Natural Feature Detection on Mobile Phones with 3D FAST

Achim Weimert, Xueting Tan and Xubo Yang

Digital Art Laboratory, Shanghai Jiao Tong University, Shanghai 200240, China

***Abstract*—In this paper, we present a novel feature detection approach designed for mobile devices, showing optimized solutions for both detection and description. It is based on FAST (Features from Accelerated Segment Test) and named 3D FAST. Being robust, scale-invariant and easy to compute, it is a candidate for augmented reality (AR) applications running on low performance platforms. Using simple calculations and machine learning, FAST is a feature detection algorithm known to be efficient but not very robust in addition to its lack of scale information. Our approach relies on gradient images calculated for different scale levels on which a modified9 FAST algorithm operates to obtain the values of the corner response function. We combine the detection with an adapted version of SURF (Speed Up Robust Features) descriptors, providing a system with all means to implement feature matching and object detection. Experimental evaluation on a Symbian OS device using a standard image set and comparison with SURF using Hessian matrix-based detector is included in this paper, showing improvements in speed (compared to SURF) and robustness (compared to FAST).**

***Index Terms*—Feature detection, natural features, mobile phones.**

I. INTRODUCTION

Feature localization and feature matching is a common task in many computer vision applications. It is frequently used in tracking, image mosaicing and object recognition. Until now, a lot of feature detectors have been proposed to satisfy different requirements. Among them, FAST [1] is known for its repeatability and efficiency, while SURF [2] is known to be robust and to exhibit good performance. Our goal is to develop a novel detector that combines SURF and FAST, creating an algorithm called 3D FAST that extends traditional FAST to provide scale level and directional information. Our new detector will be invariant to image scaling and rotation, but considerably faster to compute. We implement our algorithm on mobile phones to show its efficiency and its potential for being widely used.

As camera-equipped mobile phones become more powerful and ubiquitously available, they are a perfect platform for AR systems. They are portable, low-cost, and more important, consumers are already getting used to this kind of devices. The new challenges posed by mobile phones are mostly hardware limitations. First of all, CPU and memory still lack the power of their corresponding PC components. Not only are the number of CPU cycles and the amount of memory limited, but also data transfer rates restricted and the size and speed as well as quality of the camera images rather low.

Handling those challenges allows AR applications to be used on normal consumer mobile phones, reducing the need for special AR hardware. Therefore, users can easily be equipped with personal information and entertainment systems, allowing everyday usages that have not been possible before.

Manuscript Received on 4 June 2010
E-Mail: yangxubo@sjtu.edu.cn

II. RELATED WORK

Traditional interest point detectors like FAST [1] and Fast Corner [3] are widely used in real time tracking [4,5]. FAST tests a point by considering pixels on a Bresenham Circle around it. If the number of contiguous pixels that are brighter (or darker) than the center is significantly high, the point is classified as an interest point. By applying offline machine learning to generate a decision tree [6] for lookup, the calculation of FAST can be improved as shown in [1]. Fast Corner [3] also inspects the circle around a candidate point. In contrast to FAST, it uses a response function that considers interpolated intensities of diametric opposite points to calculate the self-similarity value rather than counting the pixels. Both algorithms are efficient but not prepared to handle scale information.

Scale space theory is a framework for multi-scale signal representation [7]. Interest points retrieved by scale space based detectors are generally more stable because of scale invariance - i.e. the detection accounts for visual effects of zooming. The most common algorithms are based on Laplacian of Gaussians (LoG) that was introduced by Lindeberg in 1998. One of them, Scale Invariant Feature Transform (SIFT) [8] was shown to outperform the others [9] with respect to matching under transformations like rotation, scale change and image blur in 2005. Generally, an input image is convolved by a Gaussian kernel with the standard deviation set to scale value. Then the scale-normalized Laplacian operator is calculated to represent a measurement for the quality of a point as a feature called blobness. Feature points are located at local extrema of blobness with respect to the 2D space and scale dimension. This method achieves high robustness and scale invariance but is expensive to calculate. The SIFT detector is inspired by LoG

but using DoG (Difference of Gaussians) to approximate LoG. It exhibits improved performance but even if run on PCs, it still lacks realtime performance [2].

Since the introduction of automatic scale selection by Lindeberg, many traditional feature response functions have been extended to scale space. The determinant as well as the trace of Hessian matrix has first been applied to scale space blob detection in [10]. Mikolajczyk refined Hessian matrix based approach in 2002 [11] to gain a robust and scale-invariant feature detector named as Hessian-Laplace. Another similar approach using Harris measure instead of determinant of Hessian is described in the same paper, named Harris-Laplacian. 4 years later, SURF [2] is introduced, again using Hessian Matrix for its good performance in computation and accuracy. This time, Hessian Matrix is operating on box filters that efficiently approximate Gaussian second order partial derivatives. Additionally, integral images are used to speed up calculations of the filter, leading to better performance (about 3 times faster than SIFT, as shown by [2]). The authors call their optimized calculation of the Hessian Matrix "Fast-Hessian" which forms the detection part of SURF.

All of the above mentioned algorithms have been studied on PCs where some of them still lag behind realtime requirements. During the assessment of feature detection and description algorithms on mobile platforms the implementation of SURF on Nokia N95 mobile phone [12] shows that the resulting system is about 22 times slower compared to the one on PC. Promisingly, Wagner introduced a realtime tracking system that is fast enough to be used in mobile phone AR applications in [5]. In Wagner's research, SIFT has been optimized by replacing the Difference-of-Gaussian for scale space search by FAST and consequently discarding scale information. To reintroduce scale invariance, multi-scale images were stored in a database, requiring offline initialization, preprocessing and a precomputed feature database in memory.

To counter the deficiencies of current approaches, we propose an algorithm which is based on FAST but designed to use gradient images as input. With mobile devices in mind, we combine our algorithm with SURF's box filter for approximation, as SURF's description scheme leads to stable and distinct features while being computationally easier than well-known SIFT. To further reduce computation time, we use integral images for efficient calculation of the box filter.

III. 3D FAST DETECTION

3.1 Traditional FAST

We base our algorithm on FAST because of its good balance between computational requirements and feature quality. Traditional FAST judges a candidate point p to be a feature by analyzing c regularly distributed points on the Bresenham circle around p. Each point on the circle has a location $x \in \{0, 1, \ldots, c-1\}$ with an intensity of $I_{p,x}$ which can be compared to the intensity of the center: I_p. Using a threshold d, p is considered a corner if at least n contiguous pixels y_i (with $i \in \{0, 1, \ldots, i_{max}\}$ and $i_{max} \geq n$) can be found on the circle so that either $I_p + d < I_{p,x}$ or $I_p - d > I_{p,x}$ holds for all n pixels - meaning either all n pixels are significantly darker than the center p or all n pixels are significantly brighter than p. Proposed values for c and n are 16 and 12. Additionally, we use a modified corner response function for each feature which is defined by summing up the differences in intensity of the selected circle points when compared to the center p:

$$Q(p) = \sum_{\forall y_i} |I_p - I_{p,y_i}| - d \qquad (1)$$

For all non-features, $Q(p)$ equals 0.

This approach is used to train a classifier for representing the algorithm as an optimized decision tree, where the path for each classification is minimized by ID3 [6]. The resulting tree is then transformed into if-else instructions, providing an efficient implementation of the described feature detector.

Additionally, as the original FAST implementation is meant to be used on pixel intensities only but features on gradient images that are located at positions where the rate of change in the image gradient is significantly high show better results. Consequently, as our algorithm is based on gradient images, it needs to be ready to handle gradient inputs. Therefore, FAST's input domain is extended to absolute gradient values by scaling its threshold d accordingly.

3.2 Introducing Scale Levels

SIFT and SURF both take advantage of scale level information to account for robustness through scale invariance. Our algorithm extends FAST to scale levels by using the FAST response function as defined in Equation 1 on gradient images. This approach includes the following three steps.

First of all, a scale space gradient image pyramid is built by approximating Gaussian second order derivatives. As proposed in [13], Gaussian kernel is the only reasonable method to setup a scale space for 1D continuous signals. Other commonly adapted techniques with a sampled Gaussian can lead to violations of scale space axioms [7] in corresponding representations, as described in [13].

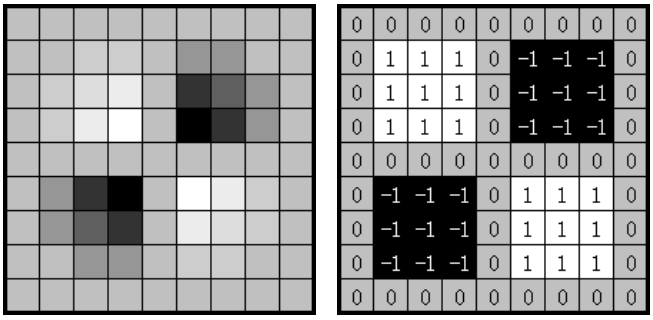

Fig. 1. left: Gaussian second order partial derivatives in xy-direction, right: approximation using box filter

As Bay et al. proved box filters to be a reasonable approximation [2] for second order Gaussian derivatives in the 2D case while being considerably easier to compute at the same

time, we use them to build the scale space pyramid. Trading memory for time, the implementation can be improved by using integral images. Figure 1 shows the 9×9 box filter and original second order Gaussian kernel with $s = 1.2$. The approximation is denoted as D_{xy} and our *2 octaves × 3 layers* gradient image pyramid is built using box filters with growing size. Because 9×9 box filter corresponds to Gaussian derivatives with $s = 1.2$, it is considered to be the kernel of the initial scale layer. 15×15 corresponds to $s = 2$ and is taken to build the first layer of the second octave. The step between consecutive filter sizes also scales accordingly. It is set to be 2 for 3 layers in the first octave and growing to 4 in the second octave.

Second, an adapted FAST that can handle gradient images is run on all levels of the image pyramid. We detect bright blobs or dark blobs using FAST algorithm with growing radius for every scale level. For each detected FAST feature, a gradient FAST response value is set to represent its quality. A quality of 0 tags a pixel as not being a FAST feature. In general, given a point $X(x, y)$ in an original image I, the gradient FAST response is defined as follows:

$$Q(X,\sigma) = \sum_{\forall i} \left| L_{xy}(X,\sigma) - L_{xy}(X_i,\sigma) \right| - \hat{d} \qquad (2)$$

where X_i stands for the neighboring pixels lying on the Bresenham circle, $L_{xy}(X,\sigma)$ is the convolution of Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point X and $\hat{d}$ being a threshold. As we approximate Gaussian derivative by box filter, the equation can be written as $Q(X,t) = \sum_{\forall i} \left| D_{xy}(X,t) - D_{xy}(X_i,t) \right| - \hat{d}$.

Third, 3D extrema are detected by considering the corner response function of 8 2D neighbors in the same layer and 9 neighbors in both the preceding and following layer. The gradient images are efficiently calculated using box filters based on integral images, similar to SURF. From the layer which includes the detected extremum, scale level information is extracted and combined with a SURF descriptor to build the new feature descriptor.

IV. NATURAL FEATURE MATCHING

4.1 SURF Descriptor

SIFT is an algorithm that includes both detector and descriptor. In the original SURF paper [2] Bay proposes to uses Hessian-Laplacian to approximate Laplacian of Gaussian, replacing Lowe's DoG approximation in SIFT. Concerning scale space extrema detection, both algorithms retrieve scale level information to ensure scale invariance. As described before, our detection is implemented based on FAST-Laplacian approximation. The next step is to compute the description vectors for detected feature points, producing feature vectors that are highly distinctive and partially invariant to distortions like illumination and projection.

To achieve rotation invariance, we identify an orientation for each keypoint based on sums of 2D Haar wavelet responses as suggested in [2]. The response in x and y direction is calculated for all pixels in a circular neighborhood at any scale. The values are weighted with a Gaussian centered at the keypoint. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of $p/3$. Of course, the size of the wavelet is scale dependent, as well as the radius of the circle. A Gaussian value lookup table is used to reduce calculation during runtime. For greater efficiency of the filtering, dynamic programming to obtain an implementation based on integral images is applied.

4.2 Matching

As noted before, description of the features is provided by the SURF descriptor which is proven to be robust and easy to match. Changes to the original SURF descriptor consist out of using the scale level obtained by 3D FAST. Finally, to provide a measurement for the quality of the detected features, we apply 3D FAST and SURF in the context of matching as suggested by Mikolajczyk et al. in [9]. Matching is performed using the methods described in [14], where the closest neighbor of a feature point from the first image is the feature point of the second image that has the smallest distance. The distance of two points is given by the Euclidean distance of their description vectors. To filter out features that do not have a match, we compare the closest to the second closest neighbor and only include matches where the distance $d1$ to the closest neighbor is significantly smaller than the distance $d2$ to the second closest neighbor, assessed by the distance ratio r in $d1/d2 < r$. As described in [14], a distance ratio of 0.8 is a reasonable choice for reliable removal of outliers.

V. EXPERIMENTAL RESULTS

We implemented 3D FAST and SURF for Symbian OS 9.2 S60 devices. Tests have been conducted and results have been measured using a Nokia N95-6 device (Dual 332 Mhz Texas Instruments OMAP 2420 CPU). The scale level pyramid consists out of 2 octaves with 3 layers each. We tested our implementation on the standard image set provided by Mikolajczyk[1]. The set includes image pairs of real scenes with image blur (trees, bikes), changes in viewpoint (graffiti, wall), zoom/rotation (boat). Due to space limitations only a selection is included here.

In the first run, we compared the speed and quality of the feature detector part of SURF (Fast Hessian) with the one of 3D FAST and traditional FAST, see Figure 2 for an example. The results of 3D FAST and SURF similarly include distinctive features: While the 3D FAST features are fewer, they are still evenly distributed over different image areas, show few overlapping features while quality is comparable (see also Table 5). Traditional FAST features are frequently located next to each other on edges and consequently not very distinctive. The average running time for SURF detector is 2.4 times of the one of 3D FAST (see Table 1 which includes the performance

[1] http://www.robots.ox.ac.uk/_vgg/research/affine/

of the detection parts of both methods).

Fig. 2. Squares indicate found features where the size corresponds to scale level and color is used to distinguish points. *left:* result generated by 3D FAST, *middle:* result generated by Fast Hessian, *right:* original color photo overlaid with the result of traditional FAST

TABLE 1: TIME CONSUMPTION OF FEATURE DETECTORS

3D FAST	842 ms
Fast-Hessian (SURF's detector)	2042 ms
FAST (2D, without scale pyramid)	83 ms

Performance of the individual tasks of 3D FAST has been evaluated and results are presented in Table 2. As we use 3D FAST for the detection, its time consumption can be split into setting up the scale pyramid and then applying our modified FAST to it. Afterwards, we use the descriptor part of SURF to derive a vector suitable for matching. Clearly, the estimations to compose the scale pyramid are responsible for a considerable portion of the running time. In contrast, calculating the corner response function by applying our modified FAST on the pyramid is extremely efficient. The SURF descriptor itself is computationally expensive and therefore rather slow.

TABLE 2: TIME CONSUMPTION OF INDIVIDUAL TASKS (3D FAST DETECTION & DESCRIPTION)

Detection	Pyramid	663 ms
	FAST on Pyramid	179 ms
Description	Calculate SURF descriptor	708 ms

In the next test, we want to assess a measurement for the quality of the features. Therefore, we run each of the detectors on two different input images taken from the same image sequence, calculate feature descriptors, do matching and then estimate the percentage of correct matches. Some examples of the process can be seen in Figure 3 which presents the results of both methods working on the same input image. Furthermore, Figure 4 shows 3D FAST's invariance to image blur. The data is presented in three categories. First of all, the percentage of features that were matched is calculated, see Table 3. Secondly, the percentage of features that were correctly matched is presented in Table 4. Finally, the percentage of correctly matched features within all matched features is given in Table 5. Consequently, this last table is most significant for judging the quality of the feature detectors for the application of matching. Features detected by the two algorithms show comparable performance, when considering the percentage of correctly matched features within all matched features, 3D FAST performs slightly better. Therefore, although the detection of 3D FAST is simpler, it still can be used for deriving stable features.

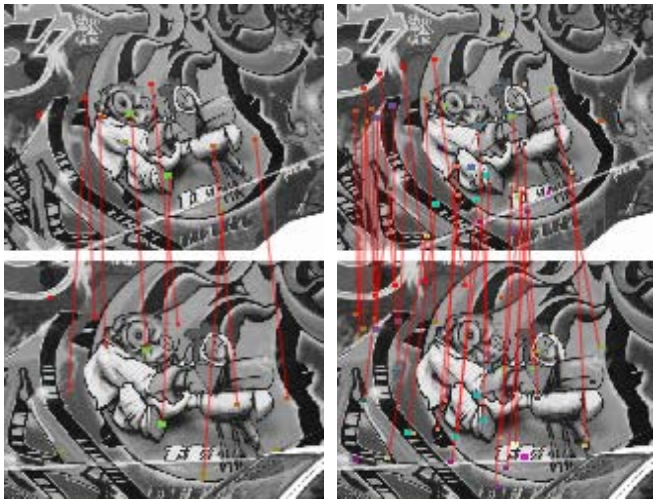

Fig. 3. Squares indicate found features where the size corresponds to scale level, color is used to distinguish points and lines are drawn between correctly matched pairs. *left:* 3D FAST, 13 matches of which 9 (≙ 69%) are correct, *right:* SURF, 42 matches of which 30 (≙ 71%) are correct

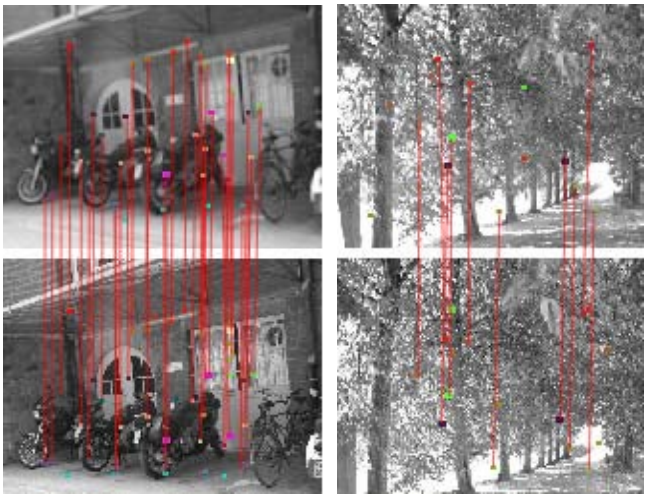

Fig. 4. 3D FAST showing resistance to image blur. Squares indicate found features where the size corresponds to scale level, color is used to distinguish points and lines are drawn between correctly matched pairs.

TABLE 3: PERCENTAGE OF MATCHED FEATURES

image set	3D FAST	SURF
GRAFFITI	21,5%	22,6%
WALL	19,3%	19,7%
TREES	23,3%	27,0%
BOAT	20,7%	22,5%
BIKES	46,2%	50,2%

TABLE 4: PERCENTAGE OF CORRECTLY MATCHED FEATURES WITHIN ALL FEATURES

image set	3D FAST	SURF
GRAFFITI	19,1%	19,3%
WALL	17,9%	18,2%
TREES	21,9%	25,0%
BOAT	17,0%	17,0%
BIKES	43,3%	45,7%

TABLE 5: PERCENTAGE OF CORRECTLY MATCHED FEATURES WITHIN MATCHED FEATURES

image set	3D FAST	SURF
GRAFFITI	52,4%	44,2%
WALL	59,7%	58,3%
TREES	84,5%	85,0%
BOAT	29,2%	25,2%
BIKES	92,5%	88,6%

VI. CONCLUSION

We have presented a new method for feature detection which is computationally easier than the currently used SURF algorithm and showing improvements in the quality of the detected features. The implementation was assessed on a mobile device. While not yet being ready for real-time applications, feasibility on limited hardware was shown.

Future work can take greater advantage of 3D FAST's strength by using a descriptor that is easier to calculate than SURF, trading accuracy for speed. To further assess 3D FAST's performance, an in-depth evaluation can be devised on PCs to check its suitability for calculation intensive real-time applications. Another important area for future work is to discard gradient images in favor of faster but less robust image intensities.

ACKNOWLEDGEMENT

This work was partially supported by the National Natural Science Foundation of China (Grant No.60970051), and the National High Technology R&D Program of China (No.2006AA01Z307).

REFERENCES

[1] E. Rosten and T. Drummond. Machine learning for high-speed corner detection, in *Proc. of the 9th European Conference on Computer Vision*, vol. 1, pp. 430–443, 2006.

[2] H. Bay, T. Tuytelaars and L. Van Gool. Surf: Speeded-up robust features, in *Proc. of the 9th European Conference on Computer Vision, Graz, Austria*, pp. 404–417, 2006.

[3] M. Trajkovic and M. Hedley. Fast corner detection, *Image and Vision Computing*, 16(2), pp. 75–87, Feb. 1998.

[4] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces, in *Proc. of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, 2007.

[5] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond and D. Schmalstieg. Pose tracking from natural features on mobile phones, in *Proc. of the 7th IEEE and ACM International Symposium on Mixed and Augmented Reality*. pp. 125–134, 2008.

[6] J. R. Quinlan. Induction of decision trees, in *Machine Learning*, pp. 81–106, Mar. 1986.

[7] T. Lindeberg. *Scale-Space Theory in Computer Vision*, Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[8] D. G. Lowe, Object recognition from local scale-invariant features, in *Proc. of the 7th IEEE International Conference on Computer Vision*, vol. 2, pp. 1150-1157, 1999.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615-1630, 2005.

[10] T. Lindeberg. Feature detection with automatic scale selection, *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79-116, 1998.

[11] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector, in *Proc. of the 7th European Conference on Computer Vision*, Springer Verlag, pp. 128-142, 2002.

[12] W.-C. Chen, Y. Xiong, J. Gao, N. Gelfand and R. Grzeszczuk. Efficient extraction of robust image features on mobile devices, in *Proc. of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 287-288, 2007.

[13] T. Lindeberg. Scale-space for discrete signals, *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3), pp. 234-254, 1990.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, vol. 60, pp. 91-110, 2004.

Achim Weimert was born in Germany in 1986. He received the Vordiplom in computer science from Technische Universität Berlin (TUB) in 2007. Taking part in a dual-degree program of TUB with Shanghai Jiaotong University (SJTU) between 2007-2009, he continued his studies to pursue a M.Sc. degree in computer science at SJTU with a focus on the field of virtual and augmented reality. Currently, he is writing his diploma thesis in the field of web browser security at the department of security in communications at TUB to complete the dual-degree program with a Dipl.-Inform. degree of TUB.

Xueting Tan was born in China in 1985. She received the bachelor degree in software engineering from Shanghai Jiaotong University (SJTU) in 2007. After 2.5 years' further education, she got the master degree in computer science and software theory in the same university. Her research interests include Augment Reality, Vision Tracking and Image Recognition.

Xubo Yang received PhD degree in computer science from the State Key Lab of CAD & CG, Zhejiang University, in 1998. He worked for three years in the Virtual Environment Group of the Fraunhofer Institute for Media Communication, Germany. Then he joined the Mixed Reality Lab of National University of Singapore from 2001 to 2003. Currently he is an associate professor and head of the Digital Art Lab at School of Software, Shanghai Jiao Tong University (SJTU). His research interests include media art computing technologies in the context of computer graphics, augmented reality, virtual reality and human computer interaction. He is member of the China Graphics Society and the China Computer Federation.