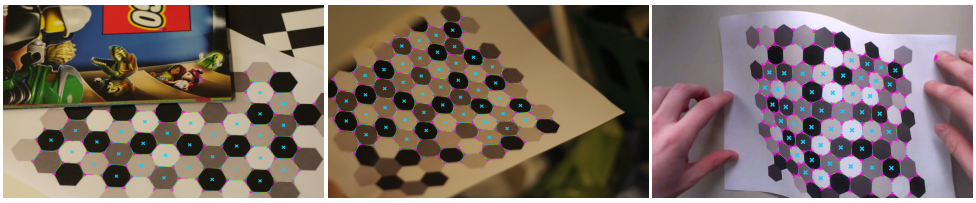
Design and Detection of Local Geometric Features for Deformable Marker Fields

Zsolt Horváth*, Adam Herout†, István Szentandrási, and Michal Zachariáš
Brno University of Technology



Abstract

A major limitation of contemporary fiduciary markers is that they are either very small (they try to represent a single point in the space) or they must be planar in order to be reasonably detectable. A deformable large-scale marker or marker field that would be efficiently detectable is the objective of this work.

We propose a design of such a marker field – the Honeycomb Marker Field. It is composed of symmetric hexagons, whose triplets of modules meet at "Y-junctions". We present an efficient detector of these image features – the Y-junctions. Thanks to the specific appearance of these synthetic image features, the algorithm can be very efficient – it only visits a small fraction of the image pixels in order to detect the Y-junctions reliably. The experiments show that compared to a general feature point detector (FAST was tested), the specialized Y-junctions detector offers better detection reliability.

CR Categories: I.4.6 [Image Processing and Computer Vision]: Segmentation—Edge and feature detection

Keywords: augmented reality, deformable marker field, local geometric features

1 Introduction

Augmented reality, robotic navigation, eye tracking and various other fields require real-time camera pose estimation based on visual information. Simple fiduciary markers (ARToolkit [Kato and Billinghurst 1999], ARTag [Fiala 2005]) are a popular and affordable solution. Marker-less solutions are becoming more and more popular recently [Wagner et al. 2008; Pan et al. 2009; Simon 2011] with the availability of sufficient computational power and with development of algorithms for fast tracking [Klein and Murray 2007; Newcombe et al. 2011] and feature point matching [Leutenegger et al. 2011; Yang and Cheng 2012].

Recent work of Szentandrási et al. [2012] offered the concept of *marker fields*. Marker field is an image pattern that can be efficiently detected in a camera image, then exactly localized, even when heavily occluded, under strong motion blur, and under bad lighting conditions. It consists of largely overlapping sub-windows that are unique within the marker field. The work of Herout et al. [2013] extended the marker fields to greyscale and color grids of squares. Firstly, this enabled generation of aesthetically pleasing designs of the marker fields. Even more importantly, it increased the descriptive power of separate sub-windows, enabling larger marker fields to be generated.

The biggest limitation of currently available marker fields [Herout et al. 2013] is thus the requirement for the marker field to be planar. The planarity assumption allows for the detection algorithm to be very efficient. However, for some application, it is restricting. In this work, we are looking for a marker field design that would be detectable even for non-planar and generally deformed surfaces. In particular, we are looking for such geometric features which could form a graph usable for identification of fractions of the field, and which would be detectable very efficiently.

One of the first marker based camera pose estimation techniques was introduced by Rekimoto et al. [Rekimoto 1998]. They used black and white fields to encode identification. The most well known square marker based system is ARToolkit [Kato and Billinghurst 1999]. Instead of binary information, they used arbitrary image placed into the marker to identify them. Fiala [Fiala 2005] introduced ARTag markers, which again used binary information with error correction to identify markers. He also replaced the adaptive thresholding step used by most systems with edge based marker detection providing better performance and also robustness against occlusion. Although square markers are the most common, there was also research on ring shaped markers [Cho et al. 1999] or even arbitrary shaped fiduciary markers [Kaltenbrunner and Bencina 2007] detectable by segmentation.

More recent research on fiduciary markers was done for example by Uchiyama and Saito [Uchiyama and Saito 2011], who introduced Random Dot Markers. Their system used the relative position between randomly placed points to identify the marker and estimate the camera pose. Marker fields were introduced by Herout et al. [Herout et al. 2012] and Szentandrsi et al. [Szentandrási et al. 2012]. Their Uniform Marker Fields were constructed based on de Bruijn tori, which enabled extremely efficient detection algorithm [Herout et al. 2013] suitable even for ultra-mobile devices. So far these fiduciary markers or marker fields were all assumed to

*e-mail: ihorvath@fit.vutbr.cz
†e-mail:herout@fit.vutbr.cz

SCCG 2013, Smolenice, Slovakia, May 1 – 3, 2013.
© 2013 ACM 978-1-4503-2480-9/13/0005 $15.00

be planar for stable detection.

Recently Uchiyama and Marchand [Uchiyama and Marchand 2011] described deformable Random Dot Markers detection algorithm. In the first step they assume the marker to be planar. They try to identify the marker and compute a RANSAC based homography of a base plane. This plane is then used further as base for non-rigid surface estimation. Random Dot Markers have several disadvantages. The recognition is sensitive to the size of the points, motion blur and defocus, 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 deformability of the marker. Similarly, deformable Random Dot Markers are only usable on non-elastic surfaces. These properties lead to unstable detection and tracking.

In recent years the attention of research shifted from marker based approaches to marker-less feature or model based techniques. Notable methods using a monocular camera in this area are monocular SLAM (Simultaneous Locatization and Mapping) based methods. This family of techniques use a prebuilt or continuously extended model of the environment. The model is then used to localize the camera in the environment using natural feature point matching and tracking. The disadvantage of these methods is high memory consumption and computational complexity. Recently Newcombe et al.[Newcombe et al. 2011] and Klein and Murray [Klein and Murray 2007] achieved real-time framerates for smaller environments using powerful GPUs.

One of the most important parts of marker-less techniques from performance and efficiency viewpoint is a fast feature point detector with high repeatability rate and an efficient feature descriptor and matching algorithm. Historically SIFT[Lowe 1999], SURF[Bay et al. 2006] and MSER[Donoser and Bischof 2006] are the most well performing feature detectors. An extremely fast machine learning based and widely used feature detector (FAST) with acceptable repeatability rate was introduced by Rosten and Drummond [Rosten and Drummond 2006].

Although both SIFT and SURF provide great performing descriptors, they are rarely used in real-time augmented reality applications due to large computational costs. Recent well performing and feature point descriptor, for example BRIEF [Calonder et al. 2010], BRISK [Leutenegger et al. 2011] or LDB [Yang and Cheng 2012], use binary descriptors often combined with machine learning to achieve best performance and high computational speeds.

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

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.

Figure 1: Design of the Honeycomb Marker Field. **top:** Every two neighboring modules (cells) of the marker field have different colors. Between every three neighboring modules thus exists a distinct Y-junction. **bottom:** A camera snapshot of the deformed marker field in a realistic environment.

2 Design & Detection of the Deformable Marker Fields

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 [Fiala 2005; Herout et al. 2012] or try to find a global solution [Szentandrási et al. 2012; Uchiyama and Saito 2011] assuming that the marker field is planar. The main problem with these approaches is that the whole marker field or its submarkers have to be planar.

Our Honeycomb Marker Fields (HMF) 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) – Fig. 1. Since the intensities 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 (Fig. 2).

2.1 Honeycomb Marker Field Recognition

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

1. **Input image pre-scanning** (Fig. 3) 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 cull or filter out unneeded parts. This step is discussed in Sec. 2.2.

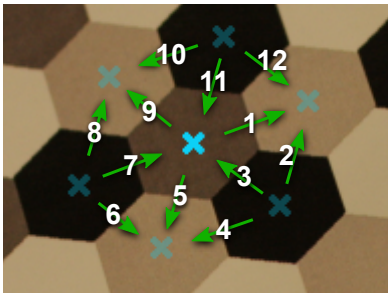


Figure 2: Identification of a fragment of the marker field based on the intensity differences between hexagons.

2. **Detection of the Y-junctions** (Fig. 4). 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 Sec. 2.3.

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 (Fig. 2) 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 Sec. 2.4.

2.2 Image Pre-Scanning

The image is processed by using sparse horizontal and vertical scanlines (Fig. 3 top) forming a grid. Edges (intensity jumps) are detected on these scanlines by using adaptive thresholding. Those grid cells whose sides do not contain any intensity edges are culled 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. The conditional subdivision is controlled by the number of edges on the block sides.

2.3 Detection of "Y" Image Features

The cells identified by the pre-scanning are searched for the Y-junctions by using a four-orientable triangular shaped scanning window – Fig. 5. 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 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:

$$d(\mathbf{a},\mathbf{b}) = |I(\mathbf{a}) - I(\mathbf{b})|, \tag{1}$$

where $I(\mathbf{x})$ is the intensity at given pixel $\mathbf{x}$ and $\mathbf{a},\mathbf{b}$ are coordinates of pixels. The response of the triangle is then represented as the

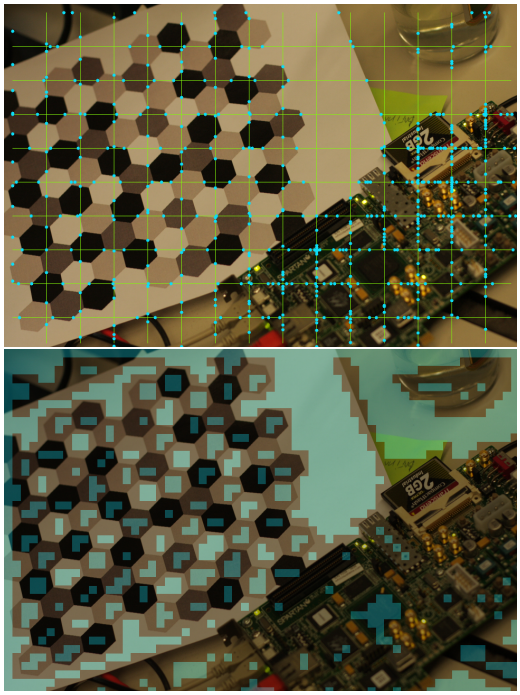


Figure 3: Grid scanning of the input image. **top:** Input image processed in sparse scanlines. The image shows the first level of the processing. Cells which contain edges are subdivided iteratively. **bottom:** Final output of the scanline grid. The square cells are classified; those that cannot contain Y-junctions are culled from further processing.

minimal value of the three differences:

$$r_t = \min\left(d(\mathbf{a},\mathbf{b}), d(\mathbf{a},\mathbf{c}), d(\mathbf{b},\mathbf{c})\right), \tag{2}$$

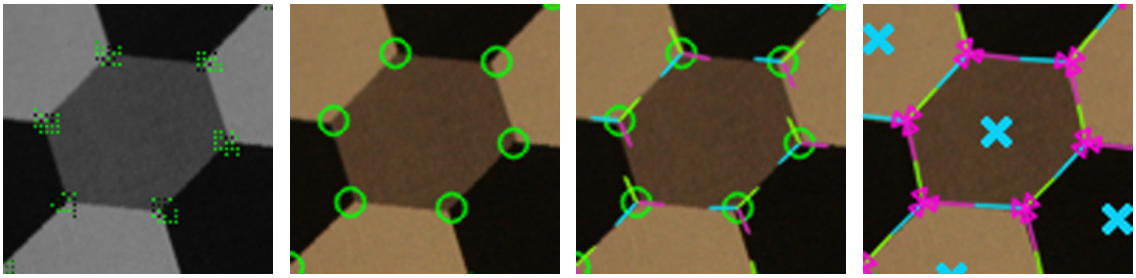
where $t \in \{1,2,3,4\}$ is the index of the triangle and $\mathbf{a},\mathbf{b},\mathbf{c}$ are its vertices (Fig. 5). The final response magnitude is computed as the maximal value of the four triangles' responses:

$$R = \max(r_1, r_2, r_3, r_4). \tag{3}$$

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 for example when detecting between a white and a black hegaxon, where the color is blurred to be changing continuously between them. In such a chase the detection triangle can be located so that one vertex lies on the grey edge between the extreme shades and it would fire a misdetection (Fig. 6). The response values of the scanning window are thresholded and the final result can be seen in Fig. 4(a).

As shown in Fig. 4(b), in the next step the center points are obtained by clustering the candidate Y-junction positions acquired from the previous step.

For each Y-junction, the directions of the edges coincident with it – referred to as the "legs" – are estimated (Fig. 4(c)). 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.



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

Figure 4: 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 recontructed by connecting the nearest junctions together.

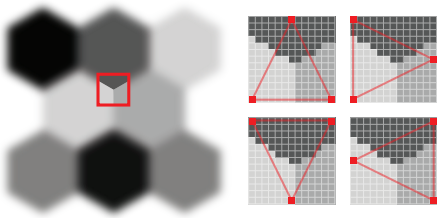


Figure 5: The scanning window contains four triangles rotated by 90 degrees, where only the triangle vertices are sampled and examined. The four-orientable window ensures robustness against rotation.

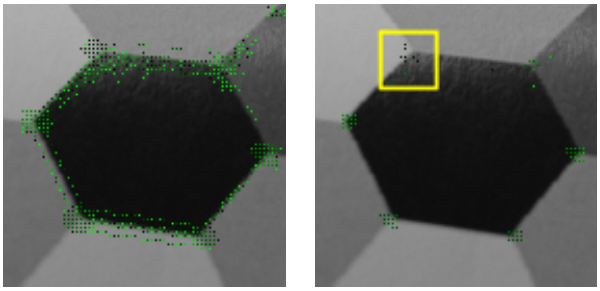


Figure 6: False responses of the detection. **left:** Not checking the magnitude of the gradient can cause false detection because the intensity change is continuous. **right:** By skipping places with high gradient magnitude these cases can be avoided. The yellow box shows when the illumination at the junction is variable and responses are smaller nevertheless the junction is detected.

2.4 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.

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 (Fig. 7):

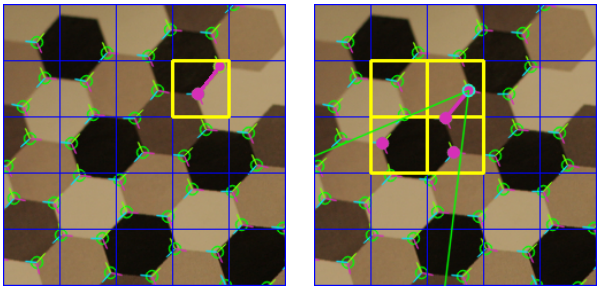


Figure 7: Grid search. **left:** The yellow outline highlights the current cell. In this case a valid junction is found in the inner cell. **right:** Adjacent cells are searched for junctions. Pink dots are the valid candidates and the one with the highest score (based on legs' direction and gradient) is selected. The green lines enclose the search area.

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:

$$cellSize = \sqrt{\frac{width * height}{n}}. \quad (4)$$

   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, see Fig. 7 left. If none are found, by using the pose of each leg an area of interest is determined (green lines in Fig. 7 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 [Green 2010], where a detailed description of these steps can be found.

To identify a unique part of the marker field actually detected, the edge gradients on the Y-junction "legs" are used. Every single Y-

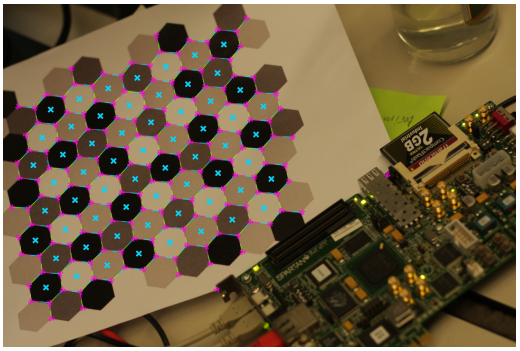


Figure 8: The reconstructed hexagonal marker field. Each full hexagon has a cross drawn in the middle.

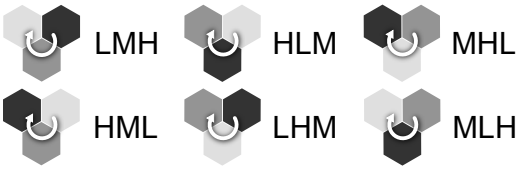


Figure 9: Descriptor of a single Y-junction: the sequence of *L*ow, *M*edium, *H*igh intensities of the adjacent hexagonal module, starting from an arbitrary hexagon in a defined order (CW in our case).

junction can be described by one of 6 combinations of grey intensities of the adjacent modules (Fig. 9). For color marker fields, a higher number of combinations exists (in particular, 6^n where n is the number of edge channels, typically $n = 3$ for RGB).

For a neighborhood of a hexagon, the LMH combinations (Fig. 9) 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 – Fig. 10. Six adjacent Y-junctions can define a hexagon in a small honeycomb marker field ($\sim 10 \times 10$ greyscale). For larger marker fields, a larger neighborhood of Y-junctions (e.g. 12 or 24) can be used.

3 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 (Fig. 11):

- **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

3.1 Detection Performance of the Y-junction 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 [Rosten and Drummond 2005; Rosten and Drummond 2006] as the reference, since both the FAST and our Y-junction detector are detecting corner-like features (contrary to for example SIFT or SURF which detect patches). The results of the comparison are shown in Tab. 2.

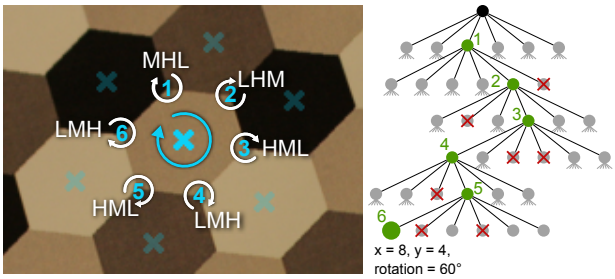


Figure 10: Recognition of a fragment of the marker field. **left:** Six Y-junctions are described by their LMH combination (Fig. 9). **right:** A decision tree of six levels (for six Y-junctions) determines the location and orientation of the hexagon within the marker field. Every non-leaf node of the tree has 6 child nodes for 6 permutations of LMH.

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 1: Performance results for various sizes of scanning window.

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.

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 pre-scan 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, see Table 1. 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 true positives, but the number of false positives is also increasing. A good compromise between precision and recall is using a 13×13 scanning window.

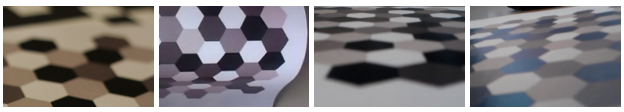
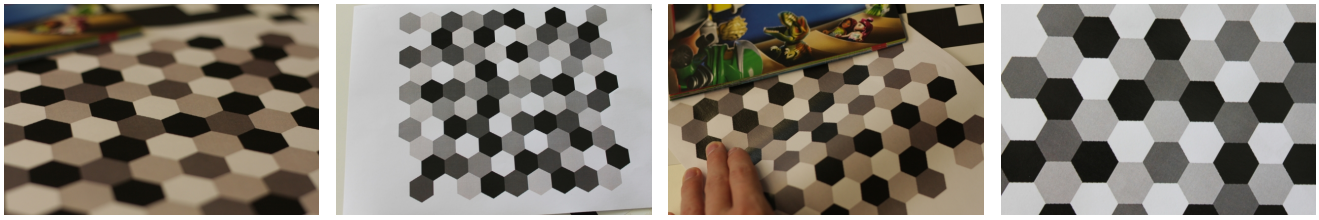


Figure 12: Examples of problematic edges (failure cases) within the pictures taken by a mobile phone of the marker field. The recognition can be challenging due to motion blur, uneven lighting conditions, high distortion, etc.



(a) Blurred (b) Clean (c) Environment (d) Zoomed

Figure 11: Sample images from the dataset.

FAST w/o 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

FAST with non-max suppression

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

Y-junction detector / HMF

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

Table 2: 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.

640×480

	Blurred	*Clean*	*Environm.*	*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

1680×1120

	Blurred	*Clean*	*Environm.*	*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

4272×2848

	Blurred	*Clean*	*Environm.*	*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

Table 3: 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.

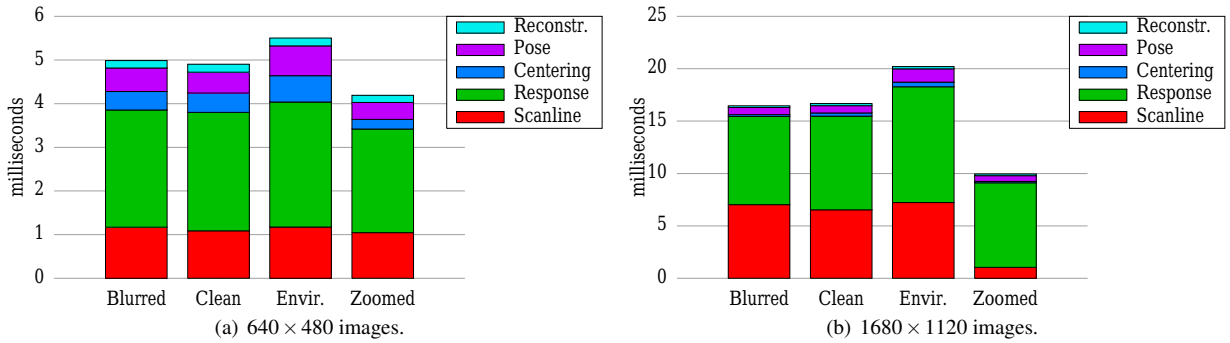
3.2 Speed Evaluation

In the second set of experiments we measured the speed of the detector. For each image in the dataset we created images in 3 different sizes (640×480, 1680×1120 and 4272×2848). Table 3 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. Since the current version of the detection algorithm is only a prototype which demonstrates the detection performance of the Y-junctions, the speed is expected to be largely improved in near future. This is why we report the relative time consumption instead of comparison to alternative detectors.

3.3 Elapsed Time Breakdown & Analysis

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 2.1 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 4 shows the times measured during the test split by categories, steps and image sizes. The first column depends only on the size of the scanned image. For bigger images the pre-scan or the scanline have to examine larger amount of pixels. The second column depends on the number of extracted blocks. The last three columns are dependent only on the number of found Y-junctions. As see in Table 4, computation times for these step are not increasing for larger images. The graphical comparison of the steps breakdown can be seen in Fig.13(a) and Fig. 13(b).

Table 5 and Fig. 14 shows the percental distribution of the computational time between different components. 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 13: Breakdown of computation time for each image category.

640×480

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

1680×1120

	Scanline	*Response*	*Center*	*Pose*	*Reconstr.*
Blurred	7.04	8.41	0.19	0.67	0.16
Clean	6.54	8.94	0.30	0.70	0.20
Environ.	7.23	11.05	0.44	1.26	0.22
Zoomed	1.05	8.06	0.14	0.55	0.16

4272×2848

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

Table 4: 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; **Reconstr.**: hexagon recontructions; for more details see section 2.3.

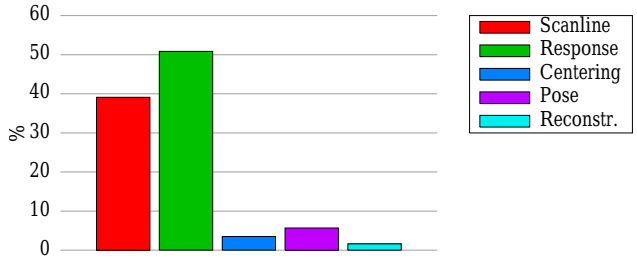


Figure 14: The breakdown of average computation time for each image category. The average is computed based on the Tab. 4.

Resolution	*Scanline*	*Response*	*Center*	*Pose*	*Reconstr.*
640×480	23	54	8	11	4
1680×1120	35	57	2	5	1
4872×2848	60	39	0	1	0
Average	39	50	3	6	2

Table 5: Breakdown of the computational time into different parts of the algorithm in percentages.

4 Anticipated Applications and Future Work

Motion capture of human beings, animals or deformable objects (cloth, ...) is still a challenging task. Reliable and complex methods require costly and space-demanding equipment. On the other hand, motion capture based on computer vision methods is cheaper, but requires considerable amount of manual processing of the captured data. The human operator needs to manually track the tracking points.

One of the intended applications of our highly occlusion-tolerant and deformable marker fields is clothes for the tracked subjects. Fractions of these clothes will be detected by the vision system (one or more high frame rate, high resolution cameras) and help localize locations on the tracked surface. The marker field is expected to cover the surface with dense localizable features (Y-junctions) that can be reliably detected in real time. Motion tracking based on deformable marker fields has the advantage that it requires no manual input.

Another application we intend to explore in near future is using the HMF as a highly deformable and occlusion-tolerant fiduciary marker for augmented reality. Today's fiducials used in the AR typically assume that the marker is planar or close-to-planar. The HMF will allow high degrees of deformation; they could be used on rigid deformed surfaces (cylinders, free-form design objects, etc.) or on non-rigid surfaces (cloths, deformed paper, furniture, etc.).

4.1 Future Work on the Detection of the HMF

The presented version of the Y-junction detector is in its early stage of development. The experiments show (Sec. 3) that detection of an artificial keypoint, namely in the nature of the Y-junctions in the HMF, by a specialized detector is more accurate than by using an existing general detector. These results encourage us to improve the Y-junction detector.

We intend to explore alternative approaches to Y-junction detection. One possibility is to detect for the edges between the Y-junctions and infer on the Y-junction locations from them. This approach could allow for even sparser scanlines processing the input image, because "hitting" an edge is very probable and the number of visited input image pixels could be significantly decreased. The Y-junction detector could also be trained by machine learning, similarly as in the case of the FAST keypoint detector [Rosten and Drummond

2006].

Our future work will also be focused on improving the construction of the graph connecting the Y-junctions. Missing parts of a graph will indicate locations in the input image, where Y-junctions and their pair-wise connections should be looked for, making a feedback loop. Exploring this feedback loop – and iteratively improving the graph by filling the missing sub-graphs should lead to very robust and fault-tolerant detection.

Along with improvement of the detector of the HMF, we are extending the generator. The generator will include tools for synthesis of visually pleasing marker fields: with attractive color maps, resembling a given 2D image, etc.

Last, but not least, the camera pose estimation in relation to different locations of the honeycomb marker field needs to be explored and followed by proposition and testing of applications.

5 Conclusions

This paper sketched out the concept of the Honeycomb Marker Fields – a highly deformable and occlusion-tolerant marker field. We focused on the design of the Y-junctions, the visual image features which appear between the modules (cells) of the marker field. As the main contribution, we presented an algorithm for real-time and robust detection of the Y-junctions, and its experimental evaluation. The specialized Y-junction detection algorithm outperforms a general interest point detector (FAST) in the detection performance. The measurements thus lead to conclusion that for applications which can accept a marker field as a fiduciary pattern, the reliability and performance can be significantly improved.

Based on the presented detector of Y-junctions and the preliminary HMF detector, we plan for the closest future to construct a robust and fast detector of the marker field and showcase it on selected applications. Thanks to the performance of the Y-junction detector, the algorithm is expected to be functional on today's ultramobile devices. The next step thus will be to implement it for some of the popular smartphone platforms.

Acknowledgements

This research was supported by the research project CEZMSMT, MSM0021630528, by the CEZMSMT project IT4I - CZ 1.05/1.1.00/02.0070, by project RODOS, TE01020155, and by project V3C, TE01020415.

References

BAY, H., TUYTELAARS, T., AND GOOL, L. V. 2006. SURF: Speeded up robust features. In *In ECCV*, 404–417.

CALONDER, M., LEPETIT, V., STRECHA, C., AND FUA, P. 2010. BRIEF: Binary robust independent elementary features. In *ECCV 2010*. 778–792.

CHO, Y., LEE, J., AND NEUMANN, U. 1999. A multi-ring fiducial system and an intensity-invariant detection method for scalable augmented reality. In *Proc. IWAR'98*, IWAR '98, 147–165.

DONOSER, M., AND BISCHOF, H. 2006. Efficient maximally stable extremal region (mser) tracking. In *Proc. CVPR 2006*, 553 – 560.

FIALA, M. 2005. ARTag, a fiducial marker system using digital techniques. In *Proc. CVPR 2005*, 590–596.

GREEN, S. 2010. Particle simulation using CUDA. *NVIDIA Whitepaper, December 2010*.

HEROUT, A., ZACHARIÁŠ, M., DUBSKÁ, M., AND HAVEL, J. 2012. Fractal Marker Fields: No more scale limitations for fiduciary markers tori. In *ISMAR 2012*, 285–286.

HEROUT, A., SZENTANDRÁSI, I., ZACHARIÁŠ, M., DUBSKÁ, M., AND KAJAN, R. 2013. Five shades of grey for fast and reliable camera pose estimation. In *CVPR 2013*.

KALTENBRUNNER, M., AND BENCINA, R. 2007. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proc. Tangible and embedded interaction*, 69–74.

KATO, H., AND BILLINGHURST, M. 1999. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IWAR'99*, 85–94.

KLEIN, G., AND MURRAY, D. 2007. Parallel tracking and mapping for small AR workspaces. In *Proc. ISMAR'07*.

LEUTENEGGER, S., CHLI, M., AND SIEGWART, R. 2011. BRISK: Binary robust invariant scalable keypoints. In *Proc. ICCV 2011*, 2548–2555.

LOWE, D. 1999. Object recognition from local scale-invariant features. In *Proc. ICCV 1999*, 1150–1157.

NEWCOMBE, R., LOVEGROVE, S., AND DAVISON, A. 2011. DTAM: Dense tracking and mapping in real-time. In *Proc. ICCV 2011*.

PAN, Q., REITMAYR, G., AND DRUMMOND, T. W. 2009. Interactive model reconstruction with user guidance. In *Proc. ISMAR 2009*, 209–210.

REKIMOTO, J. 1998. Matrix: A realtime object identification and registration method for augmented reality. In *Proc. APCHI '98*, 63–68.

ROSTEN, E., AND DRUMMOND, T. 2005. Fusing points and lines for high performance tracking. In *Proc. ICCV 2005*, 1508–1511.

ROSTEN, E., AND DRUMMOND, T. 2006. Machine learning for high-speed corner detection. In *Proc. ECCV 2006*, 430–443.

SIMON, G. 2011. Tracking-by-synthesis using point features and pyramidal blurring. In *ISMAR*, 85–92.

SZENTANDRÁSI, I., ZACHARIÁŠ, M., HAVEL, J., HEROUT, A., DUBSKÁ, M., AND KAJAN, R. 2012. Uniform Marker Fields: Camera localization by orientable De Bruijn tori. In *ISMAR 2012*, 319–320.

UCHIYAMA, H., AND MARCHAND, E. 2011. Deformable random dot markers. In *Proc. ISMAR 2011*, 237–238.

UCHIYAMA, H., AND SAITO, H. 2011. Random dot markers. In *IEEE Virtual Reality Conf. (VR)*, 271–272.

WAGNER, D., REITMAYR, G., MULLONI, A., DRUMMOND, T., AND SCHMALSTIEG, D. 2008. Pose tracking from natural features on mobile phones. In *Proc. ISMAR 2008*, 125–134.

YANG, X., AND CHENG, K.-T. 2012. LDB: An ultra-fast feature for scalable augmented reality on mobile devices. In *Proc. ISMAR 2012*, 49–57.