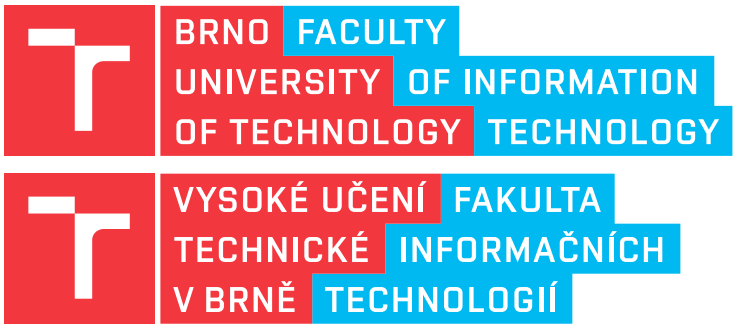


DESIGN AND APPLICATIONS OF SPECIAL-PURPOSE TWO-DIMENSIONAL VISUAL MARKERS

NÁVRH A APLIKACE DVOUROZMĚRNÝCH VIZUÁLNÍCH MARKERŮ PRO SPECIÁLNÍ ÚČELY

ING. MICHAL ZACHARIÁŠ

PH.D. THESIS
DISERTAČNÍ PRÁCE

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
FACULTY OF INFORMATION TECHNOLOGY
BRNO UNIVERSITY OF TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO 2016

I wish to thank my supervisor Adam Herout for his great coaching and mentoring during my whole Ph.D. studies. Great deal of my thanks goes also to my coworkers Rudolf Kajan, István Szentandrási, Markéta Dubská, Marek Šolony, Lukáš Polok and many others for awesome cooperation and for keeping my working spirit up. My parents for supporting me and my wife Marcela for her patience and her determination to make me finish my thesis.

THANK YOU!

DECLARATION

I hereby declare that this Ph.D. thesis is my original work and that I have written it under the guidance of prof. Ing. Adam Herout, Ph.D. All sources and literature that I have used during my work on the thesis are correctly cited with complete reference to the respective sources.

Michal Zachariáš
Brno, May 29, 2016

ABSTRACT

Contemporary visual fiduciary marker systems have a major disadvantage compared to markerless approaches – the camera movement is tightly limited to the space where the markers are. At any frame of the camera image a marker must be large enough to provide sufficient amount of information for detection and identification and at the same time, it must be small enough to fit into the camera's field of view. These requirements are contradictory.

This work presents a solution to this problem in a concept of the *Marker Field*. It is a structure whose presence can be detected in a camera image and the exact location within the field can be recognized based on just any sub-area of defined size. The sub-areas are not disjoint, but they are overlapping to a very large degree, to be identifiable from both close-up and distant views. Different implementations of the marker field concept are explained in this work, together with their intended uses and their advantages and disadvantages.

To prove and support the usability of proposed marker fields, this work's second largest part discusses their several real-life applications.

KEYWORDS

Uniform Marker Fields, Fractal Marker Fields, Honeycomb Marker Fields, Shades of Grey Marker Fields, augmented reality, real-time camera pose estimation, perfect maps, AR gaming, documents reaccess, visually appealing fiduciary markers, fiduciary markers, chromakeying, green screen, position drift correction, tabletop scene interaction

ABSTRAKT

Současné vizuální markerové systémy mají jednu zásadní nevýhodu oproti tzv. markerless přístupům -– pohyb kamery je omezen na oblast pokrytou markery. V každém snímku musí být marker dostatečně velký, aby jej bylo možné identifikovat a vypočítat pozici a rotaci kamery. Zároveň musí být dostatečně malý, aby se celý (nebo alespoň jeho podstatná část) vešel do záběru kamery. Avšak tyto požadavky jsou protichůdné.

Tato práce nabízí řešení tohoto problému za pomoci konceptu *Marker Fields*. Jde o strukturu, jejíž přítomnost je možné v obraze kamery snadno detekovat a identifikovat část, na kterou se kamera právě dívá, a to na základě jakékoli (malé) podoblasti s definovanou velikostí. Aby bylo možné podoblasti identifikovat zblízka i zdálky, nejsou od sebe odděleny, ale do velké míry se překrývají. V této práci jsou vysvětleny různé implementace konceptu marker fields, spolu s jejich zamýšleným použitím a výhodami a nevýhodami.

Jako důkaz použitelnosti marker fields v reálném světě, se druhá největší část této práce věnuje popisu jejich reálných aplikací.

KLÍČOVÁ SLOVA

Uniform Marker Fields, Fractal Marker Fields, Honeycomb Marker Fields, Shades of Grey Marker Fields, rozšířená realita, určení pozice kamery v reálném čase, perfektní mapy, hry pro rozšířenou realitu, opětovný přístup k dokumentům, vizuálně přitažlivé markery, klíčování, zelené pozadí, oprava poziční chyby, interakce s virtuálními předměty na stole

LIST OF PUBLICATIONS

[I] A. Herout, M. Zachariáš, M. Dubská, and J. Havel. "Fractal Marker Fields: No More Scale Limitations for Fiduciary Markers." In: *Proceedings of the 2012 11th IEEE International Symposium on Mixed and Augmented Reality*. Atlatnta, Georgia, US: Institute of Electrical and Electronics Engineers, 2012. ISBN: 978-1-4673-4660-3.

[II] I. Szentandrási, M. Zachariáš, J. Havel, A. Herout, M. Dubská, and R. Kajan. "Uniform Marker Fields: Camera Localization By Orientable De Bruijn Tori." In: *Proceedings of the 2012 11th IEEE International Symposium on Mixed and Augmented Reality*. Atlatnta, Georgia, US: Institute of Electrical and Electronics Engineers, 2012. ISBN: 978-1-4673-4661-0.

[III] R. Kajan, M. Zachariáš, and A. Herout. "Selective Availability for Security Monitoring." In: *Proceedings of the Software Technologies and Processes 2012*. Furtwagen, DE, 2012.

[IV] R. Kajan, I. Szentandrási, A. Herout, and M. Zachariáš. "On-Screen Marker Fields for Reliable Screen-To-Screen Task Migration." In: *Proceedings of the 2013 International Conference on Human Factors in Computing & Informatics*. Maribor, SI: Springer Verlag, 2013. ISBN: 978-3-642-39061-6.

[V] Z. Horváth, A. Herout, I. Szentandrási, and M. Zachariáš. "Design and Detection of Local Geometric Features for Deformable Marker Fields." In: *Proceedings of 29th Spring conference on Computer Graphics*. Bratislava, SK: Comenius University in Bratislava, 2013. ISBN: 978-80-223-3377-1.

[VI] A. Herout, I. Szentandrási, M. Zachariáš, M. Dubská, and R. Kajan. "Five Shades of Grey for Fast and Reliable Camera Pose Estimation." In: *Proceedings of CVPR*. Portland, OR, US: IEEE Computer Society, 2013. ISBN: 978-0-7695-4989-7.

[VII] M. Dubská, I. Szentandrási, M. Zachariáš, and A. Herout. "Poor Man's SimulCam: Real-Time And Effortless MatchMoving." In: *Proceedings of the 2013 12th IEEE International Symposium on Mixed and Augmented Reality*. Adelaide, S.A., AU: Institute of Electrical and Electronics Engineers, 2013. ISBN: 978-1-4673-4661-0.

[VIII] I. Szentandrási, M. Zachariáš, R. Kajan, J. Tinka, M. Dubská, J. Sochor, and A. Herout. “INCAST: Interactive Camera Streams for Surveillance Cams AR.” In: *Proceedings of the 2015 14th IEEE International Symposium on Mixed and Augmented Reality*. Fukuoka, JP: Institute of Electrical and Electronics Engineers, 2015. ISBN: 978-1-4799-6184-9.

[IX] M. Zachariáš and A. Herout. “Color Theme-Based Digital Art for Augmented Environment.” In: *Proceedings of the 2016 1st International Conference on Digital Arts, Media and Technology*. Chiang Rai, Thailand, 2016.

[X] M. Zachariáš, I. Szentandrási, and A. Herout. “Visual Correction of Position Drift using Uniform Marker Fields.” In: *Proceedings of 32nd Spring conference on Computer Graphics*. Bratislava, SK: Comenius University in Bratislava, 2016.

[XI] I. Szentandrási, M. Dubská, M. Zachariáš, and A. Herout. “Poor Man’s Virtual Camera: Real-Time Simultaneous Matting and Camera Pose Estimation.” In: *IEEE Computer Graphics and Applications Magazine* (2016). ISSN: 0272-1716.

CONTENTS

ACRONYMS

AR	Augmented Reality
HMD	Head Mounted Display
CRC	Cyclical Redundancy Check
FEC	Forward Error Correction
IR	Infrared
DOF	Degrees of Freedom
LLAH	Locally Likely Arrangement Hashing
VMP	Variable Moiré Pattern
FMF	Fractal Marker Field
UMF	Uniform Marker Field
SoG	Shades of Grey
HMF	Honeycomb Marker Field
RDM	Random Dot Marker
DRDM	Deformable Random Dot Marker
IMU	Inertial Measurement Unit
SLAM	Simultaneous Localization And Mapping
SPRT	Sequential Probability Ratio Test
UAV	Unmanned Aerial Vehicle
MAV	Micro Aerial Vehicle
UI	User Interface
UX	User Experience

Part I

INTRODUCTION, MOTIVATION & OVERVIEW OF EXISTING MARKERS

1

CO-AUTHORS AND COLLEAGUES

The general idea and the motivation for extending existing fiduciary markers used for Augmented Reality (AR) came from me and my mentor Adam Herout i.e. using fractal structure of markers in Fractal Marker Field (FMF) and using De Bruijn sequence in Uniform Marker Field (UMF) and its variations to overcome limitations of contemporary markers. Many practical applications shown in this work, e.g. table-top interaction, visual correction of position drift, exploring possibilities of using unobtrusive color theme-based UMFs in real-life environments and more, are my own work.

Some parts of these systems were developed in collaboration with my colleagues. It wouldn't be possible to practically test any of my own work without István Szentandrási's UMF detection and localization algorithm and Markéta Dubská's FMF detector. Other colleagues also contributed differently into our research and I feel obliged to mention them here – namely Rudolf Kajan and Jiří Havel. To make my thesis self-contained, their work is mentioned and explained, but credit goes to them and more elaborated explanations are in their Ph.D. theses.

2

INTRODUCTION & MOTIVATION

2.1 INTRODUCTION

Augmented Reality (AR) is becoming more and more popular and appealing for customers. Mainly because the expansion of the relatively cheap ubiquitous mobile devices with hardware powerful enough for doing all necessary computations needed for the pose estimation in the scene and augmenting it with digital content. Another reason would be that near-eye see-through glasses are recently becoming generally available – Microsoft's HoloLens[1], DAQRI's Smart Helmet[2] – which brings the promise of pushing the boundaries of what can be done in field of AR and related research [1].

Many AR, robot positioning, or motion capture systems (and other) rely on the use of fiduciary markers to estimate the camera's position and orientation in their 3D environment.

Markerless systems are becoming more and more popular – using PTAM [2], keypoint template matching [3], homography-base [4], camera tracking and mapping in multiple regions [5], visual Simultaneous Localization And Mapping (SLAM) [6] and other methods [7, 8]. Nevertheless, markers still offer advantages such as high precision, ease of detection, low computational demands and reliability [9].

The well-known representatives of marker approach are for example ARToolKit and ARTag. These popular designs (and designs based on them or inspired by them) of fiduciary markers consist of two components: geometrical features which help to localize the marker in the processed image and features defining the identity of the marker. 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 (see Chapter 3 for more details).

1 http://www.microsoft.com/microsoft-hololens
2 http://www.daqri.com/home/product/daqri-smart-helmet/

2.2 MOTIVATION

One big disadvantage of the AR systems based on markers against the markerless approaches, is that the camera motion is tightly limited. Systems require that any frame of the camera image must contain at least one marker (or its significant part) and the marker must be large enough in order to compute precise camera location and decode its identification. These requirements are contradictory — the markers must be large enough to provide sufficient amount of information and at the same time they must be small enough to fit into the camera's field of view (Figure 1).

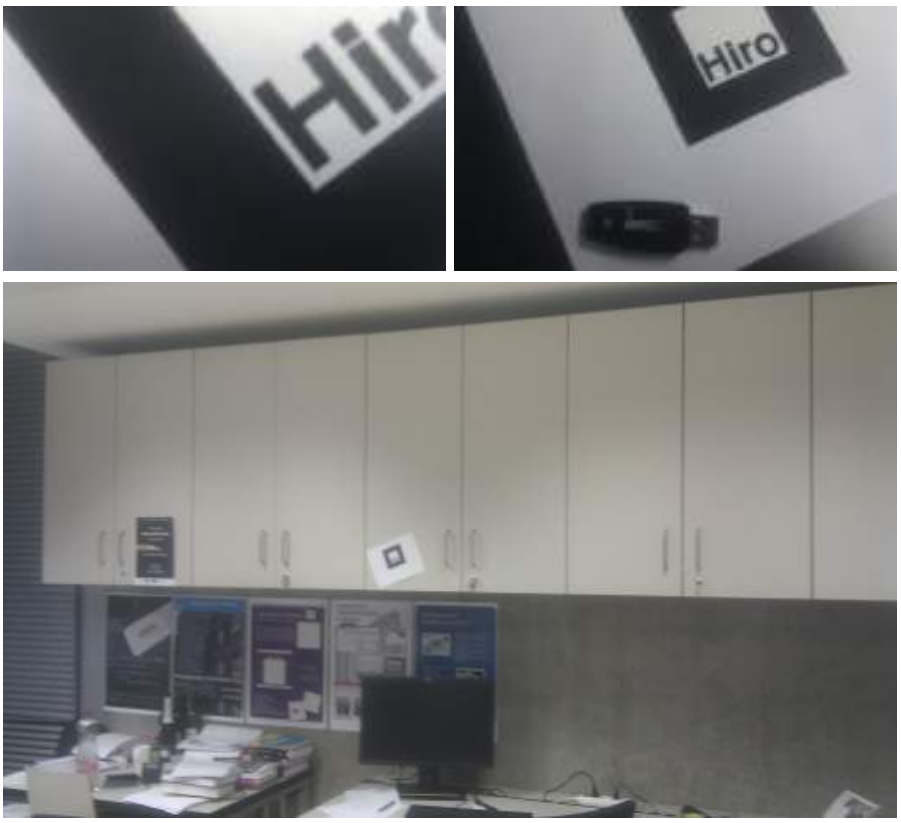


Figure 1: Examples of camera frames where markers are hard to detect.

The challenge is that the marker must cover a large area and, at the same time, it must be reliably detected even from a small visible portion of the marker. Also, the detection must be invariant to high degree of perspective distortion and to varying lighting conditions (direct light, shadows, different lighting intensities. . .). What marker design and corresponding detection algorithm can meet these requirements and, at the same time, be aesthetically appealing?

2.3 CONTRIBUTION

To overcome this major problem we present the *Marker Field* – a structure whose presence can be detected in a camera image and the exact location within the field can be recognized based on any sub-area of a defined minimal size. Such sub-areas can be viewed as individual markers in the traditional sense. However, the markers are not disjoint, but they are overlapping to a large degree, in order to allow for identification from close-up views as well as from larger distances.

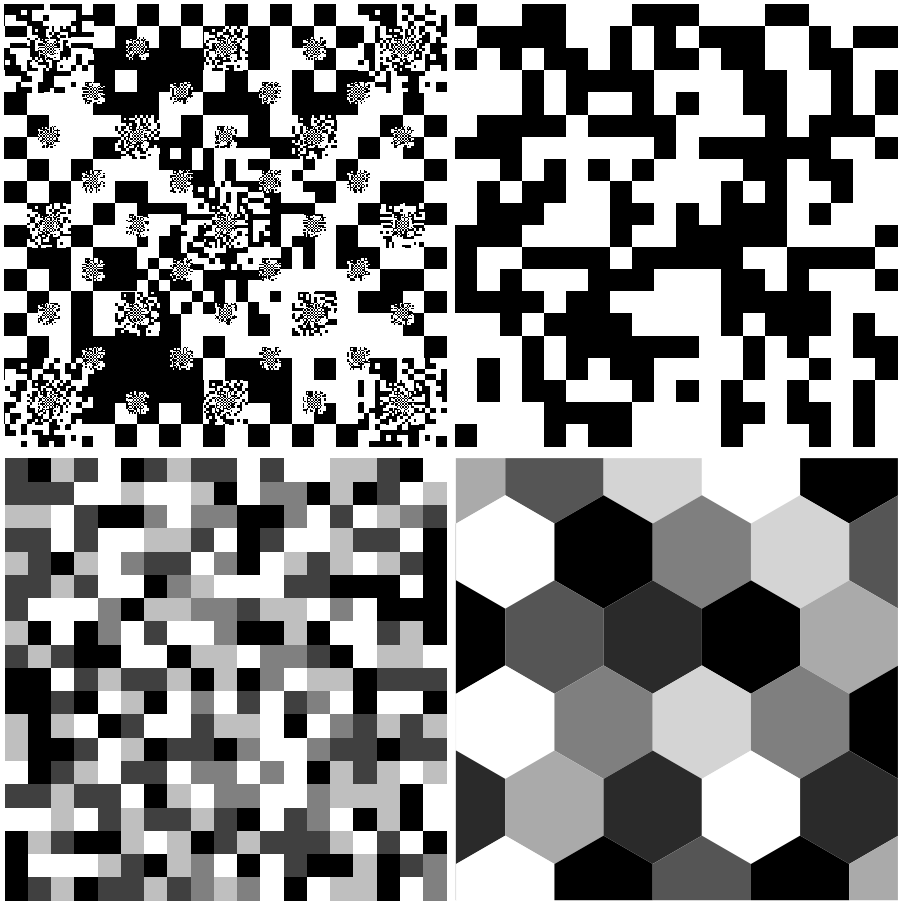


Figure 2: Examples of our marker field implementations.

The contribution of this work to the field of the AR research is the design of special-purpose two-dimensional visual fiduciary markers that overcome all or the part of the problems mentioned in the motivation (Section 2.2). This work explains in detail how are individual marker field implementations (FMF, UMF, improved UMF and HMF) designed, what are their properties and their pros and cons.

The second contribution is published and implemented specific applications of the proposed marker fields and ideas for the future work.

We are also making some of the generated markers publicly available on our website http://www.fit.vutbr.cz/research/groups/graph/MF/.

2.4 THESIS ORGANIZATION

We started upon the motivation and came up with different implementations of marker field principle. Entire Part II is devoted to design of the marker fields with different properties.

The first design is Fractal Marker Field (FMF), its fractal structure allows for almost infinite levels of nesting the markers one into another (Chapter 4).

Uniform Marker Fields (UMFs) are black and white checkerboards where individual markers are defined as n^2-windows. Each such window is unique in entire marker field, even overlapping ones and even all four possible rotations (Chapter 5).

Obvious step to improved Uniform Marker Fields (UMFs) (aka Shades of Grey (SoG)) was increasing the arity of the marker field. That means using more shades of grey (or color) than just black and white. This brought up new problems with generation and detection of such marker fields (Chapter 6).

Last implementation is Honeycomb Marker Fields (HMFs). Principle is the same as in UMF but with two exceptions – modules are not square anymore, but hexagons and detector is not looking for straight lines but "Y-junctions" that neighboring hexagons form. These exceptions allow such marker field to be detected even if it is on a deformed surface (Chapter 7).

Part III shows my work on applications for proposed marker fields. First three applications are published in conferences' proceedings, green screen application is published in the Computer Graphics and Applications Magazine and gaming carpet application is implemented but not yet published.

Using carefully selected colors, designer can create aesthetically appealing UMFs so they do not negatively interfere with the overall feel of for example interior of a building (Chapter 8).

Strategically placed aesthetically appealing UMFs can be used for indoor navigation of robots and humans. Such marker fields would serve as the anchor points for cheap visual correction of position drift (Chapter 9).

Screen-to-screen task migration is still hot research topic, using unobtrusive transparent UMF on the PC screen can simplify the context migration process (Chapter 10).

Using shades of green instead shades of grey in improved Uniform Marker Field (UMF), we can turn the marker field into the green screen for film industry (Chapter 11).

Part IV contains chapter about future work and ideas that are still in the proof of concept phase, e.g. tabletop scene interaction and digital changing room (Chapter 13). This entire work is summarized in conclusion (Chapter 14).

The next chapter briefly introduces the best known and common representatives of fiduciary markers. Historical pioneers as well as modern and state-of-the-art markers are mentioned (Chapter 3).

3

OVERVIEW OF EXISTING MARKERS

There are many possible ways how to design fiduciary marker so it can be efficiently localized, identified and then used to calculate the camera position and rotation in the scene.

First visual-based Augmented Reality (AR) identification and camera pose estimation system was proposed in 1998 by Rekimoto in his paper "Matrix: a realtime object identification and registration method for augmented reality" ([10], Section 3.1). Shortly after him, Kato and Billinghurst in year 1999 presented their pioneering AR library ARToolKit (recently acquired by DAQRI[1], an American augmented reality company) in paper "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System" ([11], Section 3.2). These two systems are typical representatives of classical approach to AR. Main property of classical system is that markers consist of two components; geometrical features which help to localize the marker in the processed image, often black-and-white shape (e.g. square, circle, thick line, . . .) and second component used for distinguishing between individual markers, i.e. identification (often inner content of the shape). Important representatives of this approach are CyberCode ([12], Section 3.3), ARTag ([13], Section 3.4) and two already mentioned above Matrix and ARToolKit.

Classical visual-based approach to AR.

Using thick black-and-white borders around markers just for localization can be disturbing for human eye and not very aesthetically appealing. Therefore markers combining localization and identification features were created. This approach is used for example by reacTIVision ([14, 15], Section 3.5).

Approaches that combine identification and localization features into one.

Because of the nature of detection techniques used for localization in these approaches, there must be silent zone around, disabling the possibility of creating a dense array of markers. Such arrays are important when a large area needs to be covered. Marker designs such as Random Dot Marker (RDM) ([16], Section 3.6), CALTag ([17], Section 3.7) and Nested Markers ([18, 19], Section 3.8) are example of this approach.

Large area markers and marker arrays.

Majority of the marker designs assume that the marker is on a planar surface. There are few exceptions that deal with deformability of the marker – extension of RDM called predictably Deformable Random Dot Marker (DRDM) ([20], Section 3.9) and system for texture mapping onto a deformable object

Deformable markers.

1 http://www.daqri.com/

by Fujimoto et al. [21] (their De Bruijn sequence marker design is based on our Uniform Marker Field (UMF), Section 3.10).

High-accuracy, special-purpose, optical markers.

Recently Tanaka et al. in their papers [22], [23] and [24] presented small high-accuracy visual markers based initially on lenticular lenses (Section 3.11) and eventually on microlens array (Section 3.12). Their targeted use is mainly object handling by robots where high precision is required even in such singular position where camera's line-of-sight is perpendicular to the marker.

Other markers.

Other representatives are virtual/augmented reality software library Alvar [25], highly occlusion tolerant RUNE-tag [26], content augmented reality marker CARMa [27], imperceptible markers called Bokode [28, 29] and unobtrusive active markers VRCodes that exploit the camera rolling shutter [30].

3.1 FIRST VISION-BASED AR SYSTEM

In year 1998 Rekimoto [10] presented first AR system for simultaneously identifying and estimating of coordinate system of real world objects and called it Matrix. This system became predecessor to all later AR systems like ARToolKit, CyberCode, ARTag, and many more. System prior to Matrix measured the position and orientation of a real world object with 3D sensors like magnetometer or ultrasonic transceiver [31]. Matrix is one of the first to use vision-based method and first using the fiduciary marker printed on the paper; it started the era of classical approach to augmented reality. Figure 4 shows an original image from the paper; almost every classical AR system works in the same way.

First video-based AR system.

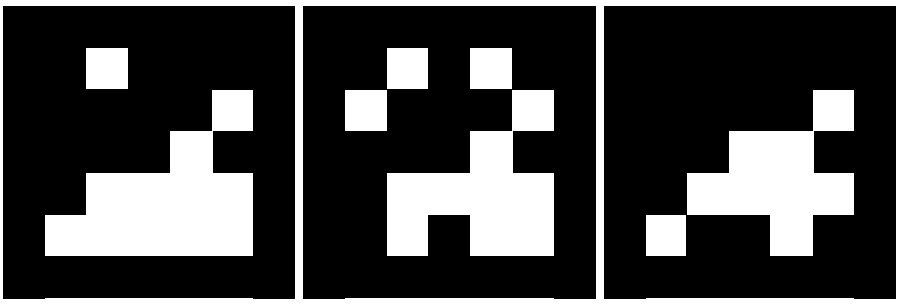


Figure 3: 2D Matrix marker design.

His proposed marker is black thick square where its interior is combination of black and white modules (bits). He uses 5×5 grid in the paper but any other size is applicable. Identification data enriched by Cyclical Redundancy Check (CRC) capabilities are stored in those bits (Figure 3).

ID of marker stored in black and white modules

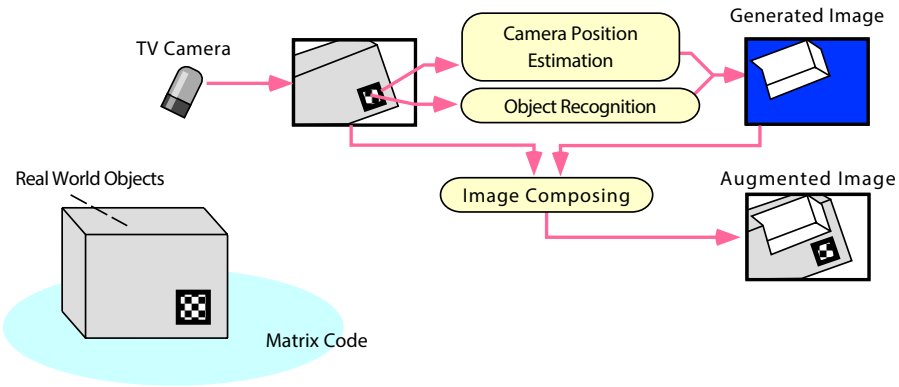
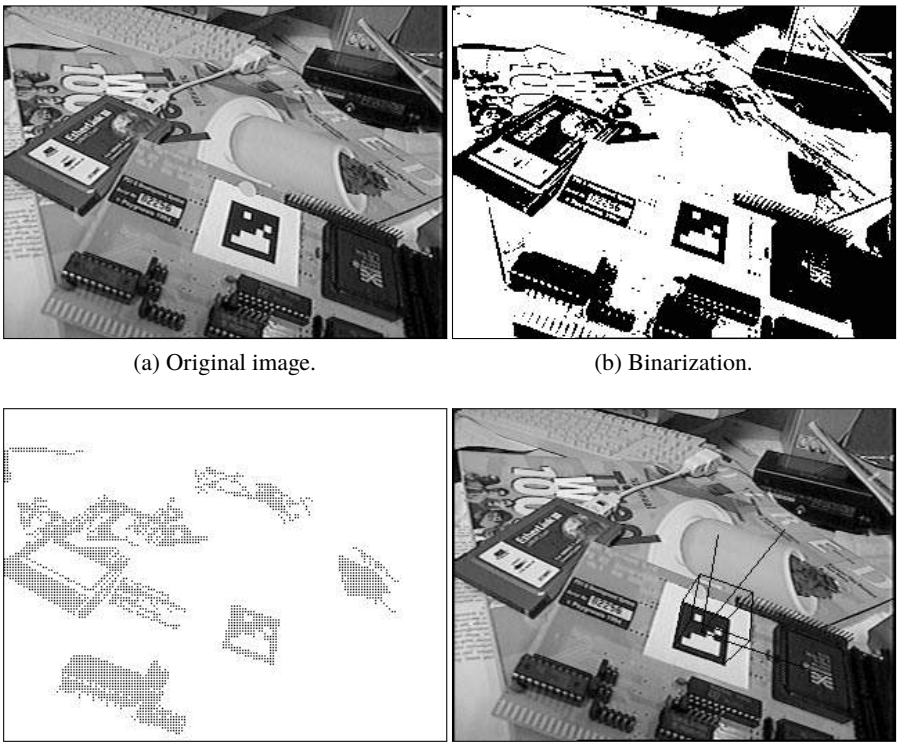


Figure 4: Overview of the first vision-based augmented reality system.

Image is binarized using adaptive thresholding, connected regions are found and rectangle bounding boxes are searched. Each such candidate is a potential area with the code. Each area is transformed to the camera space and using CRC check is determined if the marker is valid. After finding correct markers



(a) Original image. (b) Binarization.

(c) Connected components analysis. (d) Camera pose estimation

Figure 5: The Matrix code detection process.

their corners are localized and camera pose estimation is done. All these steps are present in almost any succeeding AR system (Figure 5).

3.2 ARTOOLKIT

ARToolKit is very well known and very widespread AR system. The first paper about ARToolKit is called "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System" and was published in 1999 by Kato and Billinghurst [11]. It's original intended use was an AR conferencing system; remote collaborators were represented by virtual monitors which could be freely positioned in space by moving the markers around the table desk. The system was intended to be used with Head Mounted Display (HMD) (Figure 7).

Intended use was AR conferencing system.

Localization part of the marker is thick black square on white background with white silent area around it (Figure 6). After thresholding of the input image, regions whose outline contour can be fitted by four line segments are extracted. Parameters of these four segments and coordinates of the four corners of the regions found from the intersections of the line segments are stored for later camera pose estimation.

Figure 6: ARToolKit fiduciary marker designs.

To identify the individual marker, the regions are normalized and the sub-image within the region is compared by template matching with patterns that were given the system before. Kato and Billinghurst's team used user names, letters or photos as identifiable patterns although different shapes can be used, see Figure 6 for examples.

ARToolKit uses images as marker's ID.

Figure 7: Demonstration of first intended use of ARToolKit (AR collaborating system). **Left:** Virtual monitors with the camera feeds from remote collaborators. **Right:** Person with HMD broadcasting his AR desktop.

Just as an interesting fact for the reader, ARToolKit was bought by the American augmented reality company DAQRI in 2015 and ARToolKit Pro version SDKs were released under LGP license to re-ignite innovation in the AR community. DAQRI is primarily known for their Smart Helmet system (Figure 8).

Figure 8: DAQRI Smart Helmet prototype.

3.3 CYBERCODE

Rekimoto and Ayatsuka [12] created CyberCode, a marker which consists of a very simple rectangular localization pattern (guide bar seen highlighted in the middle in Figure 10) and black squares on white background, carrying the binary marker's ID (Figure 9).

Figure 9: CyberCode fiduciary marker design.

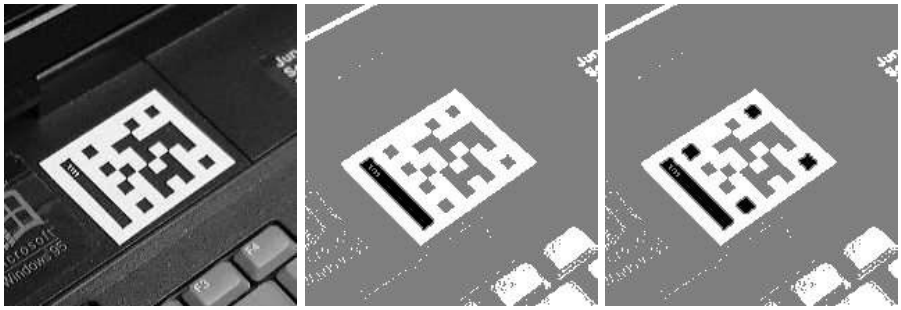


Figure 10: CyberCode localization steps. **Left:** Original greyscale image. **Middle:** Localized guide bar. **Right:** Localized four corners of the marker region followed by reading the marker's identification.

To localize the marker the system looks for candidate guide bars first. Using their position and orientation it searches for the four corners of the marker region. When all four corners are found, their position is used to estimate and compensate for the projection distortion of camera. The image inside the marker region is decoded and after checking for the error bits, the system determines whether or not the image contains a correct CyberCode (Figure 10).

3.4 ARTAG

Fiala [13] proposed a combination of two previous approaches – a distinct black square on white background (localization feature from ARToolKit), filled with square black and white modules, encoding digital marker ID with robust error correction capabilities (identification feature from CyberCode), see examples in Figure 11. His approach is very similar to Matrix but it is explored more thoroughly and of course there is a more than 10 years between their papers.

ARTag performs better under less controlled lighting conditions and some degree of marker occlusion than ARToolKit due to the adaptive thresholding method used to extract the quad shaped segments from input image. Thanks to this, multiple markers can be detected even if each one is under different light.

Each marker holds $6 \times 6 = 36$ bits of information which encapsulates a 10-bit ID and the remaining 26 bits provide redundancy to reduce the chances of false detection and identification, and to provide the uniqueness over the four possible rotations. The Cyclical Redundancy Check (CRC) and Forward Error Correction (FEC) methods are used.

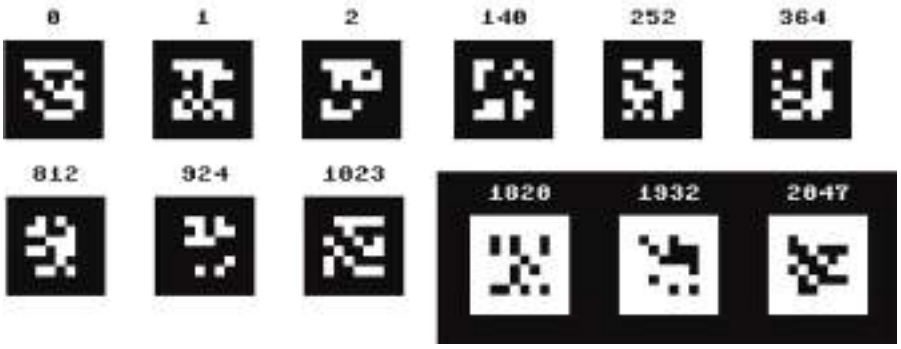


Figure 11: ARTag marker designs. Last three markers show that color can be inverted.

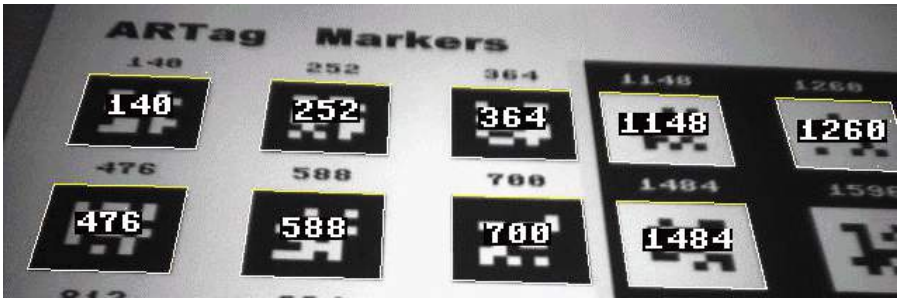


Figure 12: ARTag markers detected in an image. Overlaid with white border and ID shown above.

3.5 REACTABLE & REACTIVISION

This type of markers was specifically created for the reacTable² (Figure 15), which is a tangible user interface where physical objects represent the components of a software sound synthesizer. Markers are attached to the underside of the objects placed on a translucent table. A camera beneath the table captures images which are processed to determine the location, orientation and identity of the markers. This system is described in the papers "The reacTable*: A Collaborative Musical Instrument" [32] and "reacTIVision: a computer-vision framework for table-based tangible interaction" [14].

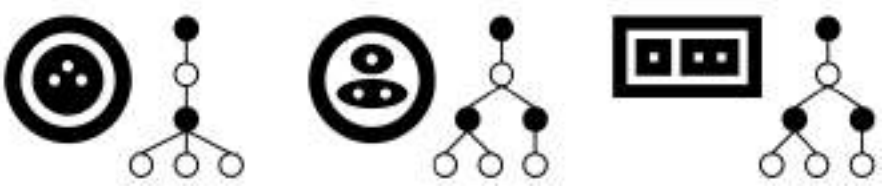


Figure 13: Some simple topologies and their corresponding region adjacency graphs.

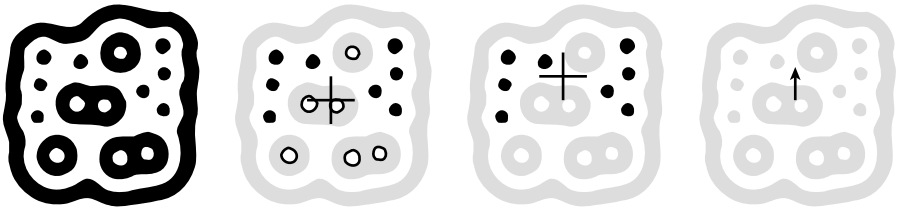


Figure 14: Calculation of location and orientation of the markers. **First:** Original marker. **Second:** Average of centers' positions of black and white nodes. **Third:** Average of centers' positions of black nodes. **Fourth:** Vector between the second and third.

reacTIVision works with region adjacency only – for identification and also localization.

The reacTIVision (also called "amoeba" for their resemblance to this animal) fiduciary markers use only the region adjacency graph and the bounding rectangles of each region to determine all necessary information about each marker, i.e. location and orientation. They were explained in "The Design and Evolution of Fiducials for the reacTIVision System" [15] and their tracking improved in "Improved Topological Fiducial Tracking in the reacTIVision System" [33].

ReacTIVision employs the topological fiducial recognition approach. A region adjacency graph is derived from a binary image of the scene through the process of segmentation. The graph can be understood as a tree representing the containership (or parentship) hierarchy of the image, that is, which black regions are contained inside which white regions and vice versa (Figure 13). Each fiducial in a set has a unique topology which can be efficiently matched against a dictionary of subtrees represented as strings.

2 http://www.reactable.com/

Figure 15: **Left:** Photo of entire reacTable. **Right:** Photos of reacTable with different "stones" embedded with markers on the bottom representing different sounds and instruments.

Genetic algorithm is employed to render the marker.

Each tree can be drawn in a huge number of ways making an exhaustive search for "optimally" rendered marker impractical. That is why a genetic algorithm is employed to optimize parameters such as marker area, aspect ratio, symmetry and centroid locations for black and white leaves.

Calculation of the marker's center and orientation.

To approximately compute the center point of the fiducial they take an average of all leaves centers (Figure 14 second). The vector from this centroid to a point given by the average of all black (or white) leaves centers (Figure 14 third) is used to compute the orientation of the fiducial (Figure 14 fourth).

3.6 RANDOM DOT MARKERS

Random Dot Markers (RDMs) were first proposed in the paper "Random Dot Markers" [16] by Uchiyama and Saito as randomly scattered dots as illustrated in Figure 16. Because the dots are directly utilized for marker identifying and tracking, this marker does not need to have any other features for localization. Dots in the markers are simply randomly generated. Overlapping dots are not desirable and degrade the robustness of the detection because they cannot be separately detected in the camera image. Such dots are therefore rejected. The number of dots in each marker does not need to be same.

Positions of dots are really just random.

Selecting proper size of the dots is an issue. The size influences minimum and maximum detection range between camera and marker. Also, the selection of the dots count per marker correlates with the number of markers that are unique enough. Tweaking these parameters controls the trade-off between visibility of the marker and robustness of the detection.

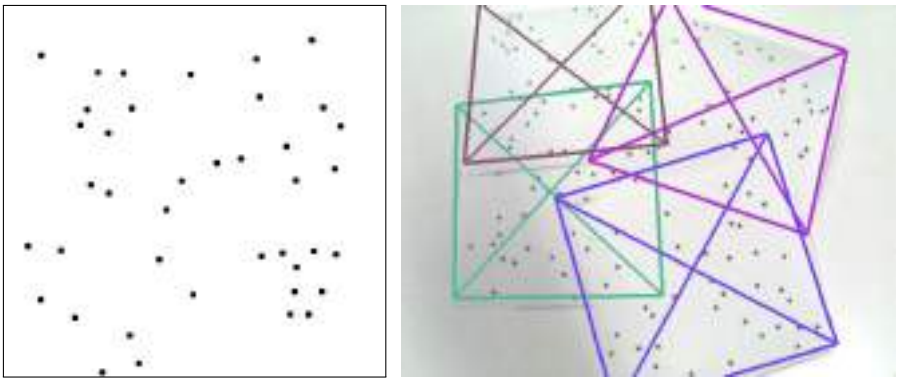


Figure 16: Example of random dot marker. **Left:** Single RDM. **Right:** Several detected and tracked RDMs showing partial robustness to occlusion.

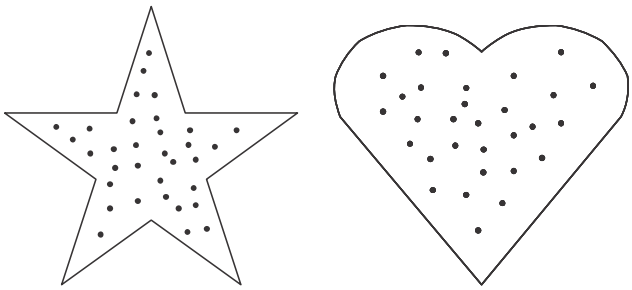


Figure 17: RDM various types of marker design.

After retrieving candidate dots (binary thresholding the camera image) and their centers from camera image, Locally Likely Arrangement Hashing (LLAH) [34] is used to compute hash of candidate marker; LLAH works with local geometrical relationship of the keypoints. This hash is afterwards searched in

database of all registered markers. If the marker is valid, all its dots are used to compute the camera pose.

According to the authors, the advantages of RDMs are *flexible marker design* (shape of the marker does not have any limitations, see Figure 17), *robustness against occlusion* (compared to the ARTag), where only one corner can be occluded) and *user interaction* which is related to the occlusion tolerance (user can "touch" the marker, occlude it, and algorithm is still able to detect the marker).

RDMs are still a live research topic.

The RDMs are still a live topic and the researchers still use it as the base for their own work "Scalable maps of random dots for middle-scale locative mobile games" [35] (Figure 18) or they work on the concept itself and improve it like Yang, Normand, and Moreau in their paper "Robust Random Dot Markers: Towards Augmented Unprepared Maps with Pure Geographic Features" [36].

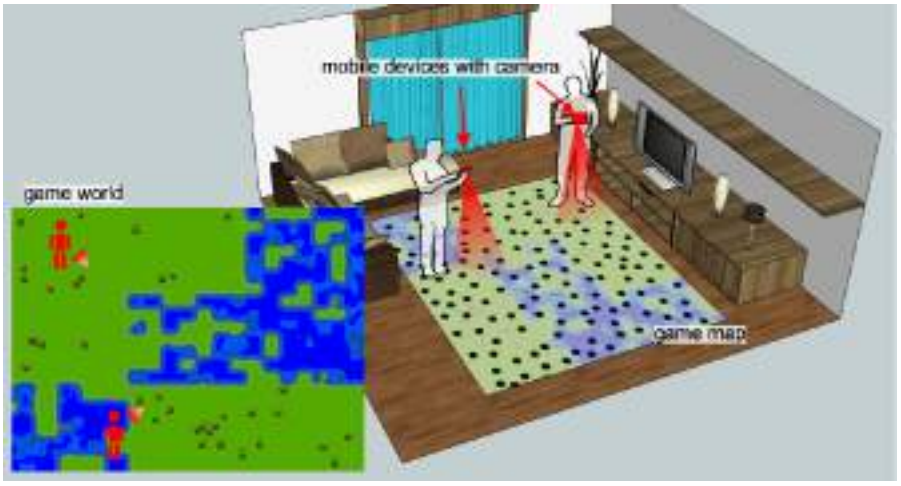


Figure 18: Scalable map of RDM used for mobile games.

3.7 CALTAG

Automatic high-accuracy camera calibration.

CALTag ("CALibration Tags") was introduced by Atcheson, Heide, and Heidrich in their paper "CALTag: High Precision Fiducial Markers for Camera Calibration" [17] as a self-identifying marker pattern targeted at very precise camera and multi-camera calibration (Figure 19). The pattern is designed to support high-precision, fully-automatic localization of calibration points, as well as identification of individual markers in the presence of significant occlusions, uneven illumination, and observations under extremely acute angles. Also no silent zone around the individual markers is needed, therefore arrays can be easily constructed just by altering black-and-white with white-and-black markers and attaching one to another.

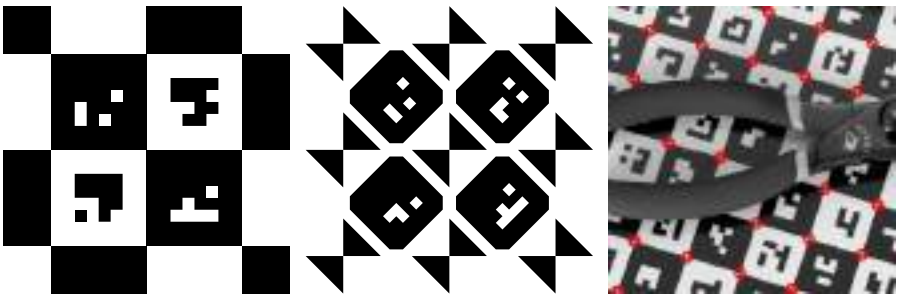


Figure 19: CALTag marker design. **Left:** Checkerstyle layout in which touching squares provide the calibration points. **Middle:** Rotated layout with additional bowtie symbols providing calibration points. **Right:** Results of detection under difficult conditions.

Advantages of this system are very accurate localization of calibration points using subpixel saddle point finders; high area density of both calibration points and markers (due to the possibility of attaching markers together without any silent zone); robustness under occlusion, uneven lighting conditions and radial distortion; reducing false detections of markers using checksums (CRC); no parameter tweaking and system doesn't require that the entire calibration pattern (array) is in the view.

CALTag is meant for camera calibration!

As mentioned above, this entire system was developed for very precise camera calibration, therefore it employs many computationally expensive steps to enhance its precision. Single calibration takes from 2 seconds for a 2 megapixel image to 8 seconds for a 12 megapixel image. That is why its use as an augmented reality system is limited. This comparison is here only because of the topic of this work. To be fair to the authors, their system was never indented for AR use.

They have also of course conducted precision measurement against ARTag. Using their own implementation of Zhang's calibration algorithm from the paper "A flexible new technique for camera calibration" [37], they obtained a

mean reprojection error of 1.073 pixels for the ARTag points, versus 0.316 pixels for CALTag points (Figure 21).

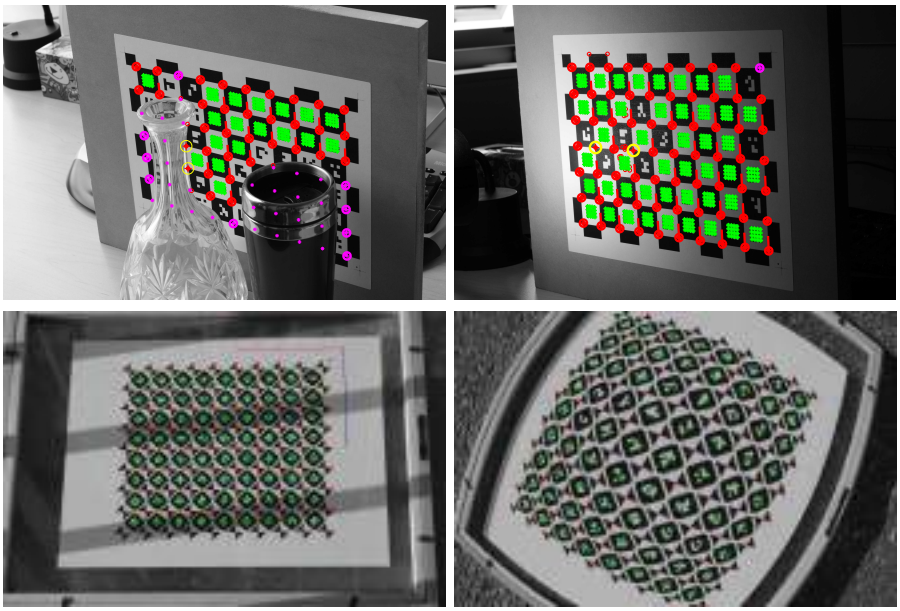


Figure 20: Detection results under difficult conditions.

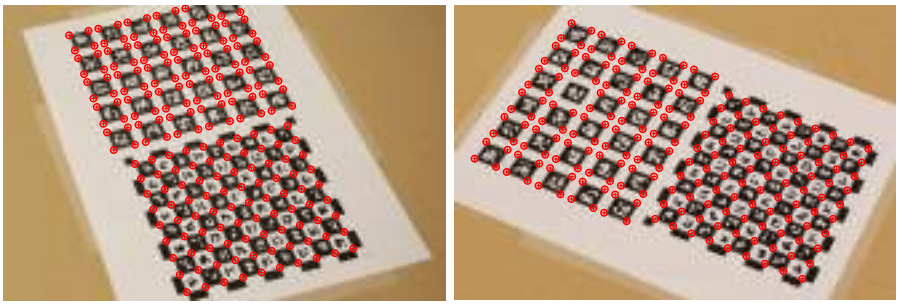


Figure 21: Two of sixteen images used for the precision test.

3.8 NESTED MARKERS

The Nested Markers system is the first one in this overview that aims at increasing the range where can the algorithm still reliably and robustly detect the marker and estimate the camera position and rotation (Figure 23).

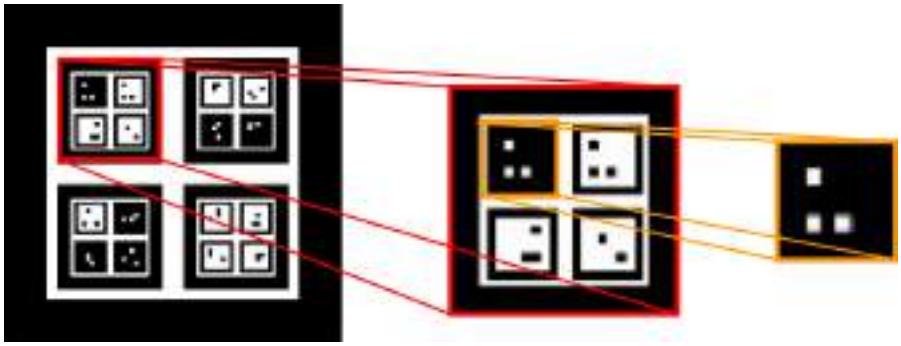


Figure 22: Recursive layered structure of the Nested Marker.

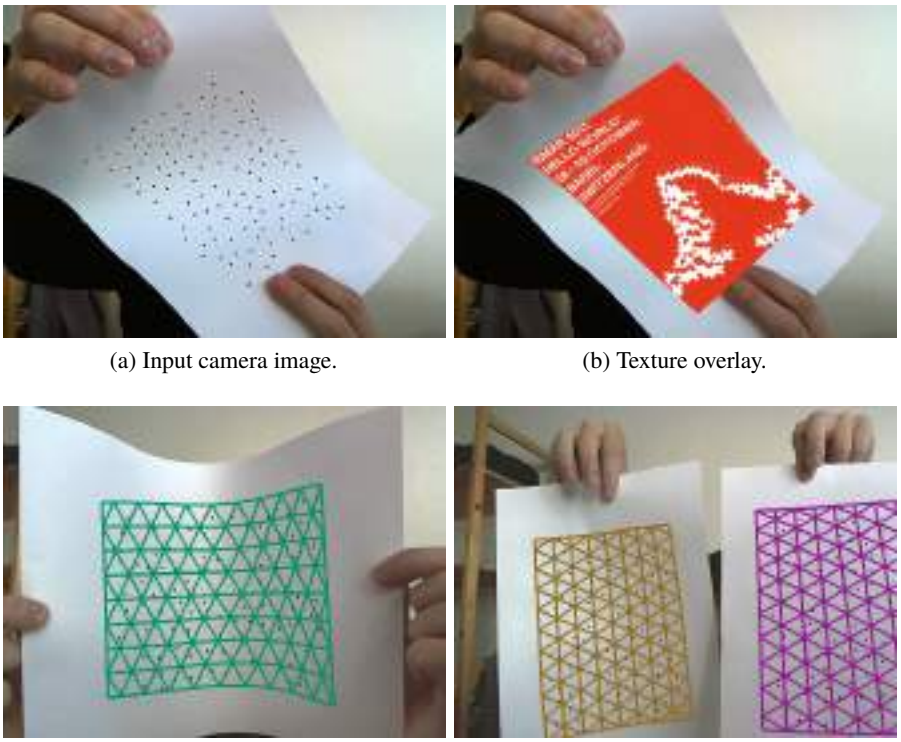
Figure 23: Example of augmented scene, showing extended range of possible camera movement.

Tateno, Kitahara, and Ohta in their papers [18, 19] introduced the Nested Marker as a recursive layered structure (Figure 22). It has one marker in the top layer and four smaller markers in the middle layer. The middle layer markers have also four lower layer markers nested inside them. Every marker at every layer is designed to be identified by its interior pattern.

The largest marker visible by the camera is dynamically selected according to the distance between the camera and the marker. The lowest layer marker has its ID stored in $4 \times 4 = 16$ grid pattern. Two types of this level markers are distinguished by background color – black or white. IDs of higher level markers are determined by average value of 2×2 grid that consists of lower level markers. When the average is “more black” it is considered black, it goes same way for white.

3.9 DEFORMABLE RANDOM DOT MARKERS

Uchiyama and Marchand [20] introduced Deformable Random Dot Marker (DRDM), extending their planar Random Dot Markers to non-rigidly deformable marker. They are using non-rigid surface recovery method based on the paper "Progressive Finite Newton Approach To Real-time Nonrigid Surface Detection" by Zhu and Lyu [38]. They formulated the problem of non-rigid surface recovery from keypoint correspondences such that the shape of 2D non-rigid surface could be computed by solving two linear equations with the finite number of iteration. The marker is initially detected as an planar RDM as in [16]. Once the marker is recognized, the detected plane is then used as the reference plane for Zhu's method.



(a) Input camera image.

(b) Texture overlay.

(c) Aditional examples of recovered surfaces augmented with triangle grid.

Figure 24: Example of Deformable Random Dot Markers (DRDMs).

3.10 TEXTURE MAPPING ON A DEFORMABLE OBJECT

Fujimoto et al. in their paper "Geometrically-Correct Projection-Based Texture Mapping Onto a Deformable Object" [21] presented novel approach for projection-based augmented reality which commonly employs a rigid projection surface and does not support scenarios where this surface can be deformed. Their AR system can project on a deformable surface that can be bent, twisted or folded.

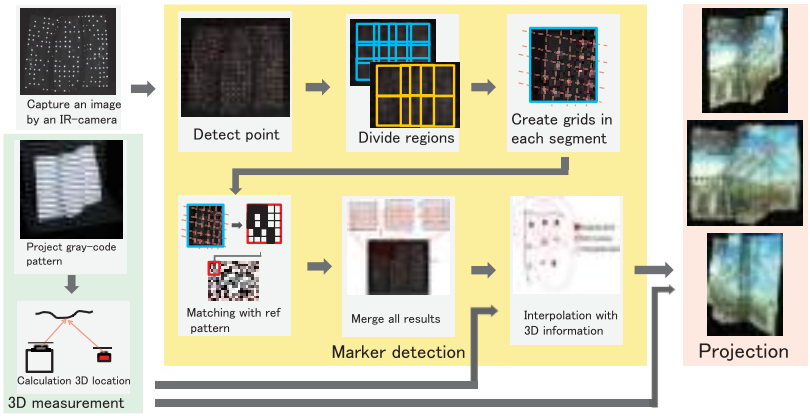


Figure 25: Overview of the entire system.

Deformable marker carrier and retro-reflective marker.

The carrier of the marker proposed by Fujimoto et al. is composed of metal mesh and acrylic foam. This combination ensures that the deformable marker retains its shape after the user manipulation. On top of this carrier is semi-transparent retro-reflective material with an actual marker embedded in it. As for the marker construction, they were inspired by our Uniform Marker Fields (UMFs) (will be discussed later in this work), except instead of squares as modules they use points. The reason is that after deformation, squares would not be so easily detectable (Figure 26).

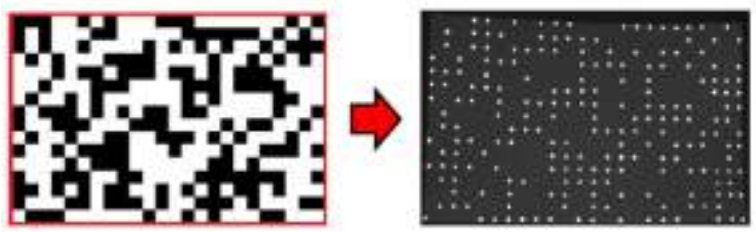
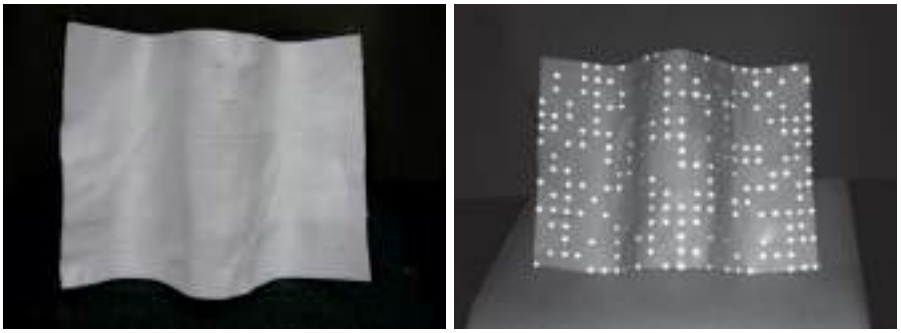


Figure 26: Change from squares as marker modules to points because of the impossibility of detection of squares after deformation.

RGB/IR camera is needed.

For video capturing they use camera with RGB and Infrared (IR) modes that cannot work simultaneously but can be toggled. When IR mode is activated infrared LED diodes are also turned on to illuminate the scene and a filter that

lets only IR light in covers the camera sensor. Resulting images of the same scene captured in RGB and IR mode are shown in Figure 27.



(a) RGB mode. (b) IR mode.

Figure 27: Images of same scene captured in RGB and IR camera mode.

Figure 28: Results of texture projection. The same texture is deformed and projected onto differently deformed surfaces.

They employ quite complicated and elaborated methods for detection and identification of points and of the entire marker. Entire process takes non-negligible processing time (approximately 8.5 seconds) so it is not possible to use it in real-time or interactive applications. The authors propose use of tracking of points instead of redetecting the surface in each camera frame for decreasing the computational time.

3.11 LENTIMARK

The LentiMark system was introduced by Tanaka, Sumi, and Matsumoto in their paper "A Visual Marker for Precise Pose Estimation based on Lenticular Lenses" [22] and then improved in "A solution to pose ambiguity of visual markers using Moiré patterns" [39]. It aims at improving the accuracy and stability of pose estimation mainly when the camera observes the marker from frontal direction. This particular singular position is problem for conventional visual based fiduciary marker systems [40, 41] (see Figure 29 borrowed from Tanaka paper to understand this problem).

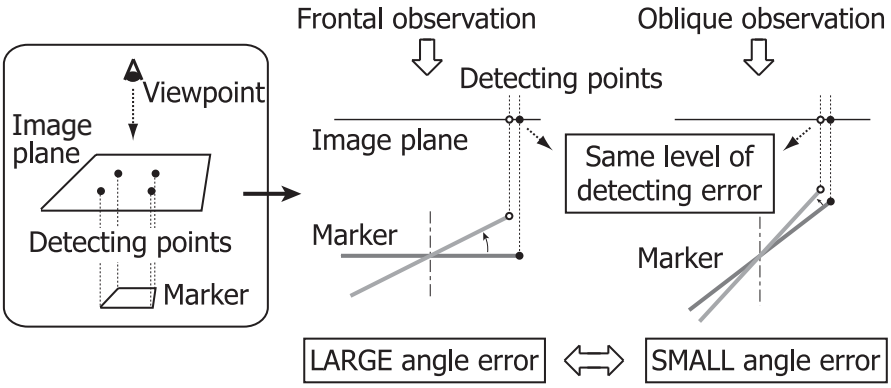


Figure 29: Difference of angle error according to direction of observation.

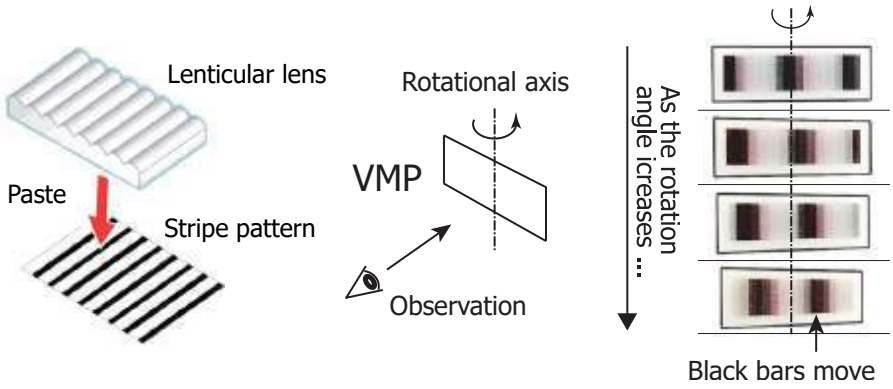


Figure 30: **Left:** structure of VMP. **Middle and right:** Changing of moiré pattern when observation angle changes.

Novel approach using visual interference patterns.

In their paper, they solved the problem by developing a very novel visual marker based on displaying moiré patterns (i.e. interference patterns) which change their appearance according to the angle of observation. It is made up of a lenticular lens and a two-tone stripe pattern. They call this Variable Moiré Pattern (VMP) (Figure 30).

VMP-unit

Using two VMPs and four anchoring points for image processing they construct a VMP-unit (Figure 31 left). The LentiMark includes already existing AR marker (they use ARToolKit in their paper but almost any other AR system

Black-peak.

could be used) for identification and orientation of the marker which VMP-unit does not provide (Figure 31 right). The darkest point on a VMP is called "black-peak".

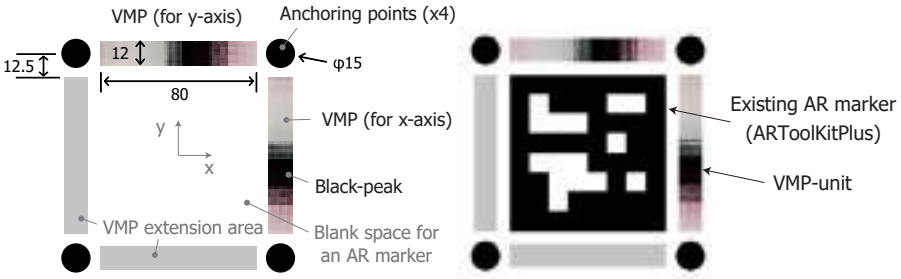


Figure 31: **Left:** Design of VMP-unit (sizes are ratios). **Right:** Entire LentiMark.

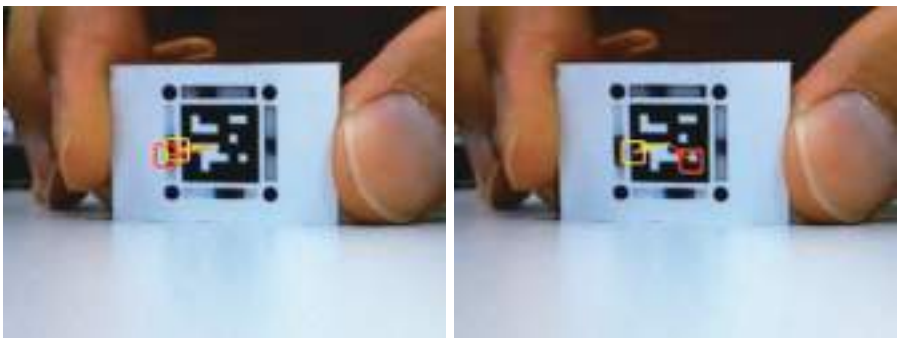


Figure 32: Comparison of LentiMark (yellow) and ARToolKit (red) pose estimation. Marker is very slowly rotated and superior stability of LentiMark can be clearly seen.

They built a prototype of LentiMark for experiment and evaluation purposes and their results confirm that system can very accurately calculate 6-DOF position by fusing the position of "black-peak" and pose estimation form ARToolKit even in case of direct frontal observation.

3.12 ARRAYMARK

ArrayMark system is an improved version of LentiMark introduced by the same authors Tanaka, Sumi, and Matsumoto in their papers "A novel AR marker for high-accuracy stable image overlay" [42], "A high-accuracy visual marker based on a microlens array" [23], and "Further stabilization of a microlens-array-based fiducial marker" [24].

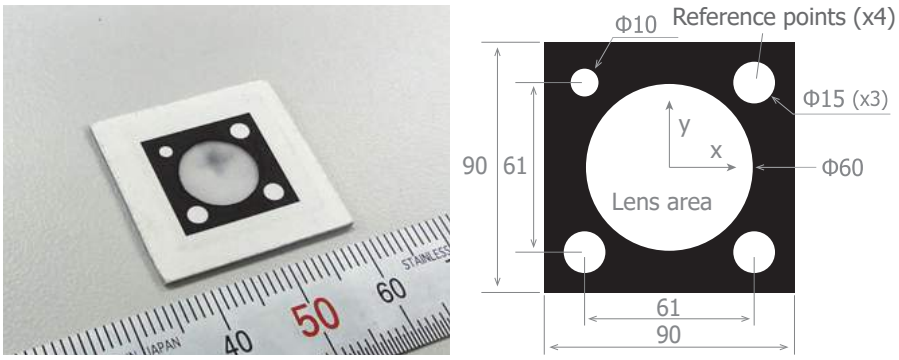


Figure 33: **Left:** ArrayMark prototype. **Right:** Design of ArrayMark. The sizes are again only ratios.

ArrayMark is easier to fabricate than LentiMark.

One of the weak points of LentiMark is the complexity of the fabrication. One LentiMark needs two moiré patterns (VMPs) with different lens direction. ArrayMark combines two 1D lenses into one 2D lens, hence the fabrication is easier. Instead of two-tone stripe pattern and lenticular lenses they use criss-cross pattern and array of micro lenses (Figure 34).

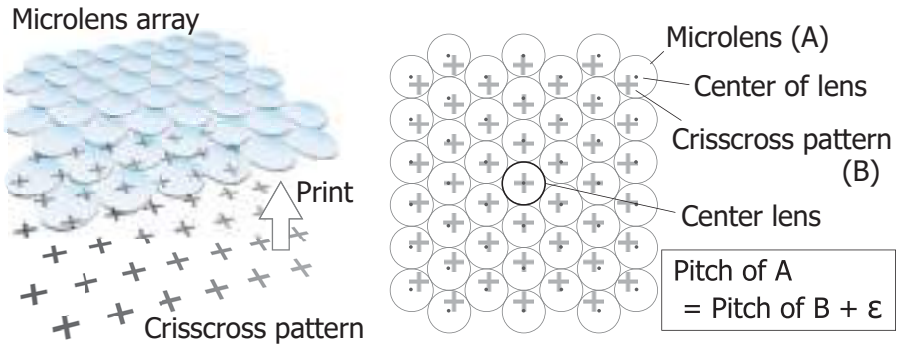


Figure 34: **Left:** Structure of lens area. **Right:** Relation between lenses and crosses.

ArrayMark doesn't have any identification capabilities like LentiMark had because its missing the ARToolKit part of its predecessor. Authors claim that they will "attach some sort of patterns aside of ArrayMark, if the marker-ID is needed in ArrayMark itself".

Again authors provided very thorough results and evaluations of accuracy of camera pose estimation, computational speed, minimum size of marker

in camera image to reliably establish camera pose, measurable angle range, robustness to distance and illumination variation.

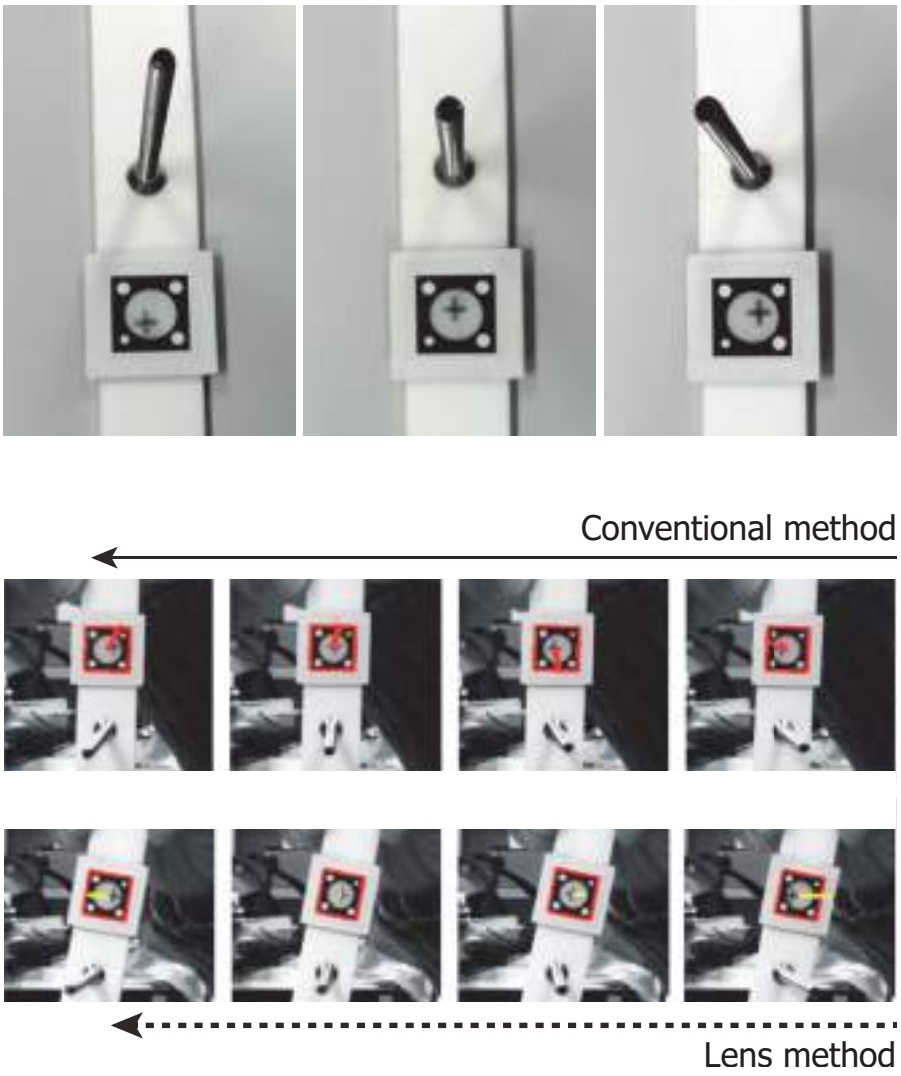


Figure 35: Evaluation of ArrayMark precision.

3.13 SUMMARY & INTERPRETATION

This overview shows that fiduciary markers were and still are important and interesting topic in Augmented Reality (AR), robotics and navigation research.

Even today when markerless approaches are still more and more feasible thanks to ever increasing power of the mobile devices and creation and wide spread of the specialized sensors for environment sensing, new research in the field of markers is done – good examples are LentiMark and ArrayMark whose research is from years 2012, 2013 and 2014. And of course our research which is still active and fruitful.

Markers are even used in commercial products like reacTable which is using "amoeba" fiduciary markers. It started as university research at 2005 and in 2009 a spin-off company Reactable Systems[3] was founded.

The main reasons why are the markers still preferred in some applications over the markerless approaches are their simplicity, robustness, precision, easier controllability, computationally cheaper detection and only simple camera is needed for their detection.

But there are still some weak points in contemporary marker designs which are already mentioned in Section 2.2. We aim at solving them in our work on marker fields described in the next part of this work.

3 http://reactable.com/

Part II

DESIGN OF PROPOSED MARKERS

INTRODUCTION TO PROPOSED MARKER DESIGNS

In this part of my thesis I present different marker fields on which I worked and where I designed different aspects – mainly the overall ideas and general properties. For example, idea of large-scale array of markers with individual markers overlapping and each one unique in entire array in Uniform Marker Fields (UMFs) (Chapter 5) or fractal structure and data payload of Fractal Marker Fields (FMFs) (Chapter 4). A lot of text in this part is inspired or directly taken from our published papers. I believe they are very well written and description of the markers' design is complete.

List of my published papers connected to the topic of markers:

FRACTAL MARKER FIELDS

Herout, Zachariáš, Dubská, and Havel: "Fractal Marker Fields: No More Scale Limitations for Fiduciary Markers" (ISMAR 2012) [I]

UNIFORM MARKER FIELDS

Szentandrási, Zachariáš, Havel, Herout, Dubská, and Kajan: "Uniform Marker Fields: Camera Localization By Orientable De Bruijn Tori" (ISMAR 2012) [II]

IMPROVED UNIFORM MARKER FIELDS

Herout, Szentandrási, Zachariáš, Dubská, and Kajan: "Five Shades of Grey for Fast and Reliable Camera Pose Estimation" (CVPR 2013) [VI]

Zachariáš and Herout: "Color Theme-Based Digital Art for Augmented Environment" (ICDAMT 2016) [IX]

Zachariáš, Szentandrási, and Herout: "Visual Correction of Position Drift using Uniform Marker Fields" (SCCG 2016) [X]

HONEYCOMB MARKER FIELDS

Horváth, Herout, Szentandrási, and Zachariáš: "Design and Detection of Local Geometric Features for Deformable Marker Fields" (SCCG 2013) [V]

4

FRACTAL MARKER FIELDS

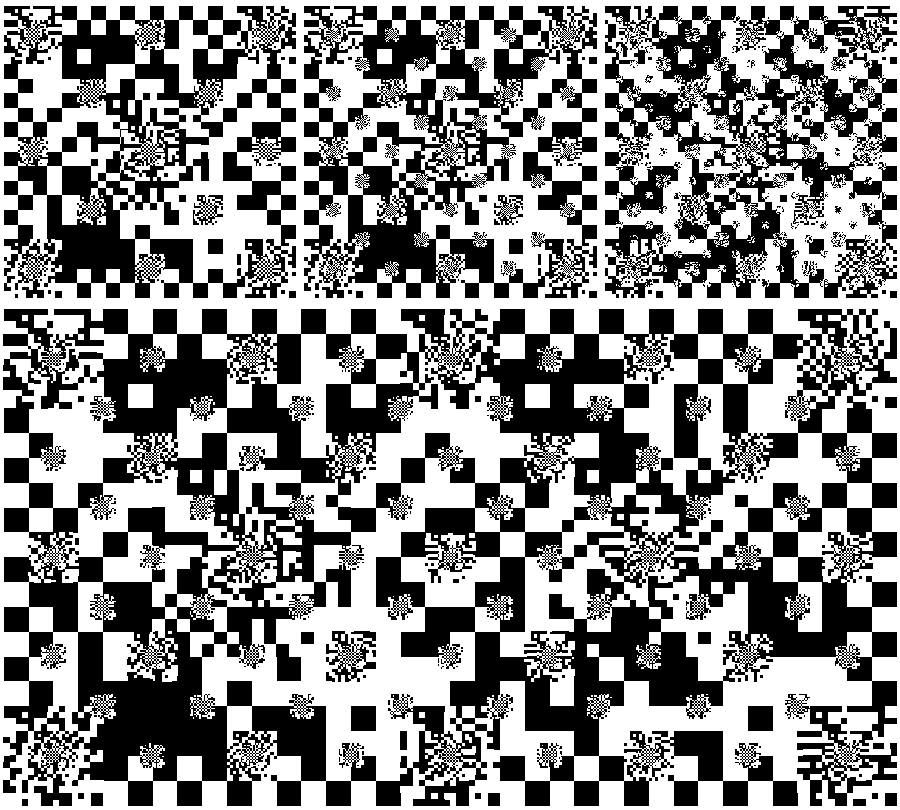


Figure 36: Examples of FMFs with different number of levels. **Top line:** Single square FMFs with 3, 4 and 5 levels from left to right. **Bottom line:** Two 5-level FMFs seamlessly connected together to make 2:1 rectangular marker.

In the paper "Fractal Marker Fields: No More Scale Limitations for Fiduciary Markers" [I] in proceedings of the *IEEE International Symposium on Mixed and Augmented Reality in 2012* (ISMAR 2012) we introduced a fractal structure of markers inspired by chessboard matrix codes such as the QR code or the DataMatrix and Nested Markers. We named it Fractal Marker Field (FMF) (Figure 36).

Virtually infinite levels of nesting.

This fractal structure allows for virtually infinite levels of nesting the markers one into another and guarantees coverage of all sizes of markers across the whole field. In this way, the marker field allows for computing the camera position regardless of how distant or close the camera is from the field and where it is located; in practical applications, many orders of magnitude of

scale are allowed and the field can be as large as any practically printable area (hundreds of square meters).

The number of straight line edges within the markers allows for reliable and precise camera calibration from the chessboard structure. The individual markers within the field contain encoded information about their identity with an error correction capability. The marker field is designed to allow for real-time detection and precise localization in quite low-resolution images using moderate computing resources.

Our structure is similar to Sierpiński carpet.

The FMF can provide unprecedented freedom of motion to camera-based augmented reality applications. The FMF is similar to the well-studied Sierpiński carpet; however, it is qualitatively different and constitutes a new, previously undocumented deterministic fractal structure with interesting applications, especially in camera localization.

4.1 CONSTRUCTION OF FMF

In this section will be explained the concept of the FMFs. Particular layout will be used as example, which seems to be the most practical according to the experiments carried out so far. Section 4.4 will discuss some alternative potentially feasible layouts of the FMF to show that a large number of possible forms of FMF are conceivable.

The FMF consists of individual markers of different scales and positions. A single marker is depicted in Figure 39.

The marker is a square chessboard grid of black and white *modules*. The grid is not a perfect and full chessboard, but the configuration of the black and white modules encodes useful information (in a way similar to the QR codes). Some modules have fixed values and form a pattern to help localize the marker and determine its orientation. Some modules (marked by red and blue in Figure 39) contain metadata about the way the payload data is stored in the data area (green color in the figure). Details of this marker layout are discussed in Section 4.2.

Any instance of the marker pattern will be denoted as $\mathcal{M}_n(u, v)$, where $n \in \mathbb{N}_0$ is the marker's *level* and $u, v \in \mathbb{Z}$ are indices of a *spatial instance*. The marker's properties n, u, v are about to be explained using the FMF construction rules. The marker pattern is repeated throughout the marker field according to the following main rules:

Pivot (center) markers are concentric and half of the size of $n-1$ pivot marker.

1. A level-n pivot marker $\mathcal{M}_n(0,0)$ contains marker $\mathcal{M}_{n+1}(0,0)$ embedded within. These two markers are concentric, which means that the image locations L of their centers are identical:

$$\mathbf{L}(\mathcal{M}_{n+1}(0,0)) = \mathbf{L}(\mathcal{M}_n(0,0)). \tag{1}$$

(a) Rule 1. (b) Rule 2. (c) Rule 3.

Figure 37: The three main construction rules: **(a)** From $\mathcal{M}_n$ (black) to $\mathcal{M}_{n+1}$ (red); **(b)** From $\mathcal{M}_n(0,0)$ (black) to $\mathcal{M}_n(u,v)$ (red); **(c)** Markers $\mathcal{M}_0(0,0)$ and $\mathcal{M}_1(0,0)$ at levels 0 and 1 (black) and the barren margins (grey). Level 2 markers (red) are just completely visible at the corners of the pivot marker.

The size of $\mathcal{M}_{n+1}$ (i.e. the side of the bounding square S) is $\frac{1}{2}$ of $\mathcal{M}_n$:

$$\mathbf{S}(\mathcal{M}_{n+1}) = \frac{1}{2}\mathbf{S}(\mathcal{M}_n). \tag{2}$$

Refer to Figure 37a for an illustration of this rule.

2. At each level n, markers $\mathcal{M}_n(u,v)$ periodically appear in the field; their locations are defined as:

Location of markers at level.

$$\mathbf{L}(\mathcal{M}_n(u,v)) = \mathbf{L}(\mathcal{M}_n(0,0)) + (2su, 2sv), \tag{3}$$

$$u + v = 2i, \text{where } i \in \mathbb{Z}. \tag{4}$$

where $s = \mathbf{S}(\mathcal{M}_n)$ is the size of the marker in the image units. Refer to Figure 37b.

3. The pivot marker $\mathcal{M}_0(0,0)$ is placed into the center of the FMF image so that its size $\mathbf{S}(\mathcal{M}_0(0,0))$ is $\frac{4}{5}$ of the image side (or of its smaller dimension if the image is not a square). This allows the L2 markers $(\mathcal{M}_2(1,1), \mathcal{M}_2(-1,1), \mathcal{M}_2(1,-1), \mathcal{M}_2(-1,-1))$ to be completely visible in the corners of the pivot marker. (Figure 37c)

Ensuring the complete visibility of L2 markers.

In this particular arrangement (i.e. for the marker layout from Figure 39 and the construction rules equations (1)–(4)), up to 4-level FMFs can be generated without markers of one level colliding with markers of another level (Figure 38 left). With a higher number of levels in the FMF, their collisions are inevitable (Figure 38 right) and it would be inevitable for any design of the marker structure. The following text is going to deal with this potential problem.

FMF greying problem.

As the number of levels approaches infinity, the infinitesimally small markers are covering the whole image, which then becomes an unreadable "grey matter". It should be noted that unlike the Sierpiński carpet, which ultimately has zero area and infinite circumference, the FMF's area is equal to one half

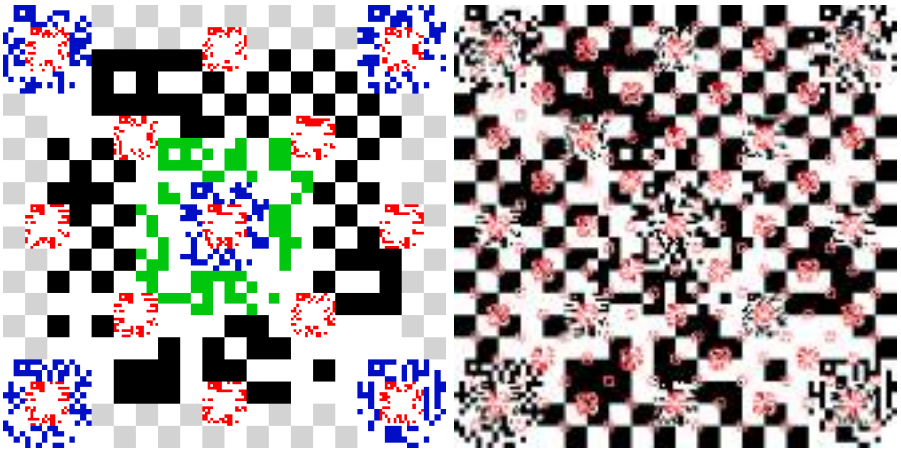


Figure 38: **Left:** Constructed pyramid (4 levels, L0 to L3); black: L0, green: L1, blue: L2, red: L3, grey: Margin filled with pure chessboard of L0-sized modules. No marker collides with any other. **Right:** Fractal Marker Field (FMF) of 6 levels (L0 through L5). Levels L4 and L5 (red) collide with lower-level markers (black).

of the complete image area, provided that the individual markers (Figure 39) are perfectly "grey" (i.e. the numbers of black and white modules are statistically equal). Section 4.3 will deal with this issue of "FMF greying". Before we approach this matter, let us discuss the marker layout and its synthesis in detail.

4.2 SINGLE MARKER IN DETAIL

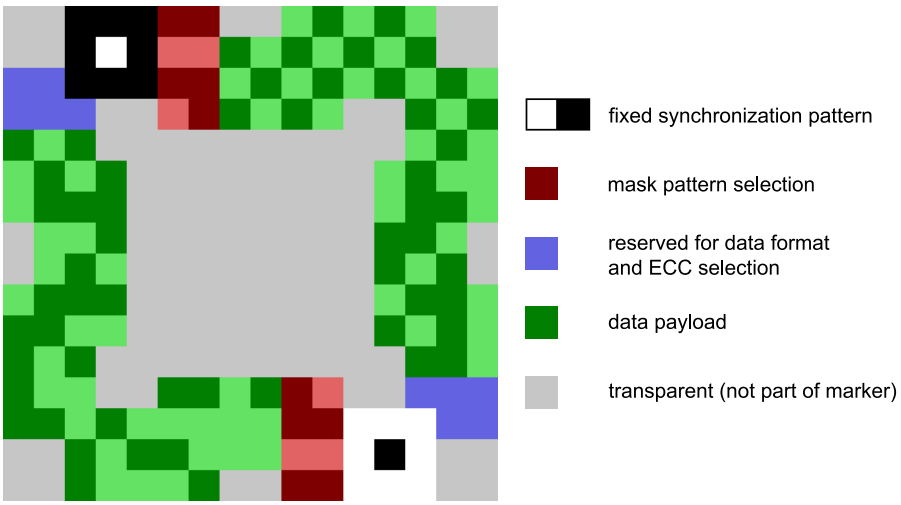


Figure 39: A single marker which is repeated at different scales/locations throughout the Field. The marker fits in a 16×16 grid.

Black and white modules in Figure 39 are always same and are used for rough determination of camera position and rotation.

Each single marker encodes information (Figure 40) about its identity in 56 bits: a unique identifier of the entire marker field (16 bits), the marker's level n in the field (8 bits), and the marker's position within the level u, v (16 + 16 bits). These 56 bits are extended by a Reed-Solomon error detection/-correction code (used in the QR codes as well) to 112 bits which can be carried by the data modules in the marker (data + Reed-Solomon = green color in Figure 39).

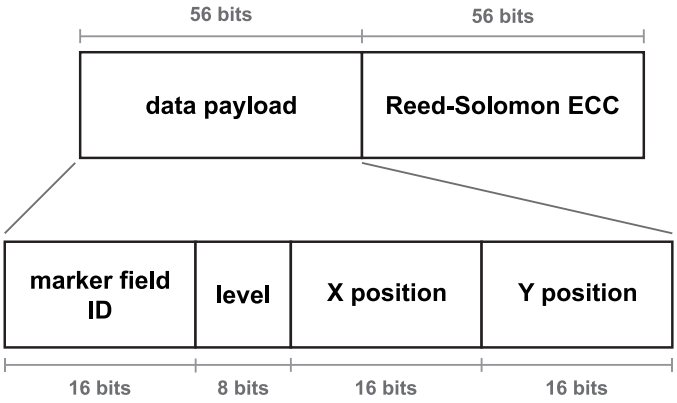


Figure 40: Break down of data payload encoded in single marker.

Each of the data modules is identified by its location within the marker (i, j), where $i, j \in \{0, \ldots, 15\}$. The 112 bits of payload are processed by one of the predefined mask patterns (Figure 41). The FMF generator tries all of the masks and selects the best performing one: ideally, the final marker should have the numbers of black and white modules balanced and both vertical and horizontal edges between the black/white modules must be present across the whole marker. The selected mask's ID is encoded by the Hamming code into 8 bits stored twice in the marker (red modules in Figure 39).

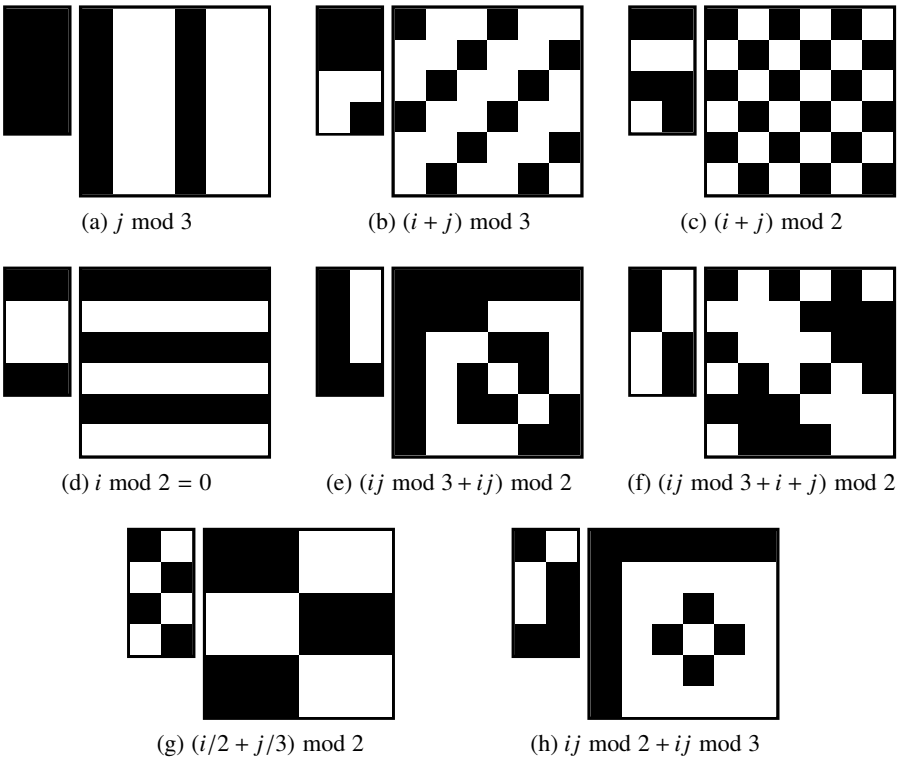
4.3 BRIGHT, DARK, AND GREY MARKERS

By using different mask patterns (Figure 41), different valid markers can be generated. The ratios of the number of black modules b and the number of white modules w in marker can be roughly divided into three groups: "dark markers" with b/w high (around 2), "bright markers" with b/w low (around $\frac{1}{2}$), and "grey markers" with $b/w \simeq 1$.

In order to avoid the unwanted situation when by increasing the FMF level count the image becomes unreadable, additional rules can be added to the three main construction rules defined above:

4. The FMF generator constructs the field sequentially, starting from level 0 and increasing the level index. The generated image is initially filled with "undefined" value. Writing a marker into the image means setting

Grey modules in Figure 39 do not interfere with already drawn data.



(a) $j \bmod 3$ (b) $(i+j) \bmod 3$ (c) $(i+j) \bmod 2$

(d) $i \bmod 2 = 0$ (e) $(ij \bmod 3 + ij) \bmod 2$ (f) $(ij \bmod 3 + i + j) \bmod 2$

(g) $(i/2 + j/3) \bmod 2$ (h) $ij \bmod 2 + ij \bmod 3$

Figure 41: Binary 6×6 mask patterns and their 2×4 identifiers using the Hamming code. Modules under the white areas (binary 1) of the mask are inverted (XOR operation) when applying on the data.

some locations of the image to "white", other locations to "black", and leaving the rest of the image in its previous state (in locations where the marker modules are grey in Figure 39).

Selecting right version of marker (bright, dark or grey).

5. Before writing the marker into the image, the generator checks the image contents underneath the marker's bounding square and decides whether to select the bright, dark, or grey version of the marker (which can be arbitrarily controlled by selecting a suitable mask pattern).

Rules 4. and 5. ensure that even for a high number of levels in the FMF, the collisions of higher levels do not make the lower levels unreadable and the structure can be virtually unlimited (Figure 42). The lower-level markers are not white and black anymore. Instead, they are darker and brighter grey; however, the contrast is still high enough to allow for reliable detection.

4.4 ALTERNATIVE MARKER FIELD LAYOUTS

The design of the marker and the set of the FMF construction rules described above is not the only possible variant. This section will mention some alternative layouts of the FMF which could prevail in particular applications.

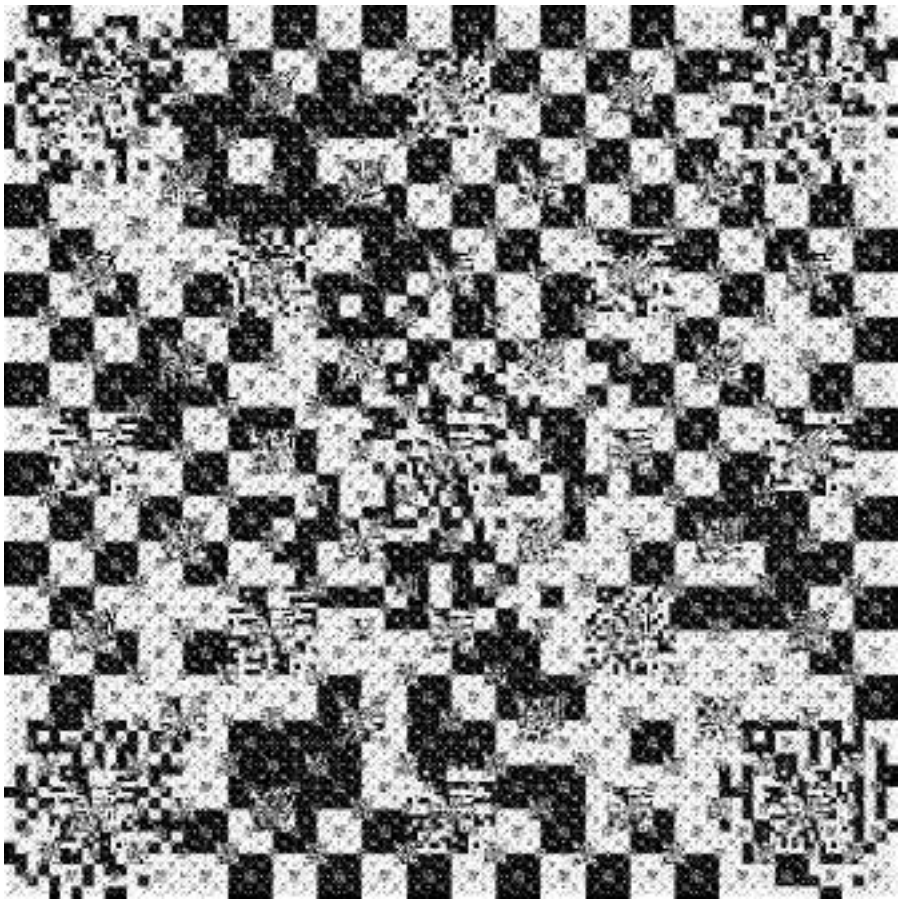


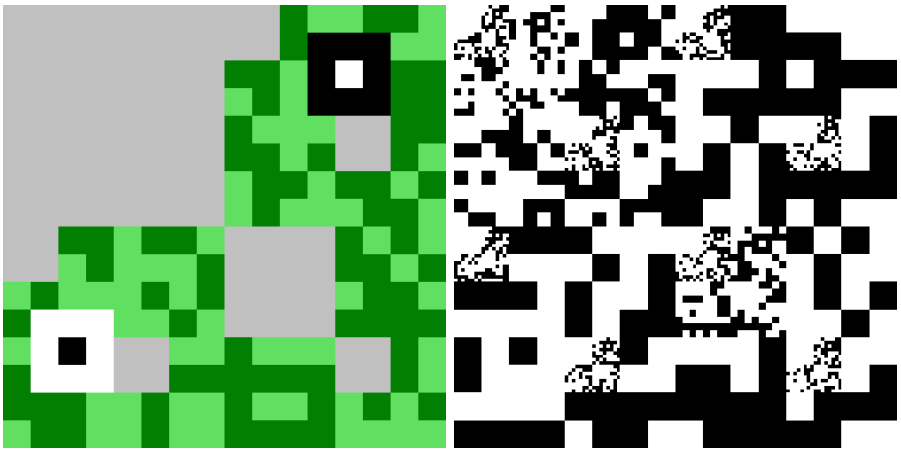
Figure 42: FMF of 10 levels (L0 through L9). This FMF uses bright and dark markers and thus it is readable despite the high number of levels involved. A marker field with 10 levels can cover the whole soccer playground with L9 markers smaller than 8 cm, which is a normal dimension of today's AR markers.

One vital alternative to the concentric marker design (Figure 39) is the "corner layout". The marker shape is shown in Figure 43a. Equation (1) from Rule 1. must be altered to

$$\mathbf{L}(\mathcal{M}_{n+1}(0,0)) = \mathbf{L}(\mathcal{M}_n(0,0)) - (\frac{s}{4}, \frac{s}{4}), \tag{5}$$

where $s = \mathbf{S}(\mathcal{M}_n)$ is the size of the marker. A sample FMF generated by using this layout is in Figure 43b. Rule 3. is modified so that the pivot marker $\mathcal{M}_0(0,0)$ spans across the whole generated field image; avoiding the necessity of the empty margin around the pivot marker seems to be the biggest advantage of this design.

The module resolution of the marker can be lower or higher than 16×16. Lowering the resolution to 8×8 would only offer a very limited space for the data payload in the markers. On the other hand, 32×32 would increase the requirements on the camera pixel resolution for the markers to be detectable.



(a) Corner-centric marker. (b) Corresponding FMF.

Figure 43: Alternative marker design. **(a)** Corner layout of the marker. **(b)** Corresponding FMF with 4 levels of markers.

Higher module resolution of the markers (at least 32×32) would allow for larger finder patterns to fit in the marker. For instance, patterns similar to those used in the QR codes could be used and searched for by the detection algorithm. However, the QR code finder patterns are only easily detectable when perspective deformation is avoided; this is not feasible for most applications of the FMF and the detection algorithm described in Section 4.5 seems to be the basis for an optimal solution, both for its robustness and computational efficiency.

4.5 DETECTION OF MARKERS IN FMF IMAGES

We have also implemented detection algorithm to prove that FMF can be efficiently detected and tracked. This section will outline a detection algorithm used in our experiments done in Section 4.6.

The algorithm is a very subtle modification of the algorithm for detection of chessboard grids and matrix codes by Dubska et al. [43]. It efficiently searches for straight lines that coincide with two dominant vanishing points by using the Hough transform and a line parametrization which is a point-to-line mapping. The algorithm processes the input image in four steps:

1. **Extraction/Selection of Edges in the Input Image.** The image is processed by a computationally cheap edge detector and information about the gradient for the edge pixels is obtained. A histogram of edge orientations is constructed during the extraction and two dominant edge orientations roughly 90° apart are selected out of all the edges (Figure 44).

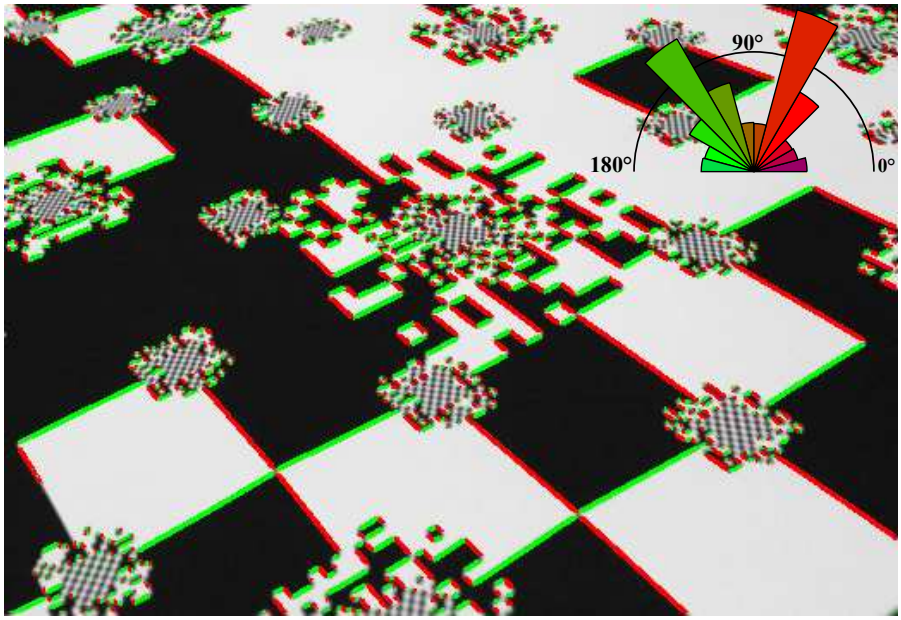
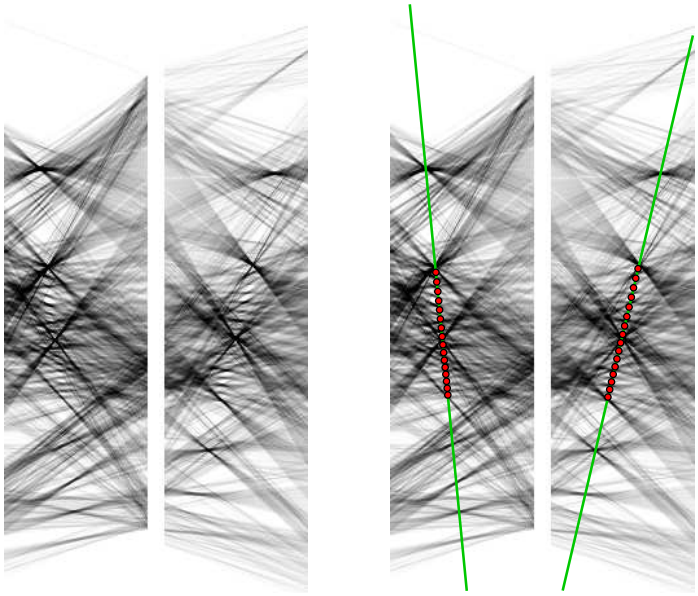


Figure 44: Edge extraction and selection. Two dominant edge orientations (red, green) are selected for further processing.

2. **Accumulation to the Hough Space.** For each of the two dominant groups of edges, a small part of the complete Hough space is being accumulated. A line parametrization by Dubská, Herout, and Havel [44] is used, because a vanishing point is there represented by a line in the Hough space (contrary to the θ-ρ line parametrization [45] typically used for line detection by the Hough transform). Since only a small fraction of the Hough space is accumulated, and only a selected subset of the edges extracted from the image is used, this step is fairly efficient. Figure 45a shows the two stripes of the Hough space, as accumulated for the image shown in Figure 44.

3. **Finding Groups of Originally Parallel Lines.** In each of the two stripes of the Hough space, N highest local maxima are detected (N is a constant, typically 8). These maxima correspond to the strongest lines in the input image and their vanishing point (Figure 45b) is represented by a line ℓ in the Hough space, coincident with the maxima. As explained in [43], stronger perspective means faster convergence of the maxima corresponding to equally spaced parallel lines to the horizon. By using the information about the positions of both the vanishing points in the input image, line ℓ can be stretched so that the maxima on it are equally spaced. By auto-correlating this stretched line and finding local maxima, dominant frequencies can be found. Each frequency characterizes one group of equally spaced maxima in the Hough space (i.e. equally spaced lines in the 3D space, as projected to the input image). In the case of the FMF, it means all parallel edge lines of exactly one marker level/scale.



(a) Hough space stripes. (b) Detected vanishing points

Figure 45: **(a)** Accumulated parts of the Hough space (for the image from Figure 44). **(b)** Detected vanishing points (green lines) and 16 lines passing through the centers of the marker's modules (red dots).

4. **Extraction of the Marker Bitmap.** For sampling the input image in the center of each module, the minima (Figure 45b) between two consecutive maxima are used. A Cartesian product of the two groups of minima in the Hough space determines individual modules of the marker. These modules are extracted and they form a low-resolution bitmap which is searched for the synchronization patterns (black and white modules in Figure 39) and further processed.

Step 3 of the algorithm does not always succeed in finding the exact 17 adjacent maxima (i.e. 16 minima) relevant for the particular marker. In order to overcome this issue, a bitmap slightly larger than 16×16 is extracted from the input image and then searched for the synchronization patterns. The computational overhead thus introduced is negligible.

Intelligent sampling and utilizing bright, dark and grey markers.

When extracting the bitmap of modules, each marker module can be disturbed by small markers of higher levels. Since the FMF generator is using the bright and dark markers, the module can be still reconstructed by sampling more than one pixel of the camera image. For low-resolution camera images, averaging all pixels belonging to the module is a simple and efficient choice. If the resolution of the camera is high, Wald's Sequential Probability Ratio Test (SPRT) [46] can be used to test a minimal number of individual pixels belonging to the module in a pseudo-random, optimally sampling manner, by using the Halton sequence [47] or a similar sampling method.

Using the Hough transform for detecting the markers in the FMF requires the lines to be strictly straight. This requires the FMF to be strictly planar and the camera must preserve linearity of straight lines.

4.6 EVALUATION OF DETECTION RESULTS

In this section, we evaluate the possibility of detecting at least one marker in an image of the FMF. We used an adjusted variant of Dubská, Herout, and Havel [43] matrix code detection algorithm, as sketched out in the previous section. This algorithm can be further developed and improved; the purpose of the measurements shown here is to prove that the markers in the FMF can be successfully detected and the concept of the FMF is usable in practical applications.

4.6.1 EVALUATION ON A SET OF REAL-LIFE IMAGES

The images used in the evaluation are snapshots of different locations within a printed-out FMF (1050×3150 mm 6 levels). Since the detector is differently sensitive to the rotation of the code and perspective deformation, the images are sorted into four categories:

- **plain** – the code is upright
- **rot** – the code is rotated
- **pers** – the code is upright but skewed by perspective
- **rot + pers** – general orientation of the code

Table 1 shows the detection rates on this set of images. By positive detection we mean the fact that at least one marker was detected in the image. Each image was processed in three different pixel resolutions: 1200×800, 600×400, and 480×320. When the FMF is viewed from a skew angle without rotation (*pers*), the detector is confused by the different frequencies of the two groups of (originally mutually perpendicular) lines.

resolution	plain	rot	pers	rot+pers	all
$\sum$	33	134	66	131	364
480×320	100	90.3	80.3	87.7	88.5
600×400	100	99.2	87.9	97.7	96.7
1200×800	100	100	93.9	100	98.9

Table 1: Detection rate (%) achieved by the detector. The head row gives the number of processed images along with the distortion type.

4.6.2 EVALUATION ON A REAL-LIFE VIDEOS

Table 2 shows the success rate on a handful of real-life videos. The videos were taken from different FMFs of realistic dimensions (200×200 mm 4 levels, 290×290 mm 5 levels, 1050×3150 mm 6 levels) and with different camera motion factors (zoom, rotation, viewing angle, . . .).

ID	frames	levels	success %	camera motion
V1	1 847	6	99.7	perpendicular view
V2	2 711	6	97.6	fixed zoom, random movement
V3	3 983	6	91.7	zoom, perspective, rapid movement
V4	791	6	97.3	zoom, perspective
V5	2 087	4	96.5	rotation, persp., random movement
V6	1 943	5	98.5	rotation, persp., random movement
V7	1 175	4/5	88.8	two different FMFs

Table 2: Detection performance on real-life videos. Success rate (*success %*) is the fraction of video frames where a marker was detected (in %).

Failures to detect the marker are typically caused by a motion blur (plus V7 contains several frames lacking any markers). At the moment, detector works without any tracking and thus the markers are detected in each frame separately. Use of time coherence and tracking once detected markers in the image could help decrease computational load and, at the same time, increase the reliability of detection. The results clearly show that the detection – even in its current baseline state – could be perfectly usable in many applications.

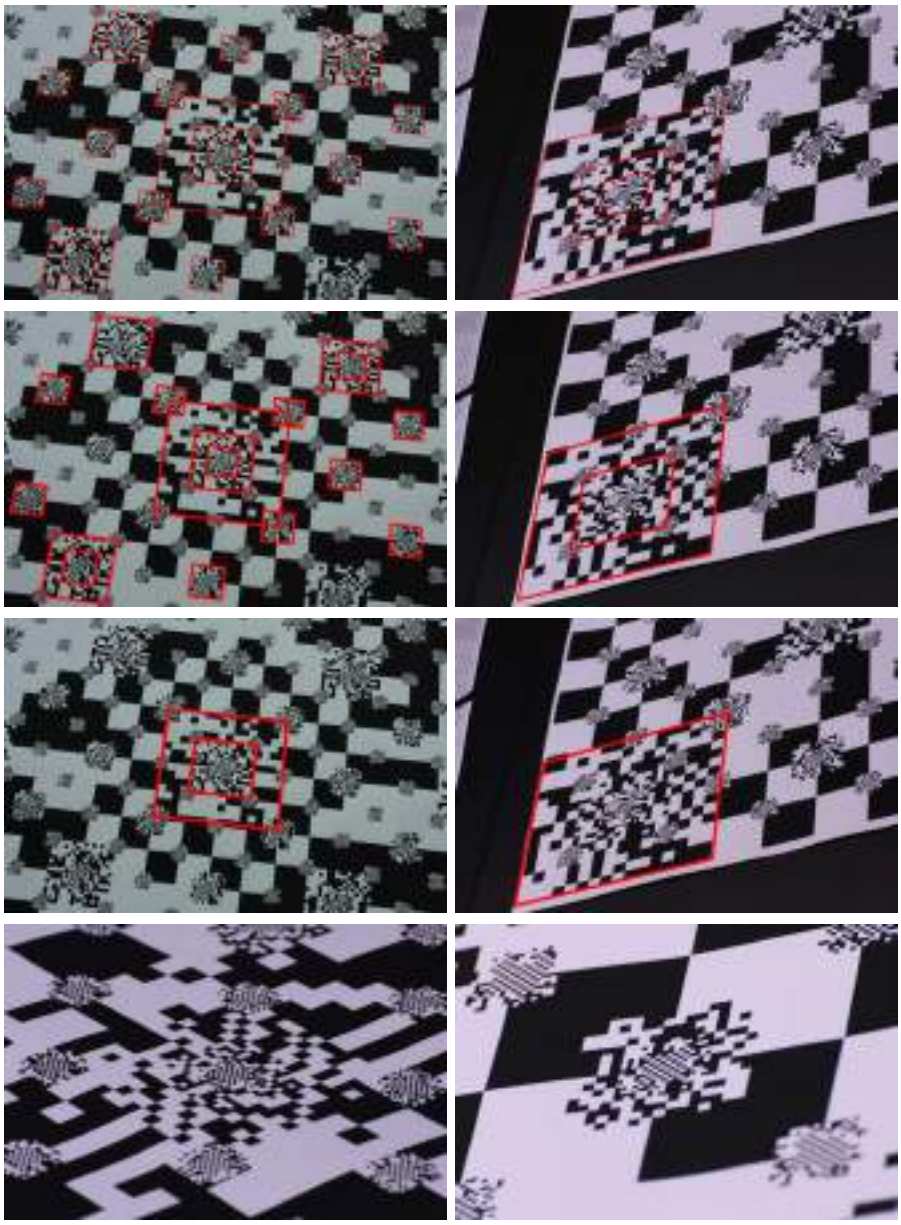


Figure 46: Examples of challenging real-life images of the FMFs mentioned in Section 4.6.1. **Row 1**: Successfully detected markers on real-life images, 1200×800 pixels. **Row 2**: Identical images, but resized to 600×400 pixels. **Row 3**: 480×320 pixels. **Row 4**: The detector failed. Red squares denote markers successfully detected, with the error-correction code valid.

5

UNIFORM MARKER FIELDS

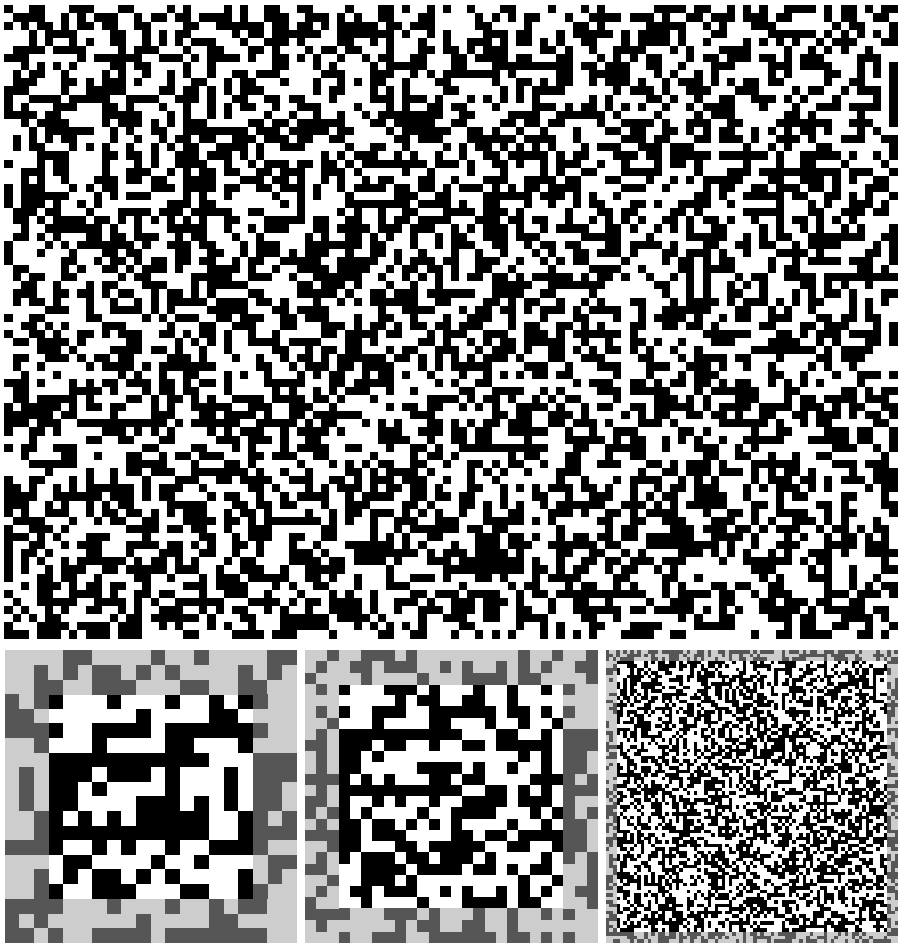


Figure 47: **Top line:** Rectangular aperiodic 110×78 UMF with $n = 4$. **Bottom line:** Three square periodic UMFs with $n = 4$ and sizes (from left to right) 14×14, 20×20 and 77×77.

In the paper "Uniform Marker Fields: Camera Localization By Orientable De Bruijn Tori" [II] in proceedings of the *IEEE International Symposium on Mixed and Augmented Reality in 2012* (ISMAR 2012) we introduce the concept of the Uniform Marker Field (UMF). This particular arrangement of the marker field is a black-and-white checkerboard whose square modules are defined as aperiodic 4-orientable binary n^2-window arrays.

Our solution is based on 4-orientable n^2-window arrays [48] – two dimensional arrays of binary values, where each $n \times n$ sub-window is present only once, including all 4 possible rotations (see Section 5.1 for more information).

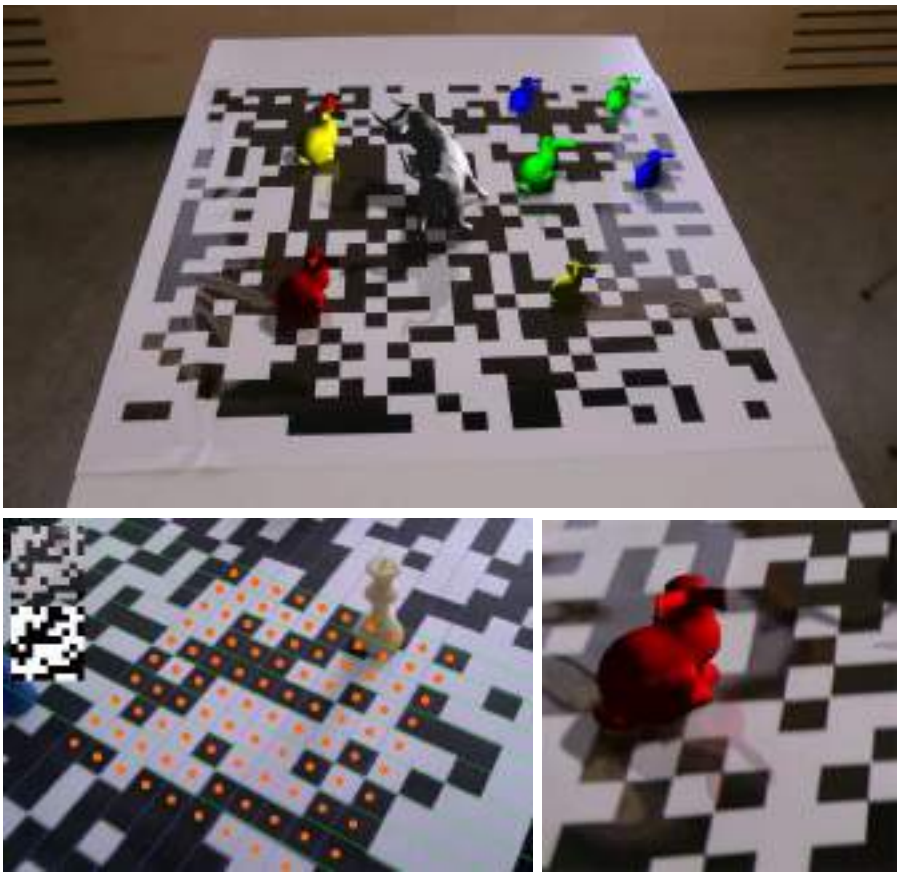


Figure 48: **Top:** The augmented scene over the real-life UMF. All the animals are augmented to the scene. **Bottom left:** The final step of the marker detection. Pixels from the input image under the orange points are sampled and matched with the marker. **Bottom right:** The detail of the augmented scene.

Major contribution is design of marker field with such properties, second contribution is that we propose a genetic algorithm for generation of such maps and we make a number of generated maps publicly available. The third major contribution is an efficient algorithm for detection of checker-board structures – such as the orientable window arrays – in a camera image. This algorithm is designed so that it can be executed on computational platforms with limited resources – such as the mobile phones and CPUs embedded in smart cameras.

UMF is uniform in the sense that the features used for detection and localization of the field and the features used for identification of individual windows (markers) within the field are uniformly distributed across the whole area of the marker field.

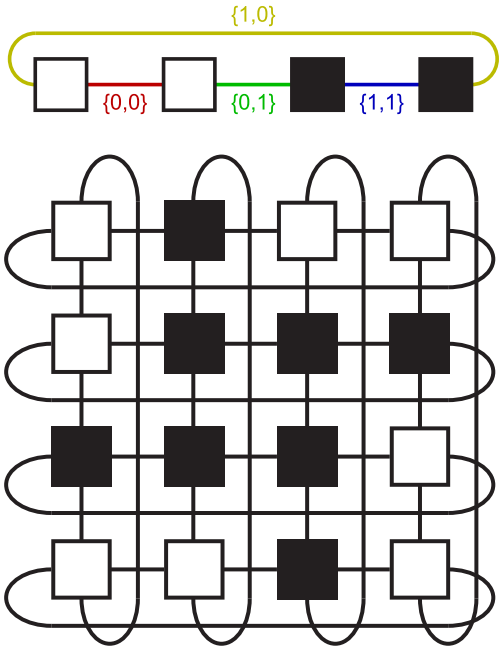


Figure 49: **Top:** One dimensional De Bruijn Sequence for $k = 2$ and $n = 2$, alphabet is $\{0,1\}$. **Bottom:** De Bruijn Torus $(4,4;2,2)_2$ where each possible 2×2 binary matrix can be found exactly once.

5.1 ORIENTABLE WINDOW ARRAYS AS UNIFORM MARKER FIELDS

This section reviews information about the orientable window arrays and discusses their usability as the UMF.

Periodic, aperiodic, orientable window array and perfect maps.

An aperiodic (m,n)-window array [48] is an k-ary 2D array of size $h \times w$

$$A = (a_{i,j} \in \{0, \cdots, k-1\}; 0 \leq i < h; 0 \leq j < w), \tag{6}$$

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 [49]. 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 is not enough. It is possible that multiple rotations of the same window can occur in the array. Orientable window arrays [48] solve this problem. Orientability can be gained by using certain equivalence relation for the window property [50].

- **1-orientable** arrays are ordinary (m,n)-window arrays defined earlier.

- **2-orientable** arrays deal with two possible orientations (e.g. "north" vs. "south" orientation). Windows in the 2-orientable perfect maps are unique in respect to rotation by 180°.
- **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 [50].

Contrary to the 1-orientable maps, 2 and 4-orientable arrays are much less explored in the literature and no good construction algorithms exist for them.

Upper bound for 2-orientable window array.

The upper bound for the size of the 2-orientable (m, n)-window array is

$$N \leq \frac{k^{mn} - k^{\left\lfloor \frac{mn+1}{2} \right\rfloor}}{2}, \tag{7}$$

where N is the window count, i.e. $N = hw$ for periodic and $N = (h - m + 1)(w - n + 1)$ for aperiodic arrays. For the 4-orientable (n, n)-window arrays (n^2-window arrays),

Upper bound for 4-orientable window array.

$$N \leq \frac{k^{n^2} - k^{\left\lfloor \frac{n^2+1}{2} \right\rfloor}}{4}. \tag{8}$$

It is known that these size bounds are not tight [48]. It is not known whether any orientable perfect map exists, i.e. the inequalities may be strict.

For 2-orientable 1D window arrays, the upper bound of the size is slightly more refined, but still not tight [51]. The sequences for $n \leq 16$ were found by brute force [48].

Only binary ($k = 2$) window arrays were considered in the first UMF paper, because they are easily printable and they can be easily and robustly detected in greyscale camera images. However, all the algorithms can be easily modified for higher values of k. See Chapter 6 for our work on this possibility.

5.2 SYNTHESIS OF n^2-WINDOW ARRAYS

In this section the novel algorithm for generation of 4-orientable n^2-window arrays is described.

According to Equation (8), for binary 4-orientable aperiodic n^2-window arrays with $n = 3$, the (square) map cannot be larger than 12×12 (our algorithm described in this section has found a number of 11×11 arrays). Thus, 3^2-window arrays can be used as UMF, but the dimension of the field is very limited and the benefits over any existing marker system are not very interesting. However, even for such a small array, the construction by exhaustive search is

Brute force search is impossible.

practically impossible (for a 12×12 array, the algorithm would need to search a state space of 2^{144} variants).

For the proposed UMF, $n = 4$ is an interesting value: n is still small enough and therefore a small fraction of the field needs to be captured by the camera in order to localize the actual view within the field. At the same time, the theoretical upper bound according to the Equation (8) for the dimensions of a square map 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. Also, rectangular arrays of similar $w \times h$ areas have been generated – refer to Section 5.2.3 for more information.

Window array with $n = 4$ is most practical; small enough for camera with up to 127×127 map size.

For $n = 5$, the arrays of reasonable dimensions can be generated (more or less) randomly; however, the required portion of the marker field to be visible by the camera (5×5 modules) is unnecessarily high. For practical purposes, the 4×4 window is therefore of most interest and in the further text, the algorithm's properties will be explored for these window dimensions.

The literature does not provide any construction method for 2 or 4-orientable n^2-window arrays. Burns and Mitchell [48] provide a baseline construction for general 4-orientable window arrays. However, this construction is far from optimal (i.e. for a given n, the w, h dimensions are small) and, more importantly, the constructed map cannot be binary, but its arity is given by combining two helper 1D 2-orientable De Bruijn sequences. 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}).

This section presents considerations leading to a genetic algorithm for synthesis of 4-orientable window arrays. The algorithm is still very computationally demanding; however, it can be parallelized (as described in Section 5.2.2) and executed on a large number of computation nodes. We harnessed a supercomputer of around 1 000 nodes and generated usable window arrays.

5.2.1 RANDOM MAPS

Our problem with conflicting windows is similar to the birthday problem.

Let us first consider a binary map ($a_{i,j} \in \{0, 1\}$) generated randomly. The probability that a randomly generated periodic array will contain 1-orientable conflicting windows can be approximated as the birthday problem

$$p(k, h, w, m, n) \approx 1 - \exp\left(-\frac{(hw)^2}{2k^{mn}}\right). \tag{9}$$

For 2 and 4-orientable arrays with square windows, the birthday conflict probability is

$$p(k,h,w,n) \approx 1 - \exp\left(-\frac{o(hw)^2}{2k^{n^2} - 2k^{\left\lfloor \frac{n^2+1}{2} \right\rfloor}}\right), \tag{10}$$

where $o \in \{2,4\}$ is the orientability. This equation neglects a significant probability that some windows can be self-conflicting. The probability that an array contains self-conflicting windows is

$$p(k,h,w,n) = \left(1 - k^{\left\lfloor \frac{n^2+1}{2} \right\rfloor - n^2}\right)^{hw}. \tag{11}$$

Figure 50 shows the probability of conflict for square aperiodic arrays with $k = 2$ and $n = 4$. The experimental measurement was done using 1000 randomly generated arrays. The number of conflicts caused by self-conflicting windows is a significant factor. The graph shows that arrays of small sizes can

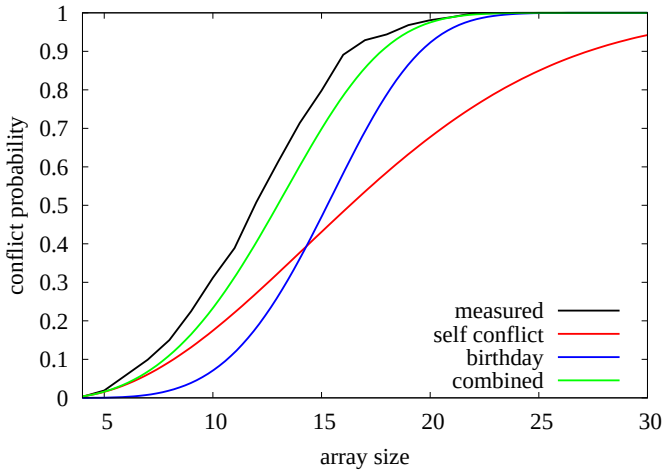


Figure 50: Conflict probability in a random square map of a given size ($k = 2$, $n = 4$, 4-orientable).

be generated randomly and chances are that they will be free of any conflicts in the 4-orientable sense.

For larger maps, conflicts are inevitable. However, a conflicting window can be replaced by an alternative content. For a randomly generated n^2-window in a periodic map, otherwise free of conflicts, the probability that the window does not conflict with the rest of the map is roughly

$$p(k,h,w,n) \approx \left(1 - o\frac{hw - (2n-1)^2}{2^{n^2} - 2^{\left\lfloor \frac{n^2+1}{2} \right\rfloor}}\right)^{(2n-1)^2}. \tag{12}$$

The internal conflicts in the block around the window are approximated using Equation (10). Figure 51 shows the probability that the new window will be conflicting after a given amount of tries.

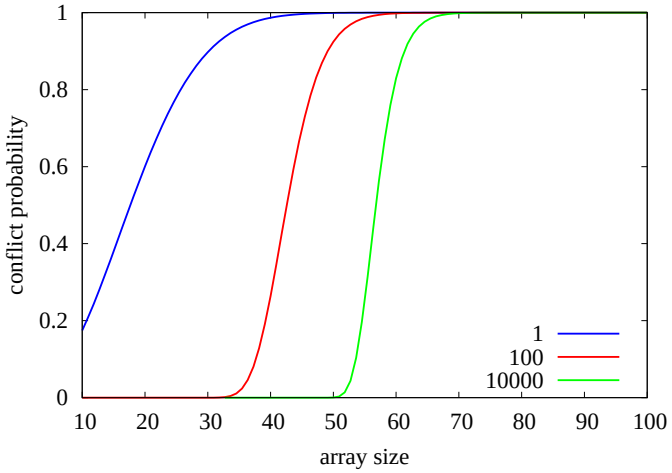


Figure 51: Approximate probability that a new window will be still conflicting after N trials of replacement ($k = 2$, $n = 4$, 4-orientable).

Small arrays can be generated randomly, large arrays can be "fixed" after reasonable count of replacing conflicting windows.

Figures 50 and 51 illustrate the possibility of generating the window array randomly. For $k = 2$ and $n = 4$, small arrays (around 15×15) can be generated totally randomly and some of the generated arrays would comply with the 4-orientable n^2-window property. Random arrays of dimensions around 50×50 can be fixed by replacing the conflicting windows with random content; after a moderate number of such trials for replacement, the array will achieve the n^2-window property. However, for arrays of higher dimensions (e.g. 90×90), the process of fixing the array by replacing the conflicting tiles can be very lengthy.

5.2.2 GENETIC ALGORITHM FOR SYNTHESIS OF WINDOW ARRAYS

The considerations from the previous section lead to a genetic algorithm which works with maps containing conflicts and improves it continually by mutations that lead to decreasing the conflict count. Such an individual can be generated randomly as a whole, or it can arise from a smaller window array extended by randomly generated rows and/or columns.

Rules of genetic algorithm.

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 is based on the number of conflicts in the given array $f(A) = \frac{1}{c(A)+1}$, where $c(A)$ is number of conflicts.

- 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.

In order to improve the algorithm's convergence we made a few modifications to the aforementioned general approach. In the mutation step, the algorithm discards from random selection all windows that are already in the map or that would cause more conflicts then the current count. The other changes to the algorithm were carried out in order to further improve the speed of convergence on the proposed client-server configuration.

Parallelization is possible to speed up the generation.

In order to distribute calculations required to solve the conflicts in randomly generated arrays, a client-server architecture can be used. The server keeps track of the population, while the clients query the server for computational tasks and send back the results.

The server stores the active population and distributes tasks for mutations to the clients. The server itself is state-less in the sense that it does not keep track of the clients and their status.

The most important change from the described genetic algorithm is that the server does not separate arrays into several generations, but works with a single population and updates it incrementally. This is necessary because the run-time of the clients might differ dramatically, and clients might post changes into different generations.

As mentioned earlier, each client queries the server for a new task after it is started or when it finishes a task. The client receives a single array and tries to solve conflicts in the array by mutation: i.e. by replacing a conflicting window with a better combination of modules. If it successfully finds a different array with the same or smaller number of conflicts, it sends the array back to the server.

5.2.3 SYNTHESIZED WINDOW ARRAYS

We generated a set of binary aperiodic 4-orientable n^2-window arrays. We make the window arrays available for public use on a web site[1] and as a part of the supplementary material of this work. The data set includes rectangular maps of different aspect ratios:

- **1:1** – square marker fields,

1 http://www.fit.vutbr.cz/research/groups/graph/MF/

- **$\sqrt{2}$:1** – marker fields suitable for office paper sizes (A series, i.e. A5, A4, A3, etc.),
- **2:1** – marker fields for rectangular areas,
- **3:1** – marker fields for wide rectangular areas.

Table 3 gives the highest resolutions of the window arrays available for the respective aspect ratios at the moment of the paper submission.

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

Table 3: Available sizes of the binary 4-orientable aperiodic 4^2-window arrays.

5.3 EXPERIMENTAL RESULTS

For testing purposes we 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. Smartphones used for collecting the data were:

- **Desire** – HTC Desire GSM (720×480)
- **Evo** – HTC Evo 3D (720×480)
- **Titan** – HTC Titan (640×480)

In order to evaluate the detection precision for different types of movements, we split the dataset into 6 categories according to the dominant movement manifested in each video:

- **zoom** – zooming in and out on the marker field
- **horizontal** – horizontal movement over the marker field
- **rotation** – rotating the camera over
- **perspective distortion** – videos where the marker field is skewed by perspective projection
- **general movement** over the marker including several of the above mentioned distortions
- **occlusion** – objects covering some parts of the marker field

Each category contains 6 videos (2 for each smartphone) of markers with different densities. All the marker fields were printed out black and white on standard A4 office papers; the low-density marker field's resolution is 14×10,

the high-density marker field was 28×19. The total number of videos in this dataset is 36. Example frames from the video dataset are shown in Figure 52.

In the first set of experiments we evaluated the success rate of detecting the checker-board marker field and recognizing a location within it. To estimate a baseline of the algorithm's robustness we did not use information from previous frames. Using such information in real applications should improve the precision of the algorithm as well as improve the processing speed.

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 movement	95.3 %	95.0 %
Occlusion	91.9 %	92.5 %

Table 4: Success rate for detecting the position in the marker with different categories and marker densities.

Table 4 contains the success rate results for our dataset. By a successfully recognized frame we mean a frame, where the detector was able to find the grid, sample the $\mathbf{x}_{ij}$ points and the resulting binarized bitmap (or its significant part) can be found in the reference marker array. The results show that our algorithm performs well even for very challenging videos (see the dataset) with rapid movement causing directional blur, rotation, and high perspective distortion. The low-density marker performed better for almost all categories. This is caused by the longer edges, hence more precise edgels. In the case of the occlusion category, the higher density marker still contains enough unoccluded fields to enable localization, unlike in some cases in the lower density marker. The difference for the horizontal category is caused by a statistical error caused by rapid movement in one of the videos.

Smartphone	Success rate	Visited pixels
Evo	95.3 %	5.50 %
Titan	96.3 %	5.79 %
Desire	97.4 %	5.68 %
Marker density	**Success rate**	**Visited pixels**
Low	96.5 %	5.83 %
High	96.1 %	5.59 %

Table 5: Success rate and percental evaluation of the visited pixels in the image by our algorithm.

Table 5 contains the success rates for the videos captured by the smartphones. Smartphone Titan gave the sharpest image and best picture quality, which helped improve the success rate. The videos captured by smartphones Evo and Desire contain an intensive motion blur which caused a drop in the performance. The results also show that the algorithm visited and used for computation only less then 6 % of the total number of pixels in the image. Since one of the slowest operations on mobile processors is randomly accessing large chunks of memory, our algorithm is well suited for smartphones.

Table 5 also shows the difference in the visited pixel percentage for marker fields with different densities. Since the edges in the image are longer in the low density markers, the algorithm visits more pixels along these edges. This helps the precision of the edgels and also the performance.

In order to get a comprehensive picture about the computational complexity of the algorithm we measured the required time of different components of the detection algorithm. Table 6 shows the percental distribution of computational time between different components. The most time-consuming part of the

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	0.74 ms	55 %
Overall	1.34 ms	100 %

Table 6: Breakdown of computational time into different parts of the algorithm.

algorithm is camera localization as implemented in the OpenCV library. The times were measured on a Intel Core i5 661, 3.3 GHz with a DDR2 memory.

Figure 52: Sample images from the dataset taken with HTC Desire GSM (720×480), HTC Evo 3D (720×480) and HTC Titan (640×480) smartphones. Orange dots are points $\mathbf{x}_{ij}$ sampled from the input image.

6

IMPROVED UNIFORM MARKER FIELDS

AKA SHADES OF GREY OR COLOR

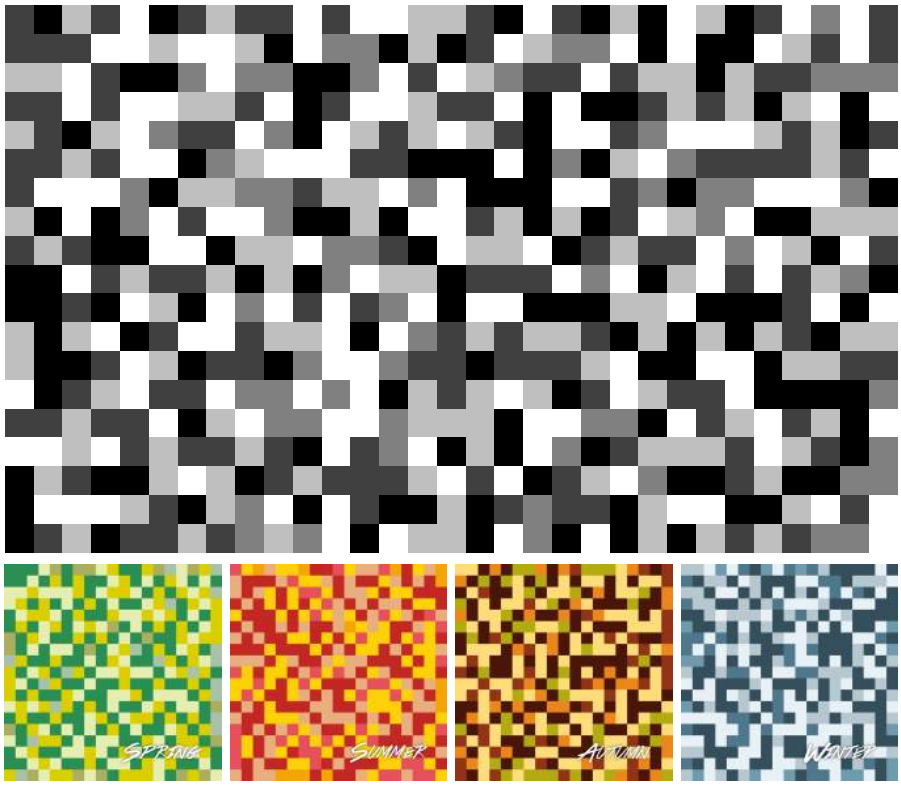


Figure 53: Example of improved UMF design. **Top:** Shades of grey UMF used for making the carpet in Chapter 12, size is 31×19 modules with $n = 3$, $k = 5$. **Bottom:** Series of UMFs inspired by colors of four seasons of the year.

Exploring the possibility of UMF with higher arity.

In the paper "Five Shades of Grey for Fast and Reliable Camera Pose Estimation" in proceedings of the *CVPR 2013* we extend the idea of Uniform Marker Fields (UMFs) by increasing the arity of the markers. That means modules are no longer just black and white but different number of shades of grey (or colors) are used; it offers more edges in the marker field to be detected and, at the same time, a smaller window of the De Bruijn torus is necessary for identifying a unique location, they are more aesthetically appealing, invariant to high degrees of perspective distortion and to varying lighting conditions (direct light, shadows, different lighting intensities).

Figure 54: The use of Shades of Grey (SoG) marker field. **Top:** Original image – input to the detector. **Bottom:** Recognized camera location and augmented scene.

We present an algorithm for the detection of the greyscale grid of squares. A unique location in the marker field is identified by the edges between the marker field modules (Figure 55). These edges need to be reliably classified – Wald's Sequential Probability Ratio Test (SPRT) [46] (similar as in Fractal Marker Fields (FMFs)) is used in order to sample a minimal number of pixels for discerning the edge.

As mentioned above we work with greyscale or color k-ary marker fields ($a_{ij} \in \{0, \ldots, k-1\}$, Figure 55). However, in comparison with binary marker fields, explained in Chapter 5, the absolute greyscale or color values of the grid modules cannot be reliably discerned under varying lighting and camera conditions. That is why we use the edge gradients between the modules for

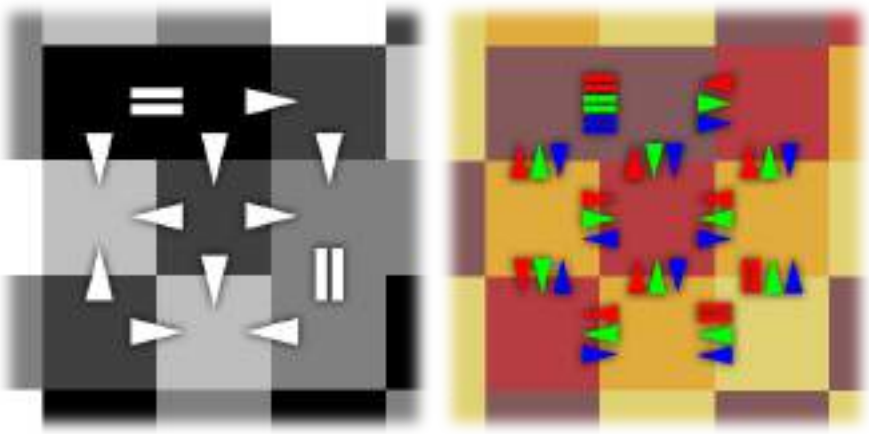


Figure 55: A fragment of the marker field. **Left:** Five shades of grey. **Right:** 8 different colors. Arrows across the edges illustrate the observable gradient which describes an individual line. In color, multiple gradients can be observed at one edge (e.g. RGB).

localization within the marker field. Horizontal (13) and vertical (14) edge gradients are defined as:

$$e^{\rightarrow}_{ij} = a_{i,j+1} - a_{ij}, \tag{13}$$

$$e^{\downarrow}_{ij} = a_{i+1,j} - a_{ij}. \tag{14}$$

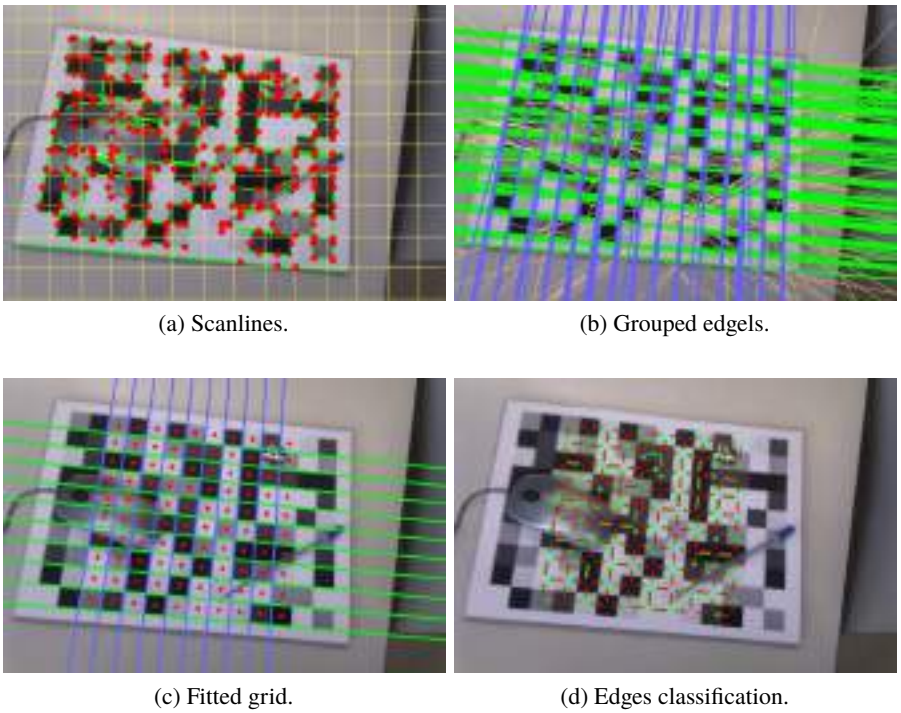
The absolute value of the edge gradient is also hard to recognize reliably and thus only the basic character of the edge is used for recognition: $\operatorname{sgn} e^{*}_{ij} \in \{-1, 0, +1\}$. The n^2-window used for localization within the marker field then is (Figure 55):

$$E_{rc} = (e^{\rightarrow}_{rc}, \ldots, e^{\rightarrow}_{(r+n-1,c+n-2)}, e^{\downarrow}_{rc}, \ldots, e^{\downarrow}_{(r+n-2,c+n-1)}) \tag{15}$$

Fitness function of genetic generator also takes "quality" of the edges into account.

Synthesis of the marker field is done in a manner very similar to the genetic algorithm sketched out in chapter 5. In this case, the fitness function must also reflect the quality of edges between the modules – edges with higher absolute value $|e_{ij}|$ are preferred. The algorithm first looks for conflict-free fields where each sub-window exists only once, including all four possible rotations. Conflict-free fields are further optimized for maximal contrast on the edges between the square modules. As with UMFs we used a cluster of computers to synthesize the marker fields and we make them publicly available on website[1] and part of the supplementary material of this work.

1 http://www.fit.vutbr.cz/research/groups/graph/MF/



(a) Scanlines. (b) Grouped edgels.

(c) Fitted grid. (d) Edges classification.

Figure 56: Detection of the Shades of Grey (SoG).

6.1 DETECTION & RECOGNITION OF PLANAR GREYSCALE GRIDS

This section describes the algorithm for detection of the greyscale checkerboard-like marker field. This algorithm supposes that the grid of squares is planar and projected by a perspective projection. The experiments (Section 6.2) show that this condition is fulfilled enough in realistic scenes observed by standard cameras. 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 (Section 6.2.2).

6.1.1 GREYSCALE GRID DETECTION

The algorithm performs the following three main steps (Figure 56):

1. Extraction of edgels (edge element or edge pixel; term borrowed from Martin Hirzer [52]) – typically, the algorithm extracts around one hundred straight edge fragments in the whole image. Exact number depends on a lot – resolution, scanline density, blurriness etc. The image is processed in sparse horizontal and vertical scanlines (Figure 56a). When a video input is being processed, the detected edgels are filtered based on the previous detected position of the marker field. In the tests we used a simple rectangular mask to

filter out the edges outside the area corresponding to the previously detected marker field.

2. Determining two dominant vanishing points among the edgels (Figure 56b). Using homogeneous coordinates for the vanishing point $\mathbf{v}$ and the pencil of lines $\mathbf{l}_i$, all the lines are supposed to be coincident with the vanishing point, i.e.

$$\forall i \quad \mathbf{v} \cdot \mathbf{l}_i = 0. \tag{16}$$

The coordinates of the lines in the real projective plane form a 3D vector space without an origin (with an equivalence relation). Points of the real projective plane correspond to hyperplanes passing through the origin, so the vanishing point can be found by fitting a hyperplane through all the lines (extended edgels) observed in the pencil. The line vectors $\mathbf{l}_i$ are scaled so that each one's magnitude corresponds to the edgel length. In this way, the longer and more reliable edgels are favored. The hyperplane's normal is found as the direction of the least variance by eigen decomposition of the correlation matrix

$$C = (\mathbf{l}_0 \dots \mathbf{l}_N)(\mathbf{l}_0 \dots \mathbf{l}_N)^T. \tag{17}$$

Since matrix C is 3×3 and symmetric, decomposition can be computed very efficiently.

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 ($|\mathbf{x}|$ denotes normalized vector)

$$\mathbf{l}_i = |\mathbf{l}_{base}| + (ki + q)\,|\mathbf{h}|, \tag{18}$$

where $\mathbf{l}_{base}$ is an arbitrarily chosen base line through the vanishing point, different from the horizon [53]. Parameter k controls the line density and q determines the position of the first line. A good simple choice for $\mathbf{l}_{base}$ is a line through the center of the image (and through the vanishing point).

In order to find k and q, the value of $(ki + q)$ is calculated for every line (extended edgel) of the input group. These values are clustered by simplified mean-shift and median difference between cluster candidates. The mean-shift box kernel size with normalized image coordinates in our tests was $w = 0.05$. Each cluster is assigned an i and then overall optimal k and q are found by linear regression (Figure 56c, blue and green lines).

For simplicity, the algorithm description supposes that a significant portion of the input image is covered by the marker field. However, steps 2 and 3 of the algorithm are conditionally applied on rectangular parts of the image (quarters, ninths); in high-resolution images, the marker field is thus found even if it covers an arbitrary fraction of the camera input.

6.1.2 EDGE CLASSIFICATION

When only a small fraction of the marker field is visible, it is crucial that the edge gradients (Equations (13) and (14)) are recognized correctly. Their recognition can be challenging due to motion blur, uneven lighting conditions, etc. Some problematic edges can be seen in Figure 57.

Figure 57: Examples of problematic edges within the pictures of the marker field. The edges are classified by deterministic sampling a varying number of pixels within the neighboring modules. Wald's SPRT is used to discern the edge by using a minimal number of such samples.

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 a small number of samples suffices to decide an edge either way ($-1, +1$; Figure 55), the decision is made, otherwise more pixels are sampled. If an edge cannot be confirmed, the location between the modules is treated as a place without an edge: $e^*_{ij} = 0$ (Figure 56d). The stopping criterion is given by Wald's SPRT [46], which is proven to be the optimal sequential test for this purpose.

6.1.3 LOCALIZATION WITHIN THE MARKER FIELD

The sub-window described by edges E_{rc} is formulated as a vector of scalars in Equation (15). This vector can be used as a key to a hash table. Values in the table represent locations in the marker field (two discrete coordinates in the terms of grid modules; enumerated orientation $0°/90°/180°/270°$). An absent record in the hash table means a wrongly recognized fragment of the marker field. Hash tables are implemented fairly efficiently in today's programming languages.

Instead of using a readymade hash table, we prefer to create a decision tree. When a compact piece of the marker field is detected in an input image, the edges are classified and used for traversing the tree. A central edge in the detected cluster of edges decides the root node, and surrounding edges follow in a predefined order (Figure 58). Any cluster of neighboring edges is recognized by the tree – the leaf node would either define the cluster's location and orientation within the marker field or reject the cluster of edges as invalid (due to misdetection). Constructing a deeper tree implies that a larger cluster

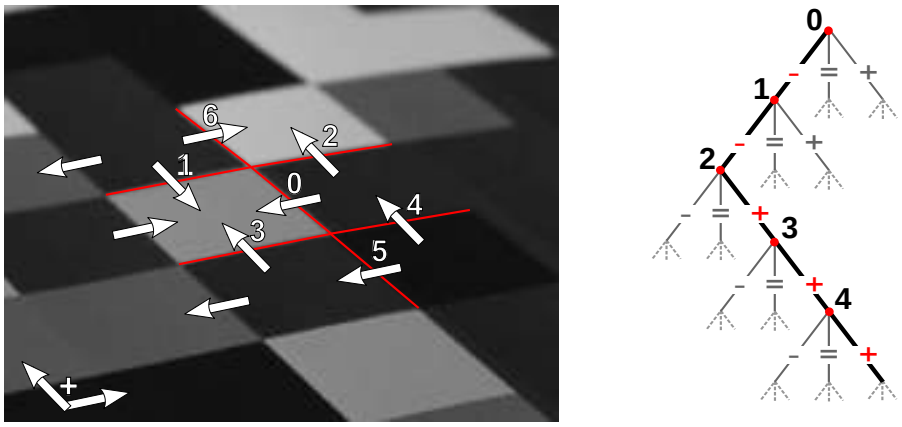


Figure 58: The decision tree used for localization in the marker field. **Left:** A compact cluster of edges detected in the image. Edges are numbered in a predefined order relative to a selected root edge (0). **Right:** Decision tree – the leaves are either invalid or contain a location + orientation to the marker field.

of edges is used for localization within the field. This allows for larger marker field resolutions. 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.

6.1.4 CORNER SEARCH AND ITERATIVE REFINEMENT

For a precise camera pose estimation we find all possible corners in the marker field (with a sub-pixel precision). The corners of the grid of squares are projected from the detected overall position and iteratively searched for in the neighborhood. Based on the marker field layout, the algorithm knows each corner's appearance including its rotation and searches for such a particular pattern. This helps mostly in cases when the image is motion blurred, the marker is not perfectly planar, or noise in the edgel data cause the grid not to fit the edges precisely. Another way of improving the precision of the pose estimation accuracy is to iteratively search for correct corners in the marker field in the image space using back-projection. In the tests we use both of the aforementioned improvements.

6.2 EXPERIMENTAL RESULTS

We compare our solution to ALVAR [25] as the most mature available AR-ToolKit follower supporting arrays of disjointed square markers. The other baseline is the Random Dot Marker (RDM) as an alternative "marker field" solution, where individual localization markers overlap in the field.

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 7: The average variance in position and rotation change using 10 frames for averaging in a 1080p 50 FPS 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 the resulting variance was 0.080).

For comparing our solution with the alternatives, we shot videos of side-by-side markers (Figure 59). The marker fields have comparable (as much the same as possible) dimensions and resolution of the individual markers (n^2-windows vs. ALVAR individual markers vs. RDM's sub-markers) and the movement is simple and well-defined to ensure fairness in the comparison.

6.2.1 SUCCESS RATE AND PRECISION

In order to evaluate the precision of our algorithm we used the local variance (in time domain) of the estimated camera pose (position and rotation) (Table 7). Low local variance means that the results of camera localization are smooth. We did not include RDM, since its stability was notably worse (Table 8) and ALVAR thus serves as a good reference for precision evaluation. Our method gave smoother results thanks to the good spatial distribution of matched points between 2D and 3D and ALVAR's inability to find the corners of the individual markers precisely in blurred images and for partially occluded corners. The number of detected corners used for the camera pose estimation is shown in Figure 60.

Apart from the precision we also show in Table 8 the success rate for each method in every category of videos. RDM were the least successful, mostly due to their high sensitivity to the motion blur. But even for a fixed camera or rotation with minimal blurring it gave the worst results. ALVAR and UMF were both very successful and gave very similar results. The only major difference was for zooming and occlusion. Figure 59 shows the clear advantage of our continuous marker field over ALVAR. ALVAR only detected two completely visible markers and one which had one edge slightly occluded. On the contrary, our method was able to detect sub-markers and corner points even between the cups.

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 8: Marker field detection success rates in %. For UMF, rates from comparison videos with RDM and ALVAR are given separately. Success rate is the fraction of video frames where at least one of the markers was correctly detected in all the video frames.

6.2.2 PIXEL FOOTPRINT AND COMPUTATION COMPLEXITY

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 9: **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.

Table 9 shows the breakdown of speed of the three tested algorithms and the breakdown of speed of our marker detection algorithm for 1080p videos using a mid-range Intel(R) Core(TM) i5 CPU 661 (3.33GHz) CPU. Our algorithm was more than 3× faster than ALVAR and visited on average about 5.3 % of all pixel points. A small memory footprint is an important property for ultra-mobile processors where the memory accesses are slow due to limited caching, etc.

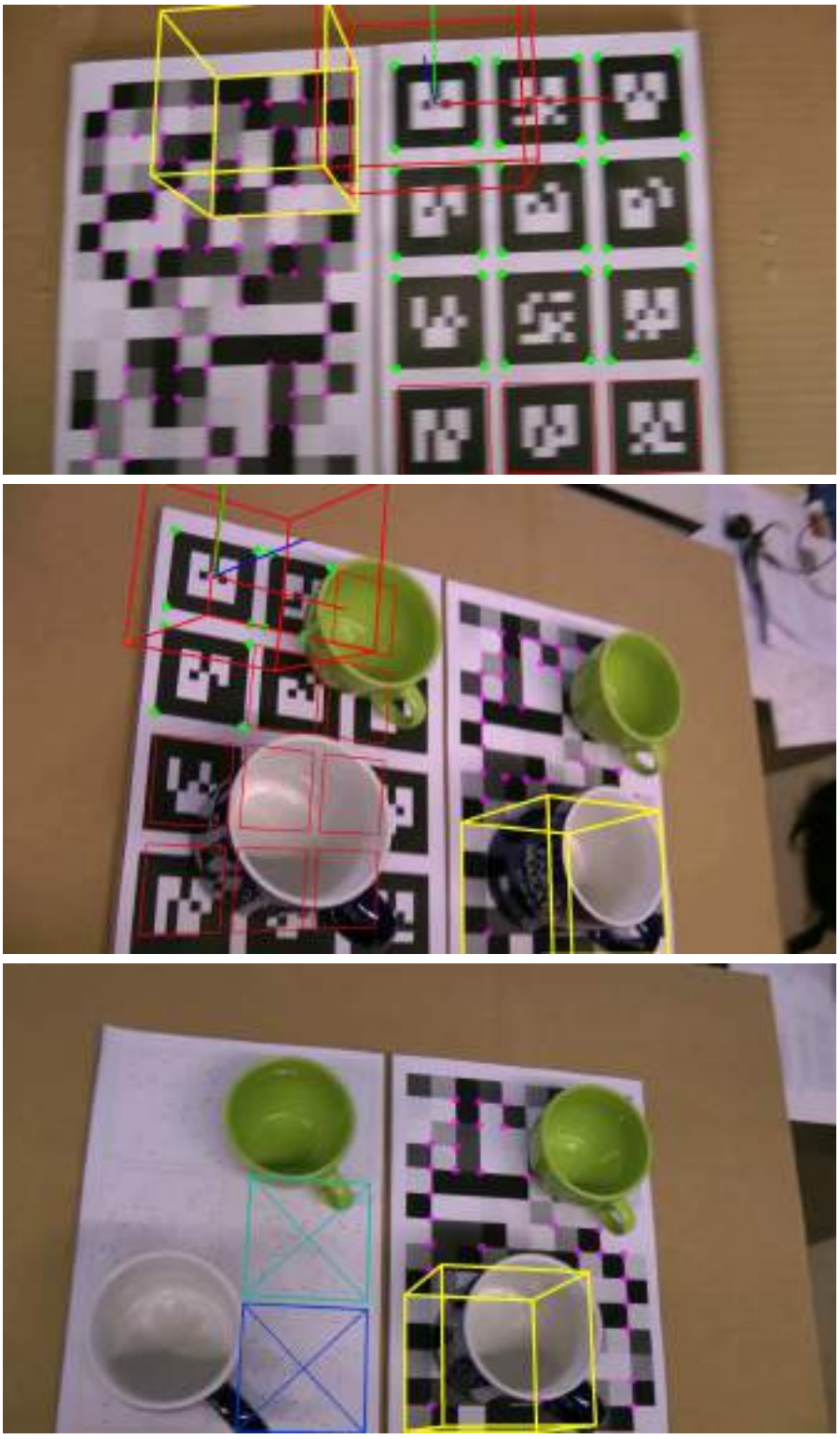
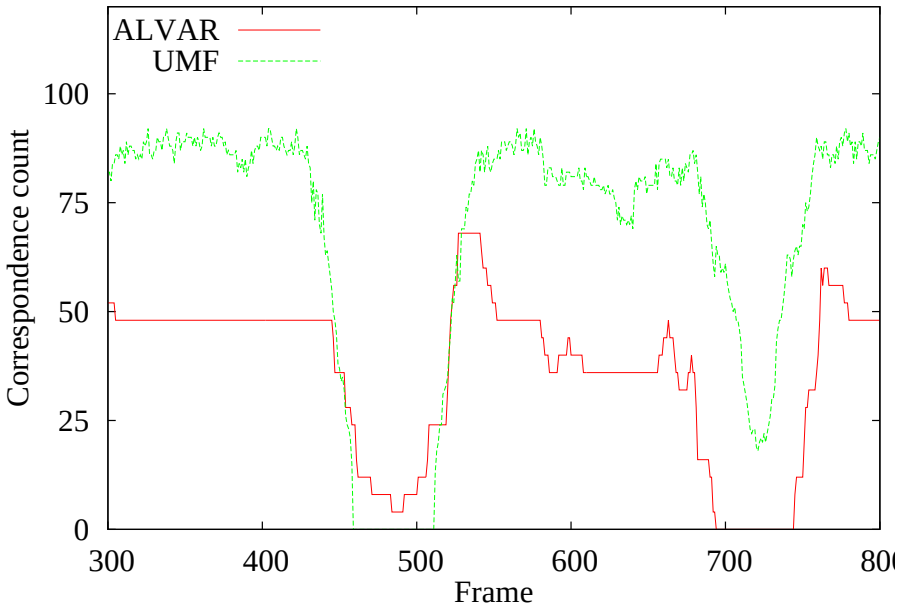
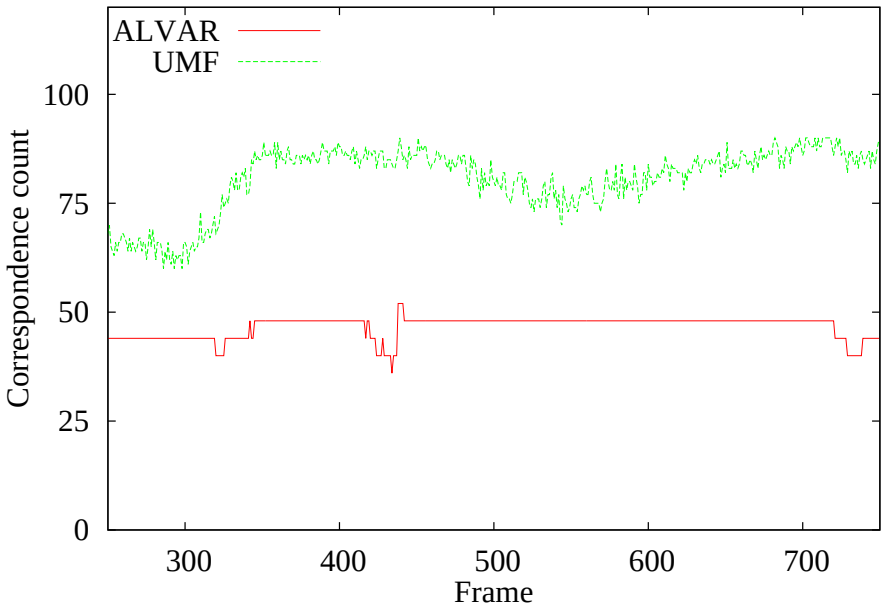


Figure 59: Illustrative frames of the side-by-side comparison video. **Top:** SoG vs. ALVAR motion blur test. **Middle:** SoG vs. ALVAR occlusion test. **Bottom:** SoG vs. RDM occlusion test. Videos were recorded in 1080p, capturing different classes of movement: zig-zag movement, upright rotation, rotation with severe perspective distortion, near/far movement, variable lighting conditions with fixed camera and general movement with occlusion.



(a) Near-far video.

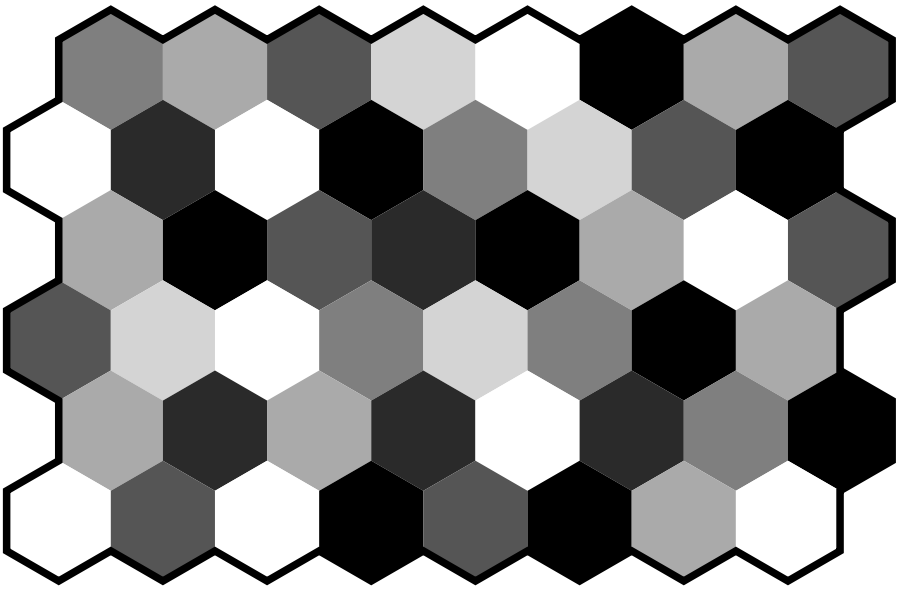


(b) Perspective video.

Figure 60: Number of correspondences used by ALVAR (red) and UMF (green) for the camera pose estimation. More points generally mean a more stable and precise camera pose estimation and better tolerance to wrongly detected points (caused by a motion blur, partial occlusion, etc.).

7

HONEYCOMB MARKER FIELDS



In 2013 we introduced the Honeycomb Marker Fields (HMFs) in our paper "Design and Detection of Local Geometric Features for Deformable Marker Fields" which is now in proceedings of *29th Spring Conference on Computer Graphics (SCCG)*.

HMF is first deformable marker field.

The biggest limitation of Fractal Marker Fields (FMFs) (Chapter 4), Uniform Marker Fields (UMFs) (Chapter 5) and improved Uniform Marker Fields (UMFs) (Chapter 6) is the requirement for the marker field to be planar. The planarity assumption allows for the detection algorithm to be very efficient. However, for some applications, it is restricting. We designed marker field that would be detectable even for non-planar and generally deformed surfaces. In particular, we designed geometric features which could form a graph usable for identification of fractions of the field, and which would be detectable very efficiently.

In 2011 Uchiyama and Marchand introduced Deformable Random Dot Marker (DRDM) in paper "Deformable random dot markers" [54]. RDM and DRDM are described in Sections 3.6 and 3.9, respectively. DRDM have several disadvantages. The recognition is sensitive to the size of the points, motion blur and focus of the camera, since it relies on detecting distinct points using adaptive threshold. They identify the marker based on the relative position of the points, which only holds for planar surfaces. Hence, this limits the

Figure 61: A camera snapshot of the deformed marker field in a realistic environment.

deformability of the marker. DRDM are only usable on non-elastic surfaces. These properties lead to unstable detection and tracking.

HMF consists of symmetrical hexagons, not squares like previous marker fields.

We propose to compose the marker field of symmetrical hexagons; the marker field then looks like a honeycomb. In this way, triplets of modules meet at junctions shaped like the "Y" letter. These Y-junctions form distinct image features, connected by edges into a graph representing the marker field. We also propose an algorithm for efficient detection of the Y-junctions.

We cannot detect straight lines anymore, therefore we make use of our Y-junction detector.

Thanks to their specific appearance, the Y-junctions can be detected by visiting only a small fraction of the image pixels. The consequence is that the algorithm is very efficient – much faster than various detectors of natural image features. At the price of using a synthetic marker field instead of tracking natural features, our solution provides very fast and robust detection and identification of fractions of the marker field. These fractions can be reliably identified and can serve for real-time and precise camera pose estimation. Contrary to the existing marker field design, the honeycomb marker field is not required to be planar, but it can be considerably bent or otherwise deformed.

7.1 DESIGN & DETECTION OF HMF

Conventional marker detectors detect square or circular patterns in the image and compute the camera pose individually for all markers. Marker field detectors, on the other hand, combine the results for individual markers (like in case of FMF or ARTag) or try to find a global solution (UMF or RDM) assuming that the marker field is planar. The main problem with these approaches is that the whole marker field or its sub-markers have to be planar.

Figure 62: Identification of a fragment of the marker field based on the intensity differences between hexagons.

HMFs are designed so that they are tolerant to high degree of deformation – non-planarity. These marker fields are designed to be detectable and reconstructable by particular feature points – the Y-junctions. They appear between hexagonal symmetric cells of different colors (shades of grey) (Figure 61 top). Since the intensities (colors) of the three neighbouring hexagons are different, the feature points are robust enough to detect under variable lightning conditions.

The marker field is synthesized so that small overlapping fragments are unique (including all 6 possible rotations). Then, only a portion of the whole marker field is sufficient to be detected and to identify a location within the field (Figure 62).

7.1.1 HONEYCOMB MARKER FIELD DETECTION

The whole detection is proposed to consist of the following steps:

1. **Input image prescanning** (Figure 63) is done by a hierarchical grid which subdivides square cells based on the detected edges on their sides. This step is able to efficiently find important blocks in the image and filter out unneeded parts. This step is discussed in Section 7.1.2.

2. **Detection of the Y-junctions** (Figure 64). The Y-junctions are detected using a four-orientable square shaped filter. For each detected Y-junction, the directions of the incident edges are also estimated. This step is described in detail in Section 7.1.3.

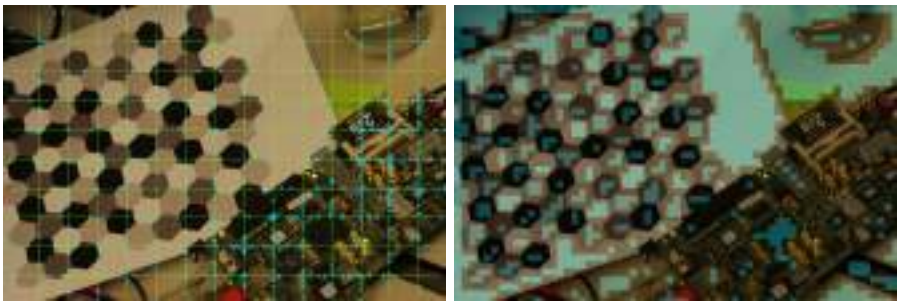


Figure 63: Grid scanning of the input image. **Left:** Input image processed in sparse scanlines. The image shows the first level of the processing. **Right:** Final output of the scanline grid. Those cells that cannot contain Y-junctions are removed from further processing.

3. **Reconstruction of the HMF grid topology** is done by connecting the couples of nearest Y-junctions. To find effectively adjacent junctions by not comparing all to all, a uniform grid is used which can reduce the complexity of the problem.

4. **Hexagon identification** is done by comparing the edge gradients (Figure 62) between a selected central hexagon and its adjacent neighbours (or a wider surroundings). The intensities form a binary code which uniquely identifies the position of the central hexagon in the marker field and its orientation. This step of the detection is sketched out in Section 7.1.4.

7.1.2 IMAGE PRE-SCANNING

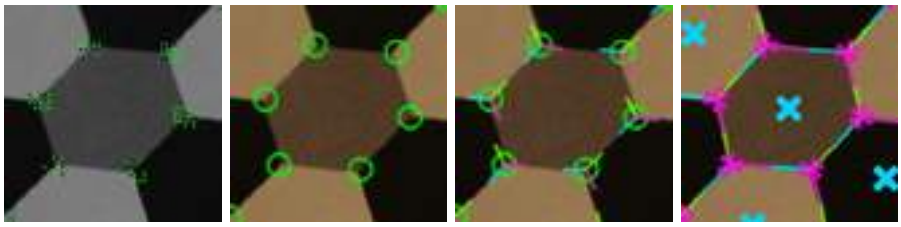
Prescanning the image to mark regions of interest.

The image is processed by using sparse horizontal and vertical scanlines (Figure 63 left) forming a grid. Edges are detected on these scanlines by using adaptive thresholding. Those grid cells whose sides do not contain any intensity edges are removed from further processing. The "interesting" blocks are conditionally further divided into four identical squares (in a manner similar to a quadtree) and the blocks of interest are found hierarchically (Figure 63 right). The conditional subdivision is controlled by the number of edges on the block sides.

7.1.3 DETECTION OF Y-JUNCTION IMAGE FEATURES

Triangle shaped scanning window is used to find Y-junctions.

The cells identified by the pre-scanning are searched for the Y-junctions by using a 4-orientable triangular shaped scanning window (Figure 65). An $n \times n$ pixel scanning window is used to obtain the response on the given area of the image. The size of the window depends on the size of the HMF cells we try to detect and the robustness against blur. If we set n to a small number, we will



(a) Detector response (b) Center point (c) Junction pose (d) Hexagon

Figure 64: Detection of the Y-junctions. **(a)** The image is processed by the detector. If the response of the filter is above a threshold, it is stored for further processing. **(b)** Thus identified points are clustered to form Y-junction centers. **(c)** The pose of each junction is defined as the three strongest edges in the nearby area. **(d)** The hexagons are reconstructed by connecting the nearest junctions together.

be able to detect small junctions, but for blurred regions the detection will fail. On the contrary, if large number is used the algorithm can not detect small HMF, but will be more blur tolerant.

For each triangle, the intensity differences are computed between the vertices by the following equation:

$$d(\mathbf{a}, \mathbf{b}) = |I(\mathbf{a}) - I(\mathbf{b})|, \tag{19}$$

where $I(\mathbf{x})$ is the intensity at given pixel $\mathbf{x}$. The response of the triangle is then represented as the minimal value of the three differences:

$$r_t = \min\left(d\left(\mathbf{a}, \mathbf{b}\right), d\left(\mathbf{a}, \mathbf{c}\right), d\left(\mathbf{b}, \mathbf{c}\right)\right), \tag{20}$$

where $t \in \{1, 2, 3, 4\}$ is the index of the triangle and $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are its vertices (Figure 65). The final response magnitude is computed as the maximal value of the four triangles' responses:

$$R = \max(r_1, r_2, r_3, r_4). \tag{21}$$

To avoid false detections of the Y-junctions, the gradient magnitude is checked at each triangle vertex. When it is above a threshold, the edges created using that vertex will be discarded in the response evaluation. This is necessary when detecting between a white and a black hexagon, where the color is blurred to be changing continuously between them. In such case the detection triangle can be located so that one vertex lies on the grey edge between the extreme shades and it would cause a misdetection (Figure 66 left). The response values of the scanning window are thresholded and the final result can be seen in Figure 64a and detail in the right part of Figure 66.

In the next step the center points are obtained by clustering the candidate Y-junction positions acquired from the previous step (Figure 64b).

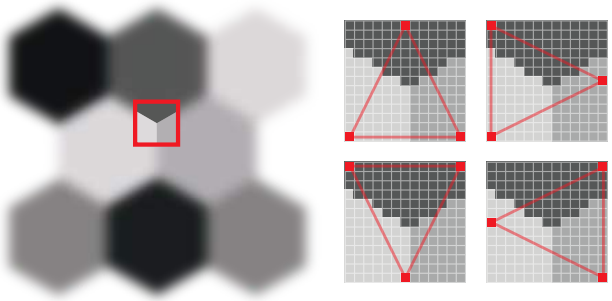


Figure 65: The scanning window contains four triangles rotated by 90°, where only the triangle vertices are sampled and examined. The 4-orientable window ensures robustness against rotation.

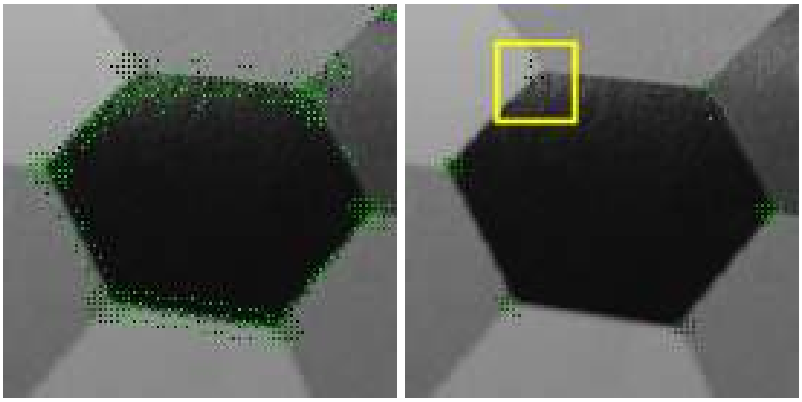


Figure 66: False responses of the detection. **Left:** Not checking the magnitude of the gradient can cause false detection because the intensity change is continuous. **Right:** By skipping places with high gradient magnitude these cases can be avoided. The yellow box shows that junction is detected even the illumination at the junction is variable and responses are smaller.

"Legs" are edges directly coincident with Y-junction.

For each Y-junction, the directions of the edges coincident with it – referred to as the "legs" – are estimated (Figure 64c). This step also helps eliminate some false detections of Y-junctions. An imaginary circle is considered around the junction on which three gradient magnitude extrema are found. These extrema have to be above a threshold. To refine the positions, a subpixel search is applied in small neighborhoods.

7.1.4 RECOGNITION OF MARKER FIELD FRAGMENTS

When the Y-junctions are detected, the following information is obtained for each of them:

- center position,
- directions of three "legs",
- edge gradient orientations on the legs.

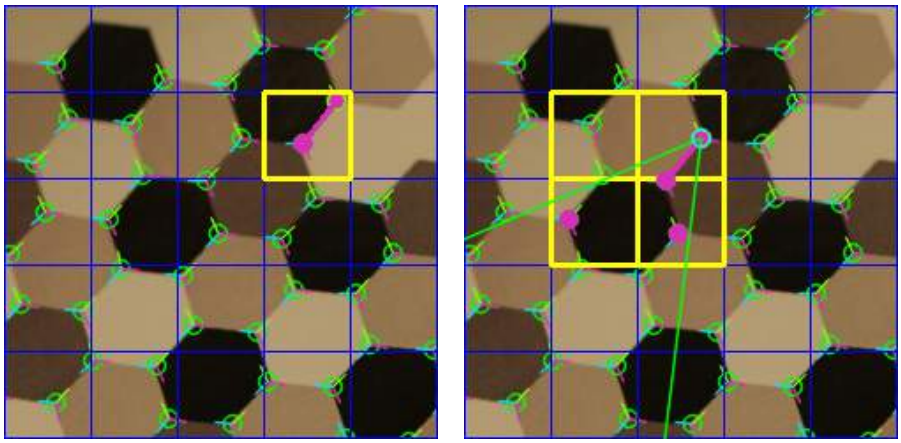


Figure 67: Grid search. **Left:** The yellow outline highlights the current cell. In this case a valid junction is found in the inner cell. **Right:** Adjacent cells are searched for junctions. Pink dots are the valid candidates and the one with the highest score (based on legs' direction and gradient) is selected. The green lines enclose the search area.

Based on this information the Y-junctions are paired and a graph is formed; the Y-junctions are the nodes and their bilateral matches are its edges. A uniform hashing grid is used to speed up the search for matching Y-junctions.

The search algorithm consist of the following steps (Figure 67):

1. **Find bounds and compute the parameters of the grid** by finding the minimal and maximal positions of the junctions. At first compute the width and height of the grid and then the cell size, based on these values and the number of found junctions (n), by using equation:

$$cellSize = \sqrt{\frac{width * height}{n}}. \tag{22}$$

   The resolution is then computed by dividing the width and height by the size of cells.

2. **Hash junctions** via their positions by using the grid.

3. **Search for neighbors.** All Y-junctions are processed sequentially and matches are found among the rest of the junctions. First, the junction's current cell is searched for valid neighboring junctions (Figure 67 left). If none are found, by using the pose of each leg an area of interest is determined (green lines in Figure 67 right) and the involved cells are traversed. These cells are then searched for valid junctions. When a valid candidate is found, the connection is scored based on the distance, pose difference and gradient directions in order to select the best match.

The hashing, sorting and cell updating parts are adopted from "Particle simulation using CUDA" [55], where a detailed description of these steps can be found.

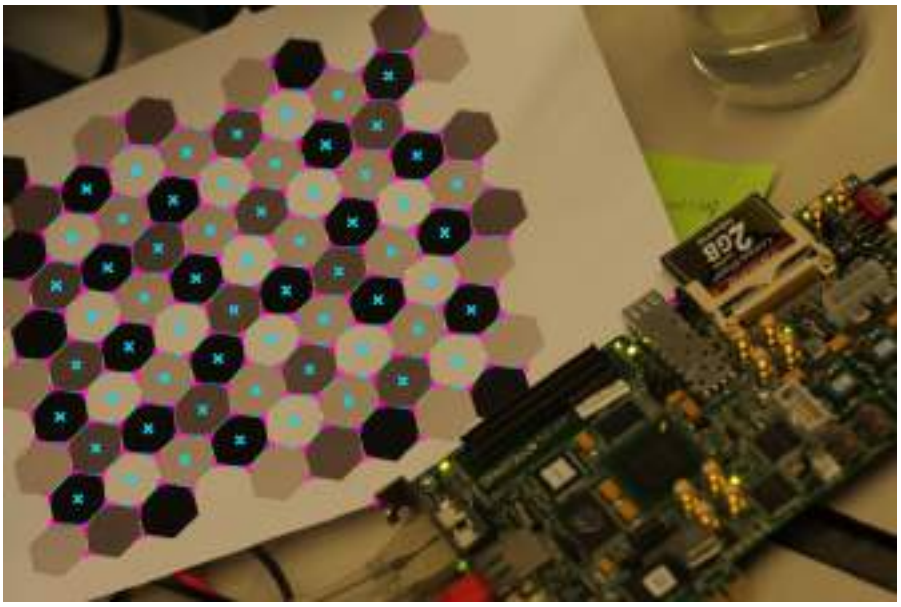


Figure 68: The reconstructed hexagonal marker field. Each full hexagon has a cross drawn in the middle.

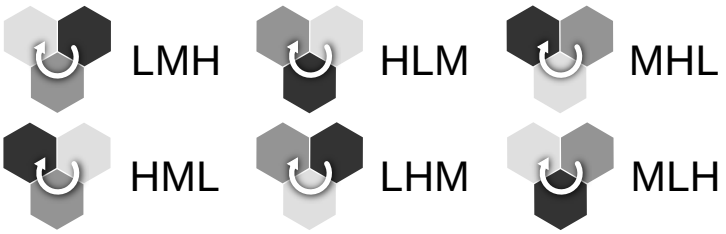


Figure 69: Descriptor of a single Y-junction: the sequence of Low, Medium, High intensities of the adjacent hexagonal module, starting from an arbitrary hexagon in a defined order (CW in our case).

To identify a unique part of the marker field actually detected, the edge gradients on the Y-junction "legs" are used. Every single Y-junction can be described by one of 6 combinations of grey intensities of the adjacent modules (Figure 69). For color marker fields, a higher number of combinations exists (in particular, 6^n where n is the number of channels, e.g. $n = 3$ for RGB).

For a neighborhood of a hexagon, the LMH combinations (Figure 69) of all involved Y-junctions define uniquely its identity and rotation. A decision tree (or a hash or other generic tool) can be used for determining the hexagon identity (Figure 70). Six adjacent Y-junctions can define a fragment (individual marker) in a small HMF ($\sim 10 \times 10$ greyscale). For larger marker fields, a larger neighborhood of Y-junctions (e.g. 12 or 24) can be used.

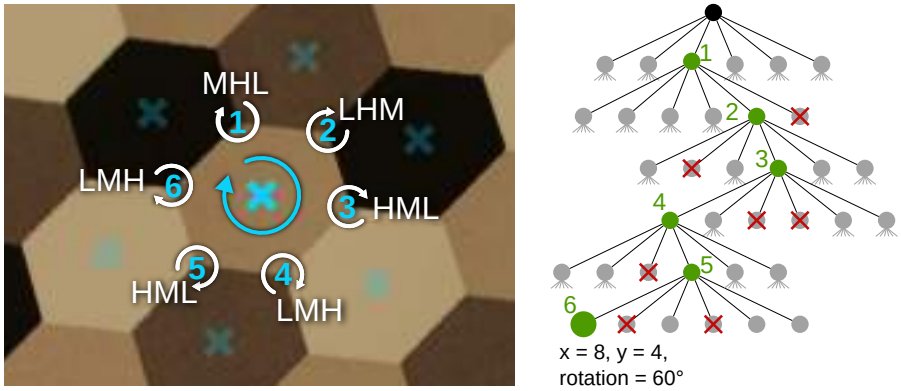


Figure 70: Recognition of a fragment of the marker field. **Left:** Six Y-junctions are described by their LMH combination (Figure 69). **Right:** A decision tree of six levels (for six Y-junctions) determines the location and orientation of the hexagon within the HMF. Every non-leaf node of the tree has 6 child nodes for 6 permutations of LMH.

7.2 EXPERIMENTAL RESULTS

To evaluate the Y-junction detector we created a dataset of 50 images. In order to evaluate the precision for different types of image distortions, we split the dataset into 4 categories (Figure 71):

- **Blurred** – a part or the full marker field is blurred
- **Clean** – only the marker field is visible
- **Environment** – the marker field is placed in "real" environment
- **Zoomed** – only a fraction of the zoomed marker field is visible

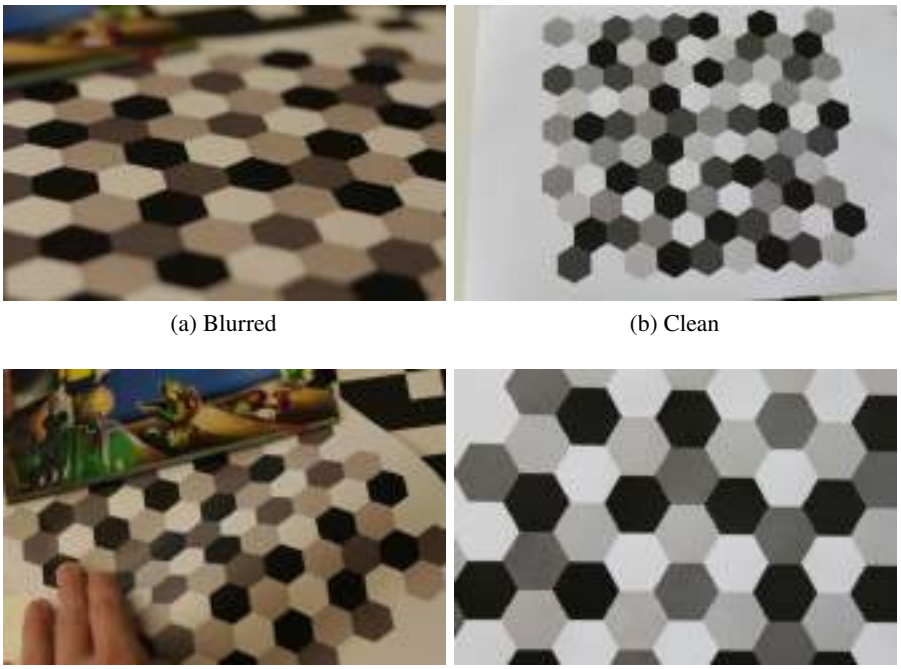
7.2.1 PERFORMANCE OF THE Y-JUNCTION DETECTOR

We selected FAST as reference detector.

All Y-junctions in all images in the dataset were manually annotated. Based on this annotation set we evaluated the success rate of our Y-junction detector. We selected the FAST detector [56, 57] as the reference, since both the FAST and our Y-junction detector are detecting corner-like features (contrary to SIFT or SURF which detect patches). The results of the comparison are shown in Table 11.

FAST finds more duplicates than Y-junction detector.

The table shows that the FAST detector finds one junction repeatedly even when the non-maxima suppression is enabled while our Y-junction detector finds much less duplicates. This is because FAST is a general corner detector, while our purpose is to find only the Y-features. The worst results (missed junctions rate – false negatives) was obtained in the blurred image category for both detectors. Nonetheless, in this category, our detector gets better results for true positives, false positives, and false negatives too.



(a) Blurred (b) Clean

(c) Environment (d) Zoomed

Figure 71: Sample images from the dataset.

While testing the success rates of the detector we used these parameters. Our detector uses 13×13 window and the threshold value is set to 20. The FAST detector in the OpenCV implementation uses 16×16 block size and the threshold was set to 20 too. In the prescan step the initial size of the blocks was 128×128 and 3 sub-levels were used.

We have tested the detector for various window sizes (Table 10). For smaller sizes the detector finds smaller number of false positives, but finds also smaller number of true positives. However by using larger window we are able to increase the number of true positives, but the number of false positives is also increased. A good compromise between precision and recall is using a 13×13 scanning window.

Window size	7×7	13×13	19×19	31×31
Precision (%)	99.5	94.2	82.2	49.3
Recall (%)	19.7	96.5	98.9	70.1

Table 10: Performance results for various sizes of scanning window.

7.2.2 SPEED EVALUATION OF THE DETECTOR

In the second set of experiments we measured the speed of the detector. We created 3 different sizes of each image in the dataset (640×480, 1680×1120

	GT	TP	FP	FN
Blurred	1504	1389	559	524
Clean	1930	4153	1930	19
Environment	1703	3041	5740	52
Zoomed	1213	2940	3511	13

(a) FAST with non-max suppression

	GT	TP	FP	FN
Blurred	1504	4944	2305	523
Clean	1930	17697	4305	4
Environment	1703	12087	28667	27
Zoomed	1213	11626	6146	4

(b) FAST without non-max suppression

	GT	TP	FP	FN
Blurred	1504	1410	87	103
Clean	1930	2002	14	44
Environment	1703	1694	265	17
Zoomed	1213	1300	8	60

(c) Y-junction detector / HMF

Table 11: Performance comparison of the Y-junction detector against the general corner detector FAST, on 800×600 images. FAST is reported with and without non-maxima suppression enabled. **GT** number of ground truth junctions (manually annotated). **TP** number of true positives detected – points detected in near vicinity of a ground truth junctions. When $TP > GT$, multi-detections appear. **FP** number of false positives – keypoints/junctions detected away from any annotated ground truth junction. **FN** number of false detections.

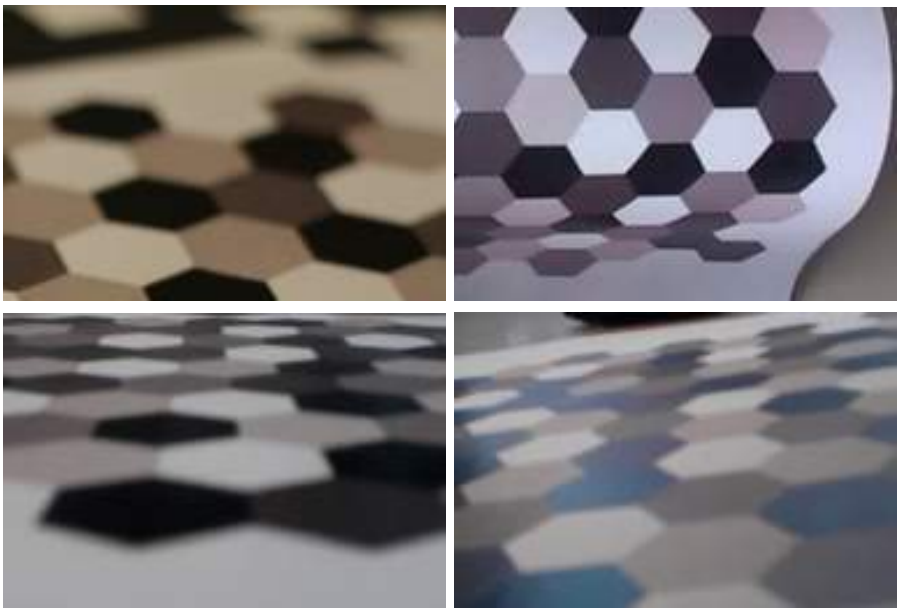


Figure 72: Examples of problematic edges (failure cases) within the pictures taken by a mobile phone of the marker field. The recognition can be challenging due to motion blur, uneven lighting conditions, high distortion, etc.

and 4272×2848). Table 12 contains the computational time results separately for the different categories. The first row (DR) in each table contains time of the pre-scan process, that is the scanline step. The second row (DRPS) contains the measured time for detector response computation with enabled pre-scan. The third row is the sum of the previous two rows, the pre-scan and response computation. The last row contains detector response measurement with disabled pre-scan. As it can be seen for smaller images (640×480) the pre-scan step can not improve the result significantly, but the SL + DRPS still smaller than DR. As the resolution increases the difference between these to values are increasing, too. For larger images the results can be further improved by the pre-scan step. The current version of the detection algorithm is only a prototype which demonstrates the detection performance of the Y-junctions. This is why we report the relative time consumption instead of comparison to alternative detectors.

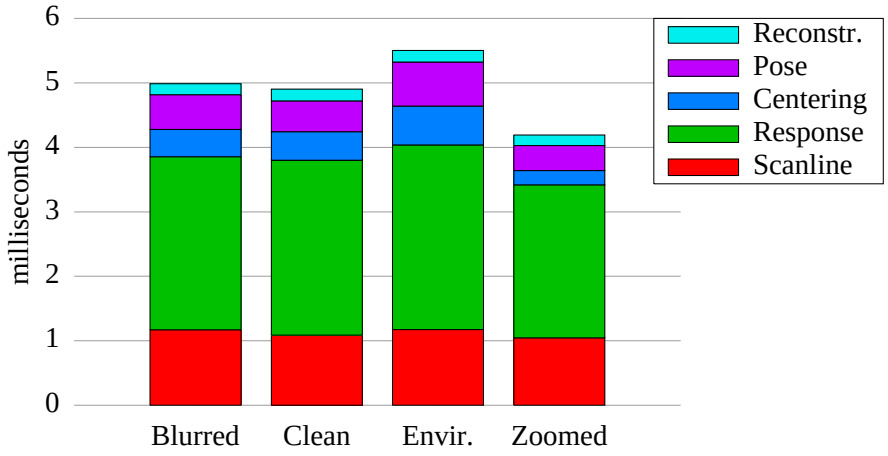
7.2.3 ELAPSED TIME BREAKDOWN & ANALYSIS

Most time consuming steps are prescanning and response evaluation.

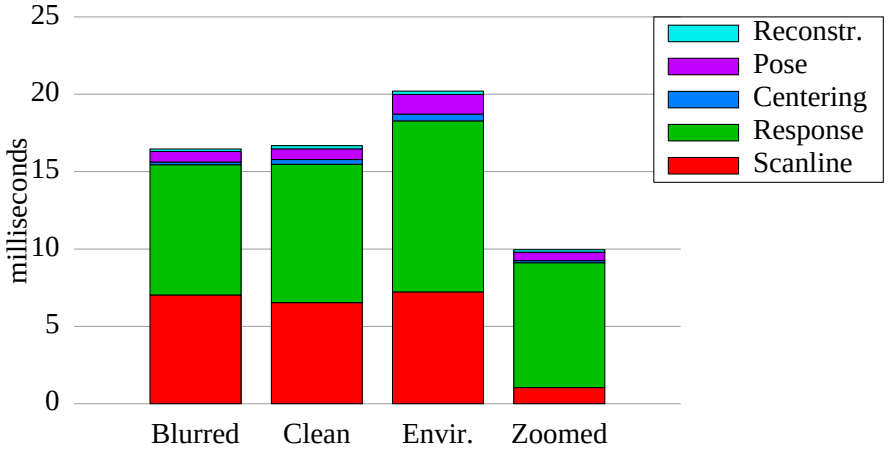
In the third set of experiments we measured the breakdown of computation time for each step of the algorithm. As it was described in Section 7.1.1 the algorithm has a number of steps: pre-scanning, response evaluation, junction center and pose estimation, hexagon reconstruction. We used 3 different input image sizes to test the complexity of the algorithm steps. Table 13 shows the times measured during the test split by categories, steps and image sizes. The

first column (pre-scan phase) depends only on the size of the scanned image. For bigger images the pre-scan have to examine bigger number of pixels. The second column (response evaluation phase) depends on the number of extracted blocks. The last three columns are dependent only on the number of found Y-junctions; times for these step are not increasing for larger images.

The most time-consuming part of the algorithm are camera pre-scan and response evaluation step. The times were measured on an Intel Core i7 920, 2.67GHz with a DDR3 memory.



(a) 640×480 images.



(b) 1680×1120 images.

Figure 73: Breakdown of computation time for each image category.

	Blurred	Clean	Envir.	Zoomed
SL	1.1	1.1	1.2	1.1
DRPS	3.0	3.1	3.4	2.6
SL + DRPS	4.1	4.1	4.5	3.7
DR	4.6	5.1	4.7	4.8

(a) 640×480

	Blurred	Clean	Envir.	Zoomed
SL	5.6	5.5	6.4	5.5
DR (SL)	8.3	9.0	11.3	8.1
SL + DRPS	13.9	14.5	17.8	13.6
DR	34.6	36.7	35.4	36.3

(b) 1680×1120

	Blurred	Clean	Envir.	Zoomed
SL	30.3	35.0	36.8	36.4
DRPS	15.9	23.0	26.9	23.6
SL + DRPS	46.3	58.0	63.8	59.9
DR	243.3	239.4	236.1	235.3

(c) 4272×2848

Table 12: Performance results in milliseconds for different sizes and categories. **SL**: pre-scan – scanline step; **DRPS**: detector response with enabled pre-scan, **SL + DRPS**: sum of SL + DRPS; **DR** detector response with disabled pre-scan.

	Scanline	Response	Center	Pose	Recon.
Blurred	1.17	2.68	0.42	0.54	0.17
Clean	1.09	2.71	0.44	0.48	0.18
Envir.	1.17	2.86	0.60	0.68	0.18
Zoomed	1.05	2.37	0.22	0.39	0.16

(a) 640×480

	Scanline	Response	Center	Pose	Recon.
Blurred	7.04	8.41	0.19	0.67	0.16
Clean	6.54	8.94	0.30	0.70	0.20
Envir.	7.23	11.05	0.44	1.26	0.22
Zoomed	7.05	8.06	0.14	0.55	0.16

(b) 1680×1120

	Scanline	Response	Center	Pose	Recon.
Blurred	32.43	16.24	0.05	0.42	0.06
Clean	35.32	23.26	0.08	0.68	0.13
Envir.	37.54	27.15	0.18	1.34	0.22
Zoomed	35.47	23.35	0.14	0.66	0.05

(c) 4272×2848

Table 13: Breakdown of computational time into different parts of the algorithm in milliseconds. **Scanline**: pre-scan step; **Response**: detector response computation; **Center**: junction center estimation; **Pose**: junction pose estimation; **Recon.**: hexagon reconstruction; for more details see section 7.1.3.

Part III

APPLICATIONS OF PROPOSED MARKERS

INTRODUCTION TO APPLICATIONS OF PROPOSED MARKERS

This part of my thesis focuses on the possible real-life applications of the markers proposed in the individual chapters of previous part. Like in the previous part, a considerable portion of the text is directly taken from the published papers.

Only applications that were published as standalone papers or have been implemented are described in this part. Ideas for other applications and future work are summarized in Chapter 13.

PUBLISHED IN THE CONFERENCE PROCEEDINGS

1. **Color Theme-Based Digital Art** (Chapter 8).
   This application utilizes possibility of selecting different colors for improved Uniform Marker Fields (UMFs), not just black and white or shades of grey, and thus creating unobtrusive, artistic and aesthetically appealing markers. Three real-life demonstrations are discussed: unobtrusive indoor navigation, public virtual displays and UMF as means of artistic expression. Published in Zachariáš and Herout: "Color Theme-Based Digital Art for Augmented Environment" (ICDAMT 2016) [IX].

2. **Visual Correction of Position Drift using Uniform Marker Fields** (Chapter 9).
   Strategically placed aesthetic UMFs can serve as anchor points for robotic indoor navigation and visually correct the drift in integrally calculated robot position and orientation. Published in Zachariáš, Szentandrási, and Herout: "Visual Correction of Position Drift using Uniform Marker Fields" (SCCG 2016) [X].

3. **Screen-to-screen Task Migration** (Chapter 10).
   Migrating the context from one device to another is still hot research topic and is still not satisfactorily solved. Using unobtrusive transparent UMF on the PC or public display screen and then scanning (obtaining the context that is "under" the virtual cursor) it with mobile device can simplify the context migration. Published in Kajan, Szentandrási, Herout, and Zachariáš: "On-Screen Marker Fields for Reliable Screen-To-Screen Task Migration" (SouthCHI 2013) [IV].

PUBLISHED IN THE JOURNAL

4. **Effortless Chromakeying** (Chapter 11).
   Using appropriately selected shades of green, the UMF can be turned into the so called green screen, nowadays massively used in the filmmaking industry. Selecting the right greens allows the detection and chromakeying (background removal) algorithms to work well simultaneously. Such greenscreen provides the camera pose estimation in addition to its default purpose. Published in Szentandrási, Dubská, Zachariáš, and Herout: "Poor Man's Virtual Camera: Real-Time Simultaneous Matting and Camera Pose Estimation" (Computer Graphics and Applications Magazine 2016) [XI].

IMPLEMENTED BUT NOT YET PUBLISHED

5. **Video Gaming on a UMF Carpet** (Chapter 12).
   In 2013 we won Czech national round of Imagine Cup, IT competition organized by Microsoft, and we placed ourselves in TOP 5 at the worldwide finals. Our contest entry was AR game called Project reSound which was played on the UMF gaming carpet. This application can be generalized, carpet can be manufactured in arbitrary size and shape, and used for example in kindergartens.

8

COLOR THEME-BASED DIGITAL ART

Figure 74: Our markers are designed so that they fit their environment and/or their design context. **Left and right bottom:** They can be large-scale designs part of the architecture. **Right top:** Or just small design patterns being part of small-scale design fragments.

In our paper "Color Theme-Based Digital Art for Augmented Environment" in proceedings of *The International Conference on Digital Arts, Media and Technology 2016 (ICDAMT)* we present our previous work on Uniform Marker Fields (UMFs) and improved Uniform Marker Fields (UMFs) in the new light of digital art and intuitive interaction in design environments. We focus on automatically, semi-automatically, or manually designing color schemes suitable for marker field synthesis (Section 8.1). Several model use-cases and examples of markers placed in small-scale (a small handheld artifact), indoor (navigation-enabling flooring, interactive AR wall), and outdoor contexts (Section 8.2).

Our work on UMFs in the light of digital art and design.

8.1 COLOR THEME EXTRACTION

This section discusses the extraction of aesthetically and functionally plausible

How to extract aesthetic and functional color schemes for marker field?

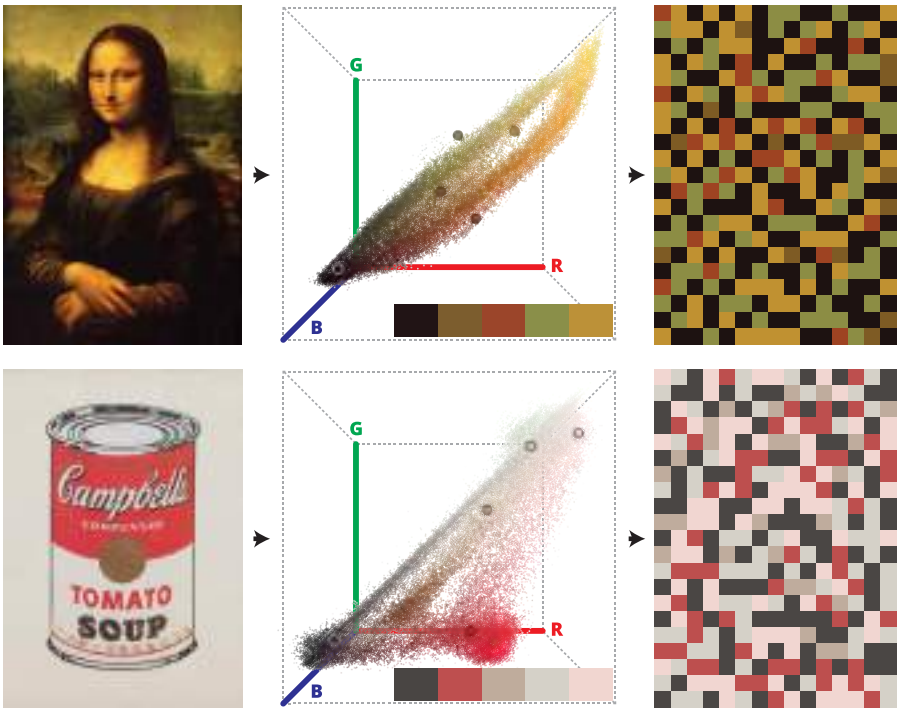


Figure 75: The proposed color theme extraction is based on color quantization (in this case the median cut algorithm) of the original image. Middle image shows three-dimensional RGB space where every dot represents each color from the original. Bigger dots are colors computed by the quantization algorithm. These colors are then used for marker synthesis. The result is shown in the rightmost image.

color schemes for the marker designs.

One of the possible approaches for color theme extraction is using Color Quantization. It is the process that reduces the number of unique colors in an image to a smaller discrete subset, keeping the aesthetic properties of the original image. It is usually used for displaying images with many colors on devices with limited palette or to reduce its size (i.e. compression). Standard techniques for color quantization described in "Color Image Quantization for Frame Buffer Display" [58] treat this problem as clustering the spacial points in some color space (mostly RGB, but another popular choice is the Lab color space), where each point represents the unique color in input image.

There are many algorithms to perform color quantization with different approaches. For example, the Dekker's NeuQuant algorithm introduced in the paper "Kohonen neural networks for optimal colour quantization" [59] reduces the color space by training a Kohonen neural network. This method produces high-quality results, but due to its complexity it is considered slow.

We are using the simple Median Cut algorithm [58]. It is very fast and it produces great results for our application. This algorithm performs these five steps:

Figure 76: UMFs with manually selected color themes where each one represents a different season of the year.

1. Find the smallest bounding box that contains all colors in the input image color space.
2. Sort them along the longest axis or by euclidean distance.
3. Split the bounding box into two new bounding boxes at median.
4. Repeat steps 1-3 until the number of bounding boxes is equal to desired number of colors in palette.
5. The representative colors (i.e. the palette) are found by finding the mean of the colors in each bounding box.

We implemented this algorithm and we fed the resulting colors into the UMF generator (Figure 75). Color quantization ensures that euclidean distance between marker's colors is sufficient enough that UMF detector can reliably work and estimate the camera position.

Manual extraction.

Another option how to choose the colors for the marker is to manually select them based on an artist design intention and ideas – e.g. four markers each representing different seasons of the year (Figure 76).

Figure 77: Greyscale UMF printed on the flooring can be used for indoor navigation by humans using augmented reality and/or by robots to precisely determine their position in the building.

8.2 DIGITAL ART DEMONSTRATIONS

[h!]

This section showcases several examples of using the color theme-based marker fields used in various environments and for different purposes. We have been experimenting with the concept of using the marker field on the floor for robot and mobile device navigation (Section 8.2.1). The discussed markers can be naturally used as a placeholder for public AR displays (Section 8.2.2). Lastly, augmenting handheld flat artifacts (Section 8.2.3).

Figure 78: Aesthetic UMFs for use as public display; inside installation with possible augmentation showed.

Figure 79: Aesthetic UMFs for use as public display; outdoor installation.

8.2.1 UNOBTRUSIVE INDOOR NAVIGATION

Modern looking navigation markers with almost unlimited length.

With our proposed solution, facilities such as hospitals, factories, business buildings, and malls can install very modern looking flooring with UMF printed on it (Figure 77). Its properties allow to synthesize very long markers for corridors and narrow areas generally. We were able to generate conflict-less greyscale marker that had 9 modules in width and 500 in length; assuming that each square would be 20×20 centimeters, such marker could cover 100 meters

of 1.8 meter wide corridor. Such flooring can be utilized for indoor navigation by humans using augmented reality and/or by robots or factory machinery to precisely and cheaply determine their position in the entire building. The aesthetic properties can be controlled and tuned by the architect to avoid unwanted compromise in the architectural design.

8.2.2 PUBLIC VIRTUAL DISPLAYS

Public yet still private displays.

Any planar wall inside or outside of a building can be outfitted with the UMF wallpaper (right portion of Figure 78) or semi-transparent material which is ideal for the glass structures (Figure 79). It can be then augmented with specific information the current environment needs.

For example, public display for visitors, interactive map of the building or current events that are taking place inside (left portion of Figure 78). A notable advantage compared to conventional public displays is that this virtual display can show different and possibly confidential information for different users.

8.2.3 UMF AS MEANS OF ARTISTIC EXPRESSION

When extracting colors from the painting, using mentioned color quantization or manual selection, we can generate markers that can be displayed next to it or around it and due to the same color theme they do not disturb the art itself (Figure 80). This offers limitless possibilities to extend/augment the art and thus create new digital art. Augmented content can be anything that fantasy of the digital artist comes up with.

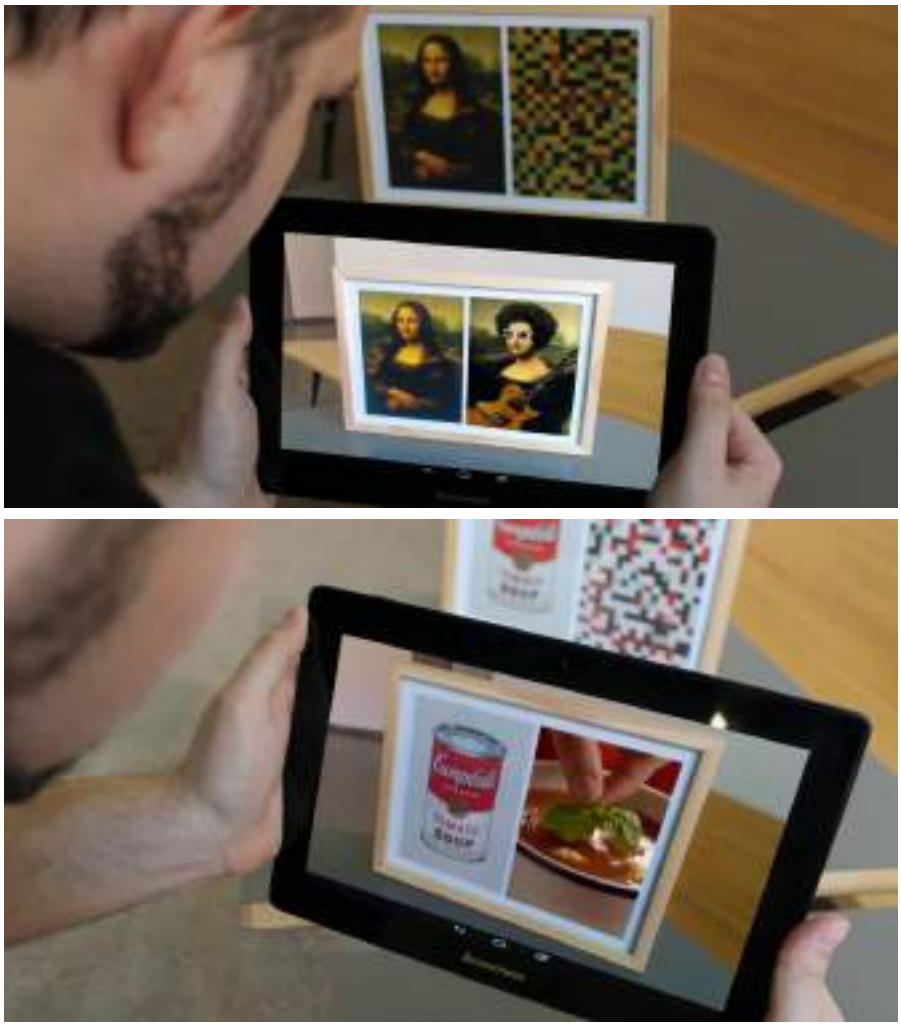


Figure 80: Art (Mona Lisa and Campbell soup) with color-themed UMF next to it.

9

VISUAL CORRECTION OF POSITION DRIFT USING UNIFORM MARKER FIELDS

In the paper "Visual Correction of Position Drift using Uniform Marker Fields" in proceedings of the *32nd Spring conference on Computer Graphic* we propose to enhance Inertial Measurement Unit (IMU) and conventional visual odometry navigation by visual location estimation based on markers conveniently displaced in the environment. We propose using visually appealing UMF (as mentioned in Chapter 8); this technique offers high precision and real-time performance on mobile devices of moderate computational power.

IMU measures the forces applied to the object.

IMU is an electronic device that measures forces applied to an object of interest in order to determine its motion and position. It measures mainly acceleration, angular velocity and sometimes magnetic field around the object. Therefore, it generally consists of an accelerometer, a gyroscope, and a magnetometer. IMUs are widely used in planes, ships and also in robotics for so-called dead reckoning navigation [60, 61] and in the large scale augmented reality systems [62]. Major disadvantage of this kind of navigation is that it suffers from drift due to the accumulation of error. Every newly measured relative position is added to the previously calculated absolute position and that is why every measurement error is accumulated point to point. Due to the limitations of today hardware, measurements can be taken only in discrete periodic intervals and variables (e.g. acceleration) are considered linear in the gaps between. This can be far from the truth and can cause additional error when integrating speed from acceleration and position from speed. This problem can be partially solved using different kinds of filtration and data fusion [63, 64]. Some systems also use visual odometry [65, 66].

In this particular application, advantage over other fiduciary marker systems is that UMF can be aesthetically pleasant and controlled by a graphics designer or an architect, as explained in the Chapter 8.

9.1 VISUAL CORRECTION OF POSITION DRIFT

We propose a system for correcting the dead reckoning navigation drift by detecting strategically placed UMFs in the environment. Its unique properties allow to easily create rectangular shape of almost any size and therefore it

can be used on various media, e.g. wallpaper, carpet, painting or flooring Figure 81.

Each such marker's absolute position in the environment and its real physical size must be known and stored in the system. Once the marker is detected by the device, ID of the marker is recognized and device's precise relative position to the marker is known. ID of the marker is searched in the system and related stored data (absolute position and physical size) are accessed. Finally, the absolute position of the device in the environment can be calculated by combining this data.

Using classical marker detection/tracking systems such as ARTag or ARToolKit to attain the same results would require creating arrays of markers with gaps between them. That creates blind spots where no marker could be detected. Overhead generating multiple markers and incorporating them into the large array is much greater then generating just one UMF that was design to function as array. Even more importantly, not many AR systems were created with aesthetics in mind and thus are not usable when designing appealing environment.

Cheaper (computationally and financially) than natural keypoints detection or SLAM.

Natural features extraction and/or any Simultaneous Localization And Mapping (SLAM) systems are dependent on the presence of natural features in environment and are generally more computationally demanding [2]. Besides the camera (required for any visual localization) some systems require additional hardware, such as stereoscopic cameras or lidar/radar which can be very expensive, must be installed on every single device, and constitutes additional battery consumption and computational overhead.

9.2 OBTAINING GROUND TRUTH DATA

In order to evaluate the precision of our visual localization concept, we had to first acquire ground truth evaluation data. To do so, we prepared the setup shown in Figure 82. Points X, Y, and Z are "reference points" which are unmovable fixed objects in the environment; distance between reference points is $|XY| = |YZ| = 157.5\,\text{cm}$. Remaining points A, B, C, D, E, and F are "test points" which we chose randomly in the space (on the floor).

We measured distances from every reference point to every test point (Table 14) and also the distances between each two reference points (Table 16a) for future comparison between calculated and measured points' positions.

Triangulation is used to calculate points' position.

Using these measurements and triangulation (knowing all three sides of each triangle), we were able to calculate the positions of all test points. With three fixed reference points, we could form three triangles for each test point. For example for point A – $\triangle XYA$, $\triangle YZA$ and $\triangle XZA$. We calculated the test points' positions using all three triangles separately and used their average as

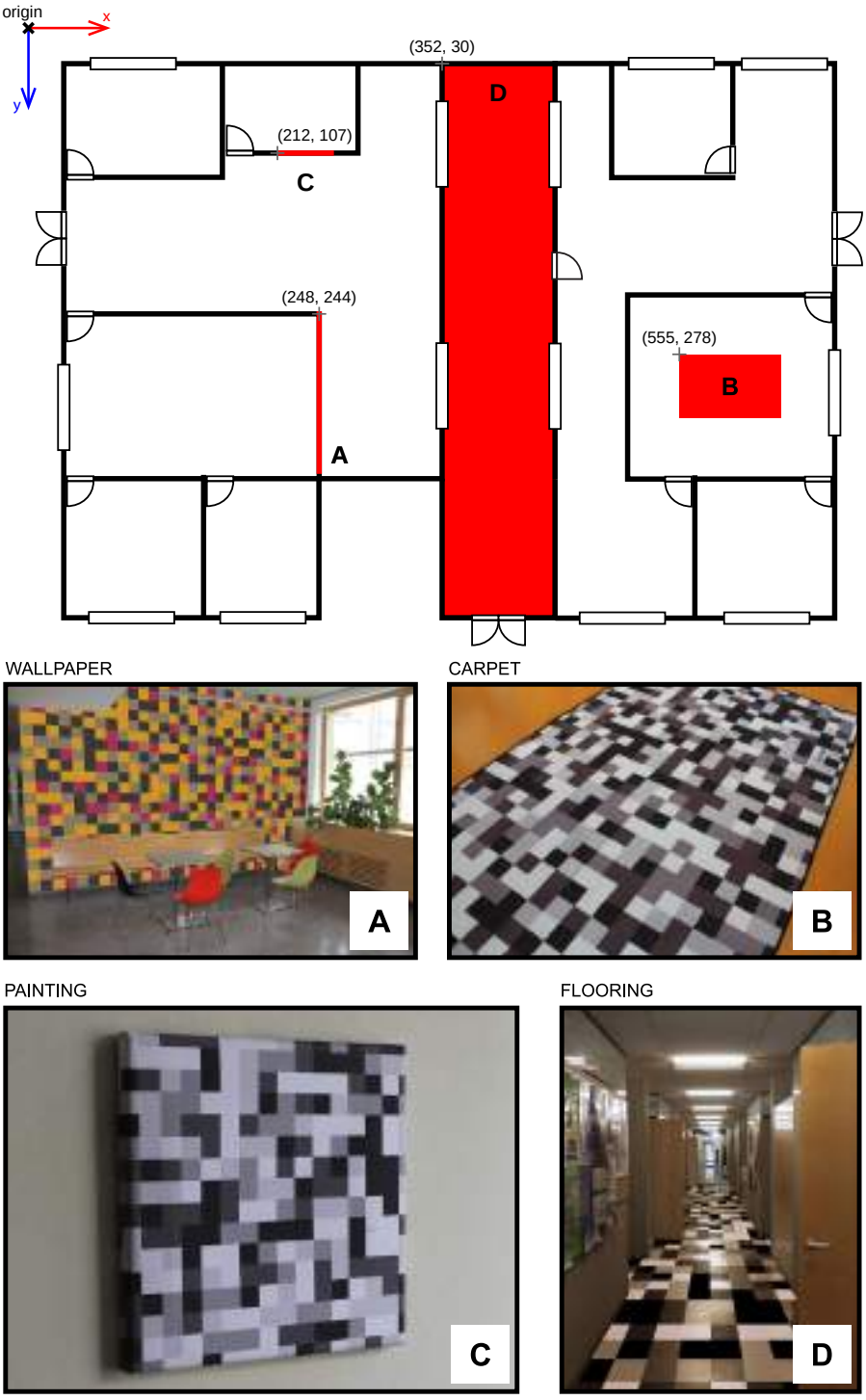


Figure 81: Floor map of an office building with the different strategically placed installations of the UMFs (red) – wallpaper, carpet, painting and flooring. These are just examples, any surface of object that can be printed or painted on can be used. Each marker has its absolute position in environment stored in the system (shown next to the marker on the plan).

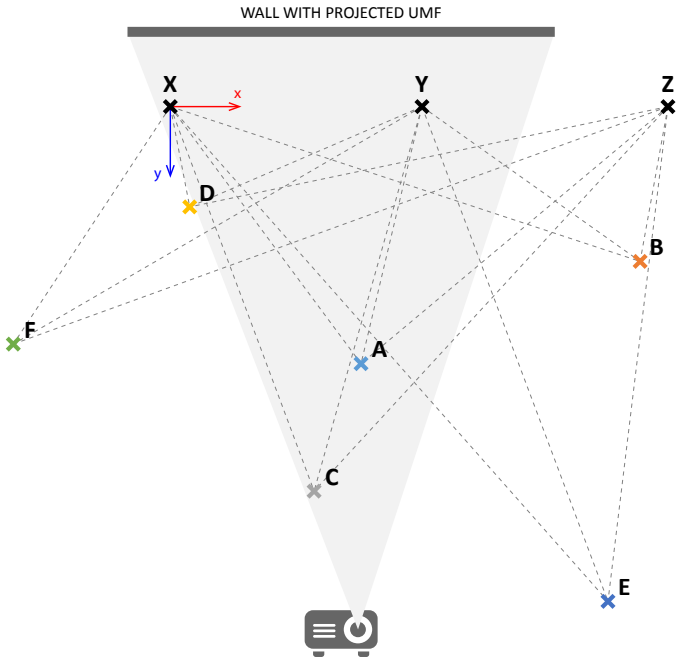


Figure 82: Setup used to gather the ground truth data and evaluation measurements.

the result shown in Table 15. Reference point X is the origin of our coordinate system (Figure 82).

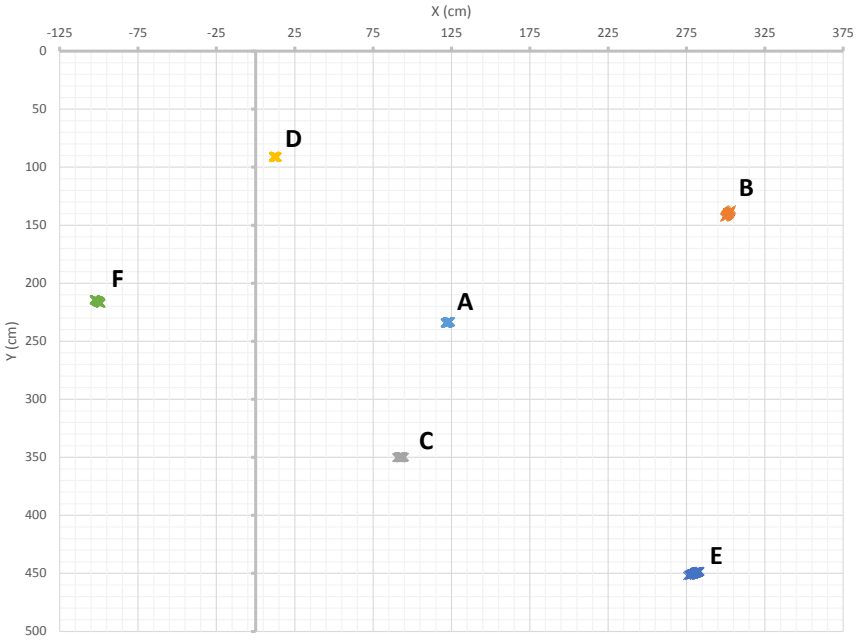
	A	B	C	D	E	F
X	264.0	332.5	236.0	92.0	530.0	238.0
Y	236.5	201.0	356.0	171.0	466.5	336.5
Z	302.0	140.0	414.5	316.0	450.0	468.0

Table 14: Measured distances in centimeters for each test point (A to F) from each reference point (X, Y and Z).

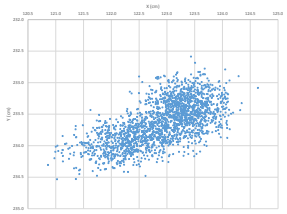
To determine what our minimal error in measurements can be, we ran 10 000 iterations of Monte Carlo method; in each iteration we changed the distances between the reference and test points by normal distribution random value from interval $\langle -0.25, 0.25 \rangle$ (half of the smallest unit on our measuring tool) and calculated new test point position. The resulting clusters are shown in Figure 83. As quantification of the error we calculated the average of variance of each cluster – 0.995 cm.

	A	B	C	D	E	F
x	123.36	301.88	92.79	12.43	281.94	-100.25
y	233.82	139.69	350.01	90.74	449.32	216.17

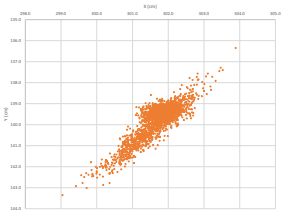
Table 15: Calculated positions of test points in centimeters.



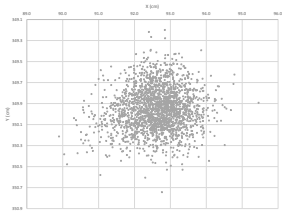
(a) All 10 000 Monte Carlo generated outputs for all 6 test points.



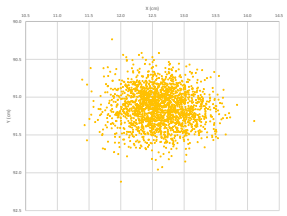
(b) Test point A



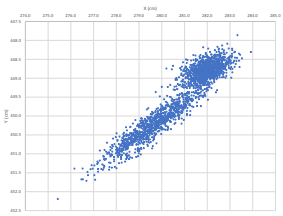
(c) Test point B



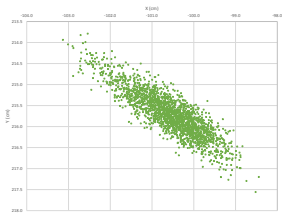
(d) Test point C



(e) Test point D



(f) Test point E



(g) Test point F

Figure 83: Output of Monte Carlo method used to determine the minimal measurement error.

	A	B	C	D	E	F
A	0	201.50	120.00	181.50	267.00	225.00
B	–	0	295.50	294.00	310.00	409.50
C	–	–	0	271.00	214.00	235.00
D	–	–	–	0	448.50	168.50
E	–	–	–	–	0	448.00
F	–	–	–	–	–	0

(a) Measured distances.

	A	B	C	D	E	F
A	0	201.81	120.15	181.04	267.57	224.30
B	–	0	296.57	293.55	310.27	409.33
C	–	–	0	271.44	213.64	234.90
D	–	–	–	0	448.57	168.61
E	–	–	–	–	0	447.69
F	–	–	–	–	–	0

(b) Calculated distances.

	A	B	C	D	E	F
A	0	0.31	0.15	0.46	0.57	0.70
B	–	0	0.07	0.45	0.27	0.17
C	–	–	0	0.44	0.36	0.10
D	–	–	–	0	0.07	0.11
E	–	–	–	–	0	0.31
F	–	–	–	–	–	0

(c) Absolute value of difference between measured and calculated distance.

Table 16: Measured and calculated distances in centimeters between each two test points (A to F) and absolute values of their difference. Average difference is 0.3 cm with standard deviation 0.193 cm.

After calculating the positions of all test points in our coordinate system, we calculated the distances between them and compared them to the measured distances (Table 16). Average difference was 0.3 cm with standard deviation of 0.193 cm.

9.3 PRECISION OF VISUAL LOCALIZATION BASED ON UNIFORM MARKER FIELDS

Dataset acquisition and its properties.

We created a dataset of images to measure the precision of the proposed approach. We took images from all 6 reference points with a smartphone

camera (Nokia Lumia 930) at 1920×1080 px resolution. For each reference point we took 15+ photos of the scene at different angles and 3 different heights from the ground. Robotic arms would allow precise camera positioning, but they have limited range and would not have covered the tested area. Instead, we used a plummet to achieve relatively high precision camera positioning. Another factor complicating the exact measurement is the camera rotation and determining the exact center of projection. These complications cause a small, but not dismissible noise to the detected positions. To achieve maximum precision during detection, the smartphone camera was calibrated including the distortion coefficients up to three radial and two tangential distortion parameters [37].

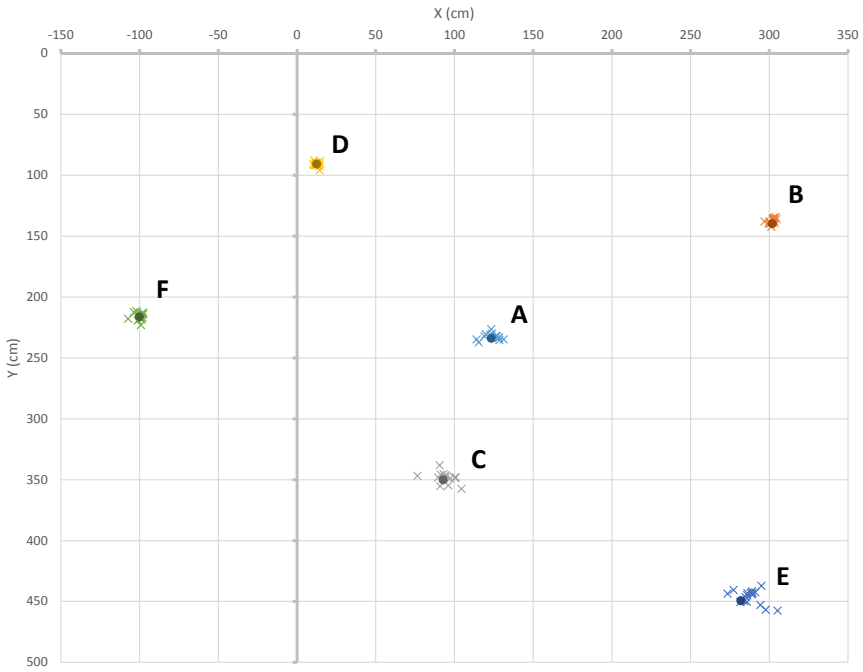


Figure 84: Detected positions obtained from our photo dataset (15+ photos for each test point). Round full points are the ground truth positions for each of the clusters.

Figure 84 shows the results. The ground truth points in Figure 84 (round full points) are for demonstration/illustration purposes only. The absolute distance of the detected points from the ground truth might contain a bias due to slight axis misalignment between the reference system and the space of the marker, camera calibration and/or mis-detections. A more rigorous approach is comparing the relative distances between pairs of detected positions, as follows.

Results of precision measurements.

The overall standard deviation of the distances between pairs of detected positions from the ground truth was 6.73 cm and the median 3.21 cm. That is error beneath 1 % in our test environment with diameter approximately 5 m.

It's important to note that the precision varies based on the distance from the fiduciary marker plane. With growing distance, the limited camera resolution and possibly imperfect camera calibration cause loss of camera pose precision. This property is also visible in Figure 84. The measurement jitter is also visibly increased for point E which observes the marker field from a large distance. Precision could be further improved by tracking the marker between frames and proper filtering (e.g. kalman).

10

SCREEN-TO-SCREEN TASK MIGRATION

In our paper "On-Screen Marker Fields for Reliable Screen-To-Screen Task Migration" in proceeding of the *2013 International Conference on Human Factors in Computing & Informatics (SouthCHI)* our goal is to deliver unobtrusive task migration between a mobile device and a desktop computer using an aesthetically acceptable UMF as overlay across a part of the computer screen. A direct visual interaction between desktop and mobile devices is quite of interest nowadays [67, 68].

Solution for keypoints-poor desktop environments.

When the screen does not contain enough unique keypoints (this is quite often in standard desktop environment!), our marker field ensures the localization (Figure 85). We show that UMFs are a good choice for this task and propose a methodology for inserting them into the screen image.

Figure 85: The goal of our work is to insert a subtle piece of information into the monitor screen that would be reliably detectable and could accurately establish the location within the screen observed by an ultramobile device (mobile phone, tablet. . .).

The study by Bales, Sohn, and Setlur [69] focuses on the methods for reaccessing the content (web pages, documents, calendar events. . .) among the personal devices. It showed that cross-device reaccess is often very spontaneous and unplanned and that support for this feature in native applications plays an important role in how users do it. Most of the contemporary solutions for content sharing focus only on documents and rely on complicated infrastructure, thus making the process quite hard for the user [70].

Dearman and Pierce [71] did survey and among most commonly observed means of content reaccess were:

- Leaving browser tabs open on the mobile device as a reminder to reaccess them on another device.
- Using handwritten or printed information, carried between the devices, and inputted on the second device.
- Utilizing shared bookmark systems access data between devices. Using these systems users save a bookmark on one device and have it available on their other device automatically.
- Unplanned reaccesses are frequently executed by entering search queries into another device.

Increasing the focus on the user and his comfort is more important than on applications and devices.

These findings suggest that very important is to increase the focus on the user and his comfort rather than on applications and devices, making devices aware of their roles and focusing on lighter-weight methods for information transfer and task synchronization / migration.

In our work we look for good way for the mobile phone user to intelligently interact with another device e.g. public display, desktop or tablet.

10.1 TARGETED REAL-LIFE SCENARIOS

There are four basic scenarios when we consider content and task migration:

- Resuming task on an mobile device. The most desired and at the same time most frustrating scenarios. Very often "going mobile" means significant reduction of comfort, pace of work and accuracy in favour of accomplishing tasks on the go. Usually this scenario involves explicit planning and preparation, for example synchronization of documents through specialized tools and services, which is time consuming.
- Resuming task on desktop computer. Besides consuming information (reading books, content from web pages, games. . .), users also often generate multimedia content on the go – they take photos, record video sequences, create notes and bookmarks. As in previous scenario, specialized synchronization services are commonly used, even though they are document-centric and are not able to capture and work with the application state.
- Sharing content between mobile devices. Only few solutions exist that allow for sharing of content among mobile device. Most commonly seen scenario is a content exchange between devices belonging to different users, content and state transfer between mobile devices of one user (e.g. task migration from smart phone to a tablet) is rare [67].

- Sharing content between desktop computers through mobile device. Despite wide availability of cloud-based (SkyDrive, Dropbox. . .) and traditional (FTP, email) file sharing solutions people still tend to rely on USB drives [71] to transfer content from one (desktop) computer to another.

10.2 DETECTION OF UMF AND CONTENT REACCESS

Marker mixed into the desktop content created new challenges for detection algorithm.

Detection of the on-screen marker is done almost equally as explained in the Chapter 6 about improved UMFs. This application of UMF created some unique problems e.g. different (as in never-seen-before) problematic edges (Figure 86) and finding the least obtrusive mixing technique (Figure 89).

From the correctly identified sub-windows (Figure 88) we calculate the location of the camera image center in the maker field. This information is sent to the content provider. The content provider then maps the position to display coordinates and updates the mixing mask and size for the marker field.

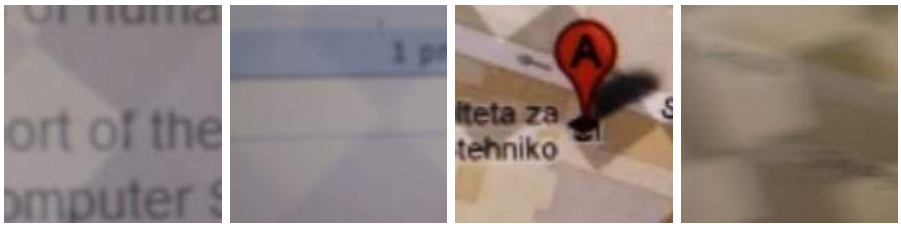


Figure 86: Examples of problematic edges within the pictures of the marker field.

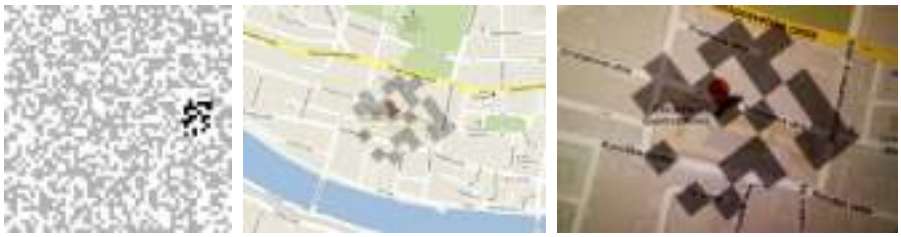


Figure 87: The use of UMF. **Left:** Highlighted region in the marker field used for mixing. **Middle:** Marker field mixed into the displayed content. **Right:** The same marker field, as seen by a mobile phone camera.

10.3 MARKER DETECTION RELIABILITY

Unobtrusiveness test.

In the paper we conducted user testing that asked about marker unobtrusiveness and comfort of use of the entire system. This was mainly the work of

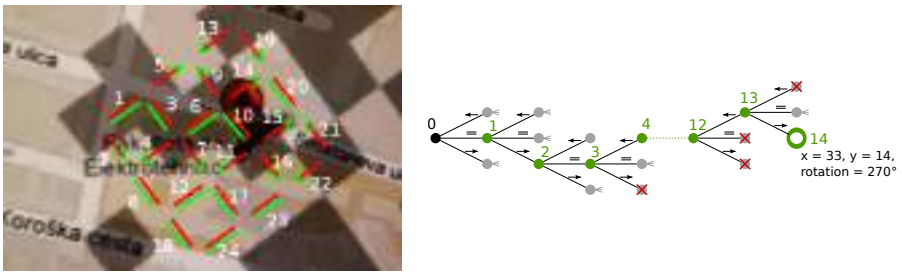


Figure 88: Correctly identified sub-window. **Left:** The order in which the edges are visited. **Right:** The decision tree. Leaves are either invalid or contain a location and orientation to the marker field.

Figure 89: Examples of mixed-in marker field with varying presets. **Top row:** Different background blur radii. **Bottom row:** Varying opacity levels.

my colleague Rudolf Kajan and is part of his Ph.D. thesis. Therefore for more information about this study check his work or the original paper.

Accuracy and reliability tests.

We tested the accuracy and reliability of our marker detection algorithm, with the marker mixed into natural screen contents. We created a setup consisting of one device in the role of content provider and one device in the role of content requester. As a content provider device we used a laptop computer with a high-resolution (1920×1080) display, and an Android 4.0.4 smartphone HTC Incredible S for the role of information requester device. The requester device was attached to a base perpendicular to the floor, in a fixed height, focused at a chosen part of the screen (Figure 90). The experiment was conducted in a room with artificial (fluorescent) lighting. Devices were connected through a WiFi connection. The requester device was held at a distance of 10 (on graphs shown as near), 20 and 40 cm (on graphs shown as far) and a pitch angle of 70°, 90°, 110°, 130°, and 150°. On the content provider device a fullscreen webpage containing text, several smaller images, and a map was displayed during the experiment. The pulsing period for the pulsing marker was set to 1.5 seconds. The frame rate of the camera was 25 FPS.

Figure 91 shows some of the reliability tests based on the display tilt and the distance between the camera and the display. For the static (non-pulsing)

Figure 90: Parameters of marker detection reliability setup – distance between mobile device and laptop, range of screen angles used during tests and height range of a mobile device in order to be focused on the same point on screen.

marker mixing we used the average transparency used in the case of the pulsing marker. We chose from each set with different pitch at least one histogram. Between 90° and 110° the results were consistent with the data presented in Figure 91 and thus they are omitted. The results show that even for 150° pitch angle (which was outside the viewing angles for the monitor) and from a large distance the pulsing marker was still reliably detectable within 5 frames (0.2 seconds) with 90 % probability.

Pulsing marker gave better results than static transparent marker.

The results indicate that the pulsing marker gave consistently better results then the static marker, especially at the extreme pitch angles (150°, 70°). This is mainly due to the fact that it is tricky to calibrate the correct transparency level for the static marker that is unobtrusive but still detectable by the smart-phone camera for all the possible display pitch angles. Notice that histograms for the pulsing markers show two main peaks (Figure 91 – 70-near-pulsing, 150-far-pulsing). The first one is near zero, since while the marker transparency is lower, the marker is detected in each frame. The second peak marks the interval, where the marker is too transparent during the pulsing period for the algorithm to be able to detect it (in the 70-near-pulsing histogram it is at 15 dropped frames which is about the third of the pulsing period).

Another observation concerning the detection reliability is that the results get better with a larger area of the visible marker in the camera view. This follows from the continuous marker design lacking localization patterns and relying only on regular structure detection. However, since the user during real-world usage would usually want to get the client closer to the content provider for more precise region selection, we do not see this as a limitation. In the near future, the framework will also be extended so that the visible

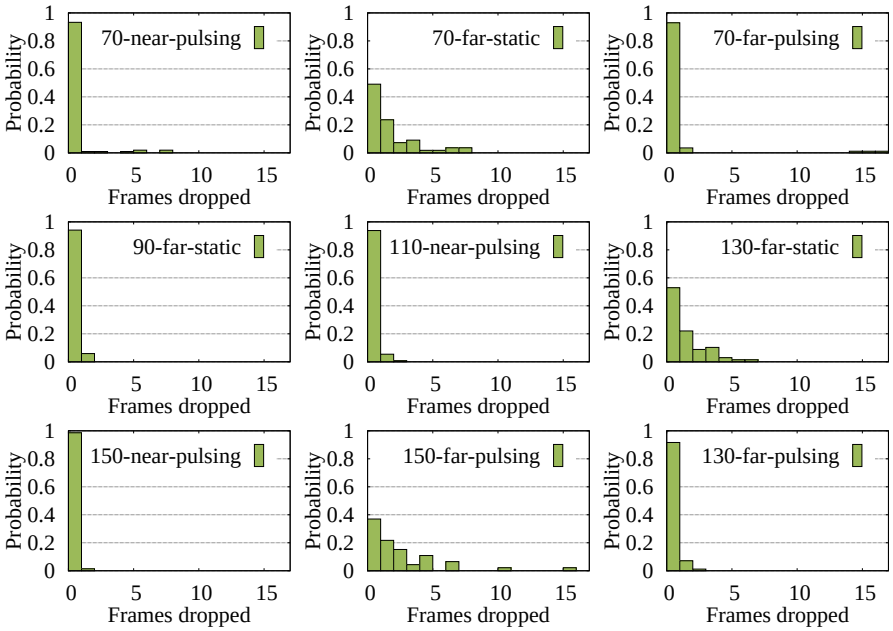


Figure 91: Selected results of the reliability test. The histograms show the probability distribution of the time (in frame count) between consecutive successful position identifications. The name of the histogram corresponds to the pitch angle, distance from the content provider plane and whether the marker was pulsing (the pulsing period calculated in frames was 37.5 frames).

marker region on the content provider would be resized dynamically based on the feedback from the client.

Our system allows for direct task migration, without any direct interaction with the desktop system – straight from the mobile device. Thanks to the usage of the marker field, the detection is reliable, because the system ensures reliable and easily detectable keypoints in the monitor screen. At the same time, the marker field proves to be unobtrusive and aesthetically tolerable by the user.

11

EFFORTLESS CHROMAKEYING

In the journal article "Poor Man's Virtual Camera: Real-Time Simultaneous Matting and Camera Pose Estimation" published in *IEEE Computer Graphics and Applications Magazine 2016* we introduce modification and use of Shades of Grey (SoG) as greenscreen for use in filmmaking industry.

Since the early years of filmmaking, the artists want to create artificial worlds to bring their ideas onto a film screen. Several approaches were developed over time, e.g. glass paintings, double exposure or using optical printer [72]. Nowadays, the most commonly used method is "chroma keying", where a specially selected color (often green or blue) is set to be transparent and substituted with a digital content [73]. Due to the use of green color, this method is also often called greenscreen.

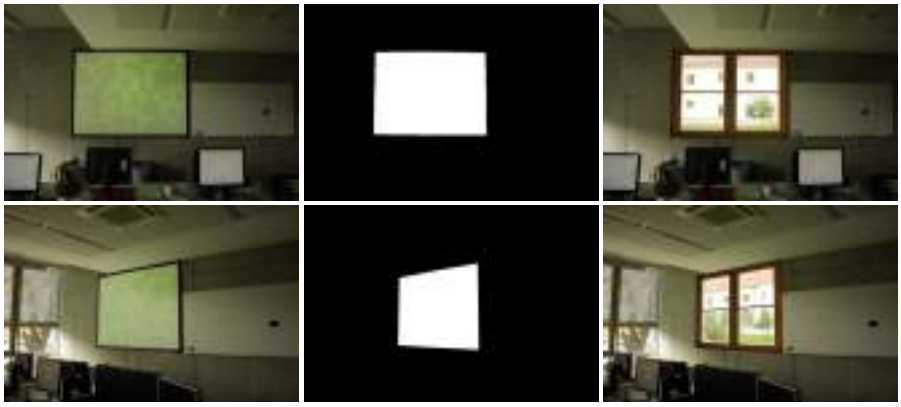


Figure 92: Matchmoving by the Uniform Marker Field (UMF). **Left:** Image captured by the camera. **Middle:** Alpha matte. **Right:** Composite image with 3D scene rendered to match the camera pose.

The camera pose can be determined by using sensors mounted to the camera (e.g. Insight VCS1[1]) or by camera rigs that can be programmed to follow a predefined track (e.g. Cyclops or Milo control rigs from Mark Roberts Motion Control[2] or TechnoDolly[3]). Camera movement recovery is a technique which estimates the movement using markers or keypoints placed and detected on the mating plate[4]. This process, also called matchmoving, often involves a considerable amount of manual work in order to match and annotate the markers.

1 http://www.naturalpoint.com/optitrack/products/insight-vcs/
2 http://www.mrmoco.com
3 http://www.supertechno.com/product/technodolly.html
4 http://www.fxguide.com/ – Art of Tracking

Using k-ary UMF enabled possibility to use such UMFs as greenscreen.

The crucial step from the UMFs towards the Shades of Grey (SoG) (improved UMF) enabled it to be used for chroma keying: the contrasts between the individual modules of the markers can be fairly low and the marker field is still reliably detectable. Therefore very similar colors (in this case shades of green) can be used. This Shades of Green marker field can be easily removed using some of the various methods for removal of background with constant color and at the same time provides instant and effortless camera pose estimation.

11.1 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 $A = G - \max(B, R)$ [74], 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 whole marker). However, more advanced algorithms are able to deal also with changing of the intensity of the green color (with constant hue). Since the changes in intensities are more suitable for the detection of the marker field, in our experiments we use this approach, instead of using different hues.

Once the marker field is detected in the image, its layout can be used for predicting exact shades of green at different locations in the camera image. These shades can be used for precise matte segmentation.

11.2 PRACTICAL SETUPS

PROJECTION

Ordinary 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 projection (Figure 93). 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.

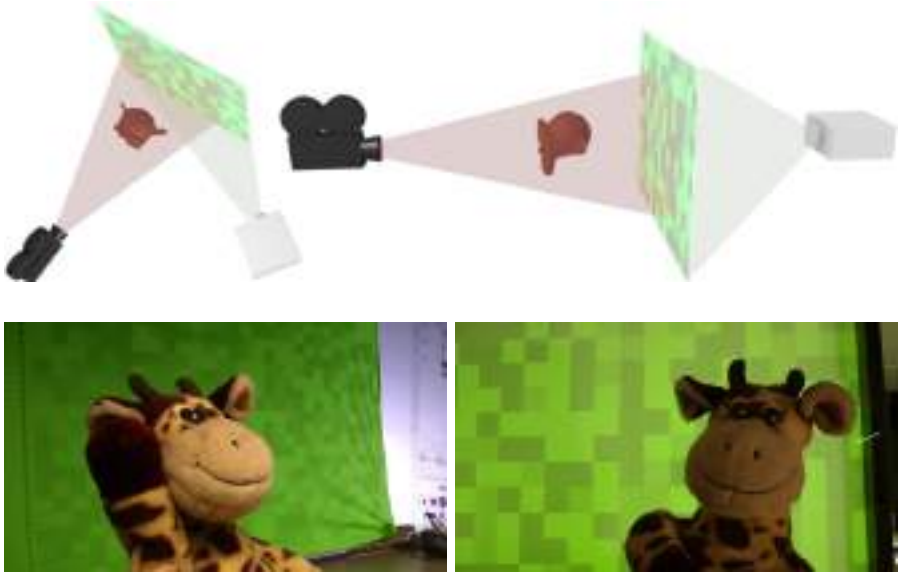


Figure 93: Projection of the marker field on green (or white or other) canvas. **Left:** Front projection – the actor's movement is limited because he must not interfere with the projection rays. **Right:** Back projection – requires more space behind the greenscreen.

SPECIAL CANVAS

An alternative to the projection of the marker field is using a special canvas with the marker field printed on it (Figure 94 top). It is possible to synthesize marker fields of arbitrary dimensions by selecting suitable kernel shape/size and a proper number of colors. The marker field needs not cover the whole matting plate; instead, only a stripe or other shape can be covered by the identifiable marker field (Figure 94 bottom). Using several sheets requires mutual and world calibration which can be done completely automatically by quickly scanning over the scene with a camera.

11.3 SAMPLE USE CASES

Our technology for online, cheap and effortless matchmoving opens space for creative expression of filmmakers. We prepared two sample applications which we are planning to release as mobile/tablet applications.

VIRTUAL CAMERA AND SIMULCAM

Thanks to the very simple, fast and precise detection algorithm, live match-moving of the camera can be done in real-time on a mobile device (tablet or smartphone). Any member of the film crew can instantly check the setting

Figure 94: Using a printed canvas as the matting plate with localization capability. **Top:** The whole canvas is covered by the marker field. **Bottom two:** A long stripe is placed over a standard green canvas for horizontal panoramic movement of the camera. Other shapes and multiple marker fields can be used as well.

Figure 95: **Top:** Virtual camera using the marker field greenscreen. **Bottom:** A simulcam concept: a tablet processes live video from a professional camera and renders the augmented preview.

of the physical scene and its match to the planned augmenting 3D model from virtually any angle without using any special equipment (Figure 95 top, Figure 96).

Embedded smartphone or tablet cameras are not suitable for capturing production quality video. The marker field detection algorithm is easily implementable just by fixed-point numbers and it is efficient. This makes it suitable for embedding into the camera firmware. Alternatively, the preview can be running on a lightweight tablet and the video signal can be fed (by firewire, USB or wirelessly) from the actual high-quality camera (Figure 95 bottom).

LIVE STORYBOARDING AND SHOT PROTOTYPING

Another use case is application for online shot prototyping and live preparation of storyboards. This app is targeted to high-end tablet devices. It includes a

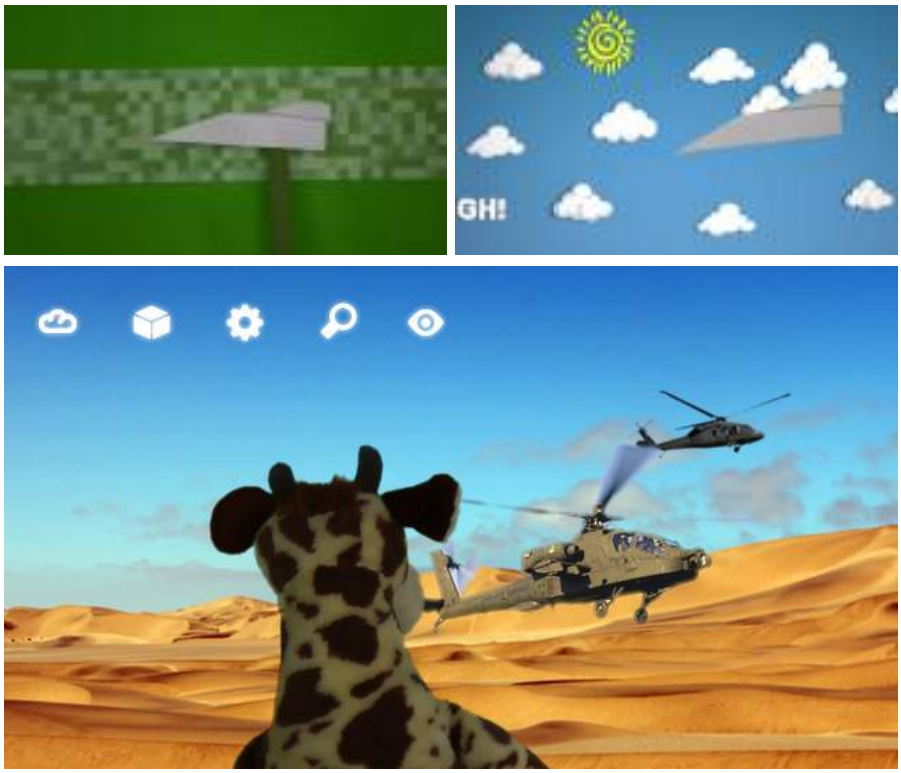


Figure 96: Sample Virtual Camera images and shots from the Live Storyboarding.

database with many images and 3D models tagged by semantic annotations and divided into thematic categories (the user can use her own 2D/3D content as well). The user (film author, script editor, 3D film artist) replaces the matted background with 2D (360° background photo) and 3D (models, animated figures) content. The interface is inspired by simple and intuitive 3D editors such as the SketchUp (Figure 96 bottom).

12

VIDEO GAMING ON A UMF CARPET

PROJECT RESOUND

In 2013 me, Rudolf Kajan, István Szentandrási, and Tomáš Milet participated in the Microsoft's software competition called Imagine Cup. Our entry was a game called Project reSound. We won the Czech national round and we advanced into the worldwide finals where we placed ourselves among TOP5 teams. More than a game it was a showcase of UMF technology and good opportunity to improve our UMF design and detection algorithm.

Figure 97: Project reSound gameplay on the UMF carpet.

It was also a good reason for manufacturing a big UMF carpet (Figure 98) and test several possible control schemes for the game and the control of an AR application in general.

12.1 PROJECT RESOUND

Music disappeared from the world.

The story of the game tells that music suddenly disappeared from the world. The player's mission is to get it back using special "resonator plates" (UMF carpet and other UMF markers) and a high-tech "reSound device" (tablet with the game). The music is represented by particle streams on the gameboard. Each stream can have a different color and each color means different part

Figure 98: UMF carpet designed and manufactured for the Project reSound game (it has been of course used for another applications too). The carpet is 1.5 m wide and 2.4 m long with modules 7.5 cm big.

(music sample) of the complete song. The player must use stream influencing game pieces to direct the particles into the appropriate bins. More particles in the bin means more volume for the corresponding sample. When all the bins are full, samples, that are now at maximum volume, create harmony and the song is complete. The player brought back one song to the world and can move to another level.

The game was implemented entirely in Microsoft XNA and after the Imagine Cup it was partially reimplemented using the popular game engine Unity. As a byproduct, we now have UMF detector plugin for Unity.

12.2 AR GAME CONTROL SCHEMES

AR game control schemes – "crosshair" scheme

First we created a control scheme based on a "crosshair" in the middle of the tablet screen. To move the game piece around the gameboard, the user was supposed to point the tablet at it, press the big virtual button on game screen, thus grabbing the piece, move it to the desired location and put it back on the board by releasing the button. The tablet we had at that time was quite heavy and we felt that this would be ideal control scheme because through the entire game the user could hold the tablet with both hands. Virtual button was very close to the screen edge so it could be pressed just with the thumb.

Classical "touch" scheme

But today's mobile User Interfaces (UIs) and User Experience (UX) are based on touch. Almost every user that played our game tried to move the game pieces by drag-and-dropping them. Even after adding a tutorial that aimed at explaining the "crosshair" scheme, majority of users had problems playing the game and tried to use touch gestures they are used to from mobile

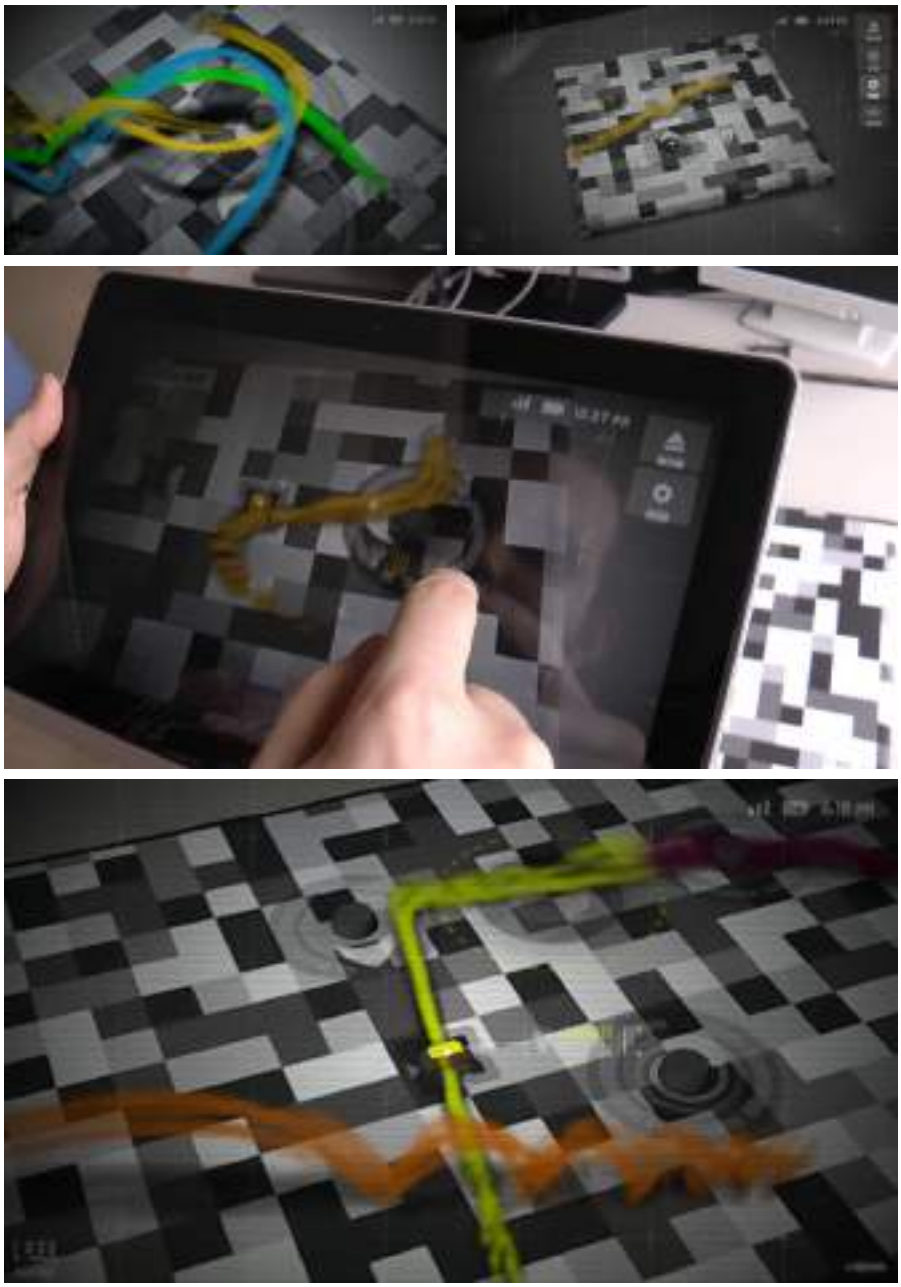


Figure 99: Photos and screenshot of Project reSound gameplay.

phones. Therefore, we implemented the classic drag-and-drop (for moving the pieces) and pinch (for changing the radius of pieces' influence) gestures.

Of course, this scheme had one big disadvantage – the heavy tablet had to be held only with one hand while the other controlled the game. It was very hard to endure it throughout the entire game and users complained about that. But contemporary tablets and mobile phones are much lighter and should not present such stress on the holding hand.

12.3 DETECTION FAILING USER NOTIFICATION

Letting the user know that the detector is failing in some user-friendly way is crucial.

Another very useful conclusion came out from users testing the game. Letting the user know that the detection algorithm is failing is crucial. Detector glitches can be caused for example by that the marker is not in the camera image, there is too extensive motion blur or marker is occluded too much even for the UMF. Therefore, when this happens, game pieces and controls are hidden and static noise effect is shown, thus notifying the user that something is wrong and he should try to fix the problem (Figure 100). Players confirmed that this is definitely better and more user-friendly than serious game glitches (e.g. big jumps of the game objects, wrong perspective. . .).

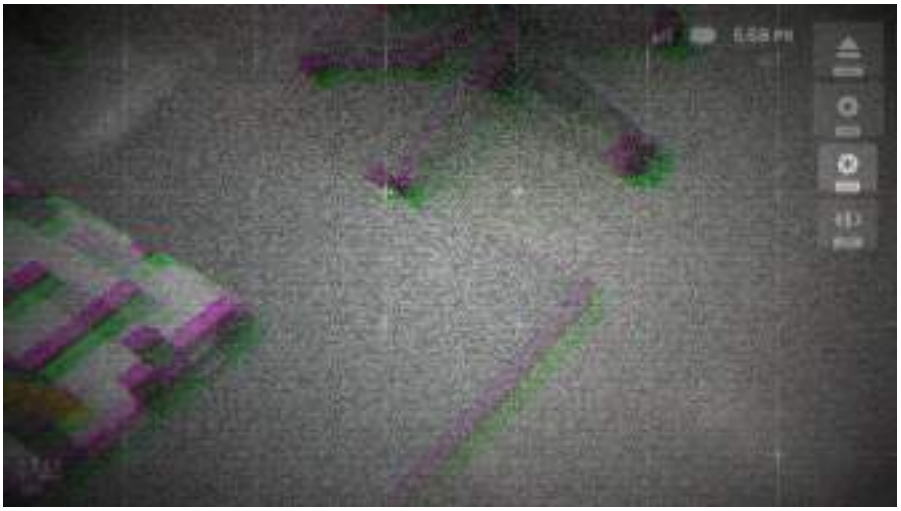


Figure 100: Static noise effect that tells the user that the marker detector is failing and his intervention is needed.

12.4 FURTHER IMPLEMENTATIONS OF THE IDEA

Gaming carpets in kindergartens and children's playgrounds.

Of course the idea of the UMF carpet does not have to be limited to specific carpet. It could be used in much larger environments, such as kindergartens and children's playground. Carpet can be manufactured to match the room in color and shape so it is continuous though the entire room. Children could play educational games just with their phones. An advantage of this system is that multiple games could be played on a single carpet by simply installing it on the kids' devices. Different kids could even play different games at the same time.

Part IV

IDEAS, FUTURE WORK & CONCLUSION

13

IDEAS AND FUTURE WORK

In this chapter, I would like to present several other possible applications for the proposed marker designs. These ideas are not yet published and can be definitely put into the "future work" category. Some of them are in the proof of concept state and some are really just ideas. They serve as a starting point for my future work.

1. **Tabletop Scene Interaction** (Section 13.1).
   Uniform Marker Fields (UMFs) are ideal for tabletop environments, which are usually very heterogeneous, because of their high resistance to occlusion and controllable looks.

2. **Unmanned Aerial Vehicle Landing** (Section 13.2).
   Visual tracking of Unmanned Aerial Vehicle (UAV) pose is quite essential task with many applications, such as autonomous navigation and scene acquisition. Fractal Marker Fields (FMFs) are ideal for navigation during landing.

3. **Digital Changing Room** (Section 13.3).
   Pieces of clothing (T-shirts, hats, trousers. . .) can be manufactured with Honeycomb Marker Field (HMF) printed on them. Such clothing can be digitally augmented with anything we want.

4. **UMF Generator Controlled with Image Pattern** (Section 13.4).
   We have also tested the possibility of creating UMFs which look as much as it is possible as an image pattern.

13.1 TABLETOP SCENE INTERACTION

One strong application of near-eye see-through glasses (recently becoming generally available – Microsoft's HoloLens[1], DAQRI's Smart Helmet[2]) is augmenting interaction in tabletop and living room scenarios [1]. These possibilities were tested even by Kato in his early experiments with AR in paper "Virtual object manipulation on a table-top AR environment" [75].

Tracking of keypoints can be used for the camera pose estimation, but tabletop environment is often very heterogeneous and the presence of a visually unobtrusive and cheaply detectable marker field can provide a reliable starting point for the tracking and offload some of the expensive computation. UMF is very suitable for this kind of application because of it huge resistance to occlusions and controllable looks (Figure 101).

Figure 101: UMFs are highly resistant to occlusion and therefore are ideal for tabletop applications. Red cup, dices and chess pieces are augmented to the scene. Everything else is the original image.

Specific applications could range from taking virtual notes on the tabletop, through seeing and organizing video conference calls like Kato and Billinghurst intended with their ARTag, to complete desktop ecosystems with widget-like applications, ranging from tiny to full-scale apps.

Special case of this application can be visual guidance for industrial robotic arms. Work table or conveyor belt could be entirely covered with UMF and help with the precise actuators positioning. This system is currently used by my department colleagues. They are exploring the possibilities of human-robot interaction and created interaction table called AR®Table with Willow Garage PR2 robot[3] and projected UMF.

1 https://www.microsoft.com/microsoft-hololens
2 http://daqri.com/home/product/daqri-smart-helmet/
3 https://www.willowgarage.com/pages/pr2/overview

Figure 102: Tabletop interaction examples. The marker field covers the whole table surface and fosters the camera pose estimation.

13.2 UNMANNED AERIAL VEHICLE LANDING

With the availability of low-cost Micro Aerial Vehicles (MAVs), Unmanned Aerial Vehicles (UAVs) gained popularity very quickly and visual tracking of position and orientation is essential task when any kind of autonomous behavior is expected. Using a Fractal Marker Field (FMF) as a landing zone would allow for an initially high altitude detection but continuous pose estimation as the aircraft descends on the landing zone. Similar approach was used in [76, 77].

For small UAVs, like commercial drones and quadcopters that are capable of vertical take offs and landings, a single FMF with the size of smallest single marker (highest level) equals to about $20 - 30$ cm should be sufficient. If we should construct such FMF with 6 levels, the L0 marker (lowest level, biggest marker) would be 10 m big (Figure 103).

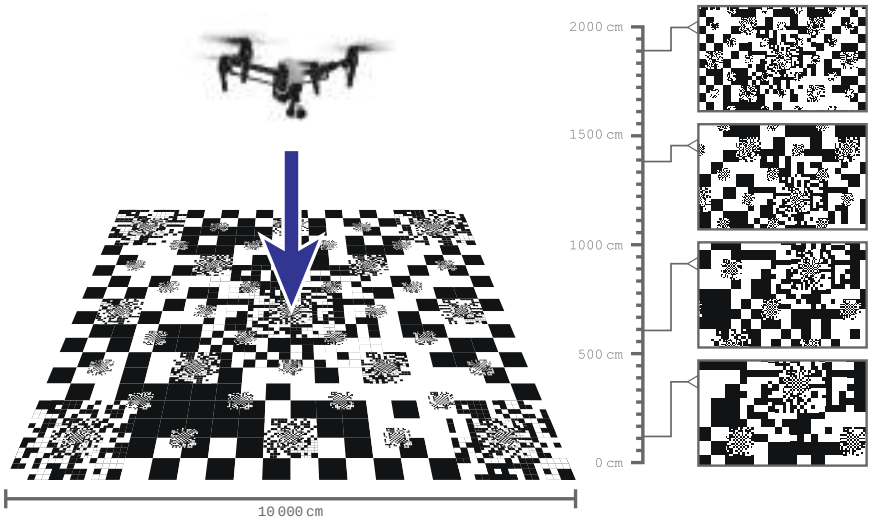


Figure 103: Vertical landing of UAV using single FMF for visual guidance. Right side of the figure shows approximation of which part and level of FMF can camera mounted on UAV see. At least one marker can be detected at any altitude.

Bigger unmanned aircrafts, e.g. American military MQ-1 Predator or MQ-9 Reaper, that have to land conventionally (cannot take off and land vertically) could also utilize the FMF aided landing. Landing stripe could be covered with series of FMFs, each with different ID. Let's assume that runway is 2040 m long and 60 m wide. If the single FMF would be also 60 m wide, then we would need 34 marker fields for entire length of the runway. That is very far away from the maximum possible number of marker fields ($2^{16} = 65\,536$).

13.3 DIGITAL CHANGING ROOM

There is a lot of "print your own T-shirt" (or hats, trousers, underpants. . .) businesses on the market since the printing technology and hardware has become available for wider the public. They print on-demand, that means that they do not hold big stock, but they make the T-shirt after a customer orders. The problem is that the customer cannot try the T-shirt before ordering.

The idea is that they could print one T-shirt with Honeycomb Marker Field (HMF). Utilizing HMF deformability, algorithm could reconstruct the clothing surface and augment it with anything we want, e.g. different designs that can be later ordered and printed exactly as it was shown in this digital changing room. A camera is capturing the scene in front of the projection screen with the person wearing the HMF T-shirt. Camera image is then processed, HMF is detected and replaced with the new image. Following figure shows the illustration of this setup for digital changing room.

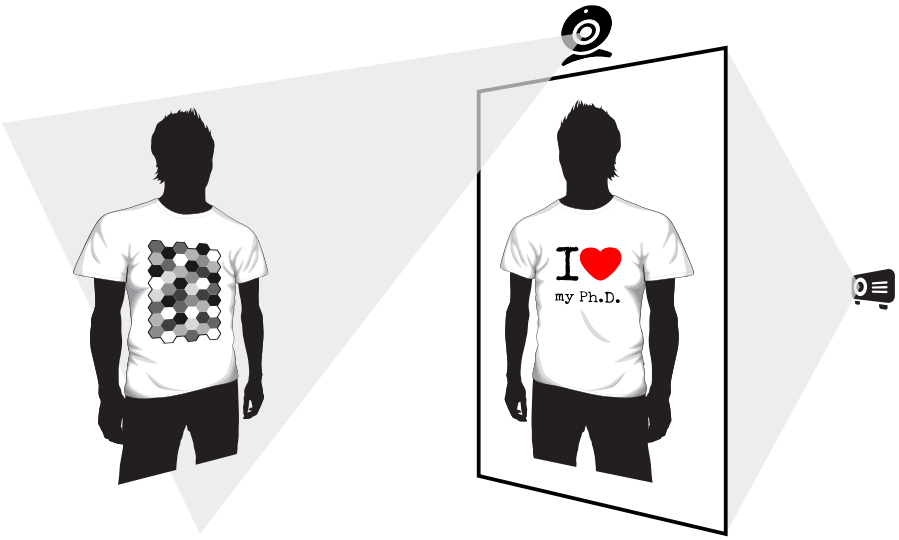


Figure 104: Possible setup for digital changing room.

13.4 UMF GENERATOR CONTROLLED WITH AN IMAGE PATTERN

Uniform Marker Field (UMF) generator can be modified so it uses image pattern as an input. It extracts the most used colors in the pattern and uses the defined number of them as colors for marker field modules. Fitness function is then modified in the way that it takes the distance of original color in the pattern to color of the UMF module into account. If more colors from the pattern are extracted, the resulting marker field is more similar to the pattern but is harder to detect because of less sharp gradients between the modules. That means there is a trade-off between similarity to the pattern and detectability.

The resulting image is similar to the pattern as much as possible (depends on the mentioned trade-off setting) and simultaneously it has all properties of proper UMF. This idea is already implemented as prototype in our UMF generator. Figure 105 shows the generated examples and depicts the trade-off.

Figure 105: Image pattern controlled UMF generation – ŠKODA Auto a.s. logo and Chrome web browser icon. **Left:** Original image pattern. **Middle:** Lower number of colors used. **Right:** Higher number of colors used.

14

CONCLUSION

This Ph.D. thesis shows that fiduciary markers are still live and important research topic, not just in Augmented Reality (AR) field but also in robot navigation and pathfinding, motion capture, and many other fields. Even though the markerless techniques are more and more sophisticated, markers can still offer advantages like high precision and reliability.

In our research, we have designed and implemented different marker fields to overcome many of today's markers' limitations: Fractal Marker Fields (FMFs) – no more problems with detection at any distance; Uniform Marker Fields (UMFs) – no white safe zones needed between markers, therefore no problem with marker coverage in the scene; improved Uniform Marker Fields (UMFs) – denser, larger, easily detectable, aesthetic and Honeycomb Marker Fields (HMFs) – all properties of improved UMF but no limitation to planar surfaces, i.e. deformable.

It would have been pointless to create new marker systems with such properties if they did not have any practical applications. And therefore this work (and our research) also shows that our marker fields have many. From aesthetically appealing indoor navigation, through screen-to-screen task migration to effortless and cheep chromakeying with simultaneous pose estimation.

This work and our research changed the view on visual fiduciary markers, what can be done with them and removed some of the major disadvantages that were deal breaker for their use in some applications.

REFERENCES

[1] Andrei State, Gentaro Hirota, David T. Chen, William F. Garrett, and Mark A. Livingston. “Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking.” In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: ACM, 1996. ISBN: 0-89791-746-4.

[2] Markéta Dubská, Adam Herout, and Jiří Havel. “Real-Time Precise Detection of Regular Grids and Matrix Codes.” In: *Journal of Real-Time Image Processing* 2013.1 (2013). ISSN: 1861-8200.

[3] Markéta Dubská, Adam Herout, and Jiří Havel. “PClines – Line Detection Using Parallel Coordinates.” In: *Proceedings of Computer Vision and Pattern Recognition (CVPR) 2011*. 2011.

[4] Richard O. Duda and Peter E. Hart. “Use of the Hough transformation to detect lines and curves in pictures.” In: *C. ACM* 15 (1972). ISSN: 0001-0782.

[5] A. Wald. “Sequential Tests of Statistical Hypotheses.” In: *The Annals of Mathematical Statistics* 16.2 (1945).

[6] J. H. Halton. “Algorithm 247: Radical-inverse quasi-random point sequence.” In: *Commun. ACM* 7 (12 1964). ISSN: 0001-0782.

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

[8] Mark Fiala. “Designing Highly Reliable Fiducial Markers.” In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32 (7 2010). ISSN: 0162-8828.

[9] Mark 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. Washington, DC, USA: IEEE Computer Society, 2005. ISBN: 0-7695-2372-2.

[10] Hirokazu Kato and Mark Billinghurst. “Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System.” In: *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 1999. ISBN: 0-7695-0359-4.

[11] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. "Virtual object manipulation on a table-top AR environment." In: *Proceedings of IEEE and ACM International Symposium on Augmented Reality*. 2000.

[12] I. S. Reed and R. M. Stewart. "Note on the Existence of Perfect Maps." In: *IRE Transactions on Information Theory* 8 (1962).

[13] Stephen G. Hartke. "Binary De Bruijn cycles under different equivalence relations." In: *Discrete Mathematics* 215 (2000). ISSN: 0012-365X.

[14] Keith M. Martin, Z.D. Dai, M.J.B. Robshaw, and P.R. Wild. "Orientable Sequences." In: *Proceedings of the 3rd IMA Conference on Cryptography and Coding*. 1993.

[15] Martin Hirzer. *Marker Detection for Augmented Reality Applications*. Tech. rep. Inst. for Comp. Graphics and Vision, Graz Univ. of Tech., AT, 2008.

[16] Frederik Schaffalitzky and Andrew Zisserman. "Planar Grouping for Automatic Detection of Vanishing Lines and Points." In: *Image and Vision Computing* 18 (2000).

[17] Wouter Pasman, Charles Woodward, Mika Hakkarainen, Petri Honkamaa, and Jouko Hyväkkä. "Augmented Reality with Large 3D Models on a PDA - Implementation, Performance and Use Experiences." In: *Proc. VRCAI 2004*. 2004.

[18] Hideaki Uchiyama and Eric Marchand. "Toward augmenting everything: Detecting and tracking geometrical features on planar objects." In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2011. ISBN: 978-1-4577-2183-0.

[19] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. "CALTag: High Precision Fiducial Markers for Camera Calibration." In: *Proceedings of VMV*. 2010.

[20] Hideaki Uchiyama and Eric Marchand. "Deformable Random Dot Markers." In: *Proceedings of ISMAR* (2011).

[21] Hideaki Uchiyama and Eric Marchand. "Toward augmenting everything: Detecting and tracking geometrical features on planar objects." In: *Proceedings of ISMAR*. 2011.

[22] H. Uchiyama and H. Saito. "Random Dot Markers." In: *IEEE Virtual Reality Conf. (VR)*. 2011.

[23] Grace Woo, Andy Lippman, and Ramesh Raskar. "VRCodes: Unobtrusive and Active Visual Codes for Interaction by Exploiting Rolling Shutter." In: *Proceedings of ISMAR*. 2012.

[24] J. Foster. *The Green Screen Handbook: Real-World Production Techniques*. The Green Screen Handbook: Real-world Production Techniques v. 978, nos. 0-52106. John Wiley & Sons, 2010. ISBN: 9780470521076.

[25] Oliver Bimber; Ramesh Raskar. *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A K Peters/CRC Press, 2005.

[26] P. S. Heckbert. "Color Image Quantization for Frame Buffer Display." In: *ACM Computer Graphics (ACM SIGGRAPH '82 Proceedings)* 16.3 (1982).

[27] A. Dekker. "Kohonen neural networks for optimal colour quantization." In: *Network: Computation in Neural Systems* 5 (1994).

[28] H. Uchiyama and E. Marchand. "Deformable random dot markers." In: *Proc. ISMAR 2011*. 2011. ISBN: 978-1-4577-2183-0.

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

[30] Simon Green. "Particle simulation using CUDA." In: *NVIDIA Whitepaper, December 2010* (2010).

[31] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection." In: *Proc. ECCV 2006*. 2006.

[32] Edward Rosten and Tom Drummond. "Fusing points and lines for high performance tracking." In: *Proc. ICCV 2005*. 2005.

[33] S. Leutenegger, A. Melzer, K. Alexis, and R. Siegwart. "Robust state estimation for small unmanned airplanes." In: *Control Applications (CCA), 2014 IEEE Conference on*. 2014.

[34] P. Paces and J. Suchy. "Statistical evaluation of multiple low-cost MEMS sensors for altitude measurement." In: *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*. 2014.

[35] Zhiwei Zhu, Taragay Oskiper, Supun Samarasekera, Rakesh Kumar, and H.S. Sawhney. "Ten-fold Improvement in Visual Odometry Using Landmark Matching." In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. 2007.

[36] Han-Pang Chiu, S. Williams, F. Dellaert, S. Samarasekera, and R. Kumar. "Robust vision-aided navigation using Sliding-Window Factor graphs." In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. 2013.

[37] Taragay Oskiper, Supun Samarasekera, and Rakesh Kumar. "Multi-sensor navigation algorithm using monocular camera, IMU and GPS for large scale augmented reality." In: *11th IEEE International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2012.

[38] A. Assa and F. Janabi-Sharifi. “A Kalman Filter-Based Framework for Enhanced Sensor Fusion.” In: *Sensors Journal, IEEE* (2015). ISSN: 1530-437X.

[39] Zhengyou Zhang. “A flexible new technique for camera calibration.” In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11 (2000). ISSN: 0162-8828.

[40] He Zhao and Zheyao Wang. “Motion Measurement Using Inertial Sensors, Ultrasonic Sensors, and Magnetometers With Extended Kalman Filter for Data Fusion.” In: *Sensors Journal, IEEE* (2012). ISSN: 1530-437X.

[41] Georg Klein and David Murray. “Parallel Tracking and Mapping for Small AR Workspaces.” In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR ’07. Washington, DC, USA: IEEE Computer Society, 2007. ISBN: 978-1-4244-1749-0.

[42] M.C. Vaz and C. Barron. *The Invisible Art: The Legends of Movie Matte Painting*. Chronicle Books, 2002. ISBN: 978-0811831369.

[43] S. Wright. *Digital compositing for film and video*. Focal Press, 2010. ISBN: 978-0240813097.

[44] Elizabeth Bales, Timothy Sohn, and Vidya Setlur. “Planning, apps, and the high-end smartphone: exploring the landscape of modern cross-device reaccess.” In: *Proceedings of the 9th international conference on Pervasive computing*. Pervasive’11. San Francisco, USA: Springer-Verlag, 2011. ISBN: 978-3-642-21725-8.

[45] Amy K. Karlson, Shamsi T. Iqbal, Brian Meyers, Gonzalo Ramos, Kathy Lee, and John C. Tang. “Mobile taskflow in context: a screenshot study of smartphone usage.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Atlanta, Georgia, USA: ACM, 2010. ISBN: 978-1-60558-929-9.

[46] David Dearman and Jeffery S. Pierce. “It’s on my other computer!: computing with multiple devices.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI 2008. Florence, Italy: ACM, 2008. ISBN: 978-1-60558-011-1.

[47] Tsung-Hsiang Chang and Yang Li. “Deep shot: a framework for migrating tasks across devices using mobile phone cameras.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI 2011. Vancouver, BC, Canada: ACM, 2011. ISBN: 978-1-4503-0228-9.

[48] Yuichiro Fujimoto, Ross T. Smith, Takafumi Taketomi, Goshiro Yamamoto, Jun Miyazaki, Hirokazu Kato, and Bruce H. Thomas. "Geometrically-Correct Projection-Based Texture Mapping Onto a Deformable Object." In: *IEEE Transactions on Visualization and Computer Graphics* 20.4 (2014). ISSN: 1077-2626.

[49] H. Tanaka, Y. Sumi, and Y. Matsumoto. "A Visual Marker for Precise Pose Estimation based on Lenticular Lenses." In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012.

[50] H. Tanaka, Y. Sumi, and Y. Matsumoto. "A novel AR marker for high-accuracy stable image overlay." In: *Consumer Electronics (GCCE), 2012 IEEE 1st Global Conference on*. 2012.

[51] H. Tanaka, Y. Sumi, and Y. Matsumoto. "A high-accuracy visual marker based on a microlens array." In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 2012.

[52] H. Tanaka, Y. Sumi, and Y. Matsumoto. "Further stabilization of a microlens-array-based fiducial marker." In: *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. 2013.

[53] H. Tanaka, Y. Sumi, and Y. Matsumoto. "A solution to pose ambiguity of visual markers using Moiré patterns." In: *Intelligent Robots and Systems (IROS 2014)*. 2014.

[54] F. Bergamasco, A. Albarelli, E. Rodola, and A. Torsello. "RUNE-Tag: A High Accuracy Fiducial Marker with Strong Occlusion Resilience." In: *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR 2011. Washington, DC, USA: IEEE Computer Society, 2011. ISBN: 978-1-4577-0394-2.

[55] Jun Rekimoto and Yuji Ayatsuka. "CyberCode: designing augmented reality environments with visual tags." In: *Proceedings of DARE 2000 on Designing augmented reality environments*. DARE '00. Elsinore, Denmark: ACM, 2000.

[56] Martin Kaltenbrunner and Ross Bencina. "reacTIVision: a computer-vision framework for table-based tangible interaction." In: *Proceedings of the 1st international conference on Tangible and embedded interaction*. TEI 2007. Baton Rouge, Louisiana: ACM, 2007. ISBN: 978-1-59593-619-6.

[57] Ross Bencina and Martin Kaltenbrunner. "The Design and Evolution of Fiducials for the reacTIVision System." In: *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts (3rd Iteration 2005)*. Melbourne, Australia, 2005.

[58] R. Bencina, M. Kaltenbrunner, and S. Jorda. “Improved Topological Fiducial Tracking in the reacTIVision System.” In: *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. 2005.

[59] M. Kaltenbrunner, S. Jorda, G. Geiger, and M. Alonso. “The reacTable*: A Collaborative Musical Instrument.” In: *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE ’06. 15th IEEE International Workshops on*. 2006.

[60] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. “CALTag: High Precision Fiducial Markers for Camera Calibration.” In: *15th International Workshop on Vision, Modeling and Visualization*. Siegen, Germany, 2010.

[61] Keisuke Tateno, Itaru Kitahara, and Yuichi Ohta. “A Nested Marker for Augmented Reality.” In: *ACM SIGGRAPH 2006 Sketches*. SIGGRAPH ’06. Boston, Massachusetts: ACM, 2006. ISBN: 1-59593-364-6.

[62] K. Tateno, I. Kitahara, and Y. Ohta. “A Nested Marker for Augmented Reality.” In: *Virtual Reality Conference, 2007. VR ’07. IEEE*. 2007.

[63] J. Rekimoto. “Matrix: a realtime object identification and registration method for augmented reality.” In: *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*. 1998.

[64] Didier Stricker, Gundrun Klinker, and Dirk Reiners. “A fast and robust line-based optical tracker for augmented reality applications.” In: *Proceedings of the international workshop on Augmented reality: placing artificial objects in real scenes: placing artificial objects in real scenes*. IWAR ’98. Bellevue, Washington, USA: A. K. Peters, Ltd., 1999. ISBN: 1-56881-098-9.

[65] Youngkwan Cho Jongweon, Youngkwan Cho, Jongweon Lee, and Ulrich Neumann. “A Multi-ring Color Fiducial System and A Rule-Based Detection Method for Scalable Fiducial-Tracking Augmented Reality.” In: *Proceedings of International Workshop on Augmented Reality*. 1998.

[66] Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura. “Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval.” In: *In Lecture Notes in Computer Science (7th International Workshop DAS2006*. Springer, 2006.

[67] Jianke Zhu and M.R. Lyu. “Progressive Finite Newton Approach To Real-time Nonrigid Surface Detection.” In: *Computer Vision and Pattern Recognition, 2007. CVPR ’07. IEEE Conference on*. 2007.

[68] Yuko Uematsu and Hideo Saito. “Improvement of Accuracy for 2D Marker-Based Tracking Using Particle Filter.” In: *Proceedings of the 17th International Conference on Artificial Reality and Telexistence*. ICAT ’07. Washington, DC, USA: IEEE Computer Society, 2007. ISBN: 0-7695-3056-7.

[69] Daniel F. Abawi, Joachim Bienwald, and Ralf Dorner. “Accuracy in Optical Tracking with Fiducial Markers: An Accuracy Function for ARToolKit.” In: *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR ’04. Washington, DC, USA: IEEE Computer Society, 2004. ISBN: 0-7695-2191-6.

[70] Liming Yang, Jean-Marie Normand, and Guillaume Moreau. “Robust Random Dot Markers: Towards Augmented Unprepared Maps with Pure Geographic Features.” In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. VRST ’14. Edinburgh, Scotland: ACM, 2014. ISBN: 978-1-4503-3253-8.

[71] L. Chen, H. Fu, W. H. Andy Li, and C. L. Taj. “Scalable maps of random dots for middle-scale locative mobile games.” In: *Virtual Reality (VR), 2013 IEEE*. 2013.

[72] C. Nitschke. “Marker-based tracking with unmanned aerial vehicles.” In: *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*. 2014.

[73] C. Nitschke, Y. Minami, M. Hiromoto, H. Ohshima, and T. Sato. “A quadrocopter automatic control contest as an example of interdisciplinary design education.” In: *Control, Automation and Systems (ICCAS), 2014 14th International Conference on*. 2014.

[74] Christian Pirchheim and Gerhard Reitmayr. “Homography-based planar mapping and tracking for mobile phones.” In: *ISMAR*. 2011. ISBN: 978-1-4577-2183-0.

[75] Robert O. Castle, Georg Klein, and David W. Murray. “Wide-area augmented reality using camera tracking and mapping in multiple regions.” In: *Computer Vision and Image Understanding* 115.6 (2011). ISSN: 1077-3142.

[76] Denis Chekhlov, Andrew P. Gee, Andrew Calway, and Walterio Mayol-Cuevas. “Ninja on a Plane: Automatic Discovery of Physical Planes for Augmented Reality Using Visual SLAM.” In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR ’07. Washington, DC, USA: IEEE Computer Society, 2007. ISBN: 978-1-4244-1749-0.

[77] Muriel Pressigout and Eric Marchand. “Model-Free Augmented Reality by Virtual Visual Servoing.” In: *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 2 - Volume 02*. ICPR ’04. Washington, DC, USA: IEEE Computer Society, 2004. ISBN: 0-7695-2128-2.

[78] Georg Klein and David Murray. “Parallel Tracking and Mapping on a Camera Phone.” In: *Proc. Eigth IEEE and ACM Int. Symposium on Mixed and Augmented Reality (ISMAR’09)*. 2009.

[79] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. “Bokode: Imperceptible Visual Tags for Camera Based Interaction from a Distance.” In: *ACM Trans. Graph.* 28.3 (2009). ISSN: 0730-0301.

[80] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. “Bokode: Imperceptible Visual Tags for Camera Based Interaction from a Distance.” In: *ACM SIGGRAPH 2009 Papers*. SIGGRAPH ’09. New Orleans, Louisiana: ACM, 2009. ISBN: 978-1-60558-726-4.

[81] M. Hossny, M. Hossny, and S. Nahavandi. “CARMa: Content augmented reality marker.” In: *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. 2013.

[82] G. Woo, A. Lippman, and R. Raskar. “VRCodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter.” In: *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*. 2012.