Fast Visual Feature Selection and Tracking in a Hybrid Reconfigurable Architecture

Alessandro Bissacco[1], Soheil Ghiasi[2], and Stefano Soatto[1]

[1] Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
{bissacco,soatto}@cs.ucla.edu
[2] Department of Electrical and Computer Engineering
University of California, Davis
Davis, CA 95616
soheil@ece.ucdavis.edu

Abstract. In this work we introduce a fast visual feature tracking system which takes advantage of dedicated hardware to perform the computationally intensive step of selection. A software system uses the output of the hardware selector to develop tracks using filtering, data association techniques, and image-based validation.

1 Introduction

Feature selection and tracking is a crucial problem in a large number of computer vision applications, such as autonomous navigation, surveillance, structure from motion and scene estimation.

Current approaches to feature detection and tracking are based on detecting patches and computing their displacement between adjacent frames using optical flow or other differential techniques. Such algorithms perform particularly well in scenarios where there is limited inter-frame displacement, as in case of video streams from high frame-rate cameras. In such settings, however, online tracking of a large number of features as required by real world applications demands for computational resources that exceed the capacity of conventional CPUs.

In this work we propose using dedicated hardware for developing an extremely efficient implementation of feature detection and tracking. We capitalize on advances in powerful and inexpensive reprogrammable hardware to perform the computationally demanding feature selection process at every frame using FPGA hardware. Then tracking reduces to the problem of filtering and data association on the detected points, which can efficiently be implemented in software on a standard PC.

2 Related work

The literature on feature tracking is very broad, here we will review only the approaches most relevant to our work.

Following the seminal work of Tomasi et al. [10], a various number extensions and variants have been proposed (see [15] and references therein).

Current feature trackers, such as [3], can track many hundreds of points at 60Hz or greater, but suffer from limitations on the speed of selection (for adding new points to the tracker) and require nearly all the available processing power of the CPU. By using separate, custom hardware to select hundreds of points per frame and then associating these points to tracks in software, we can achieve greater than 60Hz real-time tracking without burdening the primary CPU, which can use the output of the system for other tasks, such as structure from motion.

An approach closely related ours has been proposed in Nister et. al [13], using efficient frame-by-frame corner detection and performing tracking by matching detections. Points are extracted with a fast implementation of the Harris detector [14] and matching is done by comparing neighbors using normalized cross correlation. The main limitation of this method is the computational cost of the detection step and the fact that by discarding the information provided by temporal continuity, contrary to our approach, subpixel accuracy cannot be achieved.

Our tracking system consists of a number of hardware and software components. Images are initially processed by a field programmable gate array (FPGA) which runs a modified Harris corner detector, selecting the local maxima of the Harris function as feature points. These are provided to the tracking system, which uses a second-order Kalman filter to predict feature locations on the current frame. Given the prediction, previous tracks, and newly selected points from the FPGA, a data association technique is used to match new points to existing tracks. To assist the matching process, we use normalized cross-correlation to prune incorrect correspondences. Finally, the new tracks are used to predict the feature locations in the next frame.

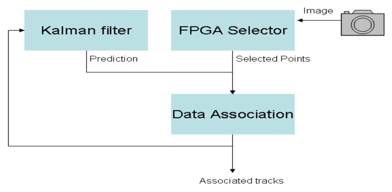


Fig. 1. The components and flow of the tracking system are illustrated above.

3 Feature Detection and Tracking

In this section we describe in details the components of our system: feature selector, filter and data association module.

3.1 Feature selection

Feature selection is implemented on a reconfigurable computer based on Field Programmable Gate Arrays (FPGA's). The method, proposed in [9], is a simplification to the Tomasi and Kanade selection technique [10] requiring only summations and multiplications on integers. We refer to the paper [9] for details.

3.2 Feature tracking

In our study of the point association and tracking problem we explored several approaches. We tested the straightforward solutions of matching selected points in adjacent frames first by nearest neighbor and then by Normalized Cross-Correlation of the surrounding patches, with very little success. The main reason for the failure of these attempts is that the output of the feature selector does not exhibit a stable behavior, points disappear and reappear from one frame to the next, and attempts of directly matching points in adjacent frames will inevitably produce poor results. Moreover, the computational load for complete NCC tests is too high for a real-time application, in particular given our goal of tracking hundreds of features at frame rates not lower than 60Hz.

We propose to enforce temporal continuity by assuming linear dynamics for the tracked points in the image sequence. If we assume that selected features are measurements of underlying tracks whose dynamics is described by a Gaussian ARMA model, then we have a standard problem of tracking and data association in a multitarget environment with less-than-unity probability of detection and presence of false alarms. This problem has been studied in the literature for decades and a number of effective approaches have been developed (see [1] for a review).

What makes our scenario different from the one considered in the target tracking literature is the presence of images, which can be used as a rich source of information for discrimination and false alarm rejection in the data association step. We explored extensions to established tracking techniques by exploiting measures of similarity between feature patches.

Tracking and Data Association. Among the various tracking algorithms proposed in the literature, the main ones are Joint Probabilistic Data Association (JPDA [1]), Global Nearest Neighbor (GNN [4]), and variants (Probabilistic Data Association [2], Suboptimal Nearest Neighbor [5], Cheap JPDA [6], Fast JPDA [8] and Suboptimal JPDA [7]) . In all these algorithms Kalman Filters are used for tracking, and the predictions from the filters are used to define the regions where the features are expected to appear in the next frame - the so called validation gates. In particular, if $\hat{y}^i$ and S^i are respectively the predicted measurement and the variance of the innovation of the Kalman filter associated to track i, then the validation gate of track i is the set of points y such that the normalized distance:

$$d^i(y) = \sqrt{(y - \hat{y}^i)^T (S^i)^{-1} (y - \hat{y}^i)} \tag{1}$$

is less than a predefined value g (typically $g = 4$).

The PDA is a single track-multiple measurements association algorithm. It uses a weighted average of the measurements in the validation gate to update a track, where the weights are the posterior probabilities that the measurements have been generated by the track assuming less than one probability of detection and Poisson distributed false measurements. The JPDA is a multiple track-multiple measurements association scheme obtained by extending PDA to multiple targets, where the probabilities of all the possible associations measurements-tracks are evaluated in order to compute the weights. This is done by forming clusters consisting of all tracks that have measurements in the intersection of their validation gates. For each cluster all the possible associations measurement-track are considered and the corresponding probabilities are computed. Since the number of association hypotheses increases exponentially in the number of tracks and points in a cluster, and in test images with 1000 selections we obtained clusters exceeding 100 tracks, a direct JPDA implementation is not possible for a real-time application. Even though we can use one of the several suboptimal solutions proposed in the literature (CJPDA, SJPDA, FJPDA), there is another main issue. JPDA shows undesirable characteristics when used in a dense target environment such as the one we consider. In particular track bias and track merging may occur when several closely spaced targets travel in the same direction. To solve this problem, in [6] a Nearest Neighbor implementation of JPDA (NNJPDA) has been suggested. In this variant, instead of using a weighted average of the observations in the validation gate to update a track, the observation with highest association probability is used.

In Global Nearest Neighbor, a track can be updated by at most one measurement and a measurement can be assigned to at most one track. The first step is to form an assignment matrix $A = [d^i(y^j)]$ where $d^i(y^j)$ is the normalized distance (1) between track i and measurement y^j. Then the solution to the track-measurement association problem is obtained by maximizing the number of assignments while minimizing the sum of the the normalized distances of the assigned track-measurement pairs. The Hungarian (or Munkres) algorithm [16] can be used to find the optimal assignment, but its complexity is $O(n^3)$, where n is the bigger between the number of measurements and the number tracks. Since we want to be able to track hundreds of points in real time, we consider a suboptimal solution, the Suboptimal Nearest Neighbor. This is a greedy approach to the matching problem: it searches the measurement-to-track pair with the minimum normalized distance, makes the assignment, removes all the pairs contaning the assigned measurement or track, then repeats the first step until no more assigment is possible. The complexity of this algorithm is $O(n \log(n))$ due to the ordering step. A further decrease in complexity can be achieved by first segmenting the input data in clusters as in the JPDA algorithm, then applying the SNN matching within each cluster.

A comparison of the performance of the discussed tracking algorithms was presented in [11]. The tests were conducted on radar data of real closely spaced maneuvering targets, and the results favor the Nearest Neighbor approaches. Consequently, given also the strict computational constraints due to the goal of a real-time application, we opted for the Suboptimal Nearest Neighbor approach.

3.3 Tracking feature patches

A natural extension of conventional target tracking algorithms for the problem of feature tracking in images is to modify the notion of normalized distance by including validation based on the appearance of the tracked patch. Our solution consists in comparing the patch surrounding the candidate selected point at the current frame with the patch at the estimated track position in the previous frame. Then the normalized distance (1) of measurement y^j from track i at time t becomes:

$$d_t^i(y^j) = \begin{cases} \sqrt{(y^j - \hat{y}^i_{t|t-1})^T (S^i_t)^{-1} (y^j - \hat{y}^i_{t|t-1})} & \text{if } \operatorname{NCC}(I_t(y^j), I_{t-1}(\hat{y}^i_{t-1|t-1})) < h \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

where h is a threshold, $\operatorname{NCC}(I_1(y_1), I_2(y_2))$ denotes the Normalized Cross-Correlation between the patch centered in y_1 on image I_1 and the patch centered in y_2 on image I_2, $y^i_{t|t-1}$, $y^i_{t-1|t-1}$ and S^i are the prediction, filtered estimate and innovation variance of track i computed at time $t-1$.

The effect of this modification is to reject from the candidate points all the selections whose appearance does not match the one of the track in the previous frame. This test allows to prune out wrong potential matches and is simple enough to allow for a real-time implementation.

3.4 Track initiation

In a scenario where track-data association is a main issue special care must be taken in the initiation of new tracks. This problem is well know and several algorithms have been proposed for its solution [1]. In view of some comparison studies on track initiator performances [12], we concluded that the logic-based method [1] was the most suitable for our application. In brief, the approach works by allowing multiple associations in the first three measurements and validating the best match in the forth step (see [1] for details).

4 Implementation and experiments

For image acquisition, we are using a Teli CS8550DiF DCAM-compliant CCD camera, which has a resolution of 640x480 monochrome pixels and a refresh of 60 full (non-interlaced) frames per second.

4.1 Feature selection

The current version of the Harris selector is implemented on a Xilinx VIRTEX WILDSTAR/PCI FPGA board. The board has the following features: 1 million system gates, 1.6 Gbytes/second I/O Bandwidth, 6.4 Gbytes/second Memory Bandwidth Processing clock 100 MHz, Synchronous ZBT SRAM with 100 MHz access.

Selection takes about 160 microseconds on average on 640×480 images, which is approximately constant with the number of features, since the algorithm runs on the entire image every frame. The board is connected to a host computer via the PCI bus. Because of limitations on the hardware, it takes about 20 milliseconds to move data from the camera to the memory on the FPGA board. This accidental hardware limitation does not allow us to display the complete end-to-end system running in real time. Therefore, for experimental purpose, we use a stored sequence of images as input to our algorithms and measure the time required for each component. An alternative FPGA board, the Micro-Line C6713Compact, has on-board Firewire (along with drivers for DCAM-compliant cameras) which writes directly into memory accessible by the FPGA. This will eliminate the memory-transfer delay and allow the system to run in real-time. Due to its high cost, however, its purchase has been ruled out since the feasibility of the system can be equally well illustrated by careful timing of each component.

A threshold parameter λ_t is used to control the number of selections per frame.

The image gradients I_x and I_y are obtained by computing the centered differences, which amounts to convolving the image with the kernels $\begin{bmatrix}-0.5 & 0 & 0.5\end{bmatrix}$ and $\begin{bmatrix}-0.5 & 0 & 0.5\end{bmatrix}^T$. This solution is computationally efficient but it is known to be rather sensitive to the noise. Significant improvements in accuracy and stability of results may be obtained in the future by using smoothing kernels in the calculation on the image gradients.

In figure 2 we show the number of selections in a sample outdoor sequence, where we can see some the fluctuations of the total number of features. Besides that, it must be pointed out that there is significant additional interframe variation in the selections due to disappearing and reappearing of features. The performance of the tracker are very sensitive to the quality of the selections, but not particularly to their number. We also show average and standard deviation of the lifetime of tracks for the previous sequence as function of average number of selected features. The flatness of the curve shows how the choice of the selection threshold is not critical for the overall lifetime of the tracks.

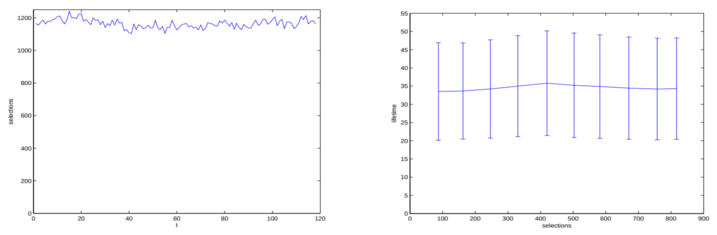
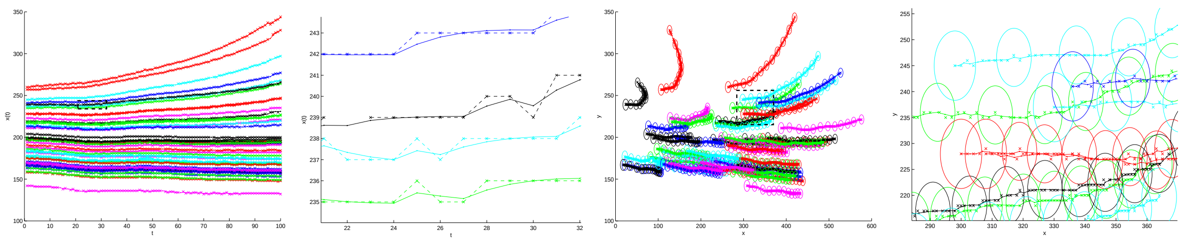


Fig. 2. Left: Number of feature selected in a outdoor sequence. Right: Tracks lifetime vs. number of selections for a 100 frame sequence: mean (solid line) and standard deviation (vertical bars).

4.2 Data association and tracking

The data association and tracking algorithm is composed of 3 modules: track initiator, filter/predictor, and data association.

Track initiation Modified logic-based track initiation scheme as described in [1], with the following parameters: frames required for successful track initiation set to 4, and maximum number of track splits in the second frame equal to 3.



(a) (b) (c) (d)

Fig. 3. Kalman filter estimates and validation gates for 50 tracks in a 100 frames clip. Estimates shown as dots connected with solid lines, measurement as crosses connected by dashed lines and validation gates as circles. (a) shows x coordinate of all the tracks, (b) is a zoom of the small dashed box in (a). We can see the smoothing effect of the filter which provides subpixel accuracy. (c) displays the prediction and validation gates of all the tracks, (d) is a zoom-in of the small dashed box in (c). It can be seen that most validation gates contain multiple measurements, which can be correctly associated thanks to the image-based validation and the temporal dynamics model encoded in the tracking filter.

Feature Selection Algorithm	Average Number of Selections	Average Number of Tracks	Average tracking time per frame (ms)	Frame rate (Hz)
Harris	1000	510	11.21	89
FPGA Tomasi-Kanade	1000	411	9.81	102
Harris	500	318	5.40	185
FPGA Tomasi-Kanade	500	250	4.74	211
Harris	200	128	2.54	393
FPGA Tomasi-Kanade	200	114	2.52	397

Table 1. Results and timing of the tracker with different selection algorithms

The track initiation is computationally the most expensive part of the algorithm. If larger number of tracks or bigger frame rates are desired, big speed gains can be achieved by removing this module and initiating a track for each new selection.

Kalman prediction and filtering. We model the position $x(t)$ of the features on the image as second order Brownian motion:

$$\begin{bmatrix} x(t+1) \\ v(t+1) \end{bmatrix} = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + w(t) \;,\; \begin{bmatrix} x(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \end{bmatrix} \;,\; w(t) \sim \mathbf{N}\left(0, Q\right))$$

$$y(t) = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} + v(t) \quad , \quad v(t) \sim \mathbf{N}\left(0, R\right) \tag{3}$$

where $y(t)$ are detected point coordinate, $v(t)$ is i.i.d Gaussian noise with variance $Q = diag(\begin{bmatrix} q_x \; q_y \; q_v \; q_v \end{bmatrix})$, $R = rI$, x_0 is the position in the first frame, q_x, q_v, r are parameters to be tuned according to the dynamics of the scene: q_x set to a small number or zero, q_v related to the velocity of the features, and r based on the confidence in the measurements (because of discretization not smaller than 0.5). A linear time invariant Kalman filter is derived from the motion model (3). The initial variance P_0 of the filter

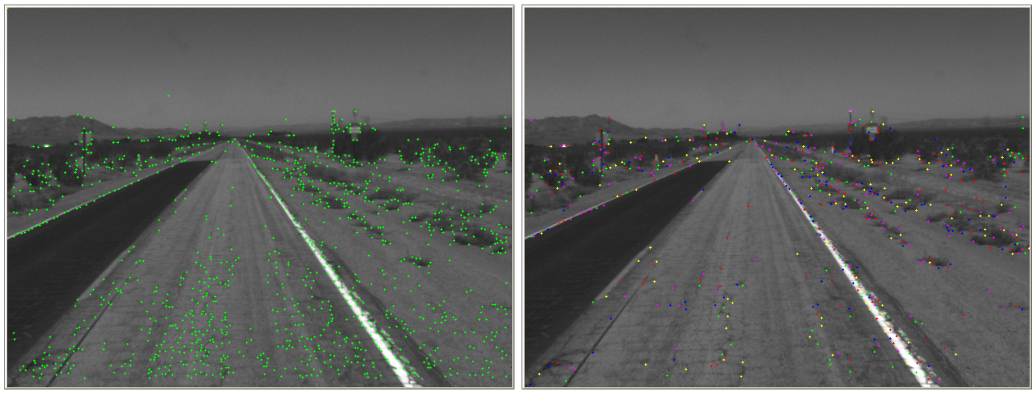


Fig. 4. Left: FPGA selection output, sample frame. There are 1162 feature selected, obtained with threshold parameter $\lambda_t = 50$. Right: Tracking output, sample frame, obtained with selections in figure 4 as input. Tracks are displayed as dots of different colors, their total number in this frame is 562.

estimate is obtained by first solving the Riccati equation for the filter, then applying n_0 prediction steps to the result. n_0 measures the number of measurements needed to reach the steady state and is a parameter of the algorithm. In this way we guarantee a monotonically decreasing error variance S, therefore a decreasing validation gate area.

Data association. For track-measurement association we implemented the Suboptimal Nearest Neighbor algorithm with modified normalized distance as described in 3.2 . A number of optimizations have been applied to the original algorithm in order to allow for real-time performance. First a JPDA-like clustering step is performed to reduce the number of the association hypotheses. Then tracks and measurements are organized in a structure with spatial information that allows to prune out pairings exceeding the maximum distance defined by the validation gate of the track. Finally, an optimized implementation of Normalized Cross Correlation has been included. Parameters of the algorithm are the size of the patch used for the NCC test and the threshold h used for calculating the normalized distance (2).

When no measurements can be associated to a track, the measurement update is step is not performed and an additional prediction step of the Kalman filter is applied.

A track is terminated when either a maximum number of consecutive predictions steps is attained, or the norm of the innovation exceeds a defined threshold, both parameters of the algorithm.

Figure 3 shows the trajectories of the filtered estimate for a small number of tracks in a video clip. It is clear from the zoomed plot how the filter successfully interpolates the noisy measurements. Figure 3 depicts also the predicted positions together with the validation gates. We can see how the gates become smaller as new measurements are associated to the track.

To illustrate the performance of our tracker, we show in Table 1 the results of tracking a sequence of 100 frames with different selection algorithms and thresholds (see figure 4 for sample frames).

The Harris feature selector used in these experiments is the implementation available in the Intel OpenCV library [3]. It is clear from the data that the Harris corner detection significantly improves the number and quality of the tracks.

The timings in Table 1 were measured on a Pentium IV 3.2 GHz PC with 1 Gbyte of RAM and do not include selection time.

5 Conclusions

In this work we developed a hybrid scheme for fast feature selection and tracking. A hardware reconfigurable architecture is dedicated to the feature selection process, which allows for high framerate selection in full resolution videos. A software module then performs tracking by using filtering and data association techniques on the selected points. We complemented state-of-the-art data association approaches with image-based validation to prune incorrect matches and allow for higher tracking accuracy. We demonstrated the excellent performances of our approach in terms of computation time and tracking results.

References

1. Y. Bar-Shalom and T. E. Fortmann. Tracking and Data Association. Academic-Press, Boston, 1988
2. Y. Bar-Shalom and X. R. Li. Multitarget-multisensor tracking: principles and techniques. Storrs, CT: YBS, 1995
3. Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. Included in OpenCV documentation, 2000.
4. S. S. Blackman. Multiple target tracking with radar applications. Norwood MA: Artech House, 1986.
5. A. Farina and F. A. Studer. Radar data processing I - Introduction and Tracking. Research Studies Press, 1985.
6. R. J. Fitzgerald. Development of practical PDA logic for multitarget tracking by microprocessor. In *Multitarget-Multisensor Tracking: Advanced Applications*, 1990.
7. A. Roecker and G. L. Phillis. Suboptimal joint probabilistic data association. IEEE Transactions on Aerospace and Electronic Systems, 29, 2 (1993).
8. B. Zhou and N. K. Bose. Development of practical PDA logic for multitarget tracking by microprocessor. In *IEEE Trans. Aerosp. Electron. Syst.*, 29(2), 352-363, 1993.
9. A. Benedetti and P. Perona. Real-time 2-D Feature Detection on a Reconfigurable Computer. In *Proc. CVPR*, 1998
10. C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.
11. H. Leung, Z. Hu and M. Blanchette. Evaluation of Multiple Radar Target Trackers in Stressful Environments In *IEEE Trans. Aerospace and Electronic Systems*, 35(2), 1999.
12. Z. Hu, H. Leung and M. Blanchette. Statistical performance analysis of track initiation techniques. In *IEEE Transactions on Signal Processing*, 45, 2, 445-456, 1997.
13. D. Nister, O. Naroditsky and J. Bergen. Visual Odometry In *Proc. CVPR 2004*.
14. C. Harris and M. Stephens. A combined corner and edge detector In *Proc. Alvey Conference*, pp. 189-192, 1988.
15. S. Baker and I. Matthews. Lucas-Kanade 20 Years On: A Unifying Framework In *Proc. International Journal of Computer Vision*, March 2004.
16. H. W. Kuhn. The Hungarian method for the assignment problem. In *Naval Research Logistics Quarterly*, 2:83, 1955