SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors

Duy-Nguyen Ta
Georgia Institute of Technology
duynguyen@gatech.edu

Wei-Chao Chen Natasha Gelfand Kari Pulli
Nokia Research Center, Palo Alto
{wei-chao.chen,natasha.gelfand,kari.pulli}@nokia.com

Abstract

We present an efficient algorithm for continuous image recognition and feature descriptor tracking in video which operates by reducing the search space of possible interest points inside of the scale space image pyramid. Instead of performing tracking in 2D images, we search and match candidate features in local neighborhoods inside the 3D image pyramid without computing their feature descriptors. The candidates are further validated by fitting to a motion model. The resulting tracked interest points are more repeatable and resilient to noise, and descriptor computation becomes much more efficient because only those areas of the image pyramid that contain features are searched. We demonstrate our method on real-time object recognition and label augmentation running on a mobile device.

1. Introduction

Robust feature descriptors such as SIFT [12], SURF [2], and GLOH [16] have become a core component in applications such as image recognition [8], multi-view stereo [25], and image registration [4, 26]. These descriptors are stable under viewpoint and lighting changes, so they are able to cope with significant amounts of image variability. At the same time, discriminative power is achieved by representing feature points as high-dimensional vectors. The combination of robustness and discriminative power makes these methods ideally suited for searching large heterogeneous image databases.

We are interested in using robust feature descriptors in real-time object recognition and tracking applications. Given a database of images of labeled objects, such as buildings in an outdoor environment, and an input video stream, we would like to recognize the objects present in the video, augment the video stream with labels indicating the objects, and maintain and update those labels as the video pans across the scene. Figure 1 shows an example of such an application. This task can be thought of as having two components. Image matching against a database is performed to identify new objects that appear in the video and object tracking is performed to update the positions of the labels of recognized objects in the consecutive video frames. Robust feature points with high dimensional descriptors perform best for image recognition, therefore we would like to compute and track them at interactive frame rates.

To track robust features across video frames, we can perform pairwise image matching given any two consecutive video frames, as done by Battiato *et al.* [1] and Skrypnyk and Lowe [24]. Figure 2 describes the algorithm flow of this approach. The main drawback of frame-to-frame matching is the wasted computation as this approach does not exploit coherence in the video. Most of the time the features are used for tracking purposes only, as there is no need to perform an image recognition step, unless a significant change is detected in the current frame. The expensive robustness properties of the descriptors are not needed for frame-to-frame matching, since consecutive video frames are likely to be very similar. Furthermore, image noise means that many of the descriptors are transient and will be thrown away when consecutive frames are matched. Therefore, performing detection and computation of robust descriptors for each frame of the video is unnecessary and can also be difficult to perform at interactive frame rates.

Alternatively, one can imagine a hybrid algorithm where motion estimation is done through lightweight features such as FAST corners [19, 20] or Harris corners [9] and object recognition is done through robust features. When enough motion is accumulated to warrant a recognition step, one can run a standard image matching algorithm against the image database to detect what is present in the scene. However, locations of the robust features such as SIFT and SURF are unlikely to coincide with the features used for tracking. This is a problem since the position of any augmentation that results from the matching step, such as building labels, can not be transferred to successive frames without knowing the scene structure. A solution to this is to compute the robust descriptors at corner locations that are used for tracking [5, 29]. These approaches, even though

(a) (b)

Figure 1. Our algorithm running in an outdoor environment. (a) Detected interest points and traced trajectories from a video sequence (Section 3). (b) Above: Video frames overlaid with their object labels zoomed in. Below: Matching images and their partial subgraph from the database. The solid and dashed edges indicate geometric and object ID relationships, respectively (Section 4)

fairly efficient in terms of tracking, require extracting multiple descriptors per corner, because the corners are not scale-invariant. The SIFT Flow approach by Liu *et al*. [11] produces a flow field of SIFT descriptors with a fairly high density, but this requires computing feature descriptors at a much higher density than produced by the difference-of-Gaussians algorithm of standard SIFT. Furthermore, the performance of SIFT and SURF is related to the distinctiveness of the computed interest points, so using descriptors computed at alternative locations can negatively impact the recognition performance of the algorithm.

Contributions. To enable tracking of robust descriptors at interactive frame rates, we propose to exploit coherency in video frames to detect interest points at locations in each frame where they are most likely to appear. Instead of tracking features in 2D images, we instead perform our tracking in the scale-space image pyramid, and we achieve the robustness of the direct frame-to-frame matching method while reducing computation significantly. The detected interest points are scale-invariant, and are inherently matched and tracked across video frames. We decouple the descriptor computation from the interest point detection so that feature descriptors are computed only for the purpose of object recognition. We demonstrate that our algorithm runs in real-time on mobile phones, and we discuss applications such as real-time augmentation and object recognition. Figure 3 shows a functional overview of our algorithm.

2. Related Work

Robust Feature Detectors and Descriptors. There are several approaches to scale-invariant interest point detection, including Difference-of-Gaussians in SIFT by Lowe [12], maximally stable extended regions (MSER) by Matas *et al*. [14], Harris-Affine and Hessian-Affine corners by Mikolajczyk and Schmid [15], and Hessians approximated using Haar-basis by Bay *et al*. [2]. Mikolajczyk *et al*. [17] performed a comprehensive comparison study on the detectors.

Robust descriptor algorithms take the output of region detectors, construct a canonical frame, and then extract a high-dimensional feature vector for each region. The descriptors are designed to be invariant to lighting, scale, and rotational changes. Examples include the very popular SIFT [12] and SURF [2]. Refer to Mikolajczyk *et al*. [16] for a comparative study on the descriptors.

There are also many successful attempts to speed up the descriptor computation algorithm. Sinha *et al*. [22] describe an efficient SIFT implementation on graphics processing unit (GPU). Grabner *et al*. [7] propose to speed up SIFT computation using integral images. Our SURFTrac algorithm improves on baseline SURF algorithm which is our primary benchmark; additional speedups similar to these approaches can be applied to gain even more efficiency.

Applications of Feature Descriptors. Many researchers have applied feature descriptors in object recognition and image retrieval. For example, Sivic and Zisserman [23] applied SIFT for efficient keyframe retrieval in video. Grauman and Darrell [8] proposed a fast kernel-based object identification method using SIFT features. Nistér and Stewénius [18] used hierarchical k-means to construct a search tree of local features for fast and efficient image retrieval.

Motion Estimation and Tracking. There is an abundant body of prior art on video tracking and motion estimation. Refer to Torr and Zisserman [28] for a summary of feature-based motion estimation methods. Optical flow algorithms such as the classic examples by Lucas and Kanade [13] and Berthold *et al*. [10] can also be used as input for a motion estimation algorithm. Zhang *et al*. [30] recover epipolar ge-

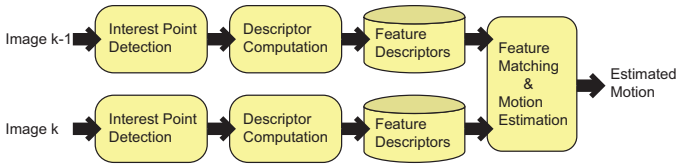

Figure 2. An algorithm using feature descriptor algorithms *as-is* for feature tracking.

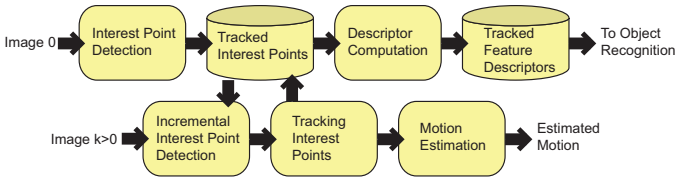

Figure 3. The SURFTrac algorithm overview. For the first image, the algorithm initializes a list of interest points by performing a full detection. The interest points are then updated and tracked upon receiving new images. The descriptors are for recognition purposes and the algorithm computes them as needed.

ometry between two images through correlation, relaxation, and robust model fitting using Harris corners. The feature-tracking aspect of our algorithm can be used as input to motion estimation algorithms, and therefore complements instead of competing with motion estimation research.

There have been many successful attempts to track robust descriptor features. Skrypnyk and Lowe [24] and Battiato *et al*. [1] propose to track SIFT features similar to Figure 2. Other algorithms change the location of descriptor computation by using different interest point detectors. For example, Liu *et al*. [11] compute dense SIFT correspondences by detecting and matching the descriptors on a dense grid. Chehlov *et al*. [5] describe a Kalman filter based SLAM algorithm that uses a corner detector followed by SIFT computation on these corners. Similarly, Wagner *et al*. [29] compute FAST corners on the object database image and extract descriptors at different scale levels, and they also demonstrate an application that performs tracking and object recognition in real time.

Our algorithm is related to these studies, but our approach is quite different because we wish to retain all of the proven great attributes of robust feature descriptors. We aim to perform image recognition similar to Takacs *et al*. [27] where a large image and object database need to be matched against the input image. This means that we do not wish to replace scale-invariant interest points with generic corner detectors when the resulting descriptors would reduce the effectiveness of object recognition. From this perspective, the quality of object recognition is equally important to the real-time tracking requirement, and the tracking algorithm should not interfere with the recognition performance.

3. The SURFTrac Algorithm

Many feature descriptor algorithms consist of two consecutive steps, namely interest point detection followed by descriptor computation. An interest point detection algorithm extracts regions of interest that tend to be repeatable and invariant under transformations such as brightness or perspective changes. In the descriptor computation step, each extracted interest point defines a circular or affine region from which one descriptor is computed. It is often possible to mix and match these two steps of the algorithm, for example, one can compute a SIFT descriptor using Hessian-Affine interest points.

To achieve scale-invariance, many interest point detection algorithms use image pyramids during the detection phase. These algorithms include Hessian-Affine, Harris-Affine, and approximate Hessian. In this process, an image pyramid is formed by downsampling the input image to a progressively lower resolution. For the purpose of our discussion, we will treat the image pyramid as a stack of same-sized images $S_k(\cdot)$, each filtered from the original image I_k with a different scale of zero-mean Gaussian as follows

$$S_k(x, y, \sigma) = I_k(x, y) * G(0, \sigma^2). \tag{1}$$

The interest point response R has the same image stack data layout, and is computed by applying the response computation function $f(\cdot)$ over the stack of images S,

$$R_k(x, y, \sigma) = f \cdot S_k(x, y, \sigma). \tag{2}$$

Local maxima in the function R represent relatively stable regions and are extracted as interest points. Because the bandwidth of function R is lower at higher values of σ, the sampling rate for maxima computation is naturally reduced at higher σ to increase efficiency. The extracted interest points are then refined with smooth local interpolation [3].

As described before, we plan to extend interest point detection algorithms such that the interest points are tracked across video frames efficiently. Although our proposal is adaptable to many algorithms using image pyramids, we will focus the remainder of our discussion on the approximate Hessian detector in SURF [2] because of its efficiency and good interest point repeatability. More importantly, by using SURF we can compute part of $R_k(\cdot)$ directly without having to produce the intermediate Gaussian stack $S_k(\cdot)$. The algorithm computes the scale-normalized Hessian matrix

$$\mathbf{H}_k(x, y, \sigma) = \frac{1}{\sigma^2} \begin{bmatrix} \frac{\partial^2}{\partial x^2} S_k(x, y, \sigma) & \frac{\partial^2}{\partial x \partial y} S_k(x, y, \sigma) \\ \frac{\partial^2}{\partial x \partial y} S_k(x, y, \sigma) & \frac{\partial^2}{\partial y^2} S_k(x, y, \sigma) \end{bmatrix}, \tag{3}$$

and the response function is the determinant of $\mathbf{H}(\cdot)$, $R_k(x, y, \sigma) = det(\mathbf{H}_k(x, y, \sigma))$. In the following sections, the use of the Haar-wavelet approximation in SURF is implied when we refer to the Hessian matrix. In the remainder of the section we describe the algorithm shown in Figure 3.

3.1. Incremental Interest Point Detection

In order to compute interest points in each video frame incrementally, we can predict regions in the Gaussian image stack where useful features are most likely to appear, and compute the response R only in these regions. Let us denote the input video sequence as $\mathbb{I} = \{I_0, I_1, \cdots I_{N-1}\}$. Given image I_{k-1} and one of its interest points $\mathbf{p}_{k-1} = (x_{k-1}, y_{k-1}, \sigma_{k-1})$, assuming we know the relative motion $M_k^{k-1}(.)$ between I_{k-1} and I_k as a homography, we can simply transform $\mathbf{p}_{k-1}$ to its location in frame I_k with

$$\mathbf{p}_k = (x_k, y_k, \sigma_k) = M_k^{k-1}(\mathbf{p}_{k-1}). \tag{4}$$

Obviously, homography is insufficient to model the motion of all tracked features. However, if the relative motion between I_{k-1} and I_k is small, we can still use the homography and expand the point $\mathbf{p}_k$ into a 3D volume search neighborhood $\mathbf{P}_k$

$$\begin{aligned} \mathbf{P}_k = \{(x'_k, y'_k, \sigma'_k) : \quad & |\sigma'_k - \sigma_k| \le \Delta_\sigma, \\ & |x'_k - x_k| \le \gamma\sigma'_k, \\ & |y'_k - y_k| \le \gamma\sigma'_k\}, \end{aligned} \tag{5}$$

where Δ_σ is the search range in the scale space, and γ is related to the motion prediction error, as well as disparity of the tracked point with respect to the primary planar structure of the scene. The search neighborhood $\mathbf{P}_k$ corresponds to a pyramidal frustum because the interest point sampling rate is reduced at higher scale levels. In practice, because of the high correlation between images, using a fixed-size search neighborhood works fairly well regardless of the actual motion. Using Equation 4 and 5, the collection of tracked interest points $\{\mathbf{p}_{k-1}^0, \mathbf{p}_{k-1}^1, \cdots \mathbf{p}_{k-1}^{m-1}\}$ from image I_{k-1} forms a joint neighborhood $\mathbb{P}_k = \{\mathbf{P}_k^0 \cup \mathbf{P}_k^1 \cup \cdots \cup \mathbf{P}_k^{m-1}\}$ where useful interest points are most likely to appear. Figure 4 illustrates the process.

In addition to $\mathbb{P}_k$, we also need to take into consideration parts of the image I_k that have not been seen before. To this end, we maintain a keyframe ID j, accumulate the motion between image I_j and I_k and transform the four corners of the image I_j to I_k. When the overlap between the keyframe and the current image drops to a certain percentage, we extract interest points from the part of image I_k that lies outside of the shrunken quadrilateral. The keyframe ID is then updated to the current frame k.

3.2. Tracking Interest Points

Next we need to match interest points between images I_{k-1} and I_k. We first assume that if a feature $\mathbf{p}_{k-1}^j$ in I_{k-1} is still present in I_k, it can only be in region $\mathbf{P}_k^j$. When more than one interest point are detected in this region, we need to choose the one that best matches $\mathbf{p}_{k-1}^j$ *without computing their SURF descriptors*. Instead, we investigate two

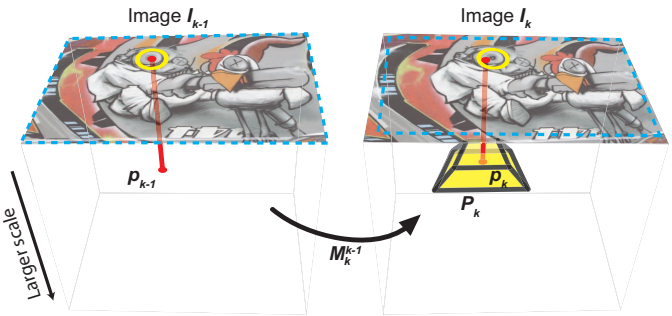

Figure 4. Incremental detection of an interest point. The predicted motion M_k^{k-1} is used to transform any interest point p_{k-1} from image I_{k-1} to image I_k. This predicted interest point p_k forms a volume neighborhood P_k in which the new interest point is extracted. The image corners are similarly transformed to the new image (blue dashed lines).

methods for computing lightweight signatures from the information that is already present in the scale space.

Local Curvature Method. In SURF, because the response function is an approximation to the determinant of the Hessian matrix, we can use the relationship between the two principal curvatures of each interest point as a signature of the interest point. Given the scale-normalized Hessian matrix in Equation 3, we compute its eigenvectors λ_1 and λ_2, $\lambda_1 > \lambda_2$, and measure the curvature ratio $r_1 = \lambda_1/\lambda_2$. This is directly related to the edge response detection method used in SIFT [12],

$$r_2 = \frac{trace(H)^2}{det(H)} = \frac{(r_1 + 1)^2}{r_1}. \tag{6}$$

Because the components of $\mathbf{H}$ are already calculated, computing ratio r_2 is more efficient than r_1. We treat the feature with the smallest difference in r_2 as the matching interest point, if this difference does not exceed a user-defined threshold Δ_{r_2}.

Normalized-Cross-Correlation (NCC). NCC is a classic technique for matching image regions and normally operates in the pixel intensity domain. With blob-type features like the ones used by SURF, this technique is not accurate because the image intensities surrounding the neighborhood of an interest point may not vary much. In addition, because the features are detected in the scale level, the neighborhood for NCC needs to be adjusted according to the scale of the interest points. Therefore NCC in the pixel intensity domain is *not* a suitable algorithm in terms of both performance and match quality. On the other hand, the values of the Hessian determinant around each interest point can be used as a more stable signature, since the values in the Hessian domain are independent of scale and relative brightness changes. This is best illustrated in Figure 5, where

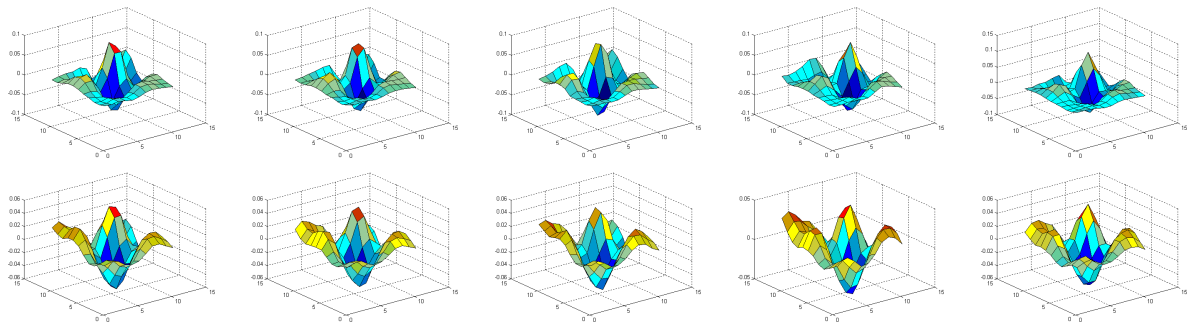

Figure 5. Approximate Hessian determinants. Each row represents the Hessian determinant values surrounding an interest point across five consecutive video frames. The patterns remain consistent and distinct, allowing easy and reliable matching.

we show values of Hessian determinants over consecutive video frames. The consistency and distinctiveness of the two patterns are evident.

To perform the NCC in the Hessian domain, for each possible pair of interest points, we construct a frustum around each interest point in domain $R(\cdot)$ corresponding to $5 \times 5 \times 3$ grid values, and compute the L2-norm between the two grids. This is much more efficient than a regular NCC because we only need to compute the dot-products at detected interest point locations. Similar to the local curvature method, we take the best matching interest point that passes a NCC threshold Δ_{NCC} as the matching interest point. Comparisons between these two tracking measures can be found in Section 5.

3.3. Motion Estimation

The interest point detection algorithm described in Equation 4 requires the estimation of relative motion. The estimation of motion M_k^{k-1} consists of the prediction and correction steps. The correction step is fairly straightforward—given the matching pairs, we use RANSAC to fit a fundamental matrix model and reject incorrect matches accordingly. The tracking process produces fairly accurate results and only a small number of iterations suffices to reject most of the false tracking results.

To compute the joint neighborhood $\mathbb{P}_k$ we need to predict the homography M_k^{k-1}. Note that the model computed in the correction step does not necessarily have to be the homography M_k^{k-1}; a more general model used in the correction stage allows more valid matches to go into the tracked interest point pool. In the simplest form, one can assume constant velocity motion and reuse the corrected motion, or assume no motion at all. If a valid model can not be found in the correction step, we do not have sufficient matches between images and in this case $\mathbb{P}_k$ falls back to the entire scale space.

3.4. Descriptor Computation

Each tracked feature descriptor is computed from the current list of tracked interest points like in the SURF algorithm. Because of the decoupling, we can choose not to compute any descriptors at all and use the SURFTrac algorithm only as a tracker. When descriptors are required, we ensure a smooth frame rate by putting the new interest points in a priority queue and computing their descriptors when the time budget allows. In addition, because the tracked points may out-live the robustness of the descriptors, especially because the interest points are not affine-invariant, we invalidate old descriptors and place them in the priority queue to be refreshed.

4. Real-Time Object Recognition and Tracking

In order to recognize and label objects in real-time, we need to compute and maintain the descriptors from the set of tracked features and query the image database. Querying the whole database can be slow especially as the size of the database grows. In order to speed up the process, we propose to organize the images in the database based on their spatial relationships, and query only subsets of the database that are more likely to contain matching objects. Figure 6 shows the overview of this process, and Figure 1 shows an example output from our system.

4.1. Image Graph Organization

We propose to organize the database images as follows. Given a collection of database images $\mathbb{V}$, we create an undirected graph $\mathbf{G} = (\mathbb{V}, \mathbb{E})$ where images form the nodes in the graph, and the edges $\mathbb{E} = \{\mathbb{E}_G \cup \mathbb{E}_{ID}\}$ describe the relationships between the images. An edge $e_g \in \mathbb{E}_G$ between two images indicates a geometric relationship when these two images can be related through standard pairwise image matching. In this case, a geometric relationship is simply the homography transform between these two images. Each image is also further identified with one or more object IDs, and two images sharing the same ID are also con-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.