Adaptive and Generic Corner Detection Based on the Accelerated Segment Test

Elmar Mair[1]*, Gregory D. Hager[2], Darius Burschka[1], Michael Suppa[3], and Gerhard Hirzinger[3]

[1] Technische Universität München (TUM), Department of Computer Science, Boltzmannstr. 3, 85748 Garching bei München, Germany
{elmar.mair,burschka}@cs.tum.edu

[2] Johns Hopkins University (JHU), Department of Computer Science, 3400 N. Charles St., Baltimore, MD 21218-2686, USA
hager@cs.jhu.edu

[3] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Münchner Str. 20, 82230 Wessling, Germany
{michael.suppa,gerd.hirzinger}@dlr.de

Abstract. The efficient detection of interesting features is a crucial step for various tasks in Computer Vision. Corners are favored cues due to their two dimensional constraint and fast algorithms to detect them. Recently, a novel corner detection approach, FAST, has been presented which outperforms previous algorithms in both computational performance and repeatability. We will show how the accelerated segment test, which underlies FAST, can be significantly improved by making it more generic while increasing its performance. We do so by finding the optimal decision tree in an extended configuration space, and demonstrating how specialized trees can be combined to yield an adaptive and generic accelerated segment test. The resulting method provides high performance for arbitrary environments and so unlike FAST does not have to be adapted to a specific scene structure. We will also discuss how different test patterns affect the corner response of the accelerated segment test.

Key words: corner detector, AGAST, adaptive, generic, efficient, AST

1 Introduction

Efficient corner detection algorithms are the basis for many Computer Vision applications, *e.g.* to find features for tracking, tracking by matching, augmented reality, registration or 3D reconstruction methods. Compared to edges and color cues, corners are more accurate and provide a two dimensional constraint. Considering corners as intersection of two edges, these features have no spatial extension and, therefore, there is no ambiguity in their location. Of course, this

* This work was performed during the author's research stay at the CIRL lab (Johns Hopkins University). We are deeply grateful to all members of this research group for the interesting discussions and the great support.

aspect is only valid if the locality of a corner is preserved and the response of a corner detector is as close as possible to the real corner location. Several different approaches to corner detection are known in literature. All try to find a solution for efficient, accurate and reliable corner detection - three rather conflicting characteristics.

The Harris corner detection algorithm is probably one of the most popular corner extraction methods [1]. It is based on the first order Taylor expansion of the second derivative of the local sum of squared differences (SSD). The eigenvalues of this linear transformation reveal how much the SSD approximation varies if the patch would be shifted along the image axes. There are solutions which interpret the eigenvalues based on a threshold [2] or without [3]. So called global matching algorithms allow features to be detected within the whole image. Therefore, a corner detector has to provide a high repeatability so that it detects the same features also after large affine transformations. The global tracker SIFT [4] uses difference of Gaussians (DoG), while the faster SURF [5] uses a Haar wavelet approximation of the determinant of the Hessian. Both methods have the drawback of being rather computationally expensive. Smith developed the so called "Smallest Uni-Value Segment Assimilating Nucleus Test" (SUSAN) [6] for corner detection. The brightness of the center pixel, the nucleus, is compared to its circular pixel neighborhood, and the area of the uni-value segment assimilating nucleus (USAN) is computed. Corner and edges can be detected by evaluating this area, or it can also be used for noise reduction. The advantages of this approach are that no noise sensitive derivation or other computationally expensive operations have to be performed. In [6] a circular disc with diameter 3.4 is used, which yields a total area of 37 pixels. A more comprehensive survey can be found in [7].

In the last decade the processing power of standard computers has become fast enough to provide corner extraction at video rate. However, running conventional corner detection (*i.e.* the Harris corner detector) and performing other intensive tasks, is computationally infeasible on a single processor. With the introduction of recent techniques such as the "Features from Accelerated Segment Test" (FAST) [8], feature extraction has seen significant performance increase for real-time Computer Vision applications. While being efficient, this method has proven in several applications to be reliable due to high repeatability (see [9]). Some applications which use FAST are, *e.g.*, Klein's PTAM [10] and Taylor's robust feature matching in $2.3\,\mu s$ [11].

In this work we are going to present a novel corner detection approach, which is based on the same corner criterion as FAST, but which provides a significantly *performance increase* for *arbitrary* images. Unlike FAST, the corner detector does not have to be trained for a specific scene, but it *dynamically adapts* to the environment while processing an image.

Section 2 discusses FAST in more detail due to its strong relation to the presented work. In Section 3, we will present the *adaptive* and *generic* accelerated segment test with increased performance for arbitrary environments. Further, we will discuss the use of different segment pattern and show some experimental results to demonstrate the achieved speed-up in Section 4.

2 FAST Revisited

The FAST principle is based on the SUSAN corner detector. Again, the center of a circular area is used to determine brighter and darker neighboring pixels. However, in the case of FAST, not the whole area of the circle is evaluated, but only the pixels on the discretized circle describing the segment. Like SUSAN, also FAST uses a Bresenham's circle of diameter 3.4 pixels as test mask. Thus, for a full accelerated segment test 16 pixels have to be compared to the value of the nucleus. To prevent this extensive test, the corner criterion has been even more relaxed. The criteria for a pixel to be a corner according to the accelerated segment test (AST) is as follows: there must be at least S connected pixels on the circle which are brighter or darker than a threshold determined by the center pixel value. The values of the other $16 - S$ pixels are disregarded. Therefore, the value S defines the maximum angle of the detected corner. Keeping S as large as possible, while still suppressing edges (where $S = 8$), increases the repeatability of the corner detector. Thus, FAST with segment size 9 (FAST-9) is usually the preferred version, and is also used in our experiments unless otherwise stated. The AST applies a minimum difference threshold (t) when comparing the value of a pixel on the circular pattern with the brightness of the nucleus. This parameter controls the sensitivity of the corner response. A large t-value results in few but therefore only strong corners, while a small t-value yields also corners with smoother gradients. In [9] is shown, that the AST with $S = 9$ has a high repeatability, compared to other corner detectors as, *e.g.*, Harris, DoG, or SUSAN. The repeatability of a corner detector is a quality criterion which measures the capability of a method to detect the same corners of a scene from varying viewpoints.

One question still remains, namely which pixel to compare first, second, third, and so forth. Obviously, there is a difference in speed, whether one consecutive pixel after another is evaluated or, *e.g.*, bisection on the circle pattern is used to test if the corner criterion applies or cannot apply anymore. This kind of problem is known as constrained twenty questions paradigm. When to ask which question results in a decision tree with the aim to reduce its average path length. In [12], Rosten uses ID3 [13], a machine learning method, to find the best tree based on training data of the environment where FAST is applied. Doing so, it is not guaranteed that all possible pixel configurations are found (see Section 4.2). Already small rotations of the camera may yield pixel configurations which have not been measured in the test images. And even if all the pixel configurations are present, a small rotation about the optical axis would cause the probability distribution of the measured pixel configurations to change drastically. This may result in an incorrect and slow corner response. To learn the probabilistic distribution of a certain scene is therefore not applicable unless only the same viewpoints and the same scene are expected. Note that the decision tree is optimized for a specific environment and has to be re-trained every time it changes to provide the best performance.

The decision tree learning used by the FAST algorithm builds a *ternary* tree with possible pixel states "darker", "brighter" and "similar". At each learning step, *both* questions, "is brighter" and "is darker", are applied for all remaining pixel and the one with the maximum information gain is chosen. Hence, the state of each pixel can be one of four possibilities: unknown (u), darker (d), brighter (b) or similar (s). In the following we call a combination of N such states a pixel *configuration.* The size of the configuration space is therefore 4^N, which yields $4^{16} \approx 4 \cdot 10^9$ possible configurations for $N = 16$. For the rest of this paper we refer to this model as restricted or four states configuration space.

FAST-ER, the most recent FAST derivation, has even a slightly increased repeatability, compared to FAST-9, at the cost of computational performance [9]. The main difference is the thickness of the Bresenham's circle, that has been increased to 3 pixels. This results again in a more SUSAN-like algorithm, which spans a circular area of 56 pixels, disregarding the inner 3x3 pixels. Again, ID3 is used to build the decision tree, restricting the evaluation to only a small part of the 47 pixels.

3 Adaptive and Generic Accelerated Segment Test

In this section we present a corner detection approach which is also based on the AST, but which is more efficient, while being more generic too. We introduce the reader step-wise to the different concepts underlying the algorithm.

3.1 Configuration Space for a Binary Search Tree

Instead of only considering a restricted configuration space, as in FAST, we propose to use a more detailed configuration space in order to provide a more efficient solution. To do this, we consider to evaluate a single question per time. The idea is as follows: choose one of the pixels to test and *one* question to pose. The question is then evaluated for this given pixel, and the response is used to decide the following pixel and question to query. Searching for a corner, hence, reduces to traversing a *binary* decision tree. Since, it is required to specify which pixel to query and the type of question to use. Consequently, the configuration space increases by the addition of two more states: "not brighter" ($\overline{b}$) and "not darker" ($\overline{d}$). Using a similar notion as [12], the state of a pixel relative to the nucleus n, denoted by $n \rightarrow x$, is assigned as follows:

$$S_{n\rightarrow x} = \begin{cases} d, & I_{n\rightarrow x} < I_n - t & \text{(darker)} \\ \overline{d}, & I_{n\rightarrow x} \nless I_n - t \bigwedge S'_{n\rightarrow x} = u & \text{(not darker)} \\ s, & I_{n\rightarrow x} \nless I_n - t \bigwedge S'_{n\rightarrow x} = \overline{b} & \text{(similar)} \\ s, & I_{n\rightarrow x} \ngtr I_n + t \bigwedge S'_{n\rightarrow x} = \overline{d} & \text{(similar)} \\ \overline{b}, & I_{n\rightarrow x} \ngtr I_n + t \bigwedge S'_{n\rightarrow x} = u & \text{(not brighter)} \\ b, & I_{n\rightarrow x} > I_n + t & \text{(brighter)} \end{cases} \tag{1}$$

where $S'_{n \to x}$ is the preceding state, I is the brightness of a pixel and u means that the state is still unknown. This results in a binary tree representation, as opposed to a ternary tree, allowing a single evaluation at each node. Note that this increases the configuration space size to 6^N, which yields $6^{16} \approx 2 \cdot 10^{12}$ possible nodes for $N = 16$.

Associated with each branch of our tree is a processing cost, which represents the computational cost on the target machine. These costs vary due to different memory access times. We specify these as follows,

– c_R: register access cost (second comparison of the last tested pixel),
– c_C: cache access cost (testing of a pixel in the same row)
– c_M: memory access cost (testing of any other pixel).

Further, for each of these, an additional cost equivalent to evaluating a greater-than operation, is required.

3.2 Building the Optimal Decision Tree

It is well known that a greedy algorithm, such as ID3, performs rather poorly when finding the optimal decision tree [14]. However, the issue of finding such a tree is a well-studied problem, where it has been shown that finding the global optimum is NP-complete [15]. There are several solutions towards finding the optimal tree [16–18], but they are either approximations to the global optimum or are restricted to special cases, making them ill-suited for this application.

In order to find the optimal decision tree we implemented an algorithm which is similar to the backward induction method [16]. We explore the whole configuration space starting at the root of the decision tree, where none of the pixels is known. Nodes of the tree are formed by recursively evaluating a possible question at a given pixel. We explore the configuration space (using Depth First Search) until a leaf is found, where a leaf is defined as the first node on the path which fulfills or cannot fulfill anymore the AST corner criteria. The cost at a given leaf is zero, while the cost at any given internal node, c_P, is determined by picking the minimum cost computed for each child pair C_+ and C_-, representing the positive and negative results of a test, by

$$c_P = \min_{\{(C_+, C_-)\}} c_{C_+} + p_{C_+} c_T + c_{C_-} + p_{C_-} c_T = c_{C_+} + c_{C_-} + p_P c_T \qquad (2)$$

where c_T represents the cost of the pixel evaluation with $c_T \in \{c_R, c_C, c_M\}$ and the p_P, p_{C_+} and p_{C_-} are the probabilities of the pixel configurations at the parent and child nodes respectively. Using this dynamic programming technique allows us to find the decision tree for an optimal AST (OAST) efficiently. The resulting decision tree can therefore be optimized for different c_R, c_C and c_M, but also for arbitrary probabilities for each pixel configuration, which is necessary for our approach described in the following section.

The binary configuration space allows for decision trees which reduce the entropy more quickly than a ternary tree, as questions which contain little information gain are deferred to later stages of the decision process. Note that the additional cost of re-evaluating the same pixel at a subsequent point in time is taken into account when computing the optimal tree.

3.3 Adaptive Tree Switching

Every image has, independent of the scene, homogeneous and (or) cluttered areas representing uniform surfaces or structured regions with texture. Hence, instead of learning the distribution of the pixel configurations from training images, like FAST, a first generalization would be to learn the probability of structured and homogeneous regions and optimize the decision tree according to this distribution. The resulting tree is complete and optimized for the trained scene, while being invariant to camera rotations. The probability of an image to be uniform can be modeled by the probability of a pixel state to be similar to the nucleus (p_s). The "brighter" and "darker" states are mirrored states, which means that, *e.g.*, a brighter pixel on the test pattern will evaluate the current nucleus pixel as darker as soon as it becomes the center pixel. Due to this mirroring the states "brighter" and "darker" are assumed to have the same probability (p_{bd}), which is chosen to sum up to one with p_s ($p_s + 2p_{bd} = 1$). Thus, the probability of a pixel configuration p_X can be computed as follows:

$$p_X = \prod_{i=1}^{N} p_i \quad \text{with } p_i = \begin{cases} 1 & \text{for } S_{n\to i} = u \\ p_s & \text{for } S_{n\to i} = s \\ p_{bd} & \text{for } S_{n\to i} = d \bigvee S_{n\to i} = b \\ p_{bd} + p_s & \text{for } S_{n\to i} = \overline{d} \bigvee S_{n\to i} = \overline{b} \end{cases} \tag{3}$$

The probability distribution of the pixel configuration is therefore a trinomial distribution with the probabilities p_s and twice p_{bd}. Note that the states $\overline{d}$, $\overline{b}$ and u are not samples of this distribution but represent a set of two and three samples respectively. While this approach provides a good solution for the trained environment, it is not generic and, as FAST, it has to be learned for each specific scene where it is applied.

A more efficient and generic solution is achieved, if the algorithm automatically adapts to the area which is currently processed, *i.e.* it switches between decision trees which are optimized for the specific area. The idea is to build, *e.g.*, two trees and specialize one for homogeneous and one for structured regions based on a small and a large value for p_s. At the end of each decision path, where the corner criterion is met or cannot be fulfilled anymore, a jump to the appropriate specialized tree is performed based on the pixel configuration of this leaf (see Fig. 1). This switch between the specialized decision trees comes with *no* additional costs, because the evaluation of the leaf node is done offline when generating the specialized tree. In this way the AST is adapted to each image section dynamically and its performance is *increased*, for an *arbitrary* scene. Any learning becomes needless.

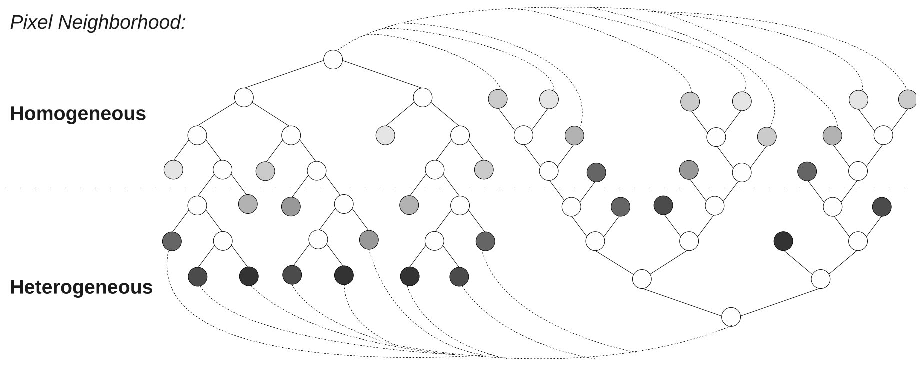


Fig. 1. Principle of the adaptive and generic accelerated segment test. The AGAST switches between two (or more) specialized trees as soon as the pixel neighborhood changes. The lighter the gray of a leaf the more equal pixels are in the configuration. The left tree achieves less pixel evaluations (shorter decision paths) in a homogeneous pixel neighborhood, while the right one is optimized for textured regions.

Because a switch between the trees at no costs can only be performed at a leaf, the adaption is delayed by one test. Therefore, the only case were the adaptive and generic accelerated segment test (AGAST) would be less efficient than FAST, is if the environment would switch from homogeneous to structured and vice versa at consecutive pixels. This is practically not possible, due to the mirroring effect of dissimilar pixels as described earlier. However, natural images usually do not have a random brightness distribution, but they are rather split into cluttered and uniform regions. If the decision trees can be strongly balanced by varying p_s, also more than two different weighted trees can be used.

4 Experimental Results

The speed and the repeatability of FAST have already been compared to state of the art corner detection algorithms in [9]. In those experiments FAST-9 has demonstrated better performance than, *e.g.*, Harris, DoG, or SUSAN. Thus, we renounce to compare our AST variation only with FAST-9. Note that our approach is also based on the AST and, therefore, it provides the same repeatability as FAST.

First, we show and discuss an experiment where we compare the performance of different AST masks on noisy and blurry images. In Section 4.2 we evaluate the corner response of different balanced decision trees; and, finally, we compare the performance of FAST with our approach.

4.1 Evaluation of Various AST Patterns

As already mentioned, SUSAN as well as FAST use a circle radius of 3.4 pixels. In [6] it is noted that the mask size does not influence the feature detection as long as there is no more than one feature within the mask. The effect of the mask size of an image operator is well studied for filters with dense masks. Their size affects the smoothing behavior so that larger filters are more robust against noise. The corresponding effect for the AST pattern size has so far not been discussed in the similar literature. While for the dense mask of SUSAN, the same smoothing criteria as mentioned above apply, it is not obvious that large circles have a similar smoothing effect for AST. Therefore, we use eight checkerboard pictures acquired from different viewpoints (see Fig. 3) to evaluate the corner response of the AST pattern shown in Fig. 2. A checkerboard provides many bright and dark corners of different sizes if viewed from different angles. Further, we add Gaussian blur and noise to determine the performance of these pattern on images of poor quality. For all the tests the same threshold is applied.

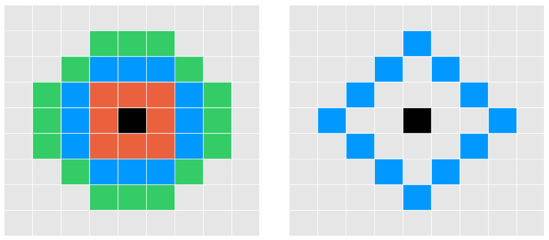


Fig. 2. Different mask sizes for the AST: a 4 pixels mask (red), a squared and diamond shaped 12 pixels mask (blue, left and right figure) and a 16 pixels mask (green). The black pixel represents the nucleus.

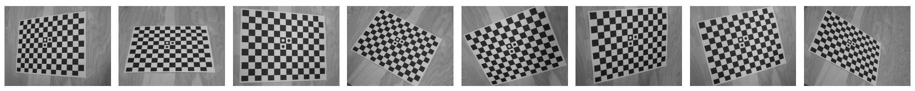


Fig. 3. Checkerboard dataset.

For pattern sizes up to 12 pixels it is possible to compute the optimal path by exploring the six state configuration space as described in Section 3.1. The computational resources of conventional computers are not sufficient to find the optimal tree for a 16 pixel pattern within the extended configuration space in reasonable time. Thus, for this size we compute the optimal tree based on the four state space, yielding a ternary decision tree. Before generating the machine code, the tree is splatted as described in [9] to cut off equal branches.

Fig. 4 shows the corner response of a 16 pixel pattern with arc lengths of 9, 10 and 11 (12 is omitted because it does not find any features at these corners), the

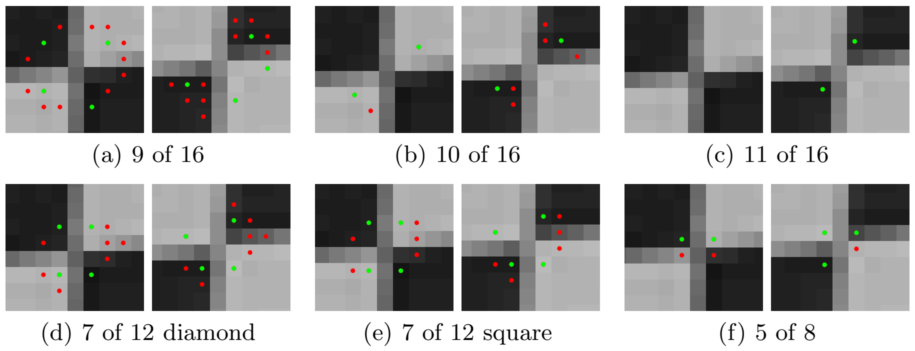


Fig. 4. The corner response for different AST pattern. Detected features are colored in red. The corners after non-maximum suppression are green.

12 pixel pattern with a square and diamond shape as well as the 8 pixel pattern. The larger the mask and the larger the arc threshold S, the more features that are found. A small arc is more discriminating and yields features only close to the real corner location, which is apparent in Fig. 4(c). Large patterns result in multiple responses around a corner location, but they may lie at a distance of about the radius of the mask from the real corner (see Fig. 4(a)). Thus, they do not preserve the corner location. They are therefore slower for two reasons: 1) the processing of a large pattern is of course computationally more expensive, and 2) they need to evaluate many features for non-maximum suppression. Smaller patterns better preserve the locality constraint of a corner. However, in the case of the pattern of size 8, the features are too close, so that a part of them get lost after non-maximum suppression. Thus, for this size such a post processing is not necessary, because only single responses are observed at a corner, and should even be avoided to prevent the loss of features.

For the next experiment the original checkerboard images were modified by adding Gaussian noise (5% and 10%) and Gaussian blur ($\sigma = 0.5, 1.0, 1.5$). Fig. 5 shows the performance of the different pattern on these images. Here the advantage of the 16 pixel mask with arc length 9, 9(16), becomes apparent. It is more robust against noise and blur. However, the same mask sizes but with larger arcs show a similar drop-off on blurry images as the smaller pattern with similar segment angle. The 16 pixel mask with arc length 12, 12(16), has a similar arc angle as 5(8), while 7(12) has a similar angle as 16(10).

The size of the arc angle controls the repeatability, as shown in [9], and the robustness against blur. The arc length and, thus, the radius of the mask influences the robustness against noise.

4.2 Corner Response Time

The corner response time of a certain decision tree is evaluated by computing the number of tests (greater-than or less-than evaluations) for all the possible

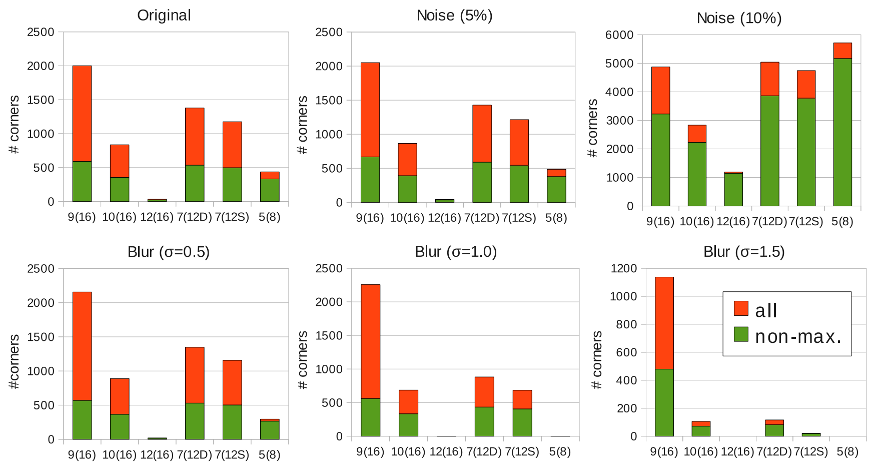


Fig. 5. These charts compare the corner response of different patterns for blurry and noisy images. To preserve the comparability we use the arc length S and in brackets the mask size to label the bars. The red bars show the total amount of features found, while the green bars represent the number of corners after non-maximum suppression. Note that the scales of the charts are not the same.

pixel configurations of a mask. To compare the weighting effects of different probabilities of a pixel to be similar (p_s), as described in Section 3.3, the pixel configurations are divided into classes representing the number of similar pixels. Fig. 6 shows the deviation of the mean and the standard deviation of the corner response time from the minimum of all tests on a class. The trees are built for 12 pixel masks exploring the six state configuration space. For zero or one similar pixels the trees with weight $p_s = 0.1$ and $p_s = 0.01$ perform fewer tests as trees with larger values for p_s. Also the standard deviation of the classes is smaller for these trees. It is apparent that the decision trees can not be balanced significantly due to the strong symmetry of this special constrained twenty questions problem. Besides, the classes with a large number of similar pixels cannot be balanced properly anymore, because the amount of possible configurations decreases drastically for them. Nevertheless, the performance of the adaptive tree is better than if only one tree is used, as we will see in Section 4.3.

No performance increase can be achieved for different p_s by exploring only the restricted configuration space, due to the reduced degrees of freedom compared to the full six state configuration space. The limitations of the latter space are also apparent in the performance of the tree M12 (4st), which shows a significantly higher average of tests as M12 (6st) in Table 1. This table compares the corner response time for trees of various mask sizes which were built using different methods. The second data row M12 (6st) shows the minimum time of the trees

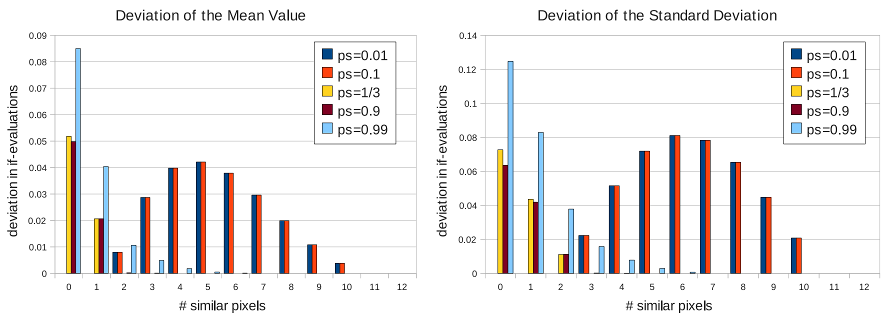


Fig. 6. This chart illustrates the performance of various decision trees, based on different probabilities for a pixel to be similar (p_s). Each decision tree was tested with all possible pixel combination for the mask.

compared in Fig. 6 which were specialized for different p_s. Thus, these values are achieved using the AGAST, switching between two trees which are optimized for $p_s = 0.1$ and $p_s = 1/3$.

Experiments have shown, that by learning a decision tree based on 120 outdoor images as proposed in [12], only about 87000 pixel configurations out of over 43 million possible ones could be found. Any learned decision tree should therefore be enhanced by the missing configurations to prevent false positive and false negative responses. The ID3 based decision tree, learned from all possible configurations with equal weights, has shown to achieve the best corner response of all trees witch were optimized using ID3 and various p_s. Indeed, it yields the identical corner response as the code provided in the FAST sources.[4]

4.3 Performance Experiments

All the timing experiments are run on one core of an Intel Core2 Duo (P8600) processor at $2.40\,GHz$. We are using five images from different scenes[5], shown in Fig. 7.

Table 2 shows the performance of various AST-decision trees with different mask sizes and built by different methods. Please note, that the achieved speed-ups do not only affect the corner detection step, but also the computation of the pixel-score for the non-maximum suppression.

To compare the performance of our decision trees with the conventional FAST-9 algorithm, we use the code from the FAST sources mentioned in Section 4.2. The FAST and optimal AST (OAST) trees are built based on a uniform probability distribution, which means that the probability for any pixel configu-

[4] http://svr-www.eng.cam.ac.uk/~er258/work/fast.html

[5] The lab scene is provided in the FAST Matlab package at http://svr-www.eng.cam.ac.uk/~er258/work/fast-matlab-src-2.0.zip

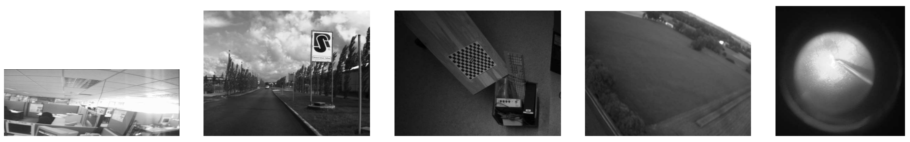
Table 1. This table compares the average tests performed for each class of configuration using various mask sizes and different methods to find the best decision tree. From left to the right: mask size 8 exploring the six states configuration space, mask size 12 exploring the six states space, mask size 12 exploring the four states space, mask size 16 exploring the four states space and the ID3-learned decision tree trained on all possible configurations. The probability of all configurations was assumed to be equal for all trees beside M12 (6st), which represents the minimum tests for all decision trees of Fig. 6. These trees were built by exploring the six state configuration space for a mask size of 12 pixels using different weights.

n_s	M8 (6st)	M12 (6st)	M12 (4st)	M16 (4st)	M16 (ID3)
0	5.54	6.53	7.80	8.1528	8.3651
1	5.32	6.17	7.27	7.6485	7.8073
2	5.07	5.82	6.77	7.1948	7.3094
3	4.81	5.48	6.33	6.7893	6.8692
4	4.59	5.19	5.93	6.4277	6.4812
5	4.41	4.94	5.59	6.1044	6.1388
6	4.26	4.74	5.28	5.8144	5.8354
7	4.13	4.56	5.01	5.5529	5.5649
8	4.00	4.41	4.77	5.3160	5.3223
9	-	4.29	4.55	5.1003	5.1033
10	-	4.18	4.35	4.9031	4.9043
11	-	4.08	4.17	4.7221	4.7225
12	-	4.00	4.00	4.5554	4.5555
13	-	-	-	4.4013	4.4013
14	-	-	-	4.2583	4.2583
15	-	-	-	4.1250	4.1250
16	-	-	-	4.0000	4.0000

ration is the same. This probability distribution yielded the trees with the best overall corner response and therefore the best performance.

As mentioned earlier, it is not possible to search for the optimal decision tree for a 16 pixel mask within the complete configuration space in reasonable time on conventional computer. Therefore, the tree is optimized in the four state configuration space and achieves an average speed-up of about 13% regarding FAST-9. For the 12 pixels mask the ideal tree can be found in the six state space and by combining the trees specialized for $p_s = 1/3$ and $p_s = 0.1$ a mean speed-up of about 23% and up to more than 30% can be gained. Using the AGAST-5 decision tree on the 8 pixels mask results in a performance increase of up to almost 50%. Of course, with the drawback of its sensitivity regarding noise and blur as discussed in Section 4.1.

The C-sources for OAST-9, AGAST-7 and AGAST-5 are available for download at `http://www6.cs.tum.edu/Main/ResearchAgast`. The trees have been optimized according to standard ratios of memory access times.



(a) Lab (b) Outdoor (c) Indoor (d) Aerial (e) Medical

Fig. 7. The scenes used for the performance test are a lab scene (768x288), an outdoor image (640x480), an indoor environment (780x580), an aerial photo (780x582) and an image from a medical application (370x370).

Table 2. This table shows the computational time of various AST-decision trees. The value in parentheses, close to the tree names, stands for the mask size which the tree is based on. The specified speedup is relative to the FAST performance. The first value represents the mean speedup for all five images while the value in parentheses shows the maximum speedup measured.

Image	Lab	Outdoor	Indoor	Aerial	Medical	Speed-Up [%]
FAST-9 (16)	1.8867	2.4242	1.8516	2.2798	1.1106	-
OAST-9 (16)	1.5384	2.2970	1.6197	1.9225	0.9413	13.4 (18.5)
AGAST-7 (12)	1.2686	1.9416	1.4405	1.8865	0.8574	23.0 (32.8)
AGAST-5 (8)	0.9670	1.4582	1.3330	1.8742	0.7727	33.0 (48.7)

5 Conclusion and Future Work

We have shown how to increase the performance of the accelerated segment test by combining specialized decision trees. The optimal trees are found by exploiting the full binary configuration space. The algorithm dynamically adapts to an arbitrary scene which makes the accelerated segment test generic. In doing so no additional costs arise. This makes this approach to the currently most efficient corner detection algorithm to our knowledge. Moreover, any decision tree learning to adapt to an environment becomes needless. By exploring the full configuration space also the processor architecture and its memory access times can be taken into account to yield the best performance on a specific target machine.

Further, we have discussed the influence of different AST mask sizes and shown that, for images of good quality, smaller mask sizes should be preferred. They reduce the processing time and emphasize the locality constraint of a corner. Dealing with blurry and noisy images, patterns with a larger radius are favored.

For future research we would like to implement an approximation for decision tree learning as proposed in [17], which considers also the length of the decision path and not only the minimization of the entropy, as ID3. In this way, we can also balance trees of pattern sizes 16 or more pixels and implement the AGAST for these masks. Further, we are looking for an efficient combination of different mask sizes to yield high robustness while preserving the real corner location.

6 Acknowledgments

We want to particularly acknowledge Raphael Sznitman for the fruitful discussions and his support. Further, we want to thank Frank Sehnke and the TUM-Cogbotlab group for allowing us to use one of their PCs, which sped up the development significantly.

This work was support by the DLR internal funding for image-based navigation systems.

References

1. Harris, C., Stephens, M.: A combined corner and edge detection. In: Proceedings of The Fourth Alvey Vision Conference. (1988) 147–151
2. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). (1994) 593–600
3. Noble, A.: Descriptions of Image Surfaces. PhD thesis, Department of Engineering Science, Oxford University (1989)
4. Lowe, D.G.: Object recognition from local scale-invariant features. International Journal of Computer Vision **60** (2004) 91–110
5. Bay, H., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. In: European Conference on Computer Vision (ECCV'06). (2006) 404–417
6. Smith, S.M., Brady, J.M.: Susan - a new approach to low level image processing. International Journal of Computer Vision **23** (1997) 45–78
7. Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: a survey. Foundations and Trends in Computer Graphics and Vision **3** (2008) 177–280
8. Rosten, E., Drummond, T.: Fusing points and lines for high performance tracking. In: IEEE International Conference on Computer Vision (ICCV'05). Volume 2. (2005) 1508–1511
9. Rosten, E., Porter, R., Drummond, T.: Faster and better: A machine learning approach to corner detection. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) (2009)
10. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan (2007)
11. Taylor, S., Rosten, E., Drummond, T.: Robust feature matching in 2.3μs. In: IEEE CVPR Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond. (2009)
12. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision (ECCV'06). Volume 1. (2006) 430–443
13. Quinlan, J.R.: Induction of decision trees. Machine Learning (1986)
14. Garey, M.R., Graham, R.L.: Performance bounds on the splitting algorithm for binary testing. Acta Informatica **3** (1974) 347–355
15. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is np-complete. Information Processing Letters **5** (1976) 15–17
16. Garey, M.R.: Optimal binary identification procedures. SIAM Journal on Applied Mathematics **23** (1972) 173–186
17. Geman, D., Jedynak, B.: Model-based classification trees. IEEE Transactions on Information Theory **47** (2001)
18. Kislitsyn, S.S.: On discrete search problems. Cybernetics and Systems Analysis **2** (1966) 52–57