in Section “Selection of Proper Shades of Green.”

Our plugin finally consists of a script attachable to the *camera object*, a custom shader material for the PC platform and native libraries for all supported platforms. The only inputs required from the user are the camera parameters for the PC platform. On Android OS, these parameters are provided by the operating system. Hence, our solution is extremely easy to use, suitable even for the less experienced designers. For more advanced users, it should provide a very easy to setup tool to preview virtual scenes. We are making our solution available for benchmarking and for practical use (<http://www.fit.vutbr.cz/research/groups/graph/MF/>).

Detection Rates

To reevaluate and compare the detection rates between grayscale and greenscreen markers, we created a small video dataset (16 videos).

Table 1. Detection rates.

Camera/UMF marker	Grayscale	Greenscreen
smartphone	94.5%	96.9%
camera	97.7%	87.7%

We used the setup with a projector projecting the marker from the front (see Figure 4 top). We used two different projectors in combination with a smartphone (1280 × 720, 30 Hz) and a handheld dedicated video camera (1920 × 1080, 50 Hz).

With tracking enabled, our algorithm was able to localize the camera in 99.9% of frames. Since tracking is a replaceable part of the pipeline, Table 1 summarizes the percentage of frames, where the camera was successfully localized without tracking for all combinations.

With a smartphone camera, the detection rates were comparable between grayscale and greenscreen markers. There was even a slight improvement, which might be caused by the reduced number of outlier edgels thanks to the background mask.

With the dedicated camera, the detection rates of the greenscreen UMF were significantly worse. This was caused by two separate problems. The dedicated camera did not do as aggressive white-balancing as the smartphone camera, hence the contrast of the greenscreen marker was significantly lower. Second, the frame-rate collision between one of the projectors and the camera caused banding and flickering (Figure 5 lower right).

Computational Complexity

For the speed performance evaluations, we tested our solution for preview purposes with VGA (640 × 480) resolution camera stream. For the tracking, the algorithm used a subsampled resolution of 320 × 240. Since contemporary cameras provide subsampled color channels (C_b, C_r channels), subsampling should theoretically not cause any loss in the detection precision. We tested our solution both on PC (Intel(R) Core(TM) i7 2.2 GHz) and ARM platform (ARMv7 Processor rev. 9, 1.5 GHz).

For our experiments, we used our Unity plugin (Section “Unity 3D Plug-In”) with a basic 3-D scene. The Android implementation was running at

33.7 FPS on the GPU, and was processing frames (sent at 13 FPS to the CPU) asynchronously in 23.4 ms including Java overhead (theoretical 40 FPS). On the desktop PC, the webcam was providing the VGA video stream at 30 FPS and each frame was processed by the detector in 6.4ms on average. The measurement results are shown in Table 2 (*chroma*: the chroma keying process for the detector; *tracking and detection (t&d)*: the detection and tracking average time; *camera pose*: camera pose estimation based on the found matches). The times include conversion from RGB to YC_bC_r for the PC and communication overhead from native to managed code for the ARM platform.

The main part of computational time on the ARM platform ($\sim 63\%$) was taken by simple image manipulation: gradient computation and Gaussian blur for the tracker, subsampling, matting mask, and mapping computation. We used the Halide language^{20,21} to create a solution for these tasks with more optimized memory access patterns. On the PC platform, this led to $\sim 41\%$ speed improvement overall, with the chroma mapping being $18\times$ faster. On the ARM platform, the speed improvement was $\sim 8\%$ overall.

Preview Precision

To evaluate the camera pose precision for the preview use-case, we created a small video dataset. The videos were shot with the smartphone camera (720p, 30 Hz) from a 2–5 m distance from the canvas. We calibrated the smartphone camera including camera distortion for maximum precision. As reported earlier,²⁴ UMF detection algorithm outperforms alternative marker-based solutions. We used our detector without any optimization and precise calibration to establish a reliable reference for each frame.

To simulate the video stream processed by the detector on the mobile platform, we scaled down and cropped each video to VGA resolution. As calibration, we only used the camera *fovy* defined by the manufacturer. The median difference in the detected camera angle was 1.5° and the median distance from the reference camera pose was 5.91 cm.

Matting Precision

In this section, we test if the presence of the UMF in the greenscreen degrades the matting



Figure 5. Left: Frames acquired with mobile camera with grayscale (top) and greenscreen (bottom) UMF. Right: Frames acquired with handheld dedicated video camera. Notice the very low-contrast greenscreen UMF compared to the frames from the mobile camera. There is also visible banding on the lower left image (handheld camera with second projector).

performance with our algorithm. In the experiment, we projected both plain green color and UMF in the shades of green on the canvas and took images with both smartphone and dedicated cameras. We evaluated the precision of our matting algorithm using GPU shaders with reference to a state-of-the-art alpha-matting approach (KNNMatting³² with manual annotation, see Figure 6). We quantified the error in the resulting alpha masks as the standard deviation of transparency values $[0, 100)$.

There was only a small difference in precision for the plain green color (standard deviation 3.62) and with the UMF marker present (4.5). This shows that using the green UMF as the background does not significantly decrease the segmentation compared to solid color. However, global methods (typically based on graph cuts) perform better on boundaries and on surfaces

Table 2. Breakdown of the processing time in milliseconds. For VGA video.

Platform	Total	(chroma	t&d	cam.)
PC	10.7	3.7	1.9	1.4
PC with Halide	6.3	0.2	1.9	1.4
ARM	25.3	4.8	14.0	2.1
ARM with Halide	23.4	4.1	12.8	2.1

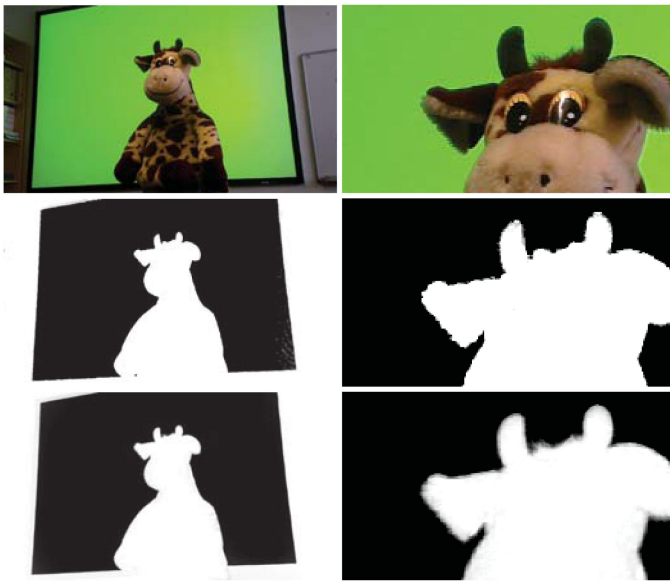


Figure 6. Top: The original image from the camera. Middle: Matting results from GPU shaders. Bottom: Reference alpha mask acquired using KNNmatting with manual annotation.

with bounced green light, at the price of much higher computational complexity.

Sensitivity to Edge Contrast

For the greenscreen to be segmented reliably, the various green colors must be close enough to a common shade, i.e., the contrast on the edges between the neighboring squares of the marker field must be low. At the same time, the marker field must be reliably recognized. An assumption fundamental to the viability of our approach is

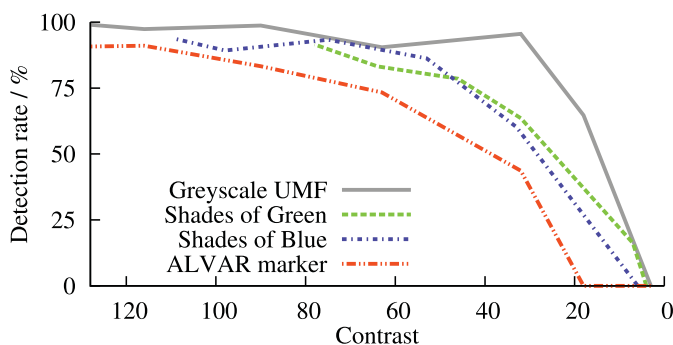


Figure 7. Detection rate as it depends on the marker “contrast.” *Contrast 255* = extreme colors (darkest, brightest) are 255 units apart in the color channels. *Contrast 0* = the colors in the field are all identical. Good news is that, as assumed, the detection performance drops at reasonably low values of the “contrast” between the shades. The experimentally found tipping point (~ 40) is a good choice for practical greenscreen canvases.

that these two contradictory requirements must be satisfied at the same time.

Figure 7 shows the results of our experiment targeted on this issue. We printed out multiple variants of a single generated marker field (14×10) with the green colors altered to change their “contrast.” We shot short video sequences with the handheld dedicated camera observing the marker field from varying angles at normal daylight conditions in an office setting. The detection rate is the fraction of correctly recognized video frames without tracking. The reported graph therefore depicts a pessimistic look on the detection performance. With the tracking feature enabled and even with 75% detection rate, camera pose tracking is theoretically restored with 99.9% probability within the next five video frames.










Realistic Dimensions of the Marker Field

The marker field is synthesized by a genetic algorithm²⁴ according to given parameters: unique subwindow dimensions (n^2 , Section “FIVE SHADES OF GRAY—THE MARKER FIELD”), number of colors/shades, dimensions of the field. The algorithm looks for conflict-free fields where each subwindow exists only once, including all four possible rotations. Conflict-free fields are further optimized for maximal contrast on the edges between the square modules. We used a cluster of computers (~ 1000 nodes) to synthesize the marker fields and we make them publicly available (<http://www.fit.vutbr.cz/research/groups/graph/MF/>).

Table 3 shows several reasonable configurations of the marker fields. Marked in bold are the most conservative variants: 3×3 subwindow, three colors. (*Note:* Substantially larger fields could be generated if necessary, but for practical reasons, we capped the generation at this resolution.) Square marker fields are suitable for printing on paper. The 16:9 marker fields are fine for panoramic canvases. Stripes are a good solution for large movements.

A lower number of colors is better for assuring sufficient edge contrast. In this paper, we used three shades for testing purposes. Higher numbers can be afforded when more variable shades of green can be permitted. To be used in chroma keying settings these shades are mapped into the $C_b C_r$ channels, as described in Section “Selection of Proper Shades of Green.”

Table 3. Synthesized marker fields of reasonable parameters.

aspect ratio	window edges	# of colors	MF dimensions
1:1		3	40×40
		4	110×110
		5	175×175
		3	175×175
		4	250×250 *
		5	250×250 *
		3	250×250 *
		4	250×250 *
		5	250×250 *
16:9		3	48×27
		4	144×81
		5	192×108
		3	160×90
		4	320×180 *
		5	320×180 *
		3	320×180 *
		4	320×180 *
		5	320×180 *
stripe		3	600×6, 150×15
		4	3000×6, 1200×15
		5	3000×6, 1500×15 *
		3	4000×8, 1200×20
		4	8000×8, 2000×20 *
		5	8000×8, 2000×20 *
		3	8000×8, 2000×20 *
		4	8000×8, 2000×20 *
		5	8000×8, 2000×20 *

CONCLUSION

In this paper, we provide an instant and reliable matchmoving solution for chroma keying-based film effects. It is based on the multicolor uniform marker fields. We propose methodology to design marker fields of shades-of-green (blue or any other practical color). Such marker fields can be detected, recognized (for accurate camera pose estimation), and precisely segmented out in order to be replaced by rendered content. The experiments show that the process is both very efficient (the applications can easily run on today's ultramobile processors) and at the same time it is inexpensive and simple to use.

Our solution can be used as a poor man's virtual camera (see the top of Figure 8). Besides this use-case, we propose a live storyboarding scenario (see the bottom of Figure 8), where the user (film author, script editor, 3-D film artist) replaces the matted background with 2-D (360° background photo) and 3-D (models, animated figures) content.

We are developing and maintaining implementations of the matchmoving and chroma keying software for popular ultramobile platforms. This software is made public with this paper.

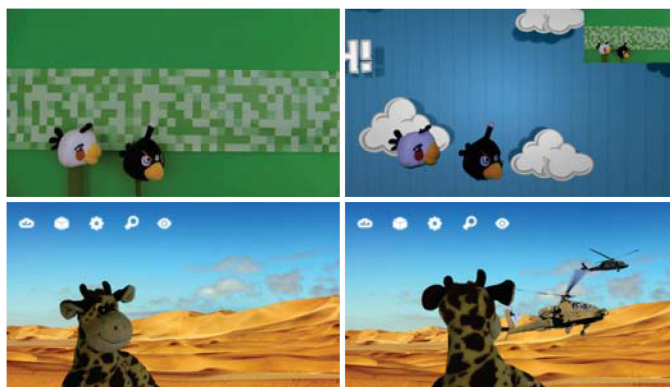


Figure 8. Sample virtual camera images and shots from the Live Storyboarding. A long stripe is placed over a standard green canvas for horizontal panoramic movement of the camera.

ACKNOWLEDGMENT

This work was supported by IT4Innovations Excellence in Science Project (IT4I XS – LQ1602).

REFERENCES

1. M. Vaz and C. Barron, *The Invisible Art: The Legends of Movie Matte Painting*. San Francisco, CA, USA: Chronicle Books, 2002. [Online]. Available: <http://books.google.cz/books?id=VkJzO8k55W4C>
2. J. Foster, *The Green Screen Handbook: Real-World Production Techniques*. Hoboken, NJ, USA: Wiley, 2010. [Online]. Available: http://books.google.cz/books?id=-KE_UzDunU4C
3. A. R. Smith and J. F. Blinn, "Blue screen matting," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 259–268. [Online]. Available: <http://doi.acm.org/10.1145/237170.237263>
4. J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson matting," in *Proc. SIGGRAPH*, 2004, pp. 315–321. [Online]. Available: <http://doi.acm.org/10.1145/1186562.1015721>
5. Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, "A Bayesian approach to digital matting," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, vol. II, pp. 264–271, Dec. 2001.
6. B.-J. Lee, J.-S. Park, and M. Sung, "Vision-based real-time camera matchmoving with a known marker," in *Proc. 5th Int. Conf. Entertainment Comput.*, 2006, pp. 193–204.
7. T. Dobbert, *Matchmoving: The Invisible Art of Camera Tracking* (ser. Wiley Desktop Editions), Hoboken, NJ, USA: Wiley, 2006.

8. R. Azuma *et al.*, "A survey of augmented reality," *Presence-Teleoper. Virtual Environ.*, vol. 6, no. 4, pp. 355–385, 1997.
9. G. Thomas, "Mixed reality techniques for TV and their application for on-set and pre-visualization in film production," in *DVD-ROM Proc. Int. Workshop Mixed Reality Technol. Filmmaking*, 2005, pp. 31–36.
10. D. Wagner and D. Schmalstieg, "First steps towards handheld augmented reality," in *Proc. 7th IEEE Int. Symp. Wearable Comput.*, 2003, pp. 127–135.
11. H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. 2nd IEEE ACM Int. Workshop Augmented Reality*, 1999, pp. 85–94.
12. E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. 10th IEEE Int. Conf. Comput. Vision*, 2005, pp. 1508–1515.
13. L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for real-time accurate 3D camera tracking," in *Proc. 3rd IEEE ACM Int. Symp. Mixed Augmented Reality*, 2004, pp. 48–56.
14. G. Reitmayr and T. Drummond, "Going out: Robust model-based tracking for outdoor augmented reality," in *Proc. 5th IEEE ACM Int. Symp. Mixed Augmented Reality*, 2006, pp. 109–118.
15. T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vision Image Understanding*, vol. 104, pp. 90–126, 2006. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1077314206001263>
16. E. Foxlin *et al.*, "Vis-tracker: A wearable vision-inertial self-tracker," *IEEE Virtual Reality*, vol. 3, pp. 199–206, 2003.
17. P. McIlroy, S. Izadi, and A. Fitzgibbon, "Kinectrack: 3D pose estimation using a projected dense dot pattern," *IEEE Trans. Visualization Comput. Graph.*, vol. 20, no. 6, pp. 839–851, Jun. 2014.
18. N. Hagbi, O. Bergig, J. El-Sana, and M. Billinghurst, "Shape recognition and pose estimation for mobile augmented reality," *IEEE Trans. Visualization Comput. Graph.*, vol. 17, no. 10, pp. 1369–1379, Oct. 2011.
19. J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg, "Global localization from monocular slam on a mobile phone," *IEEE Trans. Visualization Comput. Graph.*, vol. 20, no. 4, pp. 531–539, Apr. 2014.
20. J. Ragan-Kelley, A. Adams, S. Paris, M. Levoy, S. Amarasinghe, and F. Durand, "Decoupling algorithms from schedules for easy optimization of image processing pipelines," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 32:1–32:12, Jul. 2012. [Online]. Available: <http://doi.acm.org/10.1145/2185520.2185528>
21. J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, "Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," in *Proc. 34th ACM SIGPLAN Conf. Program. Lang. Des. Implementation*, 2013, pp. 519–530. [Online]. Available: <http://doi.acm.org/10.1145/2491956.2462176>
22. M. Dubska, I. Szentandrasi, M. Zacharias, and A. Herout, "Poor man's simulcam: Real-time and effortless matchmoving," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2013, pp. 249–250.
23. I. Szentandrás, M. Zachariás, J. Havel, A. Herout, M. Dubska, and R. Kajan, "Uniform marker fields: Camera localization by orientable De Bruijn tori," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2012, pp. 319–320.
24. A. Herout, I. Szentandrás, M. Zachariás, M. Dubska, and R. Kajan, "Five shades of grey for fast and reliable camera pose estimation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2013, pp. 1384–1390.
25. F. Schaffalitzky and A. Zisserman, "Planar grouping for automatic detection of vanishing lines and points," *Image Vision Comput.*, vol. 18, pp. 647–658, 2000.
26. C.-P. Lu, G. D. Hager, and E. Mjølness, "Fast and globally convergent pose estimation from video images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 6, pp. 610–622, Jun. 2000. [Online]. Available: <http://dx.doi.org/10.1109/34.862199>
27. V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epn: An accurate $O(n)$ solution to the PnP problem," *Int. J. Comput. Vision*, vol. 81, no. 2, pp. 155–166, 2009.
28. G. Schweighofer and A. Pinz, "Robust pose estimation from a planar target," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2024–2030, Dec. 2006.
29. S. Wright, *Digital Compositing for Film and Video*. Waltham, MA, USA: Focal Press, 2010.
30. K. Jack, *Video Demystified: A Handbook for the Digital Engineer*, 5th ed. Newton, MA, USA: Newnes, 2007.
31. C. Rhemann, C. Rother, A. Rav-Acha, and T. Sharp, "High resolution matting via interactive trimap segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
32. Q. Chen, D. Li, and C.-K. Tang, "KNN matting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2175–2188, Sep. 2013.

István Szentandrás is currently working toward the Ph.D. degree focusing on computer vision, augmented reality, and parallel computing on mobile platforms. He received the M.S. degree from the Faculty of Information Technology, Brno University of Technology, Czech Republic. Contact him at iszent@fit.vutbr.cz.

Markéta Dubská works as a Researcher at Brno University of Technology. Her research interests include computer vision, geometry, and computation using parallel coordinates. She received the Ph.D. degree from Faculty of Information Technology, Brno University of Technology. Contact her at idubaska@fit.vutbr.cz.

Michal Zachariáš received the M.S. degree from the Faculty of Information Technology, Brno University

of Technology, Czech Republic. He is currently working toward the Ph.D. degree working on augmented reality, matrix codes and human-computer interface design. Contact him at izacharias@fit.vutbr.cz.

Adam Herout works as a Professor at Brno University of Technology and leads the Graph@FIT Research Group. He is a Cofounder of angelcam.com, which provides web streaming from network cameras and real-time computer vision in the cloud. His research interests include fast algorithms and hardware acceleration in computer vision. He received the Ph.D. degree from the Faculty of Information Technology, Brno University of Technology. Contact him at herout@fit.vutbr.cz.