SuperFAST: Model-Based Adaptive Corner Detection for Scalable Robotic Vision

Gaspard Florentz[1] and Emanuel Aldea[2]

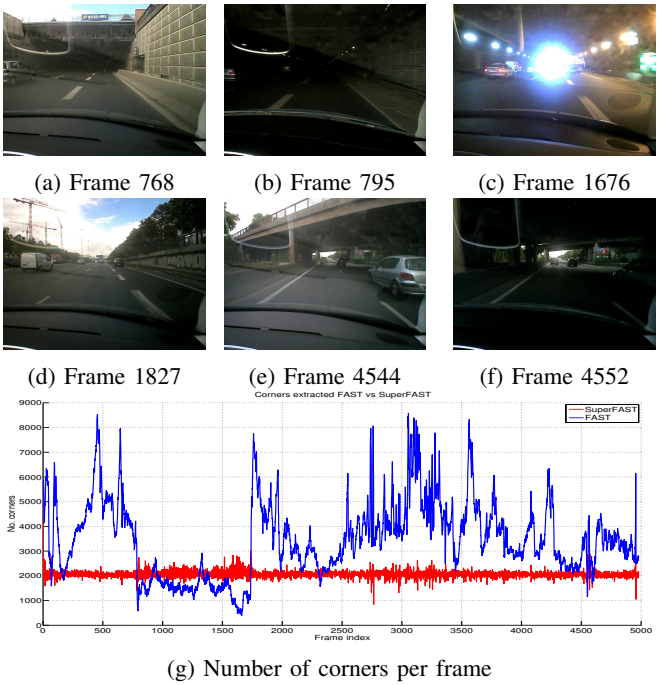
***Abstract*— In this study, we propose a novel solution to regulate the amount of interest points extracted from an image without significant additional computational cost. Our method acts at the very beginning of the detection process by using a corner occurrence model in order to predict the optimal threshold for a user-defined number of detections. Compared to existing approaches which guarantee a reasonable amount of corners by using a low threshold and then pruning the result, our approach is faster and more regular in terms of computation time as it avoids scoring and sorting the detected corners.**

Using the FAST detector as testbed, the strategy outlined in this article is evaluated in typical environments for robotics applications, and we report improved detection reliability during important scene variations. Taking into account the underlying visual navigation algorithms, we show that by regularizing the data input our solution facilitates a stable processing load, lower inter-frame computation time, and robustness to scene variations.

I. Introduction

For intelligent systems characterized by autonomy in heterogeneous environments and by high reactivity, accurate 3D perception is an essential characteristic supporting them in performing their tasks. Vision sensors have the advantage of being compact, low-cost and power efficient, hence they are widely used for exteroception. A wide range of fundamental actions such as scene reconstruction, localization, object detection and tracking may be approached by first identifying in each view a set of salient features or landmarks exhibiting invariance properties, that are generally denoted as corners; the interested reader may refer to [1], [2], [3] for examples of such applications.

In the last decade, the research community devoted a lot of effort to developing efficient corner detection and descriptor extraction algorithms. These methods are judged mostly on their ability to provide detection repeatability and descriptor invariance for a wide range of variations in lighting conditions and perspective. Owing to its excellent performance, the SIFT detector [4] remains one of the most popular solutions in all the circumstances in which its computational burden is acceptable. However, small autonomous systems with limited processing power, very common for consumer robotics, are the typical example in which SIFT use is out of the reach of the computational budget. Alternative solutions are provided by the lighter SURF detector [5], or for real time processing purposes by the Harris detector [6] and more recently by FAST [7], [8].



(a) Frame 768 (b) Frame 795 (c) Frame 1676

(d) Frame 1827 (e) Frame 4544 (f) Frame 4552

(g) Number of corners per frame

Fig. 1: The graph presents the number of corners detected for an urban video sequence by FAST detection ($\tau = 20$) and by SuperFAST. For challenging situations (tunnel entry/exit, underpass) FAST output varies significantly and most applications rely on low thresholding and filtering. Comparable in terms of cost to a base FAST execution, SuperFAST predicts an optimal threshold and provides a stable number of detections.

Despite its computational advantage, this last algorithm does not provide a descriptor, and subsequent processing tasks rely typically on patch based matching or on light descriptors with limited invariance properties. Vision algorithms compensate these shortcomings through data association and robust statistics, which can cope with a large fraction of unreliable detections at a significant computational cost. On the other hand, real time constraints require detections to be as scarce as possible. Due to these considerations, FAST thrives in terms of usefulness in a narrow band of detections per image which is architecture, application and content dependent; the ideal number of corner responses is mainly defined according to experimental observations (see

[1]Gaspard Florentz is Research Project Leader with Parrot S.A., 174 quai de Jemmapes 75010 Paris, France gaspard.florentz@parrot.com

[2]Emanuel Aldea is with the Autonomous Systems Department, IEF, Univ. Paris-Sud/CNRS, 91405 Orsay, France emanuel.aldea@u-psud.fr

for example [9]). In order to obtain a precise number of detections or to approach this number as much as possible, FAST must be combined with a heuristic selection process which introduces by itself a significant additional cost.

Our work aims to address this problem of keeping the corner detection count at an optimal level under varying scene conditions. Figure 1 shows in blue the typical output variation of the FAST detector on an outdoor sequence, with a threshold set at $\tau = 20$. In order to guarantee a minimal number of detections in homogeneous scenes (tunnels, underpasses, over-exposition), a low threshold is required. However, this results in a significant computational burden for highly contrasted scenes, where the large detection output must be pruned and processed. We manage to adapt the detection threshold to the image content in order to stabilize the output (Figure 1, in red), while actually speeding up the overall process. We also illustrate that by regularizing both the extraction time and the amount of data to be processed by the subsequent tasks, we also improve the stability and the average computation time of the overall visual processing algorithm. We argue that beside repeatability and invariance, another feature that is of utmost importance for a detector in the context of real-time robotics is its adaptiveness to the algorithm employing it.

II. Thresholding and adaptive detection

In the following section, we discuss the importance of adaptive thresholding and its implications when using FAST as a lightweight detector. The detector works by labelling pixels $\{x_1, \ldots, x_{16}\}$ located on a discrete circle around the analysed location p in image I in three categories, using the following decision function:

$$S_p(x_i) = \begin{cases} \mathrm{d}, & I(x_i) \leq I(p) - \tau \\ \mathrm{s}, & I(p) - \tau < I(x_i) < I(p) + \tau \\ \mathrm{b}, & I(p) + \tau \leq I(x_i) \end{cases} \quad (1)$$

where the threshold τ controls whether surrounding pixels will be considered significantly brighter or darker than the analysed location. An image location is considered to be a corner if a contiguous number $m \geq N$ (where typically $N = 9$) of pixels on the discrete circle are labelled either d or b.

As any corner detector, FAST will get for a fixed threshold τ more responses in highly contrasted areas and much fewer in homogeneous regions of an image, typically varying by an order of magnitude. Depending on the application, this behaviour may cause two fundamental problems:

1) **temporal instability**: in a heterogeneous video sequence, images with a high number of corners require more processing time from other algorithm modules, and for real time processing applications the computational cost may exceed the available resources. Conversely, the same threshold may detect an insufficient number of corners in other images. This degrades the performance of pose estimation algorithms (optimizations will be insufficiently constrained) and mapping algorithms (failure to extend the cartography)
2) **spatial instability**: even if an image exhibits a high number of corners on the whole, the detections may not be uniformly distributed. The scarcity of corners in parts of the image has the same negative impact as in the previous case.

In order to mitigate the effect of these two phenomena, an adaptive method must be considered. A widely employed solution is to use a fixed, low threshold which is permissive enough to generate an acceptable number of responses in a low contrast context. In this case, guaranteeing a target number of detections amounts to filtering the initial responses based on their saliency (either in the form provided by FAST, or by using other measures [10]) and discarding the necessary amount. However, choosing a low threshold comes with a cost which is mainly related to the longer processing time required by the detector. FAST is particularly affected by a lower threshold as its cascading test on surrounding pixels will be able to advance further in the decision tree. Moreover, such a method amplifies the variation in processing time per frame as it relies on attributing a score to each corner and sorting a varying number of corners for selection. Another important observation is that since this solution still uses a fixed threshold, albeit providing a more stable output, it does not behave fundamentally as an adaptive method.

An alternative approach (used for example in [1]) addressing the temporal instability is to vary the threshold based on a simple proportional feedback between the current detection and the desired result. Although this solution does adapt to image content, it has two main limitations. Firstly, it does not cope robustly with sudden variations either in terms of scenery or in terms of global illumination (such as a tunnel entry or exit). Low cost camera sensors are widely employed in consumer systems and their auto exposure correction may aggravate this problem, as it assumes a linear response of the sensor and is also based on average image statistics. Secondly, the assumption supporting the proportional feedback is the linearity between τ and the output.

Conditioned by the presence of one of the solutions for adaptive detection, the problem of spatial instability is usually addressed by two strategies. If a global, low threshold is used, the detection result may be filtered by an algorithm which favours an even distribution of corners across the image, such as by using a quad-tree for selection [9]. Alternatively, detections with independent thresholds may be performed on non overlapping regions of the image, a process which is called bucketing [1], [11], [12].

In conclusion, FAST based algorithms and more generally corner detection methods aim ideally to provide a stable, uniformly distributed number of responses per image. However, predicting the number of responses for a given threshold requires a content dependent model, and existing solutions either incur a computational cost by using a conservative threshold, or they adapt the threshold using a feedback mechanism which is ineffective in varying conditions.

III. A CONTENT ADAPTIVE CORNER OCCURRENCE MODEL

In the following section, we propose a fast method for estimating the number of detections in an image for a given threshold τ based on a corner occurrence model. In Figure 2 we illustrate the dependence between the threshold τ and the number of detections in a typical image (in blue), for the original resolution and for the next three down-sampled levels in the image pyramid. For illustration, we vary the threshold in the range $\tau \in [5, 80]$; in applications τ may vary from $\tau_{min} = 10$, a low level which is frequently used in order to guarantee a minimum number of detections, to high values which are adapted to contrasted regions; some particularly sharp scenes may feature a large number of detections even for thresholds as high as $\tau = 50$. Based on the profile of the corner distribution, we found that the square root exponential model [13] which depicts the spatial interaction (contamination) with respect to the isolation distance in biological systems fits remarkably the empirical distribution observed in typical human environments. In our case, the model is expressed as:

$$N(\tau) = C \exp\left\{ -\sqrt{\frac{\tau}{\sigma}} \right\} \tag{2}$$

where C is the initial population and σ is the scaling parameter.

In the same figure, we provide in red the best fit for the experimental observations according to the model; the graphs highlight clearly the remarkable correlation across all the scales and across the threshold range.

A. Fitting and prediction

The proposed model features two parameters, and a fast extrapolation may be performed using two measurements. In order to optimize the process, we rely on the following specific properties of the FAST detector.

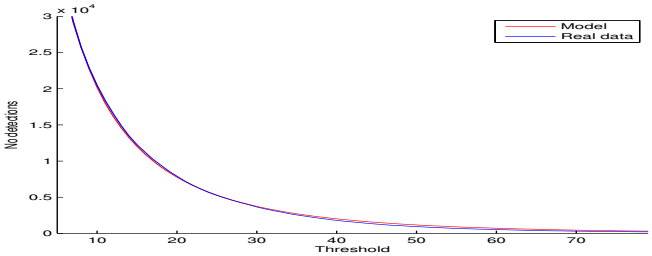
For an unknown scene, extrapolation may be performed using points corresponding to high threshold values, since detections will execute faster in this range. Actually, we perform the detection only for τ_1 and then we iterate through the results and count the responses for the higher threshold τ_2, where τ_2 is chosen in order to guarantee a robust extrapolation, typically $\tau_2 = \tau_1 + 10$. Assuming that N_1 and N_2 are the number of detections for the two control thresholds, we get after some simple algebraic manipulations for the model parameters:

$$\begin{cases} \sigma &= \frac{(\sqrt{\tau_1} - \sqrt{\tau_2})^2}{\ln^2(N_1/N_2)} \\ C &= N_1 \exp\left\{ \sqrt{\frac{\tau_1}{\sigma}} \right\} \end{cases} \tag{3}$$

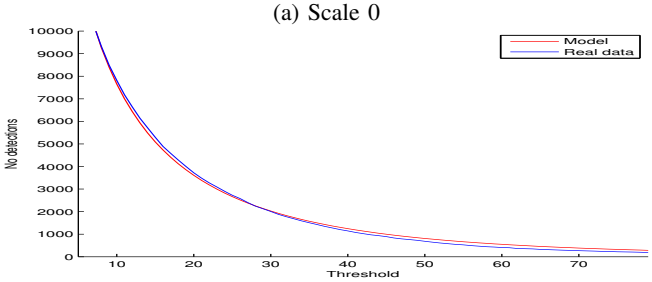
and for the threshold $\hat{\tau}$ predicted to provide an ideal number $\hat{N}$ of corners:

$$\hat{\tau} = \sigma \ \ln^2 \frac{C}{\hat{N}} \tag{4}$$

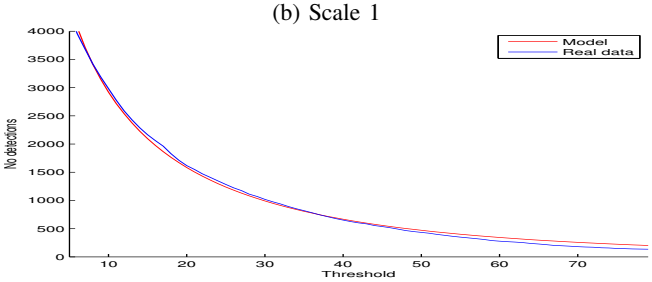
However, in poorly contrasted areas, the number of detections may be too low to provide reliable extrapolation data points, or the detection result might even be null for the extrapolation thresholds τ_1 and τ_2. In this case, we assume that the content exhibits low contrast and we set $\tau_1 = \tau_{min}$.



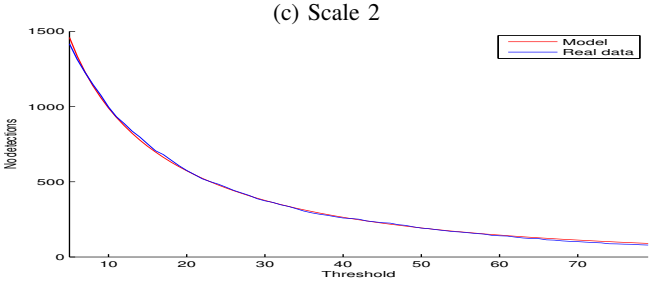
(a) Scale 0



(b) Scale 1



(c) Scale 2



(d) Scale 3

Fig. 2: For a regular image we plot in blue, for the initial resolution (scale 0) and for three progressively down-sampled versions, the observed distribution of corners depending on the detector threshold, and in red the best fit using the model proposed in Equation 2. The model approximates very accurately the experimental data at all scales.

B. Temporal consistency and multi-scale analysis

A frequent phenomenon which is encountered in the presence of rapid scene variations with low-cost cameras is the erratic impact of auto exposure; as a typical example, we present in Figure 3 two consecutive frames which are visually similar. However, in terms of corner detection, the difference is extensive with an output of 6067 versus 3200 corners for the same threshold $\tau = 20$. Even though

(a) Frame 2763 (b) Frame 2764

Fig. 3: Two consecutive frames which are visually similar. However, the number of FAST detections varies by a factor of 2 ($\tau = 20$)

auto-exposure has a beneficial effect in the presence of illumination changes, it introduces important dynamics for which FAST is particularly sensitive as it does not perform normalization with regard to contrast.

It is therefore essential in this situation to predict reliably the modification of the detection threshold. We propose to rely on a pyramidal image representation and to adapt the threshold prediction based on the assumption that effects such as scene change, camera auto-exposure or white balance have a similar effect on each level of the pyramid. More precisely, we rely on the fact that:

- properties of image content at a coarse scale may be used as a predictor for properties of image content at finer scales
- properties of image content for a scale at time t may be used as a predictor for properties of image content at time $t+1$ for the respective scale

Variations in corner detection results at a coarse scale may be promptly integrated in the prediction for the subsequent levels of the pyramid, based on the correlation assumption between pyramid levels stated above. As threshold estimation is more critical for robotic tasks for the finer scales, we propose a multi-scale analysis in which we apply a correction on a level given the prediction errors observed at coarser levels. The overall strategy is to choose the first extrapolation threshold τ_1 as close as possible to the optimal unknown threshold, and then rely on the observed mismatch in terms of detected corners for adjustment at the following time step.

Beside the interest for propagating prior information for thresholding in varying contrast settings, pyramid representations are essential and commonly used as well in robotic vision for coarse-to-fine egomotion estimation or cartography (see for example [2]). Therefore, relying on a pyramid representation for threshold prediction should integrate in the underlying application without determining a notable computational penalty. For our purposes, the total number of pyramid scales being considered is l. In the following paragraphs, we explain how we use the temporal and the hierarchic correlation assumptions for predicting optimal thresholds.

For the current time step t, the algorithm starts by setting for the coarsest level I^{l-1} a predicted threshold $\tau_{1,t}^{l-1}$ equal to the optimal threshold which was estimated at $t-1$ for the same level:

$$\tau_{1,t}^{l-1} = \hat{\tau}_{t-1}^{l-1} \tag{5}$$

This is due to the fact that at the coarsest level we do not have any other scale information; a slightly costlier solution would be to set $\tau_{1,t}^{l-1}$ at a constant low value in order to guarantee a good detection rate but we did not find any significant improvement by adopting this strategy. Then, the FAST detector is called using $\tau_{1,t}^{l-1}$. In order to cope with significant contrast variation with respect to the previous frame, we resort to a conservative low threshold if the result of the detection using $\tau_{1,t}^{l-1}$ is inadequate, especially since FAST computation time is negligible at this level. The occurrence model is fitted and then, based on the desired number of detections $\hat{N}^{l-1}$ for the coarsest level we estimate a posteriori an optimal threshold $\hat{\tau}_t^{l-1}$ and also a penalty between the value used for corner detection and the value predicted by our model:

$$\Delta\tau_t^{l-1} = \hat{\tau}_t^{l-1} - \tau_{1,t}^{l-1} \tag{6}$$

For any other scale I^k , we assume that coarser levels have been analyzed and we predict a threshold $\tau_{1,t}^k$ in a similar manner, except the fact that we also integrate the penalties provided by the higher levels. We search $\tau_{1,t}^k$ as $\tau_{1,t}^k = \tau_{1,t}^k(\hat{\tau}_{t-1}^k, \{\Delta\tau_t^q\}_{k<q<l})$ which gives at first order:

$$\tau_{1,t}^k = \hat{\tau}_{t-1}^k + \sum_{q>k} \lambda_{k,q}\Delta\tau_t^q \tag{7}$$

The weights $\{\lambda_{k,q}\}_{0\leq k<q<l}$ control how the hierarchical and temporal cues are integrated for the current scale prediction. They are estimated for each scale pair under the assumption that detection variation created by phenomena such as auto-exposure follows a constant pattern over time. For a given level k, the set $\{\lambda_{k,q}\}_{k<q}$ is updated at each time step as the solution of the following optimization:

$$\begin{aligned} \{\lambda_{k,q}\}_{q>k} &= \arg\min \sum_t \left[\hat{\tau}_t^k - \tau_{1,t}^k\right]^2 \\ &= \arg\min \sum_t \left[\hat{\tau}_t^k - \hat{\tau}_{t-1}^k - \sum_{q>k} \lambda_{k,q}\Delta\tau_t^q\right]^2 \end{aligned} \tag{8}$$

Using a simple minimization of Equation 8 with respect to variables $\{\lambda_{k,q}\}_{k<q}$, we determine the set of unknowns for each level k by inverting a symmetric matrix equal in size to the number of coarser levels. The terms of each matrix related to the minimization of the sum of squared errors are updated at each time step.

Alternatively, the parameters $\{\lambda_{k,q}\}_{k<q}$ may be computed offline on a characteristic sequence but we choose to compute them online since their determination is computationally light and we prefer to limit the parameters requested by SuperFAST to the number of detections desired at each level. Also, another solution to cope with intensity variations could be implemented by taking into account camera auto-exposure settings if available, since both these settings and the parameters $\{\lambda_{k,q}\}_{k<q}$ are involved in handling contrast variations.

C. Bucketing

The model proposed in the previous paragraphs assists the corner detection process in providing an optimal number of detections for each frame. However, the optimization of the number of responses is performed for the whole image, and thus the problem of spatial instability is not addressed. The challenge raised by uniform regions of the image is that in these locations successful detection requires very low thresholds; a more uniform repartition of corners may be facilitated by a bucketing strategy.

Instead of considering for image I a corner detection model determined by a parameter pair (C, σ), we divide I in $r \times c$ non-overlapping rectangular cells $I_{i,j}$ with $0 \leq i < r$ and $0 \leq j < c$. Under these circumstances, each cell benefits from an independent occurrence model $(C_{i,j}, \sigma_{i,j})$ which is adapted to the local variability.

The extra cost required by bucketing is minimal; it is mainly determined by the estimation of a set of $r \times c$ parameter pairs instead of a single global model.

In order to apply bucketing to the pyramid representation, at scale k with $0 \leq k < l$ we consider a cell representation for I^k of size $r_k \times c_k$ and in the following we denote the cell elements as $I^k_{i,j}$. The strategy introduced in Section III-B is identical, except the fact that correlations are performed at cell level. The temporal prediction $(\hat{\tau}^k_{t-1})_{i,j}$ is provided by the corresponding cell at $t-1$ and the pyramid penalties are provided by the corresponding cells in the coarser levels:

$$(\tau^k_{1,t})_{i,j} = (\hat{\tau}^k_{t-1})_{i,j} + \sum_{q>k} \lambda_{k,q} \Delta\tau^q_t \tag{9}$$

IV. Experimental results

In the following section, we estimate the reliability of our model, and we compare the prediction performance of SuperFAST to that of different FAST based detection strategies for an indoor and an outdoor sequence[1]. Regarding the hardware platforms, we run our tests on an architecture featuring an i5 quad core processor at 3.3 GHz, and also for the benchmarking section on an ARM quad core Cortex-A9 processor at 1.7 GHz.

For all tests, the video resolution of the frames is VGA and we use a pyramid representation with $l = 4$ levels. Whenever bucketing is considered, we rely on the following cell repartition:

$$\{(r_k \times c_k)\}_{0 \leq k < l} = \{(4 \times 6), (2 \times 3), (1 \times 1), (1 \times 1)\}$$

The desired number of corner detections per pyramid level is defined as $\{\hat{N}^k\}_{0 \leq k < l} = \{7000, 2000, 1000, 500\}$ for experiments depicted in Sections IV-A to IV-C.

A. Model accuracy

The performance of the results provided by SuperFAST in real-time applications is bounded by how accurately the square root exponential model is able to approach the corner distribution. In order to estimate the error of the model with respect to the observations, we proceed as follows.

For the indoor sequence and for each pyramid level k and time step t we measure the number of FAST points $N^{exp}(\tau, t, k)$for a threshold in the range $\tau \in [10, 80]$. Using the experimental observations, we determine the best possible parameters $(C, \sigma)_{t,k}$ according to the proposed model (Equation 2), through nonlinear optimization. Let $N(\tau, t, k)$ be the resulting function. We define the error bound as the infinity norm between the model predictions and the measurements:

$$\epsilon_{t,k} = \max_{\tau} |N^{exp}(\tau, t, k) - N(\tau, t, k)| \tag{10}$$

We present in red in Figure 4 the evolution of the error defined above for the finest scale of the pyramid. In order to provide an intuitive interpretation of the result, let us consider τ_ϵ the threshold value that maximises the model error for a given scale k and time step t. In blue we plot the value of $N^{exp}(\tau_\epsilon + 1, t, k) - N^{exp}(\tau_\epsilon, t, k)$ which is an approximation of the local derivative of N^{exp} at τ_ϵ, which is influenced by the discretization of τ. The results show that the error introduced by the model is bounded by the discretization of τ, and also that relatively to the local variation of number of corners our model approaches the observations very accurately.

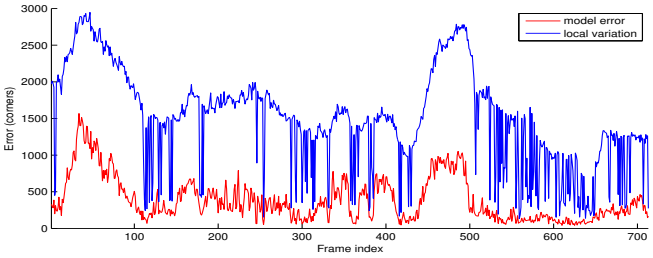


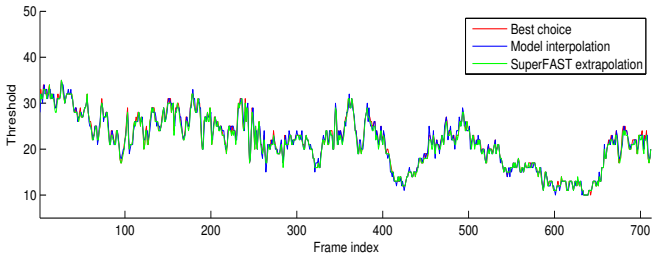
Fig. 4: For the finest scale, we present in red the error (infinity norm) between the observations and the best fit according to the model, as well as the local variation of the measurements at the maximum error point (in blue). The error introduced by the model remains below or around this level. The behaviour is similar for the other three scales considered in the experiments.

Next, we compare in Figure 5 the thresholds selected in order to obtain a specific number of responses per pyramid level. The red line indicates the best possible threshold, chosen according to the observed detections $N^{exp}(\tau, t, k)$, and in blue we plot the thresholds selected according to the best fit model. Again, we note that the model follows remarkably well the experimental data and is effective in guiding the choice towards the optimal threshold; the average error between the model threshold and the best choice is $\epsilon_1 = 0.32$ for the entire indoor sequence.

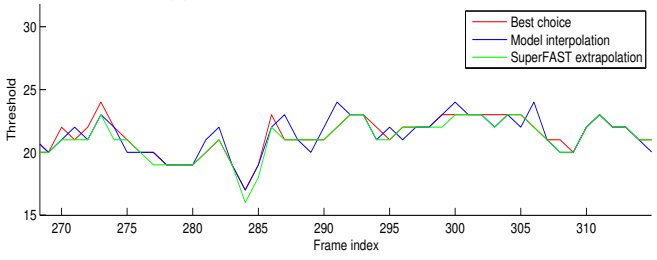
B. Model extrapolation and threshold prediction

Although we show that the SuperFAST model approaches the corner occurrence distribution very accurately, in the

[1]An implementation for testing is available at http://hebergement.u-psud.fr/emi/superfast/



(a) Threshold selection at scale 0



(b) Zoom of the first figure

Fig. 5: At the highest scale, we compare the threshold value selected in order to obtain a specific number of points for different strategies. In red we use the experimental data to select the best possible threshold. In blue, we provide the threshold selected by the best model fit for Equation 2 from experimental data (average error $\epsilon_1 = 0.32$). In green, we provide the result of SuperFAST selection (average error $\epsilon_2 = 0.54$). The behaviour of SuperFAST is similar for the other three coarser scales considered.

actual algorithm we do not have available the set of values of $N^{exp}(\tau, t, k)$, and in practice we extrapolate the model based on a single measurement $N^{exp}(\tau_1, t, k)$ (a second evaluation regarding $N^{exp}(\tau_2, t, k)$ is performed only on the result of the former evaluation, and is much faster). Then, as we explained in Section III-B, our algorithm tries to predict a threshold which is as close as possible to the optimal value, relying on temporal and multi-scale analysis.

In Figure 5 we plot in green the threshold selection in the actual SuperFAST execution, against the optimal choice (in red) inferred based on the actual observations $N^{exp}(\tau, t, k)$. The results illustrate that the extrapolation and the prediction strategies allow for a precise estimation of a threshold compared to the best possible choice, and also compared to the best model fit through nonlinear optimization. This time, the average error between the SuperFAST threshold and the best choice is $\epsilon_2 = 0.54$.

C. SuperFAST regularization of the detection

Figure 6 presents for comparison the number of detections during the video sequence by adopting different strategies. In Figure 6a, we show the variation of detected corners using FAST with a fixed threshold $\tau = 25$. Then, in Figure 6b we present the output of SuperFAST if we take into account only the temporal prediction, i.e. we use the same strategy as in Equation 5 for the whole image pyramid. Compared to the standard FAST behaviour, the temporal prediction regularizes significantly the number of detected corners. For comparison, we also show for the finest scale the result of using a proportional controller (P-controller), where the proportional gain has been determined using a grid search aiming to minimize the overall error for the same sequence. On initial consideration, both methods perform similarly and experience problems when consecutive frames exhibit strong saliency variability (average errors for optimal threshold estimation are respectively 1.20 for SuperFAST with temporal prediction and 1.29 for the P-controller). However, by looking more in detail we note that our method contributes significantly on the following points:

- the optimal proportional gain depends on the scene, and the result of the sequence-optimized P-controller shown in Figure 6b is not representative of a typical instance which usually performs much more poorly.
- the gain also depends on the number of corners to extract; the optimal gain varies by two orders of magnitude for the same scene depending on this input whereas SuperFAST does not need a parameter. On the illustrated sequence, the P-controller is instable compared to SuperFAST when the scene is less salient.
- the P-controller has an implicit dependence on multiple previous frames which can lead to consecutive failures; for an underlying navigation algorithm, a pattern of recurrent failures is much worse than occasional, spread misses.

Moreover, a key feature of our approach is to provide a good a posteriori estimation of the optimal threshold which can be used to improve the performance within a coarse to fine detection.

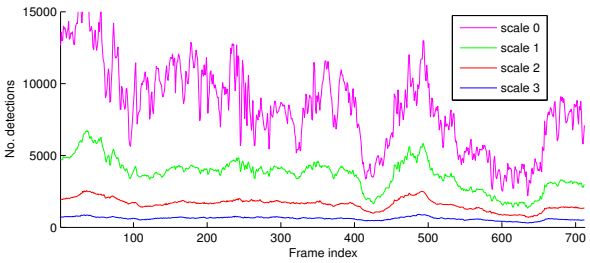
Finally, Figure 6c depicts the SuperFAST regularization when the hierarchical correction is taken into account as well, according to Equation 7. The role of the occurrence model is critical when we exploit it additionally for hierarchical prediction. In this way, SuperFAST is much more reactive, being able to adapt to sudden variations of image content as it performs a coarse-to-fine analysis in the image scale space (Figure 6c compared to Figure 6b).

TABLE I: Average detection time required by FAST9 with a fixed threshold $\tau = 18$, and by SuperFAST with bucketing on the indoor and outdoor sequences.

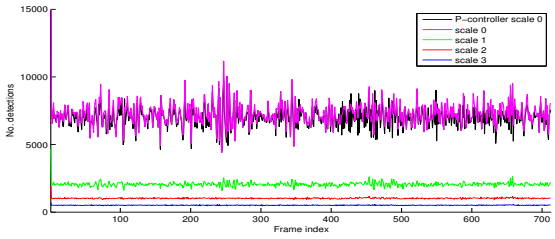
Arch. type	**Algorithm**	**Detection time in ms** ($\mu \pm 3\sigma$) **Indoor**	**Outdoor**
i5	FAST9-18	1.01 ± 0.57	0.56 ± 0.24
	SFAST-B	0.86 ± 0.47	0.59 ± 0.21
Cortex-A9	FAST9-18	4.86 ± 2.31	2.86 ± 0.77
	SFAST-B	4.14 ± 1.18	2.97 ± 0.73

D. Detection times

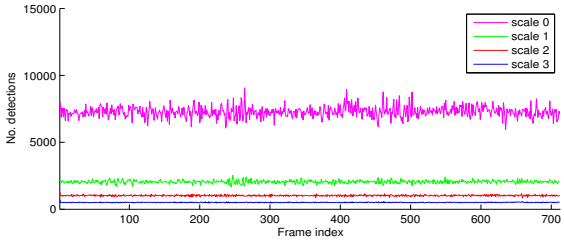
In Table I we present the average detection time required by FAST9 with a fixed threshold $\tau = 18$ and by SuperFAST with bucketing on the indoor and outdoor sequences. The implementation of FAST9 is SIMD-optimized for each



(a) FAST with $\tau = 25$



(b) SuperFAST with temporal prediction



(c) SuperFAST with temporal and scale prediction

Fig. 6: The figures above plot the number of corners detected for the four scales using different strategies, for the indoor sequence. In Figure 6a, FAST is employed with fixed thresholding. In Figure 6b, SuperFAST exploits only the temporal prediction at each scale based on the occurrence model. Finally, in Figure 6c SuperFAST makes use of temporal and scale prediction and provides the most stable output.

architecture and SuperFAST relies for the actual detection calls on the same optimized FAST9 implementation. Both algorithms are tested on the i5 and ARM architectures, with a single core being used each time.

For SuperFAST, we have requested a number of corners per scale (uniformly distributed within the cells) more adapted to real-time processing: $\{\hat{N}^k\}_{0 \leq k < l} = \{2000, 1300, 600, 500\}$. Both algorithms are tested on the i5 and ARM architectures.

In terms of detections, the output of the algorithms is in both cases similar to the situation depicted in Figure 1. In the outdoor sequence, SFAST-B is slightly slower than FAST for two reasons. The lower cells of the image are poor in corners, and the algorithm relies on lower thresholds than FAST for detection. Conversely, the indoor sequence features more corners overall, and a fixed $\tau = 18$ is costlier than the adaptive thresholds.

The results show that for a similar computational burden, SuperFAST provides a much more reliable and regularized output. In the following paragraphs, we will underline why this result is highly relevant for real-time robotic vision.

E. SLAM benchmark

Lastly, as an illustration of a full robotic vision application we run a visual SLAM on the indoor sequence. Our SLAM follows a typical strategy similar to [1], [2], with a tracking thread which performs matching between map points and current frame corners for pose estimation, and a mapping thread responsible for map augmentation and correction.

For each test, the same SLAM algorithm relies on different FAST based corner detection strategies performed on the i5 and the Cortex-A9 processors depicted at the beginning of the section. The algorithms tested are: an implementation SIMD-optimized for each architecture of FAST9 with a fixed threshold $\tau = 18$, the same FAST9 with a fixed threshold $\tau = 12$ and with bucketing, and SuperFAST with bucketing and relying for the actual detection calls on the optimized FAST9 implementation. For FAST9 with bucketing, we have requested the same number of corners per cell as in the case of SuperFAST.

Regarding the pruning that we use for FAST9-12B, the most effective strategy we identified works as follows. Firstly, we detect all FAST corners in the image using a threshold $\tau = 12$. Then we attribute a cell to each corner. Finally, for each cell where the number of corners is higher than the desired value and for each corner in these cells we compute a FAST score (the highest threshold for which the pixel is still considered a corner) using SIMD optimized code, and we use a partial sort to select the best (we either keep the best or remove the worst depending on the smallest amount of values to sort).

in order to select the best corners, we use the discretization of the FAST score and count the occurrences for each score vale. Then we add those occurrences in decreasing order of score until the number of corners is above the desired value, this method being faster than partial sort algorithms.

In Table II, we report the average corner detection time, and the average total SLAM processing time for each frame (detection, localization, mapping). We also provide the average number of potential inliers that are filtered by the localization module with a significant computational cost, and also the average number of corners that are validated as inliers. Finally, we provide the mean reprojection error.

The benchmark shows that SuperFAST has a significant positive impact on the SLAM performance; for a power efficient architecture for which the Cortex-A9 is a typical example, we improve SLAM performance in terms of precision and in terms of speed, from 56 fps to 93 fps on average. The result is mainly due to the selection of corners which favors a higher proportion of inliers, as it can be also observed from the experimental data.

We report for SFAST-B and FAST9-12B a similar ratio of potential inliers to inliers. The marginally higher number of inliers in FAST9-12B is mostly due to the significant frame-rate loss which leaves more time to the mapping module for adding key-frames - we observe on average a single

TABLE II: We present below a benchmark of different FAST based corner detection strategies used by a visual SLAM application which is executed on an i5 and a Cortex-A9 processor for the indoor sequence. The algorithms considered are FAST9 with a fixed $\tau = 18$, FAST9 with a fixed $\tau = 12$ and with bucketing, and SuperFAST with bucketing.

Architecture type	**Algorithm**	**Measurements ($\mu \pm 3\sigma$)**				
		Det. time (ms)	**SLAM Time (ms)**	**Potential Inliers**	**Valid Inliers**	**Repr. Error (px)**
i5	FAST9-18	1.01 ± 0.57	2.69 ± 1.18	198.65 ± 180.13	82.60 ± 94.01	0.39 ± 0.33
	FAST9-12B	3.85 ± 2.34	5.54 ± 1.88	150.05 ± 88.17	91.92 ± 69.41	0.34 ± 0.11
	SFAST-B	0.86 ± 0.47	2.28 ± 0.54	147.78 ± 87.66	88.41 ± 61.45	0.34 ± 0.10
Cortex-A9	FAST9-18	4.86 ± 2.31	17.78 ± 10.96	205.33 ± 288.66	85.98 ± 72.13	0.43 ± 0.61
	FAST9-12B	15.22 ± 9.56	23.57 ± 7.74	156.08 ± 86.19	95.37 ± 70.23	0.34 ± 0.11
	SFAST-B	4.14 ± 1.18	10.72 ± 2.33	137.45 ± 82.04	85.35 ± 62.07	0.33 ± 0.11

frame latency before a key-frame is fully integrated to the cartography instead of a 2-3 frame latency in the others cases.

Overall, in the context of an entire visual navigation application, we achieve with SuperFAST the same level of accuracy as with the most accurate alternative, but with a much higher and more regular frame processing time.

F. Discussion

In view of the results presented in the current section, we consider that the major contribution of this method is two-fold. Firstly, the proposition and the validation of a corner occurrence model allows us to estimate adaptively a detection threshold based on temporal and on scale consistency, with no extra computational cost for bucketing. The computational gain compared to alternative corner selection strategies is significant, especially for power-efficient systems.

Secondly, a major improvement in terms of computational efficiency is observed if we take into account the whole architecture of a vision based navigation algorithm. SuperFAST facilitates for a SLAM family application:

- a stabilized data input size, hence a balanced load for the different processing modules (matching, outlier rejection, odometry estimation, cartography update etc.)
- better adaptation and reactivity in heterogeneous environments, with a more intuitive parametrization
- an effective solution for adapting the computational burden to a specific architecture by setting the desired number of detections at a certain level (less corners meaning faster processing time)
- an indicator for the current visual saliency of the environment, through the adaptive threshold that is selected

Finally, the strategy we propose should be particularly adapted in the context of emerging high-definition cameras, such as 4K video capture devices. The experiments above allow us to highlight, for example in Figure 6a, the fact that for higher resolutions the detection and computational time instabilities are more marked, and this is precisely the context in which SuperFAST contributes significantly.

V. Conclusion and perspectives

In this paper we proposed a solution for adaptive thresholding for corner detection based on a corner occurrence model. The model allows us to predict accurately a threshold value based on previous observations performed on the time and scale axis. The effectiveness of the model, as well as that of our prediction mechanism, have been successfully validated in indoor and outdoor environments typical for robotic vision applications.

For future work, we envision to investigate further the statistical process that allows the square root exponential model to approach the corner occurrence distribution for the FAST detection criterion. Ultimately, we hope to extend our analysis for other widely used corner detection algorithms.

ACKNOWLEDGMENT

The authors acknowledge Benoit Pochon for his support, and Guillaume Papin for the architecture optimized implementations of the algorithms. The work presented has been patented by Parrot S.A.

References

[1] A. Huang, N. Roy, A. Bachrach, P. Henry, M. Krainin, D. Maturana, and D. Fox, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the International Symposium of Robotics Research (ISRR)*, 2011.

[2] G. Klein and D. W. Murray, "Parallel tracking and mapping for small ar workspaces," in *ISMAR*, 2007.

[3] S. Scherer and A. Zell, "Efficient onboard RGBD-SLAM for autonomous MAVs," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 1062–1068.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[6] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[7] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *ICCV*, 2005, pp. 1508–1515.

[8] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[9] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "Rslam: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, vol. 94, pp. 198–214, 2010.

[10] K. Schauwecker, R. Klette, and A. Zell, "A new feature detector and stereo matching method for accurate high-performance sparse stereo matching." in *IROS*. IEEE, 2012, pp. 5171–5176.

[11] R. Voigt, J. Nikolic, C. Hurzeler, S. Weiss, L. Kneip, and R. Siegwart, "Robust embedded egomotion estimation," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2011, pp. 2694–2699.

[12] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, June 2010, pp. 486–492.

[13] A. Bateman, "Contamination in seed crops III. Relation with isolation distance," *Heredity*, vol. 1, pp. 303–336, 1947.