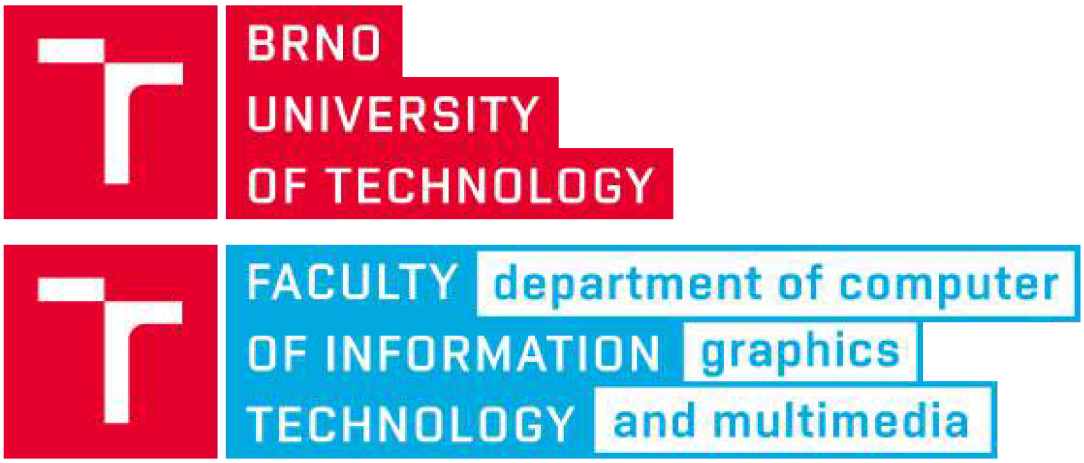

REAL-TIME CAMERA POSE ESTIMATION FOR AUGMENTED REALITY

URČENÍ POZICE KAMERY V REÁLNÉM ČASE PRO ROZŠÍŘENOU REALITU

DOCTORAL THESIS

DISERTAČNÍ PRÁCE

AUTHOR — Ing. ISTVÁN SZENTANDRÁSI

AUTOR PRÁCE

SUPERVISOR — prof. Ing. ADAM HEROUT, Ph.D.

ŠKOLITEL

BRNO 2016

Curriculum vitae

Personal Data

full name	István Szentandrási
born	June 11, 1988, Galanta, Slovakia
email address	iszent@fit.vutbr.cz
telephone number	+420 776498837
languages	hungarian, slovak, english, czech

Education

2010 – today **Doctoral programme** — *Computer Science and Engineering*
Brno University of Technology
Thesis: Real-Time Camera Pose Estimation for Augmented Reality

2008 – 2010 **Masters study programme** — *Computer Graphics and Multimedia*
Brno University of Technology
Thesis: Modern Methods of Realistic Lighting in Real Time

2005 – 2008 **Bachelors study programme** — *Information Technology*
Brno University of Technology
Thesis: Graphics Applications in Python Using OpenGL

Experience

- Image and video processing
- 3D rendering
- Augmented Reality

- C(++), Java, PHP, JavaScript
- OpenCV, Android, Unity 3D

Teaching

Courses, Laboratories, and Projects

2010 – 2016 Leading diploma and bachelor thesis

2010 – 2013 *Graphic and Multimedia Processors*
3 laboratories – CUDA, OpenCL, OpenGL shaders

2011 – 2013 *Computer Graphics (ERASMUS)*
5-10 lectures – Computer graphics, OpenGL

2014 – 2016 *Application Development for Mobile Devices*
2 lectures – Android development

Contents

Chapter 1

Introduction

Augmented Reality (AR) is viewed as a variation of Virtual Environments. While immersed in a Virtual Environment, the user is limited to seeing only virtual objects. Augmented reality, on the other hand, simply enhances the real world with virtual objects. Azuma [4] proposed a commonly accepted definition of augmented reality systems from 1997. Such system is required to have three characteristics: combines real and virtual environment, it is interactive in real time and registered in 3D. This definition does not allow simple 2D overlays or (non-interactive) movie effects. On the other hand, augmented reality based applications on contemporary mobile devices clearly fit all the required characteristics.

This work is focused on real-time camera pose tracking. The current state of the art in several areas, such as marker-based tracking, feature point matching and tracking has achieved great maturity. Objectively, there is still progress to be made in this field to enhance robustness, reduce computational complexity and increase scalability to mobile devices. In this work I included an overview of the above mentioned tracking of the camera pose relative to real-world objects. I pinpointed current limitations and established possible future directions with a focus on methods with extremely low computational and memory requirements.

None of the existing marker-based approaches fulfills the requirements set as the goals of this work: scalable size, reliable and robust detection, and efficient algorithm suitable for ultra-mobile devices. ALVAR, ARTag, CALTag and similar approaches have very efficient algorithms, but allow only small individual markers and require complex setup and calibration, if a larger area is to be used. Random Dot Markers' marker design theoretically allows for scalable sizes, but the detection algorithm is far from efficient.

These limitations lead to the development of Uniform Marker Fields by me and my colleagues. My most important contribution was the research of efficient algorithmic approaches and their maximally efficient implementation on multiple computing platforms. The marker design and synthesis were done as a joint research with Michal Zachariáš, Adam Herout, Jiří Havel – my contribution to these parts was secondary. Based on the developed technology of UMF, we opened space for a few distinct applications.

1.1 Summary of Contributions

This thesis contributes to the state of the art of fast camera localization using artifical markers. This work describes the design decisions and proposed algorithms for efficient detection of the Uniform Marker Fields and its utilizations.

Efficient detection of planar grid structures using vanishing points.
In the industrial manhattan world, the occurrence of tile-based structure is frequent. Many marker based approaches rely on tiles of black and white fields to encode information (ARTag, ALVAR, QR code) or to get reliable points for the camera pose estimation. However, these approaches use only corners or special local image features to localize the markers (silent areas, length-ratio on line segments, circular patterns, etc.). The proposed method in this thesis uses a global approach to detect the grid of tiles as a whole. This is the key part of the camera pose estimation algorithm for the Uniform Marker Fields.

Novel approach to real-time virtual camera.
Contemporary virtual camera systems used in movie production to replace image segments with virtual objects use complex and expensive hardware and software setups. A challenging component in these systems is the real-time camera pose estimation for live scene previews and storyboarding. This thesis describes an approach that works on commodity mobile devices in real time.

Estimating relative pose for human computer interactions.
The growing number of user-owned smart devices equipped with camera opened the door towards new inter-device interaction techniques. Visual one-time transfer of data with limited size between devices already exists (QR codes, VR codes). This thesis discusses a novel interaction technique that uses continuous information flow for interaction. This is achieved by establishing and tracking the smart devices' camera pose relative to the information provider.

Cross-platform efficient implementation of proposed methods in real-world use cases.

With technological advancement, the number of available computational platforms grows. Mobile architectures focus on low power consumption with rich support for auxiliary sensors, while desktop architectures aim for maximum possible performance and ease of development. The methods described in this thesis were implemented with both these platforms in mind. An efficient, low-memory footprint algorithm is especially important on mobile platforms, where the computational power is relatively low.

1.2 Authorship

Although most of the work presented in this thesis is my own, some parts resulted from a collaboration with colleagues.

Adam Herout has contributed to my work with many ideas and consultations. He proposed first the usage of de Bruijn sequences as a perspective direction to solve the limitations of state-of-the-art marker designs. The initial visual design and synthesis of such markers (Uniform Marker Fields) were done as a joint research with Michal Zachariáš, Adam Herout and Jiří Havel – my contribution to these parts was secondary. My contribution related to Uniform Marker Fields was the proposal of an efficient detection method, its refinement into a practical algorithm and experimental evaluation. Jiří Havel's research and consultations were pivotal during this process, who laid down the mathematical bases of efficient vanishing point detection.

Markéta Dubská proposed the first basic principle of using fiduciary markers as part of a greenscreen for cheap camera pose estimation in movie production. I refined this idea into a practical algorithm and tested in the experiments presented in this thesis.

Rudolf Kajan proposed the system for continuous inter-device communication. This system included the module for relative pose estimation using the camera stream as a bases for interactions. The parts relevant to this module in this thesis are my own work.

Chapter 2

State of the Art

Augmented reality systems are complex systems consisting of many individual sub-parts or sub-modules from a broad range of fields concerning computer vision, computer graphics, hardware sensors, robotics, etc. The research on augmented reality is in consequence highly fragmented.

The overwhelming majority of contemporary AR systems solutions use exclusively visual information. Even the strict definition of augmented reality systems by Azuma [4] does not specify the characteristics of devices used for registering 3D position. Besides optical sensors, as the most common sensor used at present, magnetic, acoustic, inertial, GPS, mechanical and other sensors can be used. Using captured images alone for 3D registering is sometimes insufficient and require relatively large computation power.

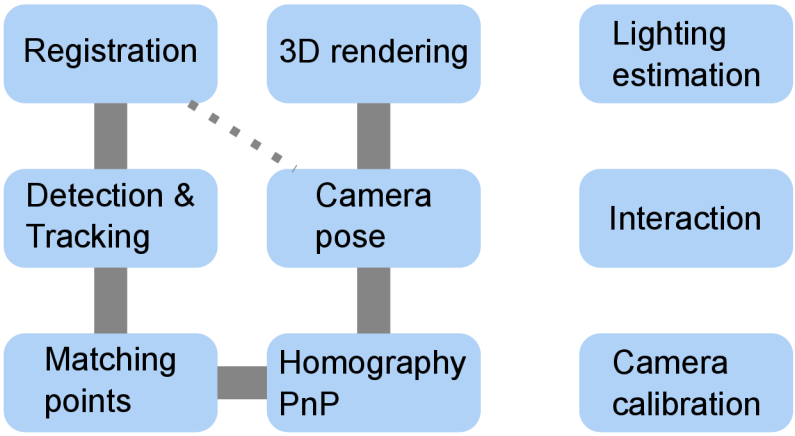

Figure 2.1: Simplified model of an augmented reality system. For each iteration, the system starts with gathering sensory input (*Registration*) and provides the user with an augmented view (*3D rendering*). Visual data, as the main source for camera pose estimation, needs to be further processed (*Detection & Tracking*, *Matching points*, *Homography – PnP* calculations). Other sensory information, like GPS or IMU, can be directly used for camera pose estimation. An AR system optionally includes several other modules. I gave three examples: *Lighting estimation* for realism, *Interaction* with virtual objets, *Camera calibration* for improved precision.

For the remainder of this work, I will focus on augmented reality systems using cameras as main input sensors. Figure 2.1 contains a simplified model of such a system. The first step is registration – acquiring the input from different sensors. The visual

data is then further processed to find important edges, corners, other reliable feature points and markers in the image. These points are then matched based on the model to 3D positions. The correspondences between the 2D and 3D points can be used to calculate a homography, to get the global 6 degrees-of-freedom camera pose (position, rotation). Other sensor input can optionally be used to improve precision (dashed line between Registration and Camera pose in Figure 2.1). Using the knowledge about the camera's internal parameters and its position and rotation, the system is able to augment the captured image or video with virtual objects.

The three further modules: Lighting estimation, Camera calibration and Interaction are also important to achieve realistic results, good user experience and precise camera position estimation. Unfortunately, the research on interaction with the virtual environment is still in early stages.

2.1 Perspective-*n*-Point Problem

In some computer vision applications, simple 2D homography between consecutive frames might be enough to track the camera movement. In augmented reality, however, most applications rely on acquiring the full camera position and orientation relative to a known origin. In these calculations, the camera's intrinsic parameters are assumed to be available. The problem of determining the camera pose given the correspondences between 2D and 3D points and the intrinsic parameters is known as *Perspective-n-Point*, where *n* refers to the number of correspondences.

Generally, P*n*P solving algorithms try to solve the equations given by the pin-hole camera model:

$$\mathbf{p_i} \approx \mathbf{K}(\mathbf{R}, \mathbf{t})\mathbf{m_i} \tag{2.1}$$

for unknowns 3×3 orthogonal rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ for each correspondence $(\mathbf{p_i}, \mathbf{m_i})$. Points $\mathbf{p_i}$ are the undistorted 2D projected points expressed as a column vector in homogeneous coordinates and 3D homogeneous column vectors $\mathbf{m_i}$ of the model points. $\mathbf{K}$ is the camera intrinsic matrix:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.2}$$

where f_x, f_y are the focal lengths expressed in the same units as the projected points and (c_x, c_y) is the principal point.

Solving P*n*P accurately is computationally expensive even with known camera calibration. Quan et al. [30] described an often used solution for fixed $n = 3$ (combined with RANSAC for $n > 3$). Usually a fourth point is used for disambiguation between possible solutions. This is used with markers like ARTag [12] or ARToolkit [18], which have exactly 4 corner points. For $n > 3$ stable non-iterative approaches have complexity

of $O(n^5)$ [30] or even $O(n^8)$ [3]. From iterative approaches, Lu et al. [23] described a very accurate algorithm, though slower than non-iterative algorithms without a good initial pose. More recently, Lepetit et al. [22] proposed a non-iterative algorithm combined with Gauss-Newton optimization algorithm with $O(n)$ complexity – EPnP.

In an augmented reality setting, the temporal camera pose dependence between successive frames provides a good enough initial guess for iterative methods to reduce the number of needed iterations. Methods like EPnP are still useful during initialization and when the tracking gets lost.

2.2 Vision-based Tracking Techniques

Vision-based tracking have been the most active area of research in augmented reality. It allows to calculate the camera position with high accuracy compared to other sensor based techniques. They represent closed loop systems, since they can use results from previous steps and correct errors dynamically.

Vision-based tracking methods can be separated into two main classes based on the used information from the image: feature-based and model-based. The feature-based methods try to find a correspondence between 2D image feature points and 3D world frame coordinates. Feature-based method can be further split into two groups based on the type of the features used for detection: fiduciary marker based tracking methods (Section 2.3) and natural image feature based tracking.

Model based tracking methods explicitly use the features of tracked objects, which have a 3D model known beforehand. This technique is often combined with methods based on natural features. The texture of objects provides more easily trackable features and is usually more dominant than the shape of the objects.

Most modern model based tracking methods, build their own models based on points, edges, or lines. There are two main families of approaches, depending on how the image features are being used. The first family tries to match projections of target objects based on lines and edge positions, as the algorithms described above. The second set of approaches rely on local information in the image region.

Model based approaches that rely only on geometric properties of the objects are not scalable enough for larger scenes. In an outside area they fail to register finer geometry, like the windows on buildings. Simon [33] proposed a hybrid approach combining 3D model and texture information (Figure 2.2 left).

Most of the 3D reconstruction methods like monocular SLAM, DTAM [26], PTAM [19] (Figure 2.2 right), etc., could be also classified under model based tracking methods. Even though these methods can be extended to be used in augmented reality system, they require large amount of memory and computational power. As a consequence, it is unrealistic for them to work on contemporary mobile devices and used for consumer oriented augmented reality applications.

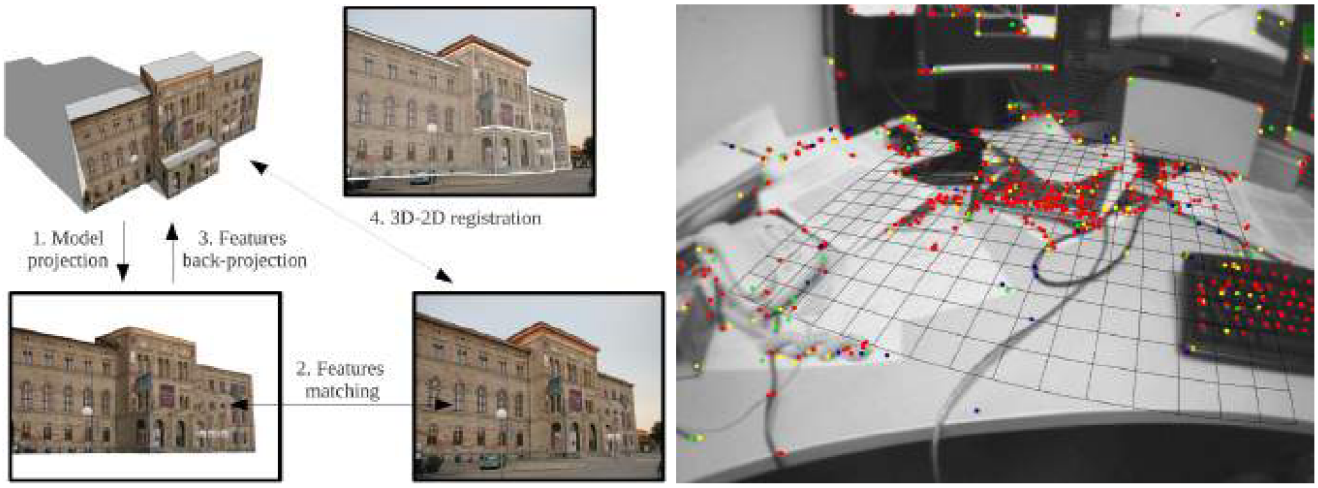

Figure 2.2: **Left**: One iteration of model based tracking combined with feature matching on textures by Simon [33]. **Right:** Parallel tracking and mapping [19] (PTAM) demonstration. The successful point observation and the maps dominant plane is shown in the image.

Vision based tracking methods for augmented reality work with high precision and robustness in confined areas with complex geometry and textures, but fail with fast motion or in large-scale areas. Using other type of sensors, like GPS or inertial sensors, works well in these situations. Notable research on combining auxiliary sensors with visual information is the work of Oskiper et al. [28].

2.3 Approaches Based on Fiduciary Markers

Historically and also in recent literature many augmented reality based research is using *fiduciary markers* to reliably establish the camera position within the scene. Popular designs of fiduciary markers consist of two components: geometrical features which help localize the marker in the processed image and features defining the identity of the marker (for example [12, 18]). That allows for placing several (or many) markers into one scene and their efficient detection. Usage of several markers displaced within the scene is necessary to allow for free movement of the camera within the scene.

The ARToolKit library was first presented in 1999 [18]. Even though the detection algorithm has disadvantages and newer, more robust methods were introduced, it is still used in research in augmented reality as a fast and simple solution. The detection algorithm of the markers is based on binarization with adaptive threshold and matching the rectified content against a library. The corners of the marker are then matched to their known 3D positions. Based on these correspondences, the algorithm estimates and iteratively refines the camera pose to get better precision.

ARTag introduced by Fiala [12] has tried to solve these problems (Figure 2.3).

They combined *Data matrix* coding for marker identification with the rectangular thick border shaped markers used in ARToolKit. In the detection algorithm, they replaced the adaptive thresholding of image regions with thresholding of extracted edges. These changes improve detection performance and require lower computational complexity. The most notable change introduced by ARTag is replacing the template image with digitally encoded information.

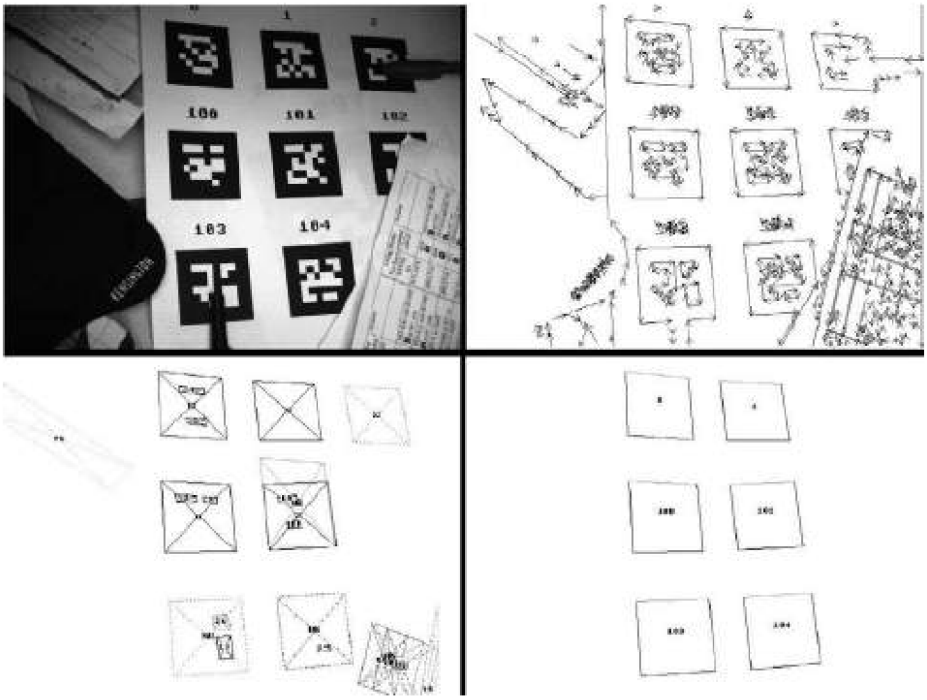

Figure 2.3: ARTag [12] marker detection algorithm.

In a follow-up work, Fiala [13] combined several markers on a single plane of known relative position and rotation to improve the reliability of the camera pose detection. He used this approach to provide a "magic mirror" system with acceptable accuracy and delay. One disadvantage of this approach is that individual markers are still detected and decoded separately, wasting computational time. The second disadvantage is that the relative position and rotations between markers contributing to a single reference frame have to be annotated manually. Only after each marker was detected, their corners are extracted and combined together based on the annotations to improve camera pose estimation precision.

Nested Markers [35] tried to solve the contradictory requirements of small enough marker size to fit in the frame and large enough to realize accurate geometric registration by nesting a number smaller markers into a larger one.

One of the latest improved techniques based on individual square-based markers was proposed by Herout et al. [15], who introduced Fractal Marker Fields (Figure 2.4). They provide the ultimate solution to the contradictory requirements faced by pure marker-based approaches. These marker fields provide guaranteed density of visible markers in

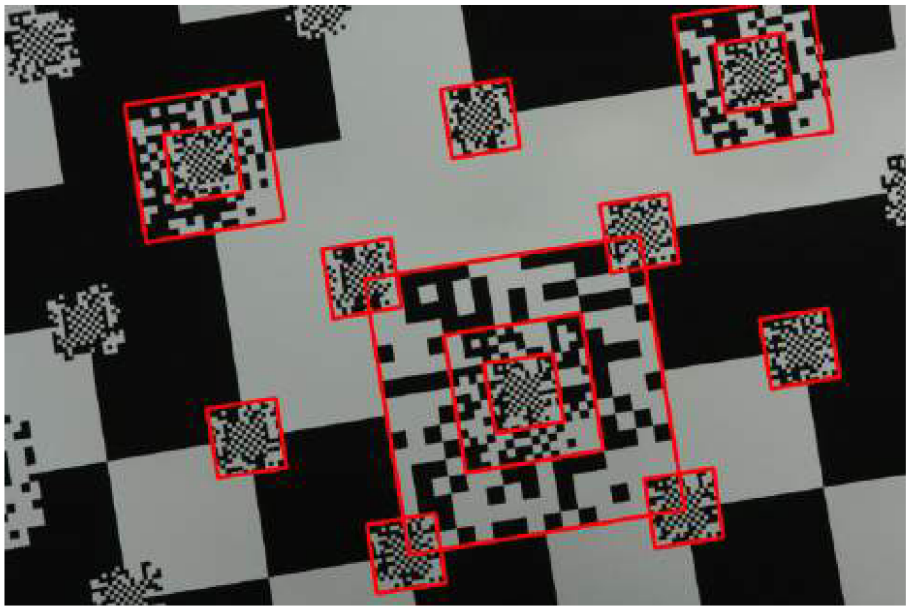

Figure 2.4: Fractal Marker Fields [15] with detected sub-markers.

every scale, solving the main problem of limited distance range useable for detection.

Fractal Marker Fields would be an ideal solution in large-scale situations, where markers are acceptable. Practical applications of the allowed freedom in scale is limited. In most real-world augmented reality applications: human-computer interaction, 3D visualization, medical training, etc. – 2-3 scale levels at most are sufficient. The biggest disadvantage of Fractal Marker Fields is the dependence on computationally complex detection algorithms.

Uchiyama et al. [37] used randomly scattered dots as fiducial markers (Random Dot Markers – Figure 2.5 left). Compared to traditional markers with square patterns, Random Dot Markers require slightly larger area, so that the camera could recognize the individual points for detection. On the other hand, random dot markers are more robust against occlusion.

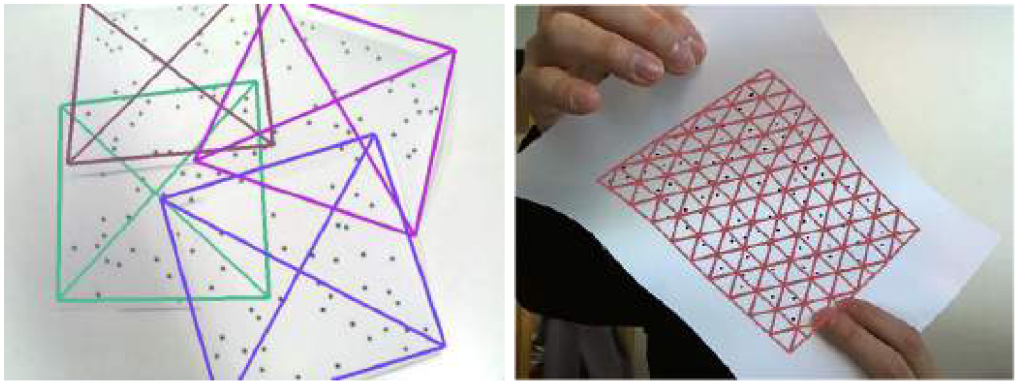

Figure 2.5: Random Dot Markers [37] demonstrating robustness against occlusion. **Right**: Deformable Random Dot Markers [36] with detected mesh.

The advantage of Random Dot Markers is that it is not constrained by a square area and it has excellent robustness against occlusion. Theoretically any shape can be used for the marker. The disadvantages of their approach lie in the memory-intensive keypoint extraction, questionable choice of descriptors, and the sensitivity of the detection algorithm to the chosen dot size. Since the geometric descriptor uses only local arrangements of points, the Random Dot Markers could also be applied to slightly curved surfaces (Figure 2.5 [36]) to recover the deformation.

The amount of research on fiducial marker based tracking has been on decline in recent years. This suggests that the research in this area has achieved high maturity. Marker based approaches are slowly replaced by natural feature based tracking. However, where simplicity, precision and computational efficiency is critical, marker detection based methods are still superior and widely used method.

2.4 Augmented Reality Applications

In the state of the art describing mostly new and improved methods of camera pose tracking, scene modeling and visually correct rendering, there are many applications envisioned for Augmented Reality. The fields of these applications also varies widely, ranging from medical training to children book coloring. Most of the research, though, is focused on individual pieces required to create a full Augmented Reality experience. Industry-ready use cases of augmented reality are almost completely non-existent. In the state of the art also comprehensive user evaluation of the proposed systems is missing. The enabling technologies and the commodity of high-performance smartphones represent a landmark, which could boost this research area in the near future. This thesis also focuses on these realistic use cases and presents several use cases of AR including a cheap match-moving solution and efficient inter-device content acquisition.

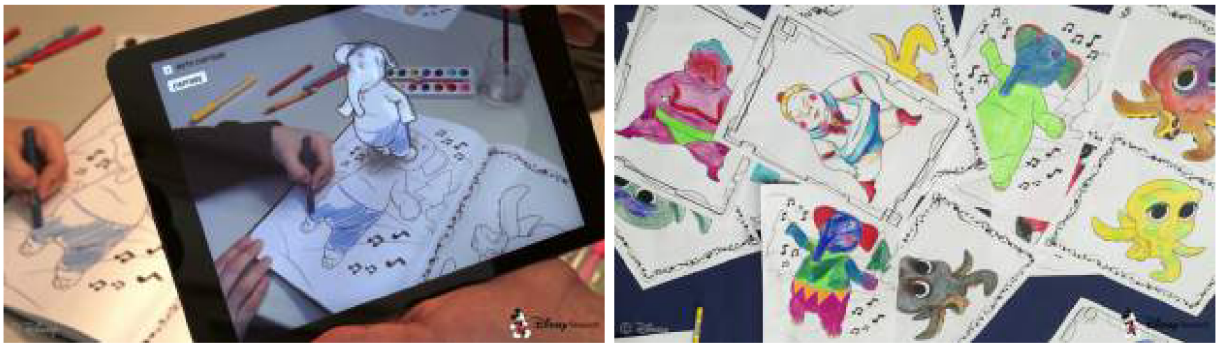

Figure 2.6: Interactive coloring book with live animated characters textured based on the drawing. The templates for augmentation are on the right.

Recently, Magnenat et al. [25] from Disney Research showcased an application for an interactive coloring book (see Figure 2.6) using BRISK feature point detector and

descriptor. The disadvantage of using feature point descriptors on binary images is the low discriminatory power between feature points. In their work, they had to use complex heuristics to filter out outliers and in some cases they added complex visual patterns around the drawing to achieve reliable camera pose estimation and tracking.

A demonstration of the growing computational power of smartphones and gradually maturing state of dense tracking algorithms is the work of Ondruska et al. [27]. They demonstrated an application supporting full volumetric surface reconstruction and dense tracking in real time on mobile phones.

A frequent AR application in the state of the art is mentoring or training. Recently, Zhu et al. [40] demonstrated a wearable real-time AR mentoring system to assist in complicated maintenance and repair tasks using a hybrid approach with a high-latency visual landmark matching and feature tracking modules, and a low-latency IMU prediction module.

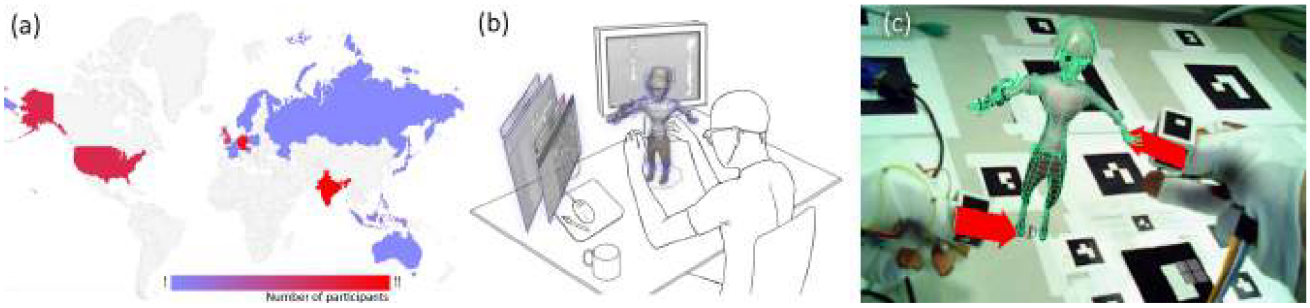

Figure 2.7: A user interface design and prototype for professional 3D media production AR system [20]. **(a)**: the distribution of participants in their survey during the design process; **(b)** the envisioned AR system; **(c)**: the implemented prototype.

One of the goals set forth in this thesis was to showcase the possibility to create real-time functional AR systems with practical use even in the real world and media production. A similar notion also lies behind the recent work from Krichenbauer et al. [20]. They explored the possibilities to create an immersive 3D UI for 3D computer graphics content creation. For robust 6DOF camera pose estimation and hand-tool localization, they used fiducials in their prototype implementation. These examples further accentuate the fact that fiduciary markers are still the go-to solution for prototype systems, when fast and robust solution is required, and that fiduciary markers are still a relevant research area.

Chapter 3

Uniform Marker Fields

This chapter presents the technological core of my PhD research. My main contributions in this thesis are centered around Uniform Marker Fields generation, detection algorithms and experimental evaluation. Some of the work resulted from cooperiation with others: Adam Herout and Michal Zachariáš in relation to marker design and generation and Jiří Havel in relation to de Bruijn tori theory and line parametrization.

3.1 Orientable Window Arrays as Marker Fields

Perfect maps are 2D arrays in which every possible rectangular subarray of a given size occurs exactly once. The perfect map can be either periodic or aperiodic. An *aperiodic* (m, n)*-window array* [7] is an k-ary 2D array of size $h \times w$

$$A = (a_{i,j} \in \{0, \cdots, k-1\}; 0 \leq i < h; 0 \leq j < w), \tag{3.1}$$

in which each subarray $A_{r,c}$ of size $m \times n$ occurs exactly once.

If all possible subarrays are used (i.e. $(w+n-1)(h+m-1) = k^{mn}$), the (m, n)-window array is called aperiodic *perfect map* [32]. Opposite edges of the array can be connected together for a *periodic window array*. Of course, the windows created by the connection must also be unique (i.e. $wh \leq k^{mn}$).

Unfortunately, when the orientation of the array is not known, the simple (m, n)-window property of a window array is not enough. It is possible that multiple rotations of the same window can occur in the array. *Orientable window arrays* [7] solve this problem.

4-orientable arrays can distinguish all four rotations of the array (e.g. "north", "east", "south", "west"). The 4-orientability is reasonable only for square windows, that must be unique in respect to rotation by $90°$. It is self-evident that 4-orientable arrays are always also 2-orientable [14].

Contrary to the 1-orientable maps, 4-orientable arrays are much less explored in the literature and no good construction algorithms existed for them before our proposed algorithm in [34] and [16]. Similar work on 4-orientable 2D window arrays to the best of my knowledge have been missing in the literature.

Binary 4-orientable window arrays can be visualized as 2D checkerboard structure, where the white and black modules are reorganized to match the values in a window

array (Figure 3.1). As a more general definition, *Uniform Marker Fields* are visual patterns – fiduciary markers – made up of square modules forming a regular grid, where windows of size n^2 are unique in every rotation.

3.2 Synthesis of Binary n^2-Window Arrays

Binary 4-orientable aperiodic binary n^2-window arrays with $n = 3$, a (square) map cannot be larger than 12×12. Our algorithm described in this section has found a number of 11×11 arrays (Figure 3.1 left). Thus, 3^2-window arrays can be used as Uniform Marker Fields, but the dimension of the field is very limited and the benefits over any existing marker designs are not very interesting. The theoretical upper bound for the dimensions of a square map with $n = 4$ is 127×127. By the algorithm presented in this section, 4-orientable 4^2-window arrays as large as 92×92 have been found by using a supercomputer (Figure 3.1 right).

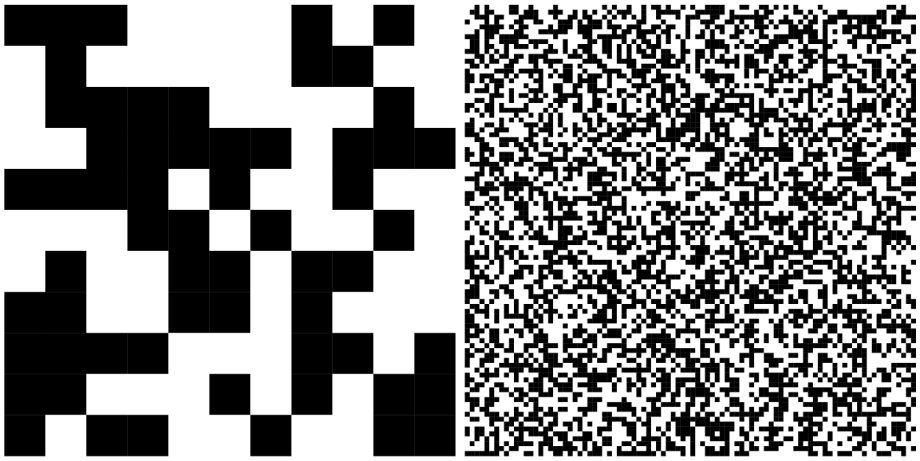

Figure 3.1: Largest generated 4-orientable n^2-window arrays using our synthesis algorithm, with $n = 3$ on the left and $n = 4$ on the right.

The literature does not provide any efficient construction method for 2 or 4-orientable n^2-window arrays. Exhaustive search is not feasible as a construction method: for example, for a 4^2-window array 90×90 modules large, the area of the map to be searched for is 8100 modules and the state space is just too large (2^{8100}).

We proposed a genetic algorithm which works with maps containing conflicts and improves it continually by mutations that lead to decreasing the conflict count. We also applied several heuristics for faster convergence. In order to distribute calculations required to solve the conflicts in randomly generated arrays, we used a **client-server** architecture.

The genetic algorithm can be characterized by these terms:

- For the initial population we use a number of copies of the same array, or various arrays are generated randomly.

- The fitness function of an individual is based on the number of conflicts in the given array $f(A) = \frac{1}{c(A)+1}$.
- The fitness threshold, where the algorithm is stopped is set to 1 (the algorithm is looking for conflict-free maps).
- For selecting members for the next generation, rank selection is used.
- Mutation is defined as replacing a window with randomly generated content. The windows are selected randomly; the conflicting windows have a higher probability of being selected for replacement.

We generated a set of binary aperiodic 4-orientable n^2-window arrays. Table 3.1 gives the highest resolutions of the window arrays available for the respective aspect ratios in the data set.

aspect ratio	available dimension
1:1	92×92
$\sqrt{2}$:1	110×78
2:1	122×61
3:1	159×53

Table 3.1: Available sizes of the binary 4-orientable aperiodic 4^2-window arrays.

3.3 Detection of binary UMF

My main contribution related to Uniform Marker Fields is the efficient detection algorithm. The Uniform Marker Field construction does not distinguish between marker design features intended for general marker *localization* and features for marker *identification*. Checker-board modules serve simultaneously as the localization and identification features. This approach is more space efficient and provides more uniformly distributed points of interest for 2D-3D correspondences.

The detection algorithm was designed so that it visits as small a fraction of the image pixels as possible, and assumes that a significant portion of the input image is covered by the marker field. The algorithm performs the following main steps:

1. **Extraction of edgels** – edgels are described by an image point and edge orientation (vector) or by two endpoints.
2. **Determination of two dominant vanishing points** among the edgels. The vanishing points define the horizon (a line connecting the vanishing points). Using

homogeneous coordinates for the vanishing point $\mathbf{v}$ and the cluster of lines $\mathbf{l}_i$, all the lines must be coincident with the vanishing point, i.e.

$$\forall i : \mathbf{v} \cdot \mathbf{l}_i = 0. \tag{3.2}$$

The vanishing point is found as the direction of the least variance by eigendecomposition of the correlation matrix

$$C = (\mathbf{l}_0 \dots \mathbf{l}_N)(\mathbf{l}_0 \dots \mathbf{l}_N)^T. \tag{3.3}$$

3. **Finding the grid of checker-board edges** as two groups of regularly repeated lines coincident with each vanishing point. The lines in each group corresponding to the edges of the grid squares can be computed using the horizon as ($\hat{\mathbf{x}}$ denotes normalized vector)

$$\mathbf{l}_i = \hat{\mathbf{l}}_{base} + (ki + q)\,\hat{\mathbf{h}}, \tag{3.4}$$

where $\mathbf{l}_{base}$ is an arbitrarily chosen base line coincident with the vanishing point, different from the horizon. First, $(ki + q)$ is estimated for each line. The values are clustered to recover the density k of the grid and offset q.

4. **Extraction of checker-board modules** using the grid and localization of the camera view within the 4-orientable n^2-window array. Points

$$\mathbf{x}_{ij} = \mathbf{l}^{(1)}_{(i+1/2)} \times \mathbf{l}^{(2)}_{(j+1/2)}, \forall i, j \in \mathrm{N} \tag{3.5}$$

are intersections of lines right between the edge lines: points in the middle of the checker-board square modules. These locations are sampled from the input image. Once the sampled values have been filtered using an adaptive threshold, each 4×4 window's location inside the sampled region is found using a hash function.

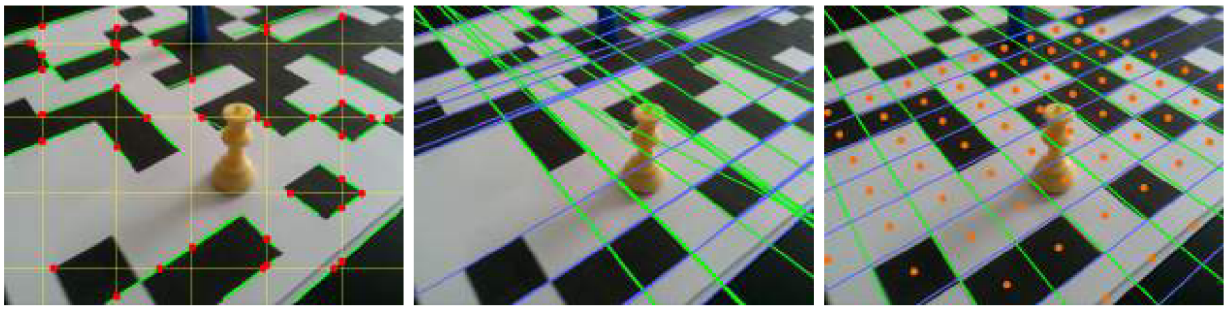

Figure 3.2: Left to right: Edgel extraction along scanlines; dominant group of lines; finding the grid of the marker.

For testing purposes of this initial solution, I collected a set of videos acquired by 3 different smartphone cameras at resolution 640×480 or 720×480 with 24 frames per second, each 20 to 30 seconds long. In order to evaluate the detection precision

for different types of movements, I split the dataset into 6 *categories* according to the dominant *movement* manifested in each video. I used two different marker fields: a low-density marker field (14×10) and a high-density marker field (28×19). To estimate a baseline of the algorithm's robustness we did not use information from previous frames. Table 3.2 contains the percentage of frames, where the camera pose was successfully recovered. The results show that our algorithm performs well even for very challenging videos with rapid movement causing directional blur, rotation, and high perspective distortion.

Category	Low density	High density
Zoom	94.5 %	92.0 %
Horizontal	97.3 %	99.4 %
Rotation	99.9 %	99.0 %
Perspective	99.8 %	99.4 %
General movmement	95.3 %	95.0 %
Occlusion	91.9 %	92.5 %

Table 3.2: Success rate for detecting the position in the marker with different categories and marker densities.

I measured the required time of different components of the detection algorithm. Table 3.3 shows the percental distribution of computational time between different components.

Algorithm part	time	percent
Scanlines	0.21 ms	16 %
Edgel extraction	0.22 ms	16 %
Vanishing points and Grid	0.11 ms	8 %
Module extraction	0.06 ms	5 %
Camera localization (OpenCV)	0.74 ms	55 %
Overall	**1.34 ms**	**100 %**

Table 3.3: Breakdown of computational time into different parts of the algorithm, measured on an Intel Core i5 661, 3.3 GHz with a DDR2 memory.

3.4 Five Shades of Gray

In a follow-up work, we generalized the Uniform Marker Fields construction to grayscale or color k-ary marker fields ($a_{ij} \in \{0, \ldots, k-1\}$, Figure 3.3). In comparison with binary marker fields the absolute greyscale or color values of the grid modules cannot

be reliably discerned under varying lighting and camera conditions. We used the *edge gradients between the modules* in a single $n \times n$ window as the unique window array property for localization within the marker field. Horizontal (3.6) and vertical (3.7) edge gradients are defined as:

$$e^{\rightarrow}_{ij} = a_{i,j+1} - a_{ij}, \tag{3.6}$$

$$e^{\downarrow}_{ij} = a_{i+1,j} - a_{ij}. \tag{3.7}$$

The absolute value of the edge gradient is also hard to recognize reliably and thus only the basic character of the edge is used for recognition: $g^*_{ij} = \operatorname{sgn} e^*_{ij} \in \{-1, 0, +1\}$. The n^2-window used for localization within the marker field then is (Figure 3.3):

$$G_{rc} = (g^{\rightarrow}_{rc}, \ldots, g^{\rightarrow}_{(r+n-1,c+n-2)}, g^{\downarrow}_{rc}, \ldots, g^{\downarrow}_{(r+n-2,c+n-1)}), \tag{3.8}$$

where G_{rc} is the unique window at position (r, c) inside the window array. Given this ternary classification of edges, grayscale markers can be seen as a generalized version of $k = 3$-ary n^2-window arrays, and color marker fields as $k = 3c$-ary n^2-window arrays, where c is the number of channels in the used color model.

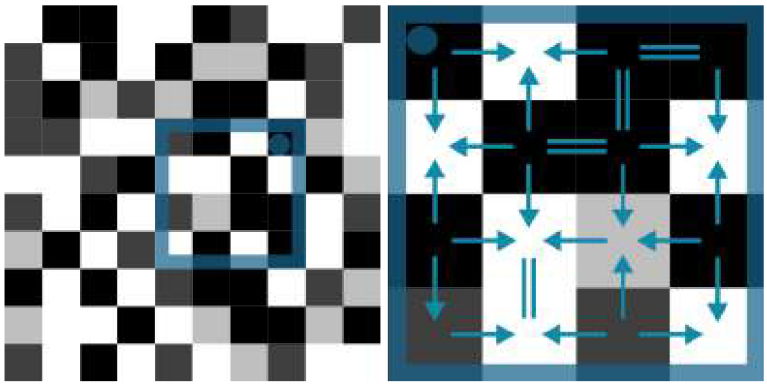

Figure 3.3: Uniform Marker Fields with several shades of grey. The highlighted blue sub-window is unique in the map considering the edge directions as seen on the extracted region.

Synthesis of the marker field is done in a manner similar to the genetic algorithm sketched out in Section 3.2. In this case, the fitness function additionally also reflects the quality of edges between the modules.

Detection

Figure 3.4 summarizes the detection algorithm for grayscale UMF. We added several improvements to the detection algorithm. We used a simple rectangular mask to filter out the edges outside the area corresponding to the previously detected marker field. We

added one additional step to filter out outliers inside each group of lines using RANSAC-like approach before the vanishing point calculations. We also used improved clustering of $(ki + q)$ values using mean-shift.

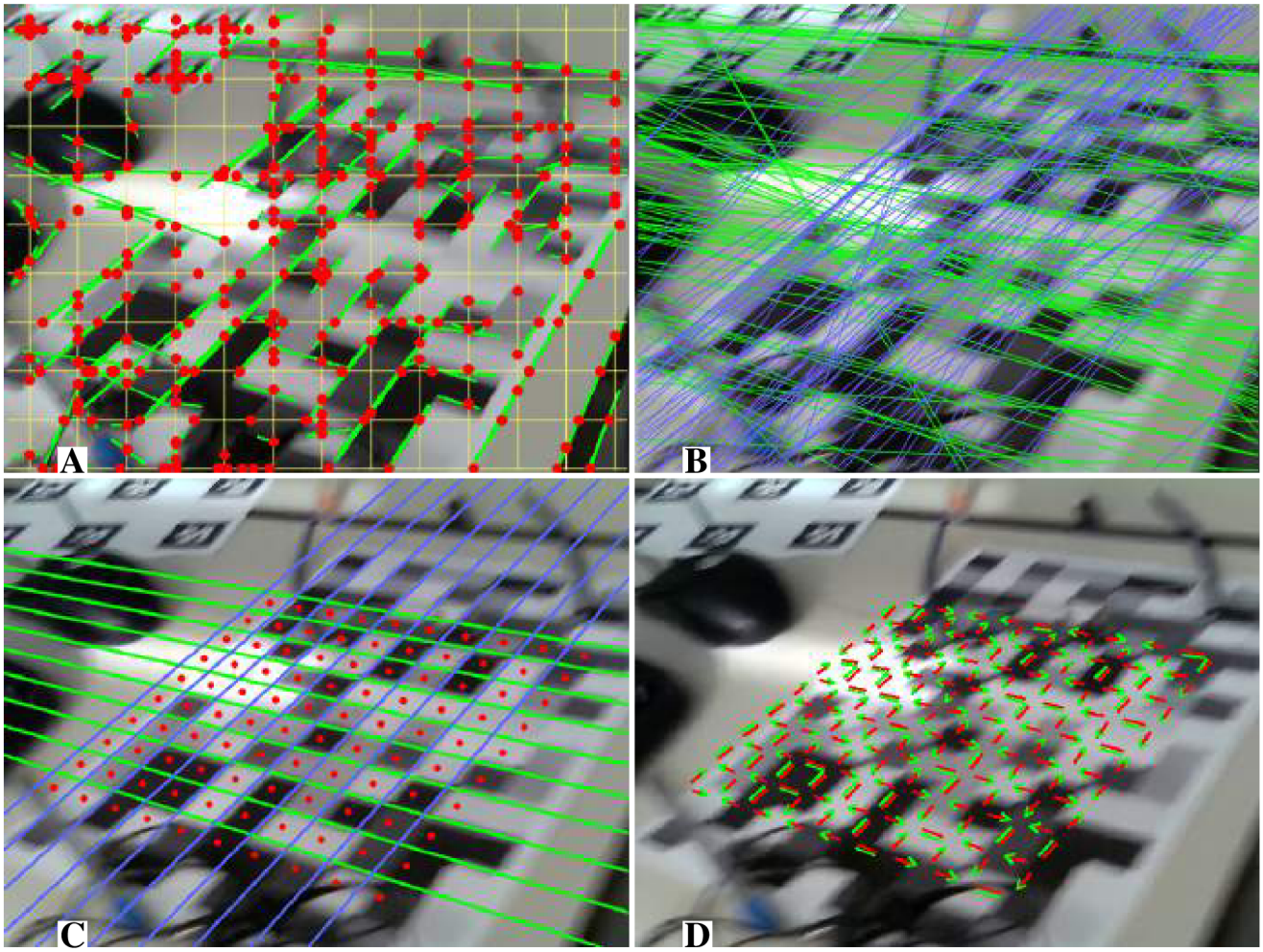

Figure 3.4: Detection of the greyscale grid of squares. **A:** The image is processed in sparse scanlines. On each scanline, edges are detected (Red) and extended to edgels (Green) by iteratively finding further edge pixels in the direction perpendicular to the gradient. **B:** The edgels are grouped into two dominant groups using RANSAC; two vanishing points are computed by hyperplane fitting. **C:** Based on the vanishing points, the optimal grid is fitted to the set of the edgels (orange dots denote the estimated centers of grid modules). **D:** Edges between the modules are classified.

Due to the design decisions for the generalized Uniform Marker Fields, the proposed algorithm after step 3 diverged significantly from the original algorithm. 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 and compares them to vote for edge direction. The stopping criterion is given by Wald's sequential probability ratio test [38], which is proven to be the optimal sequential test for this purpose.

The sub-window described by edges G_{rc} is formulated as a vector of scalars in (3.8).

Instead of using a ready-made hash table, we prefer to create a **decision tree**, which could be constructed fault-tolerant. For a precise camera pose estimation the algorithm finds all possible corners between the square modules in the marker field (with a sub-pixel precision).

Experimental Results

We compared our solution to ALVAR [1] as the most mature available ARToolKit follower (ARTag is no longer publicly available). ALVAR supports arrays of disjointed square markers. The other baseline is the Random Dot Markers (RDM) [37] as an alternative "marker field" solution, where individual localization markers overlap in the field and exhibit robustness against occlusion.

For comparing our solution with the alternatives, we shot videos of side-by-side markers (Figure 3.5). We evaluated the precision of our algorithm using local variance (Table 3.4) and performance as the percentage of frames with successfully estimated camera pose (Table 3.5).

Method	**RDM**	**ALVAR**	**UMF**
Average position variance:	8.5 cm	3.48 cm	3.28 cm
Average rotation variance:	0.049	0.035	0.024

Table 3.4: The average variance in position and rotation change using 10 frames for averaging in a 1080p 50FPS video. The rotation variance is expressed as variance of quaternions, since the euler angles are unstable due to the gimbal lock. (Note: RDM gave highly unstable results and the low average variance in rotation is caused mainly by the low detection rate. For the *rotation* test video it gave 0.080 variance.)

Method	**RDM**	**ALVAR**	**UMF**
Lighting	89.7	100.0	100.0
Perspective	42.7	100.0	100.0
Near/Far	75.8	91.3	93.4, 94.6
Rotate	94.7	100.0	100.0
Zig-Zag	29.6	98.3	97.5, 97.4
Occlusion	38.5	93.0	94.0, 96.5
Overall	**61.8**	**97.1**	**97.8**

Table 3.5: Marker field detection success rates in %. For Uniform Marker Fields, rates from comparison videos with RDM and ALVAR are given separately, if different. Success rate is the percental ratio of video frames where at the different markers were correctly detected.

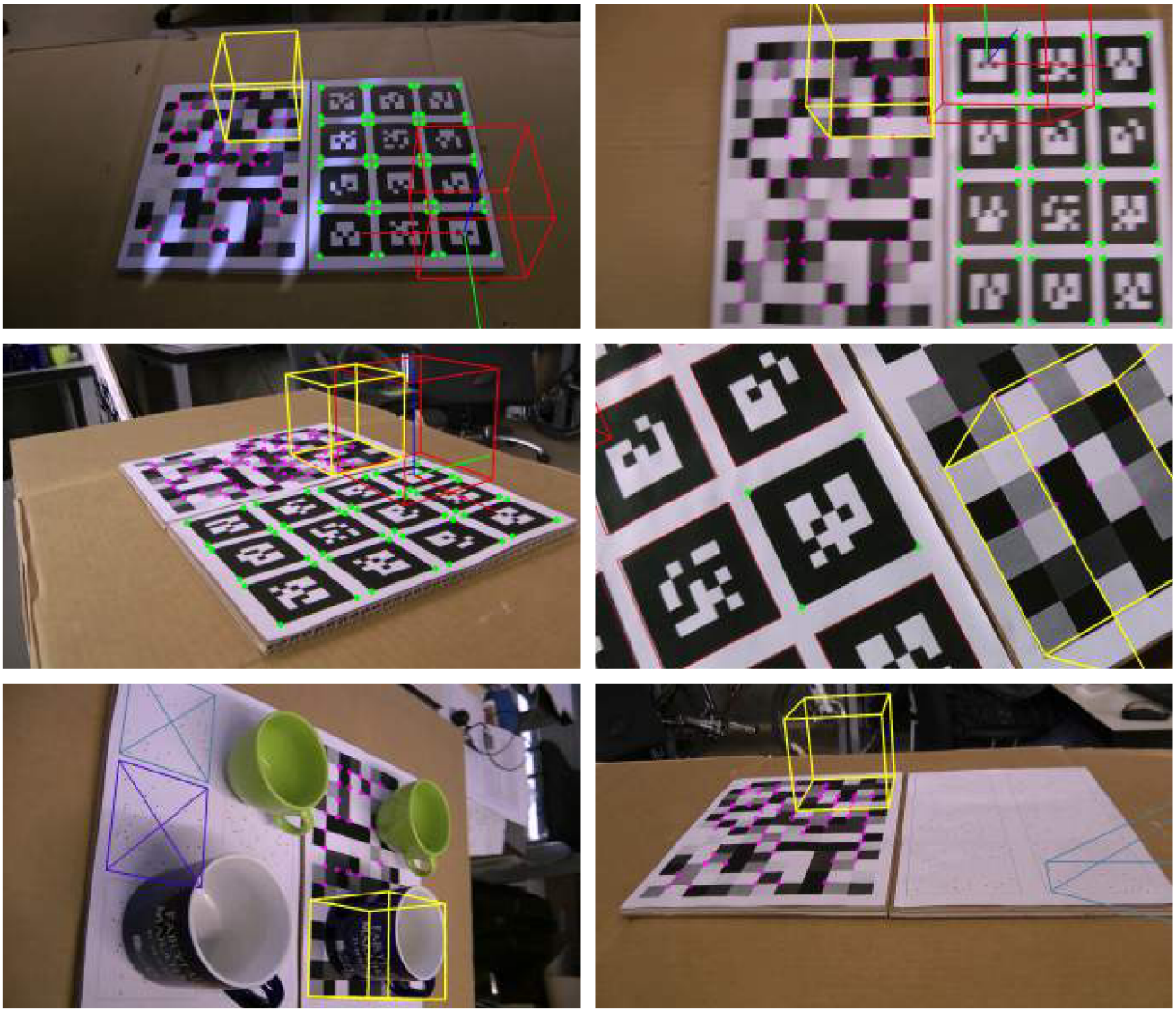

Figure 3.5: Sample images from the dataset. Purple dots are the detected corner points for UMF used for camera pose estimation.

Table 3.6 shows the speed of the three tested algorithms and the breakdown of speed of our marker detection algorithm for videos with $1920 \times 1080\,px$ resolution. Our algorithm was more than $3\times$ faster than ALVAR and visited on average about $5.3\,\%$ of all pixel points.

We used a cluster of computers (~ 1000 nodes) to synthesize the marker fields with highest possible resolutions. For $n > 3$ the several marker fields of size 250×250 were found.

RDM	ALVAR	UMF	(edge	grid	match	cam	sref)
164.4	30.1	8.8	(3.8	1.1	0.3	0.7	2.9)

Table 3.6: Breakdown of speed in milliseconds for 1080p videos using a mid-range Intel(R) Core(TM) i5 CPU 661 (3.33GHz) CPU. **edge:** edgel detection in scanlines; **grid:** reconstructing the grid using RANSAC and vanishing point detection; **match:** edge direction detection and position decision making; **cam:** camera pose estimation based on the found matches; **sref:** processing in subwindows and position refinement by iterative search for more corner points.

3.5 Indoor Localization by UMF

We created a dataset of images to measure the precision of the Uniform Marker Fields detector in collaboration with M. Zachariáš et al. [39] as part of our research concerning indoor navigation. We marked 6 different view points relative to a projected grayscale Marker Field with $14.3\,cm$ module size.

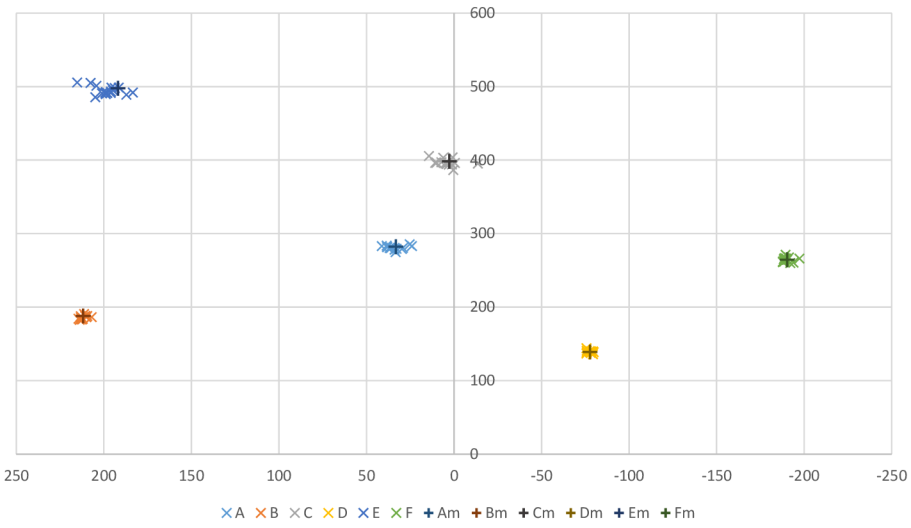

Figure 3.6: Detected positions from the (15+ photos for each test point) - A, B, C, D, E, F points. The + points are the illustrative reference point positions - $A_m, B_m, C_m, D_m, E_m, F_m$.

We took images from the 6 reference points with a smartphone camera (Nokia Lumia 930) at $1920 \times 1080\,\mathrm{px}$ resolution. For each reference point we took $15+$ photos of the scene at different angles and 3 different heights from the ground.

Figure 3.6 shows the results. The ground truth points in Figure 3.6 are for illustration purposes only. The overall standard deviation of the distances between pairs of detected positions from the ground truth was $6.73\,\mathrm{cm}$ and the median $3.21\,\mathrm{cm}$.

Chapter 4

Fast Grid Detection

One important assumption during the design of UMF and its detection algorithm is that it covers a significant portion of the input image. For large scenes, this assumption is unrealistic. In this chapter, I propose an efficient search for candidate positions for regular grid patterns. While I demonstrate this approach for detecting QR codes, the algorithm is not limited to any specific marker or Matrix Code.

QR codes [2] are a very popular case of matrix codes (or 2D barcodes). They are receiving an increasing popularity among smartphone users and are becoming the standard when it comes to short data migration into their devices. Their detection in high-resolution images of real-world complex scenes is desirable.

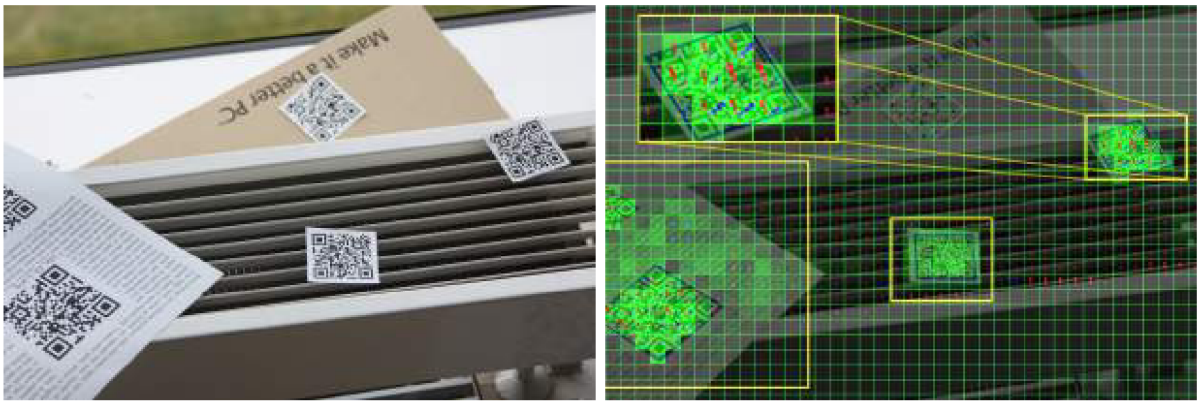

Figure 4.1: **Left:** The original 15MPix image with multiple codes present in the image. **Right:** The grid with histograms of oriented gradients.

The QR code was designed in such a way, so that it can be easily localized by finding the predefined structures at its three main corners. There have been many attempts (e.g. [6]) to speed up and improve the detection of the QR codes the way it was intended using the *Finder Patterns* (FIP), but even for recent solutions [6], it takes 50-150 ms to process a 640×480 image in one pass and several passes are needed.

We proposed a hierarchic segmentation approach based on the distribution of the histogram of gradients in tiles. The whole algorithm for detection of QR codes in a high-resolution image is depicted in Algorithm 1, where $\mathbf{H}_n$ is the histogram with n bins, and $\mathcal{T}_l(u, v)$ is a tile at level l and position (u, v). The probability of a segment being part of a QR code $P(S_i)$, depends on the distribution of gradients and two most dominant edge orientations.

Algorithm 1 QR code detection in high-resolution images

Input: Image I
Output: Detected QR codes
compute $\mathbf{H}_n(\mathcal{T}_1(u,v))$ by edge extraction
compute $\mathbf{H}_n(\mathcal{T}_l(u,v)), l \in \{2,\ldots,l_{max}\}$ from lower-level histograms
for all $l \in \{1,\ldots,l_{max}\}$ **do**
    compute feature vectors $\mathbf{v}_l(u,v)$ from the histograms
    compute the segments $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}, k \in \mathbb{N}$
    for all $S_i \in \mathcal{S}$ **do**
        compute segment probability $P(S_i)$
        if $C(S_i) == 1$ **then**
            run QR code detection algorithm
        end if
    end for
end for

In order to evaluate the performance, we collected a dataset of challenging real-life images. Since no standard dataset was publicly available for evaluation of QR code detection algorithms, we acquired the images ourselves.

The results show that our candidate search has very low false negative rate (7.4 %) and acceptable false positive rate (52.9 %). We compared our solution combined with QR code detection library proposed by Herout et al. [11] with publicly available ZBar[1] library. Our solution gave comparable detection performance, while being 4 times faster (Figure 4.2).

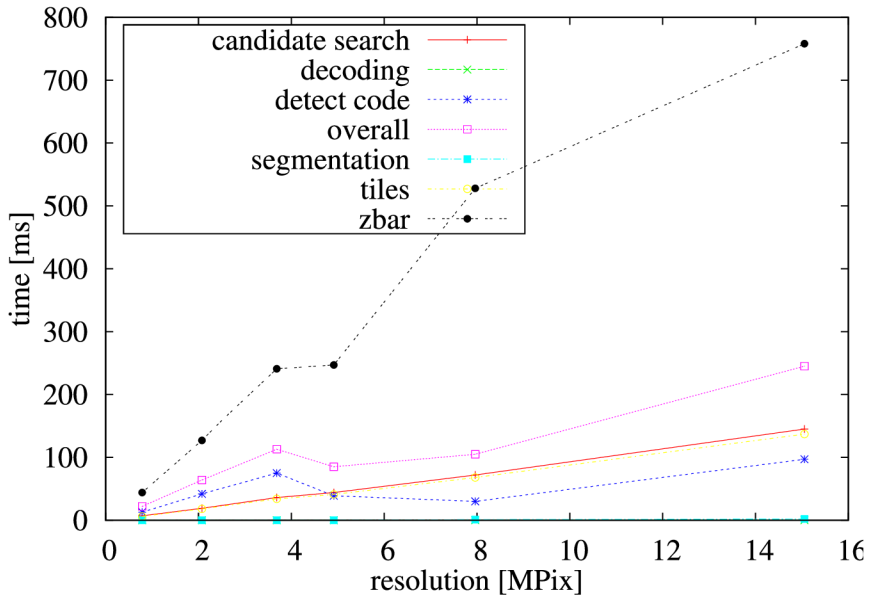

Figure 4.2: The required time for processing in ms. The graph also shows average required times for different parts of our solution.

[1] zbar.sourceforge.net

Chapter 5

On-screen Markers

Our Uniform Marker Field solution enables certain domain-specific augmented reality applications. One of them, where the markers are placed onto a computer screen so that it can be detected and recognized by an ultramobile device (typically a smartphone), is described in this chapter. This work was done in collaboration with other colleagues (primarily Rudolf Kajan) who provided the user interaction expertise and the use-case itself. I was responsible mostly for the client-side communication, and smartphone localization method and its evaluation.

With the appearance of large and cheap high-resolution network-connected displays, and smartphones becoming a widespread personal accessory, the ubiquity of screen real-estate naturally drew the interest of many researchers to examine the possibilities for interaction between these devices.

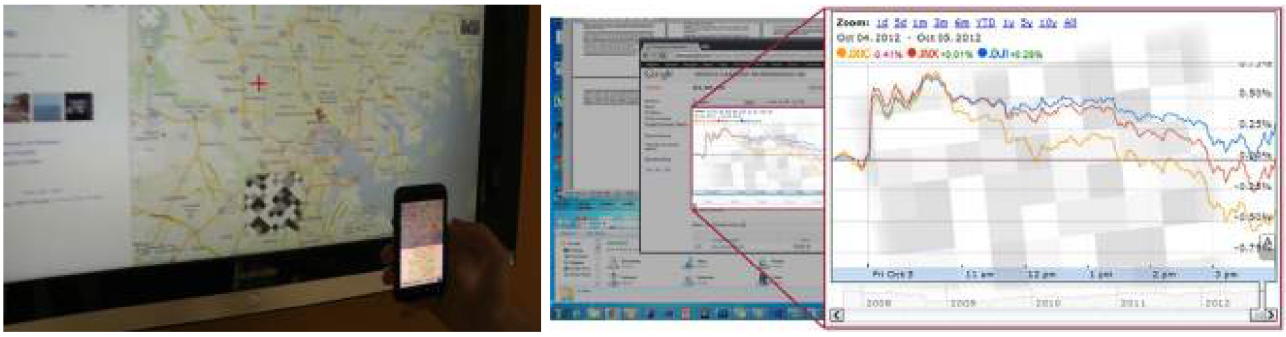

Figure 5.1: In our work, we aimed at exchange of information between a large screen (a desktop computer, a public kiosk, etc.) and a mobile device. This information exchange should be visual and intuitive: based on the metaphor of "video recording" with the mobile camera. We achieved this by inserting a cutout from a Uniform Marker Field into the monitor screen that would be reliably detectable and could accurately establish the location within the screen.

For a vast majority of applications, the initial assumption is still that users interact with just a single computing device throughout the day. The practical consequence of this assumption is the lack of collaboration among devices and lack of user-centric activities that may span multiple devices as well as multiple applications. While there are initial steps in this direction [5], they must support a wider variety of activities and fully recognize the members of a user's device collection.

Pierce et al. [29] introduced an infrastructure based on instant messaging which provides mechanisms for applications to send information, events and commands among devices. Chang and Li proposed DeepShot [8] – a framework for capturing work state which uses natural visual features and tracks them. Despite various techniques to balance the features' density in the camera view, it is impossible to ensure the presence of enough visual features in the whole camera view. In the case of observing a computer screen, the problem is even more difficult, because unlike the real world, the monitor screen tends to contain surfaces of exactly constant color

In our research, we wanted to go further and provide users with a lightweight solution for information transfer, able to work with different types of information and contexts, respects the need for privacy and supports additional metadata generated through interaction which is useful for future interactions on other devices.

5.1 Proposed System Architecture

We have designed a highly responsive system, which allows for intuitive task migration without the need of manual application state inspection or copying of "raw" pixels without any additional semantic information (as done in Deep Shot [8]). The task migration process from the system architecture's point of view is a two-way communication between a *content provider* and a *content requester* device (see Figure 5.2).

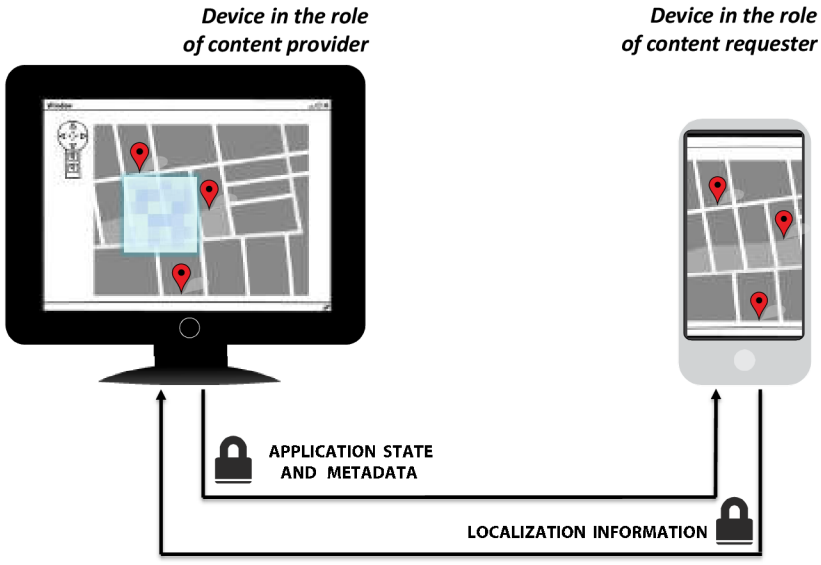

Figure 5.2: The task migration process between a content requester and a content provider device. The content provider device creates an unobtrusive maker field overlay which enables fast and accurate within-screen view localization of the requester device. This localization information is used to select either full application state or to migrate selected content to the requester device.

5.2 Marker Field's Design and Detection

In our approach we tried to minimize the required time to localize the client relative to the content provider with high stability. We achieved this by using grayscale Uniform Marker Fields (Section 3.4).

Showing the whole marker on the whole display would be highly obtrusive. Instead, only a small part of marker is shown, which is still reliably detectable by our detection algorithm (see below). We tested constant transparency or pulsing between transparency levels ($25 - 75\%$ during performance evaluations) to achieve high detection rates and make the marker less obtrusive.

In order to minimize the outliers caused by the most commonly occurring horizontal and vertical lines in display content (window borders, menus, vertical panels), we rotated the whole marker by angle α ($\alpha = \frac{\pi}{4}$ in our tests). To avoid introducing additional long edgels into the content provider's display, we also used smoothed or sinus border mask.

There were no major changes necessary to the detection algorithm (Section 3.4). The content provider's orientation is assumed to be mounted on the wall without any rotation. A minor improvement is that the extracted edgels in the mobile device's view are filtered based on the its orientation acquired from its built-in accelerometer or gyroscope and the marker orientation (α) on the content provider. To make the edge classification more robust against transparency, we checked more sample points than the stopping criterion by Wald's sequential probability test [38] for reduncancy.

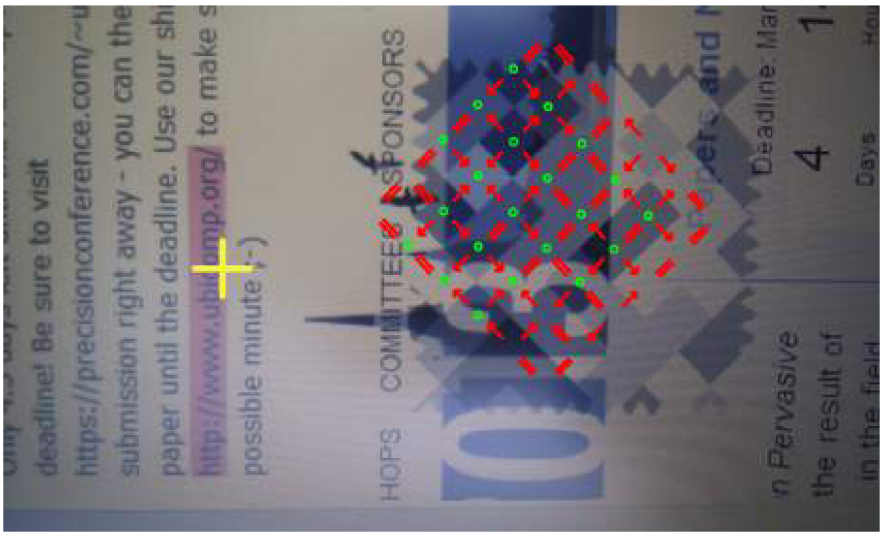

Figure 5.3: A succesfully detected Uniform Marker Field as seen by the requester device's camera with a crosshair shown to aid the user with targeting. The purple highlight is the content provider's reaction to targeted content based on continuous interaction.

The algorithm also does not compute the full homography, since the detected grid and marker position is sufficient to compute for arbitrary image pixel the position inside

the marker field (Figure 5.3). Given the decoded marker position, $\hat{\mathbf{l}}_0$ in each pencil represents either a set of rows starting at index l_{0r} or columns starting at l_{0c}. The position $(m_r; m_c)$ in the marker field of an arbitrary point $\mathbf{p}$ in the camera image can be determined by solving:

$$\mathbf{p} \times \mathbf{v}_r = k\hat{\mathbf{l}}_{0r} + (m_r - l_{0r})\hat{\mathbf{h}}, \tag{5.1}$$

$$\mathbf{p} \times \mathbf{v}_c = k\hat{\mathbf{l}}_{0c} + (m_c - l_{0c})\hat{\mathbf{h}}, \tag{5.2}$$

where $\mathbf{v}_r$ and $\mathbf{v}_c$ are the determined vanishing point for the pencils representing rows and columns, and $\hat{\mathbf{h}}$ is the horizon. We used Kalman Filter to achieve smooth interaction.

5.3 Implemented Solution – Chrome and Android

As a proof of concept and as the testing prototype for user testing and exact experimental evaluation, we created a pilot version of the whole system. It consisted of:

- The content provider background service for Microsoft Windows,
- Google Chrome extension as the application-side provider module,
- Android application as the client.

My contributions in this prototype was the client side communication and Uniform Marker Fields detection.

The application uses the video stream from the camera to identify the position and orientation of the *content requester* relative to the *content provider*. The marker field detection algorithm was implemented in native code through Android NDK toolset that allows implementation application parts using native-code languages such as C and C++.

The detection algorithm computes the position inside the marker and also the position of a virtual cursor (see Figure 5.3). These coordinates are sent to the *content provider*, which uses them to extract semantic content and move the visible marker field fragment on the *content provider* device.

5.4 Experiments: Empirical Tests and User Study

We conducted an initial user study to observe how would people use our prototype. Our main goal was to find out how obtrusive was the usage of marker fields for task migration for participants and whether this approach is feasible also for inexperienced users.

In general, our system was perceived very positively, with 86% of participants stating that it would definitely help them with content reaccess. 72% of participants would use it to obtain information from public displays. In this case, the biggest concern were privacy issues.

We tested the reliability of our marker field detection algorithm, with the marker mixed into natural screen contents. The results show, that we were able to detect reliably the content requester's position and viewing angle within 5 frames with 95 % probability on average over all angles and marker types (constant opacity, pulsing, different borders, etc.).

For accuracy measurements, the mobile device was fixed with the visible fix-sized marker segment moving around on the screen. Table 5.1 shows the standard deviation of the determined position of the crosshair on the content provider's display in pixels. We did not use Kalman filter for these measurements.

[pixels]	**75°**	**90°**	**105°**	**120°**
$10cm$ constant	11.0	10.0	10.0	27.6
$10cm$ pulsing	6.3	7.6	8.9	11.8
$20cm$ constant	27.7	19.4	21.9	26.4
$20cm$ pulsing	17.0	24.1	24.3	27.0
$30cm$ constant	34.4	27.2	22.6	23.4
$30cm$ pulsing	25.0	27.2	23.9	23.0

Table 5.1: Standard deviation of the detected positions in content provider's coordinate system in pixels.

The accuracy of the algorithm without corner detection and full homography calculations is relatively low. On the other hand, an unstabilized hand-held mobile device would cause even larger variance in position. As a solution we used a Kalman filter, modelling position and speed of the detected position (measurement variance set to $\sigma^2 = 400px$). The accuracy was sufficient to select blocks of text, map regions, images or menu entries.

The overall average time required by our baseline implementation for mobile platforms – excluding the system overhead to acquire the image – was $24.5\,ms$ (∼40 FPS) for 800×480 resolution. The results show a significant speed increase when compared to task migration solutions based on visual features - authors of the DeepShot [8] task migration framework report 7.7 seconds (SD 0.3 seconds) for processing the request. Our approach allows for real-time information feedback for a selected screen area.

5.5 Continuous Task Migration using Natural Features

In our follow-up work in collaboration with Rudolf Kajan, Adam Herout and Alena Pavelková we improved on the proposed system above. We created a full augmented reality experience on the mobile phone. We combined our natural feature point based detection with natural image tracking using the Vuforia library[1].

[1] `https://www.vuforia.com/`

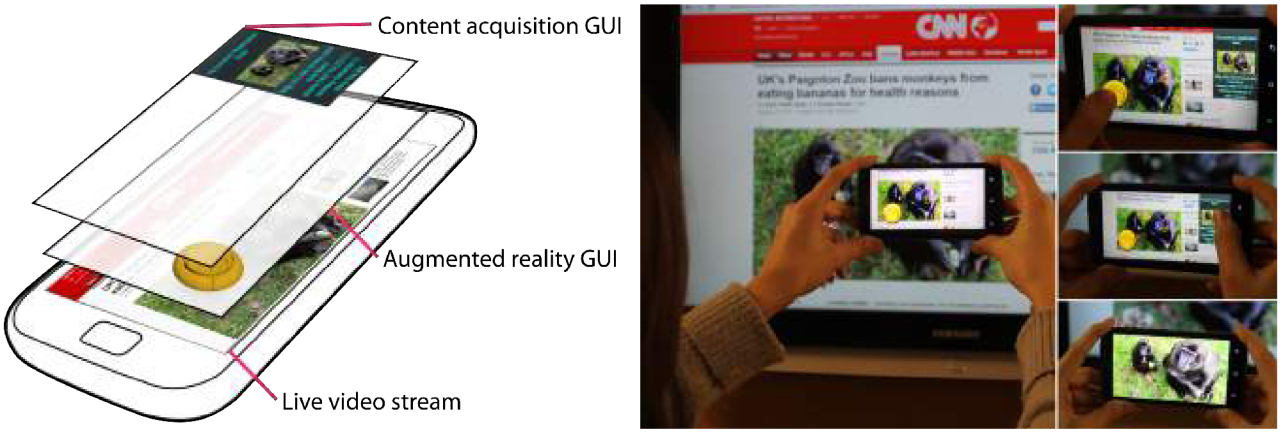

Figure 5.4: **Left:** GUI layers of the content requester application. **Right:** The migration process.

During the initialization phase and in case of fast camera movement, we employed natural features based detection similar to [8]. The difference is, that our solution does not stream the video, as it would generate high network traffic. Instead, we use natural features detection as a fallback method, and send frames only in large intervals (1 second).

A major disadvantage of pure natural features based methods is that they rely on rich features being present on the target display. This assumption is rarely met in the highly manhattanic world of desktop and web applications. As a solution, we utilize a *virtual cursor* using the Vuforia library on the content requester side combined with a small natural image target on the content provider. The on-screen position of the target follows the content requester's view.

The results show that after $4s$ the cursor tracking algorithm was able to restore tracking with $99\,\%$ probability. Our system required on average $2.5\times$ less bandwidth than the theoretical minimum bandwidth used up by a pure natural features-based approach. However, $88.4\,\%$ of the time during interactions (cursor tracking) our system requires just $0.5\,kB/s$ bandwidth, which is approximately $35\times$ less than a natural features based approach.

In order to measure accuracy of content selection with our system, we have used targeting tasks based on ISO 9241-9 standard [24]. When compared to commonly used pointing devices, our system had a lower throughput (TP 0.9-1.1 bps), and relatively low error rate (ER 4-21%) for primary migration targets - images, text paragraphs, links. In [24] the reported values were: joystick TP 1.8 bps ER 9%, touchpad TP 2.9 bps ER 7%, trackball TP 3.0 bps ER 8.6%, mouse TP 4.9 bps ER 9.4%.

Chapter 6

Poor Man's Virtual Camera

This chapter describes another important and distinct usecase of the Uniform Marker Fields which constitute the core of my work. In this case, I, with my colleagues, proposed to use the marker fields in the film-making domain for a structured greenscreen canvas. We created a real-time mobile solution for virtual production for preview purposes and as a fast, simple, and cheap solution for low quality production. Previously (e.g. Cyclops or Milo[1], TechnoDolly[2], Insight VCS[3], [10], [21], etc.) such task required complex and costly setup of infrared cameras, additional tracking extensions for the main cameras and external servers. These provide real-time visualization only for the virtual scene and not the augmented result.

We proposed a method based on camera pose estimation using Uniform Marker Fields as part of the greenscreen. During the shooting, the camera position is established and a preview of the mixed scene is rendered in real time on the device. This solution is unprecedentedly cheap – it is available for a wide range of filmmakers, including amateurs.

This research has been a result of collaboration between Dubská, M., Herout, A., Zachariáš, M. and myself. My main contributions in this research are the following:

- Proposed color mapping for AR use and color selection for the greenscreen marker fields.
- Proposed automatic color calibration for matting by sampling the marker field.
- Mobile prototype implementation of the 3D preview application.
- Real-time performance even on mobile platforms using multi-platform optimization (Halide [31]).
- Evaluation and testing of the prototype system.

[1] http://www.mrmoco.com
[2] http://www.supertechno.com/product/technodolly.html
[3] http://www.naturalpoint.com/optitrack/products/insight-vcs/

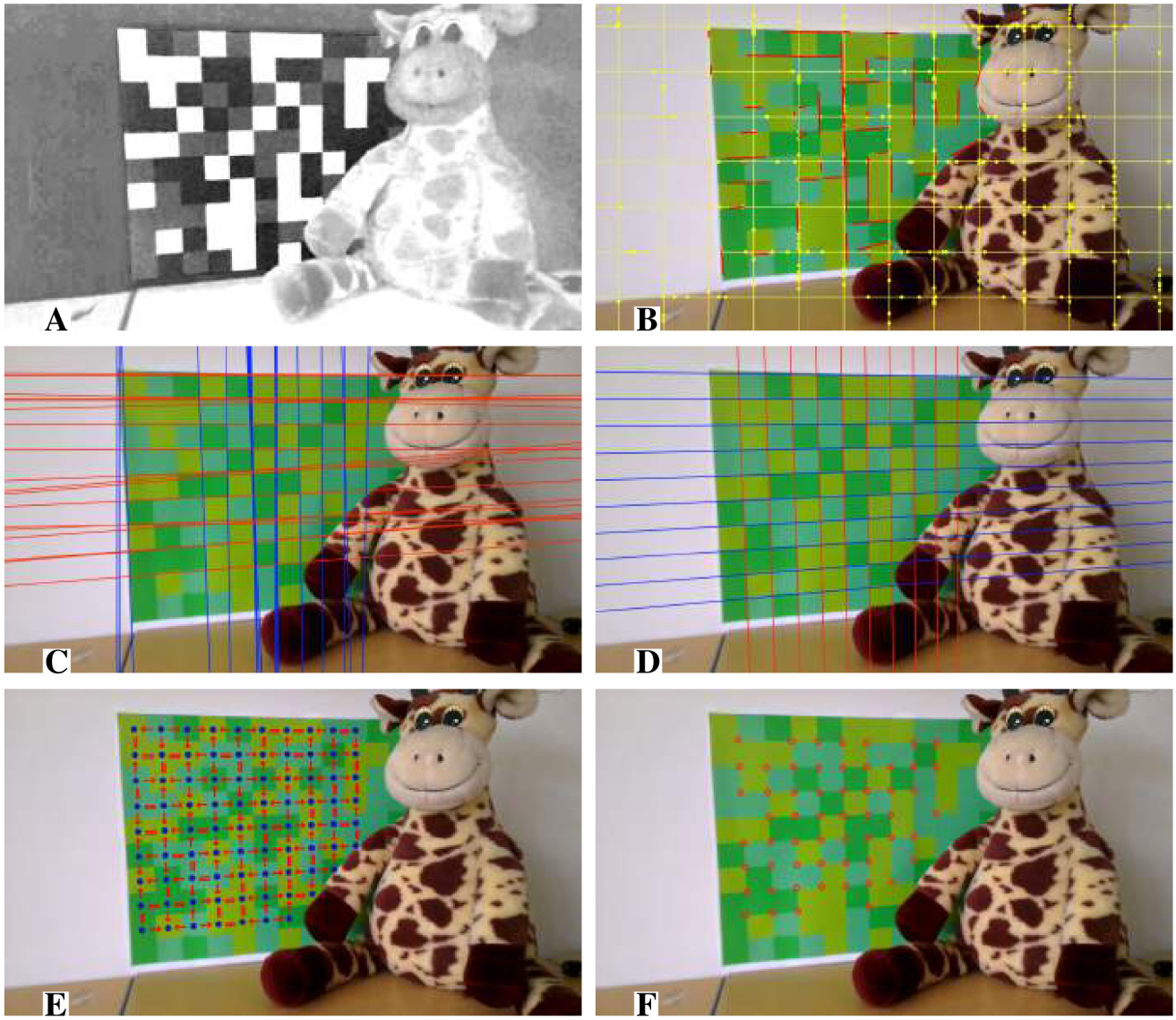

Figure 6.1: Detection of the grid of squares composed of suitable shades of green. **A:** The YC_bC_r image is mapped to grayscale for the detection algorithm. **B:** The grayscale image is processed in very sparse scanlines (for better visualization we use the source image). On each scanline, edges 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 3D camera pose.

6.1 Greenscreen Marker Field

In a chromakeying setting, the proposed algorithm first computes the chromakeying mask to segment out the background containing a fiduciary marker. 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.

Contemporary mobile device cameras provide raw data in this YC_bC_r color space (or a variant of it). Choosing this colorspace to encode gradient direction between modules, initially means no information loss for the matting process due to conversion and saves computational time. Encoding the marker into the C_bC_r channels provides more robustness against intensity changes (shadows) and white balance. For matting, in our experiments we are using the method based on [17].

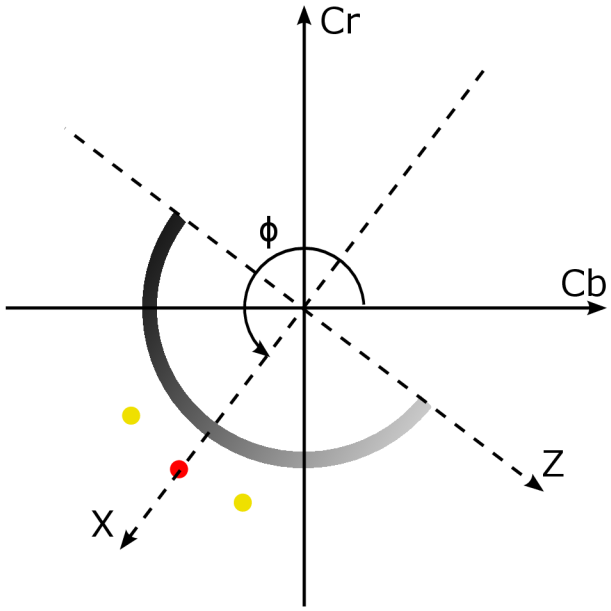

Figure 6.2: The normalized C_bC_r mapping to XZ space. The red dot represents the main keying color. The red and yellow dots are used for the UMF fields. The half arc demonstrates the mapping of the XZ space to grayscale for the detector.

The detection algorithm was adopted from Section 3.4. For the detection algorithm we encoded the edge direction between modules of the UMF into the C_bC_r channels. A good mapping is:

$$I_m(x,y) = \text{atan2}\frac{X(x,y)}{Z(x,y)}, \tag{6.1}$$

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 (Figure 6.2). An alternative simple choice for mapping could be:

$$I_m(x,y) = 2(C_b - C_r) + 1, \tag{6.2}$$

with $C_b, C_r \in (-1, 1)$. The detection algorithm then uses the resulting I_m mapped image as a grayscale image to detect UMF in further processing. The chromakeying mask was also used to filter out foreground edgels during the edgel extraction step (Figure 6.1B).

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 proposed to progressively optimize the exact key colors once the marker was successfully detected in the image.

6.2 Implementation and Results

We targeted live streaming applications using a webcam for PC or an integrated camera on smartphones. We created a plugin for Unity 3D, that integrates with our proof-of-concept implementation. Unity 3D[4] is a free and cross-platform 3D game engine, that supports all our targeted platforms (Figure 6.3).

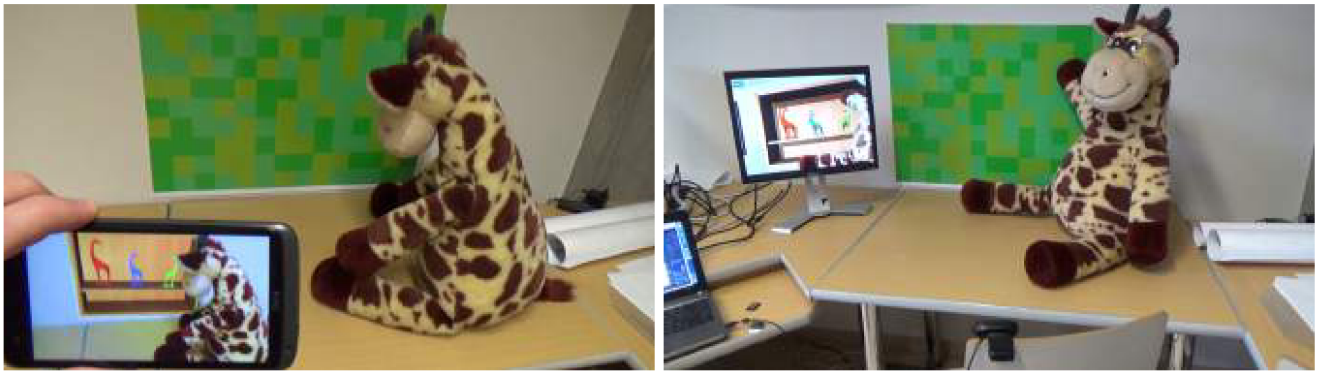

Figure 6.3: Camera pose estimation and chromakeying using greenscreen markers on mobile and PC platforms.

To re-evaluate and compare the detection rates between grayscale and greenscreen markers, we created a small video dataset (16 videos). We used a setup with a projector projecting the marker from the front. We used two different projectors in combination with a smartphone (1280×720, 30Hz) and a hand-held dedicated video camera (1920×1080, 50Hz). With tracking enabled, our algorithm was able to localize the camera in $99.9\,\%$ of frames. Since tracking is a replacable part of the pipeline, Table 6.1 summarizes the percentage of frames for all combinations where the camera was successfully localized without tracking.

[4] http://unity3d.com

Camera/UMF marker	grayscale	greenscreen
smartphone	94.5 %	96.9 %
camera	97.7 %	87.7 %

Table 6.1: Detection rates between grayscale and greenscreen markers projected on canvas.

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 sub-sampled resolution of 320×240. Since contemporary cameras provide sub-sampled color channels (C_b, C_r channels) used in our mapping to grayscale representation, sub-sampling 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).

The measurement results are shown in Table 6.2 (*chroma:* the chromakeying 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. We used the Halide language [31] to create a solution for these tasks with more optimized memory access patterns.

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

Table 6.2: Breakdown of the processing time in milliseconds. for VGA video.

To evaluate the camera pose precision for the preview use-case, we created a second small video dataset. The videos were shot with the smartphone camera (720p, 30Hz) from a 2-5 m distance from the canvas. We used our detector without any optimization and with 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 $5.91\,cm$.

We evaluated the precision of our matting algorithm using GPU shaders with reference to a state-of-the-art alpha-matting approach (KNNMatting [9] with manual an-

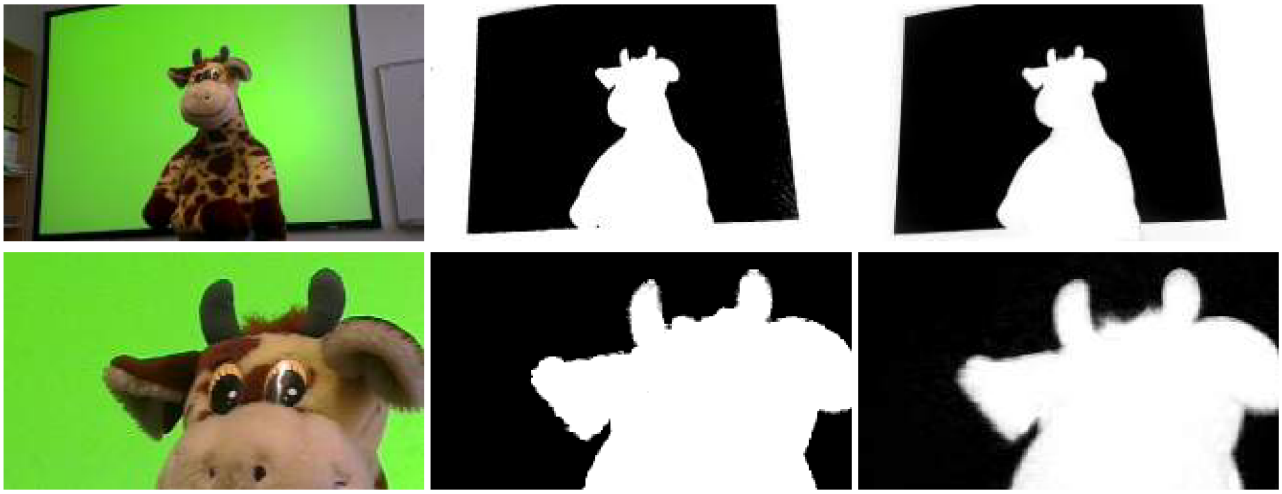

Figure 6.4: **left:** The original image from the camera. **middle:** Matting results from GPU shaders. **right:** Reference alpha mask acquired using KNNmatting with manual annotation.

notation, see Figure 6.4). We quantified the error in the resulting alpha masks as the standard deviation of the difference 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 (standard deviation 4.5).

Chapter 7

Conclusion

This thesis presents in its core an efficient camera pose estimation for real-time augmented reality applications. Most importantly, I introduced the concept of Uniform Marker Fields, which overcome limitations of existing marker designs. The detection algorithm proposed in this thesis was designed to be highly efficient and have a small memory footprint. The algorithm relies on long edgels (connected edge pixels) to estimate two vanishing points and recover the marker fields' grid structure by line parametrization. The interrelation between neighbor modules is used to recover the marker fields rotation and position. This information provides 2D-3D correspondences for module corners, which can be used to compute a full 6 degrees of freedom camera pose. Comprehensive evaluation shows that the described algorithm for Uniform Marker Fields was faster than alternative marker-based approaches with comparable or better performance.

The efficient implementation of the detection algorithm and its various modifications enabled real-time camera pose tracking even on mid-range commodity smartphones. I have demonstrated this on several applications of Uniform Marker Fields. These included the on-screen markers used for document reaccess, task migration, and other user-centric tasks. The last important application was the use of the UMF in film-making domain as a structured and "intelligent" green screen.

My PhD research was centered around the topic of camera pose estimation (mostly based on markers), but it naturally visited other related fields, such as mobile app development, algorithmic optimization, rendering, human-computer interactions, and a few distinct and specific applications. I would also like to give credit to my colleagues for providing knowledge and assistance in the vast research fields associated with this thesis.

Coming into this research, I was focusing on computer graphics and rendering during my earlier studies. I was pleased with the opportunity to further my education and broaden my horizon. Not only was I fortunate enough to dive into several fresh and rapidly developing research fields, I also contributed my own ideas and created original solutions. My work is already cited and used in several publications authored by other researchers.

Bibliography

[1] ALVAR tracking subroutines library web page, 2012. http://www.vtt.fi/multimedia/alvar.html.

[2] Qr-code bar code symbology specification. *ISO/IEC 18004:2015*, 2015.

[3] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):578 – 589, may 2003.

[4] Ronald T Azuma. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[5] E. Bardram. Activity-based computing: support for mobility and collaboration in ubiquitous computing. *Personal Ubiquitous Comput.*, 9(5):312–322, September 2005.

[6] L.F.F. Belussi and N.S.T. Hirata. Fast QR code detection in arbitrarily acquired images. In *Conference on Graphics, Patterns and Images, SIBGRAPI 2011*, pages 281 –288, aug. 2011.

[7] J Burns and C J Mitchell. Coding schemes for two-dimensional position sensing. *Institute of Mathematics and Its Applications Conference Series*, 45:31, 1993.

[8] Tsung-Hsiang Chang and Yang Li. Deep Shot: a framework for migrating tasks across devices using mobile phone cameras. In *Proc. SIGCHI*, 2011.

[9] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(9):2175–2188, Sept 2013.

[10] T. Dobbert. *Matchmoving: The Invisible Art of Camera Tracking*. Wiley Desktop Editions. John Wiley & Sons, 2006.

[11] Markéta Dubská, Adam Herout, and Jiří Havel. Real-time precise detection of regular grids and matrix codes. *Journal of Real-Time Image Processing*, 11(1):193–200, 2013.

[12] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Mark Fiala. Magic mirror system with hand-held and wearable augmentations. *IEEE Virtual Reality (VR)*, 0:251–254, 2007.

[14] Stephen G. Hartke. Binary De Bruijn cycles under different equivalence relations. *Discrete Mathematics*, 215:93 – 102, 2000.

[15] A. Herout, M. Zachariáš, M. Dubská, and J. Havel. Fractal marker fields: No more scale limitations for fiduciary markers. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 285–286, Nov 2012.

[16] Adam Herout, István Szentandrási, Michal Zachariáš, Markéta Dubská, and Rudolf Kajan. Five shades of grey for fast and reliable camera pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1384–1390. IEEE Computer Society, 2013.

[17] Keith Jack. *Video Demystified: A Handbook for the Digital Engineer, 5th Edition*. Newnes, Newton, MA, USA, 5th edition, 2007.

[18] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IWAR'99*, 1999.

[19] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[20] M. Krichenbauer, G. Yamamoto, T. Taketomi, C. Sandor, and H. Kato. Towards augmented reality user interfaces in 3d media production. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 23–28, Sept 2014.

[21] B.-J. Lee, J.-S. Park, and M. Sung. Vision-based real-time camera matchmoving with a known marker. In *Entertainment Computing - ICEC 2006*. 2006.

[22] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(N) solution to the PnP problem. *Int. J. Comput. Vision*, 81(2):155–166, February 2009.

[23] C.-P. Lu, G.D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(6):610 –622, jun 2000.

[24] I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy measures for evaluating computer pointing devices. In *CHI*, 2001.

[25] Stéphane Magnenat, Dat Tien Ngo, Fabio Zund, Mattia Ryffel, Gioacchino Noris, Gerhard Rothlin, Alessia Marra, Maurizio Nitti, Pascal Fua, Markus Gross, et al. Live texturing of augmented reality characters from colored drawings.

Visualization and Computer Graphics, IEEE Transactions on, 21(11):1201–1210, 2015.

[26] R.A. Newcombe, S. Lovegrove, and A.J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. of the Intl. Conf. on Computer Vision (ICCV), Barcelona, Spain*, volume 1, 2011.

[27] Peter Ondruska, Pushmeet Kohli, and Shahram Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *Visualization and Computer Graphics, IEEE Transactions on*, 21(11):1251–1258, 2015.

[28] T. Oskiper, S. Samarasekera, and R. Kumar. Multi-sensor navigation algorithm using monocular camera, imu and gps for large scale augmented reality. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 71 –80, nov. 2012.

[29] Jeffrey S. Pierce and Jeffrey Nichols. An infrastructure for extending applications' user experiences across multiple personal devices. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 101–110, New York, NY, USA, 2008. ACM.

[30] L. Quan and Z. Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774 –780, aug 1999.

[31] Jonathan Ragan-Kelley, Andrew Adams, Sylvain Paris, Marc Levoy, Saman Amarasinghe, and Frédo Durand. Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Trans. Graph.*, 31(4):32:1–32:12, July 2012.

[32] I. S. Reed and R. M. Stewart. Note on the existence of perfect maps. *IRE Transactions on Information Theory*, 8:10–12, 1962.

[33] G. Simon. Tracking-by-synthesis using point features and pyramidal blurring. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 85–92, 2011.

[34] I. Szentandrási, M. Zachariáš, J. Havel, A. Herout, M. Dubská, and R. Kajan. Uniform Marker Fields: Camera localization by orientable De Bruijn tori. In *11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

[35] Keisuke Tateno, Itaru Kitahara, and Yuichi Ohta. A nested marker for augmented reality. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[36] H. Uchiyama and E. Marchand. Deformable random dot markers. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 237–238, Washington, DC, USA, 2011. IEEE Computer Society.

[37] H. Uchiyama and H. Saito. Random dot markers. In *IEEE Virtual Reality Conf. (VR)*, 2011.

[38] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[39] M. Zachariáš, I. Szentandrási, and A. Herout. Visual correction of position drift using uniform marker fields. In *Spring conference on Computer Graphics*, Bratislava, SK, 2016. UNIBA.

[40] Zhiwei Zhu, V. Branzoi, M. Wolverton, G. Murray, N. Vitovitch, L. Yarnall, G. Acharya, S. Samarasekera, and R. Kumar. Ar-mentor: Augmented reality based mentoring system. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 17–22, Sept 2014.