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.

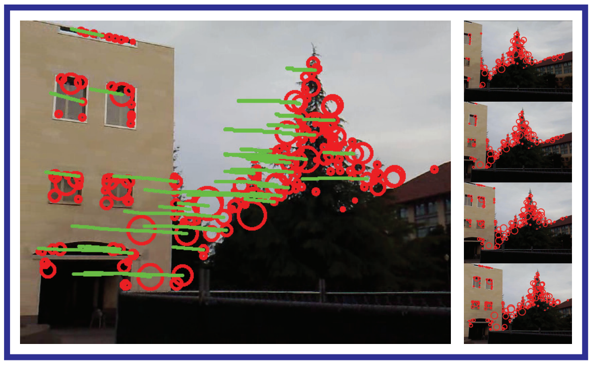
1. Introduction

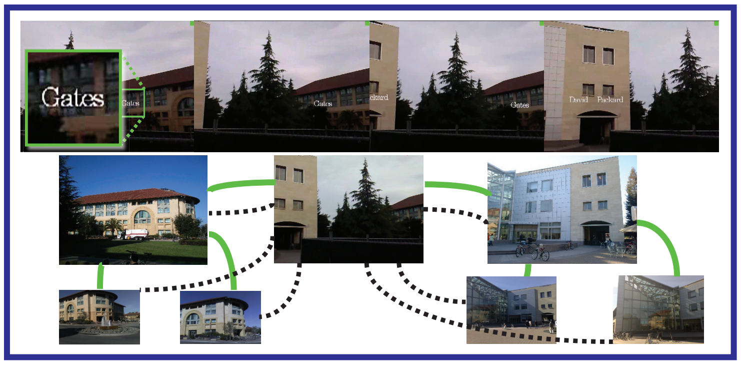
Robust feature descriptors such as SIFT [12], SURF [2], and GLOH [16] have become a core component in applications such as image recognition [8], multi-view stereo [25], and image registration [4, 26]. These descriptors are stable under viewpoint and lighting changes, so they are able to cope with significant 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-

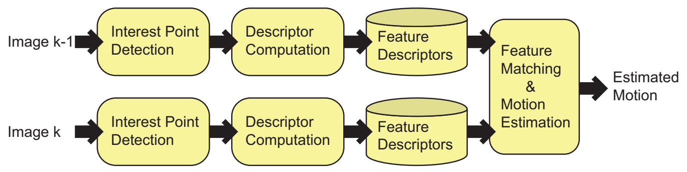


Figure 2. An algorithm using feature descriptor algorithms *as-is* for feature tracking.

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.

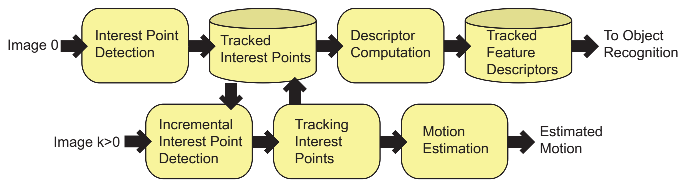


Figure 3. The SURFTrac algorithm overview. For the first image, the algorithm initializes a list of interest points by performing a full detection. The interest points are then updated and tracked upon receiving new images. The descriptors are for recognition purposes and the algorithm computes them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

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 $\mathbf{P}_k$

$$\mathbf{P}_k = \{(x'_k, y'_k, \sigma'_k) : \begin{array}{l} |\sigma'_k - \sigma_k| \leq \Delta_\sigma, \\ |x'_k - x_k| \leq \gamma\sigma'_k, \\ |y'_k - y_k| \leq \gamma\sigma'_k\}, \end{array} \tag{5}$$

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

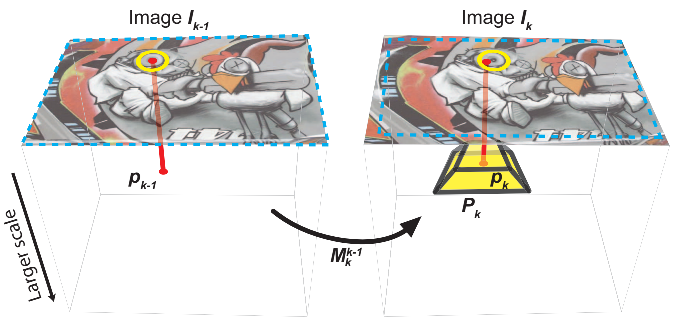


Figure 4. Incremental detection of an interest point. The predicted motion M_k^{k-1} is used to transform any interest point p_{k-1} from image I_{k-1} to image I_k. This predicted interest point p_k forms a volume neighborhood P_k in which the new interest point is extracted. The image corners are similarly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

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

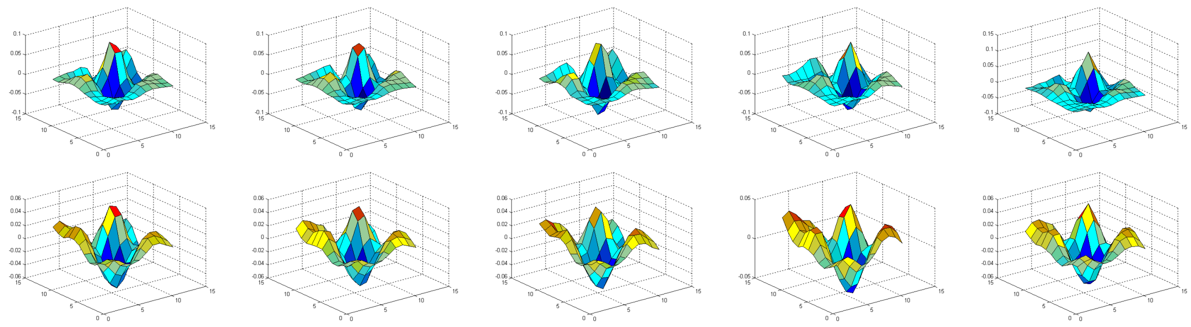


Figure 5. Approximate Hessian determinants. Each row represents the Hessian determinant values surrounding an interest point across five consecutive video frames. The patterns remain consistent and distinct, allowing easy and reliable matching.

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-

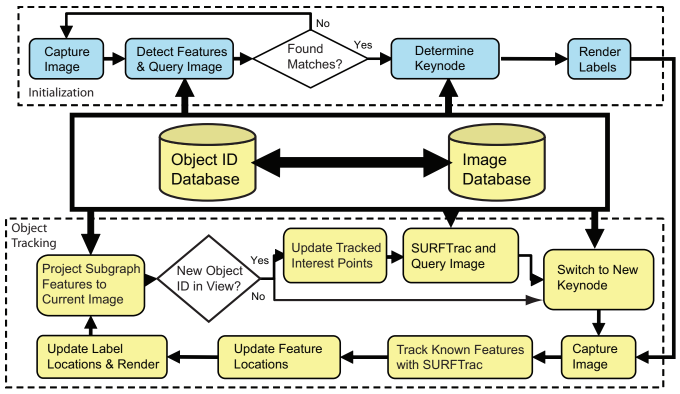


Figure 6. Real-time tracking and recognition pipeline.

nected by an additional edge $e_{id} \in \mathbb{E}_{ID}$. This organization is similar to Simon *et al.* [21] where a graph of images is constructed for hierarchical browsing purposes. It is particularly suitable for our purposes where the images are taken from real-world environments such as streets and buildings. An example of the image graph is shown in Figure 1(b).

4.2. Initialization

During initialization, we compute the full SURF feature descriptors from the first video image and match them against images in $\mathbf{G}$ using a method similar to Takacs *et al.* [27]. This method constructs an approximate nearest neighbor tree for all the image features in the database followed by geometric verification (RANSAC). Upon successfully identifying the matching images, the best image $v_k \in \mathbb{V}$ is marked as the current *keynode*, and the set of images in-play is reduced to only those images that are connected to v_k by a path in $\mathbf{G}$, as shown in Figure 1(b). Once a keynode image and its object ID are identified, we can continuously match and update the keynode at a relatively low cost since we are fairly confident all potentially relevant objects are included in the current database subgraph.

The next task involves computing the placement of the labels. In this step, we group all the matching features according to their respective object IDs, and render one object ID label at the geometric centroid of each feature group. The locations of the labels remain static with respect to the feature group until a new keynode image is chosen.

4.3. Object Tracking

At every new video frame, we run SURFTrac to update the interest points, compute the homography against the previous video frame, and update the label location accordingly. Because SURFTrac continues to run in every input video frame, we only need to run the object recognition algorithm occasionally for newly revealed objects. In order to avoid running the matching algorithm frequently, we reproject all features in $\mathbb{V}_k$ to the current video frame, and if features corresponding to new object IDs fall onto the video frame, we add these new features to the list of tracked interest points, and run SURFTrac algorithm again before querying into $\mathbf{G}_k$.

5. Results and Discussions

We now move on to discuss the performance and attributes of our implementation of SURFTrac algorithm. In the following experiments, unless otherwise noted, we predicted the motion M_k^{k-1} between consecutive frames using constant velocity assumption, and rely on the SURFTrac local neighborhood to locate the updated interest points. We also chose to set Δ_{NCC} and Δ_{r2} to infinity, which means we always select one best matching feature for each local neighborhood. We implement our methods on top of the SURF algorithm in [6] to obtain fair performance comparisons between these two algorithms.

5.1. Interest Point Tracking

To evaluate the incremental interest point detection algorithm, we first produce a synthetic image sequence by rendering a rectangular texture in 3D and varying the virtual camera parameters to simulate the pure translation and rotation of the real camera. Using a synthetic sequence allows us to measure the impact of each individual extrinsic parameter independently. We use either the NCC or the Local Curvature method to track the interest points. Then, we apply homography RANSAC to filter out the false-positive matches. We measure the accuracy of the tracking method as a percentage of the matches that passed RANSAC. The accuracy under different camera movements are shown in Figure 7 where the camera moves in x direction with a constant velocity. For the local SURFTrac neighborhood, we always use $\Delta_\sigma = 1$ and $\gamma = 6$, which for the base level corresponds to 16×16 pixels. With an accuracy up to 90%, the NCC method proved to be much better than the Local Curvature method.

Figure 8 shows how the accuracy changes according to the amount of camera movement around different axes. SURFTrac appears to cope with most of the camera movements quite well within reasonable range of motion. However, we can see that the amount of change in roll severely affects the SURFTrac accuracy in both NCC and Local Curvature methods.

5.2. Mobile Phone Performance

We implemented SURFTrac on a Nokia N95 mobile phone to analyze and compare the speed of SURFTrac against regular SURF in both interest point detection and tracking using a frame-by-frame matching method similar to Skrypnyk and Lowe [24] and Battiato *et al.* [1], as described in Figure 2. The N95 contains an ARM-11 based

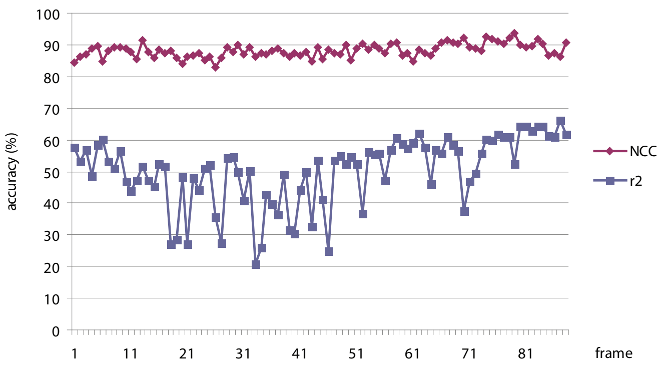


Figure 7. Accuracy comparison between NCC and Local Curvature (r2) matching methods.

Texas Instrument OMAP2 processor running at 330MHz. Our test results use 256×192 live video from the N95 camera. The local neighborhood $\mathbf{P}_k$ is chosen experimentally to balance the tradeoff between the computational cost and the tracking consistency, and varied from $12 \times 12 \times 3$ in the lowest scale to $32 \times 32 \times 3$ in the highest scale.

Table 1 summarizes the average processing time per frame in different methods. Tracking features with the SURF algorithm takes 0.678 seconds even with fairly optimized code. Detecting interest points alone with SURF takes 0.357 seconds, which is over $3\times$ slower than SURF-Trac even though the latter provides additional tracking results. Overall for tracking purposes, we achieve over $5\times$ speedup compared to SURF.

Without geometric verification, the Local Curvature method is slightly faster than the NCC method. However, when geometric verification is enabled, both methods are equally efficient because in the case of Local Curvature method, RANSAC requires more iterations to converge due to its much larger false-positive ratio.

5.3. Outdoor tracking

Figure 1 shows the outdoor tracking and label augmentation results on the N95 phone. We use a small set of keynodes in the database with about three thousand features matched across all keynodes. We use NCC for this purpose because it generally outperforms the Local Curvature method. We observe that, in this application context, the user tends to hold and move the mobile device parallel to the ground plane, and therefore the amount of camera roll is fairly insignificant. Our implementation reached 1.4 FPS during initialization or recovery when the full-frame feature detection and querying with database is needed. After the first keynode is matched, the tracking and subgraph matching runs at about 6-7 FPS, including capturing and rendering of the image and label overlay.

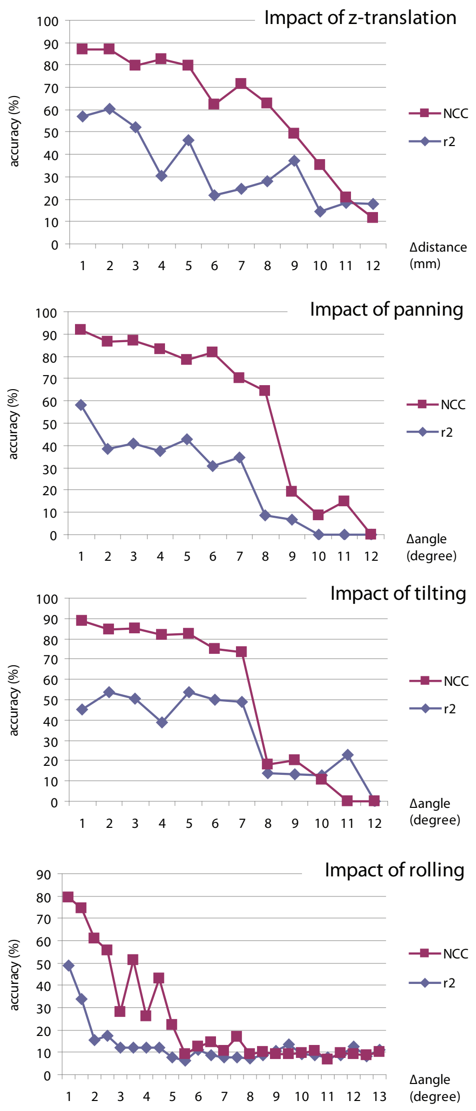


Figure 8. Impact of change in each camera parameter on neighborhood matching results of NCC and LC (r2).

5.4. Limitations

Our tracking method inherits similar limitations as the traditional active search strategy which depends very much on the search region's size, the camera motion and the distribution of features in the scene. In the outdoor environment, the tracker fails when, for example, the camera pans to an empty space between two buildings. In addition, because the interest points are localized and interpolated quadratically in the scale space, their positional accuracies do not rival traditional corner detectors where sub-pixel accuracies are commonly achieved.

In terms of label placement, because we have multiple keynodes per building, a label may jitter when a new keynode is selected. A label can also drift after a while, but its

Methods		Time (sec)
SURF	Matching and Tracking	0.678
	Interest Point only	0.357
SURFTrac	NCC only	0.115
	NCC + RANSAC	0.133
	Local Curvature only	0.111
	Local Curvature + RANSAC	0.134

Table 1. Speed comparison of SURFTrac algorithm versus SURF algorithm. For tracking purposes we achieve over $5\times$ speedup compared to SURF.

position will be recovered appropriately when the tracker switches to a new keynode from the database.

6. Conclusion

We presented the SURFTrac algorithm, an efficient method for tracking scale-invariant interest points without computing their descriptors. We also demonstrate a framework for outdoor tracking using SURFTrac and achieve near real-time performance on mobile phones while tracking and recognizing the scene objects at the same time.

Our framework has large potential for improvement for outdoor mobile augmented reality applications. We would like to investigate better methods for tracker initialization and recovery, minimize speed disruption when the subgraph querying strategy fails, and experiment with more sophisticated motion estimation methods.

Acknowledgments: The authors would like to thank Ming-Hsuan Yang, Jana Kosecka, and Radek Grzeszczuk for their useful comments and suggestions.

References

[1] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato. Sift features tracking for video stabilization. In *Proc. of International Conference on Image Analysis and Processing (ICIAP)*, 2007.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *ECCV (1)*, pages 404–417, 2006.

[3] M. Brown and D. Lowe. Invariant features from interest point groups. In *British Machine Vision Conference*, pages 656–665, 2002.

[4] M. Brown and D. Lowe. Automatic Panoramic Image Stitching Using Invariant Features. In *International Journal of Computer Vision*, volume 74, pages 59–77, 2007.

[5] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *2nd International Symposium on Visual Computing*, November 2006.

[6] W.-C. Chen, Y. Xiong, J. Gao, N. Gelfand, and R. Grzeszczuk. Efficient Extraction of Robust Image Features on Mobile Devices. In *ISMAR '07: Proc. of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[7] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Asian Conference on Computer Vision*, pages 918–927. Springer, 2006.

[8] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *Proc. of ICCV*, pages 1458–1465, 2005.

[9] C. Harris and M. Stephens. A combined corner and edge detection. In *Proc. of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[10] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[11] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT Flow: Dense correspondence across different scenes. In *Proc. of the 10th European Conference on Comput. Vision*, Oct. 2008.

[12] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[13] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[14] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[15] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision*, 60(1):63–86, 2004.

[16] K. Mikolajczyk and C. Schmid. Performance Evaluation of Local Descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[17] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[18] D. Nistér and H. Stewénius. Scalable Recognition with a Vocabulary Tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2006.

[19] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, October 2005.

[20] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[21] I. Simon, N. Snavely, and S. M. Seitz. Scene Summarization for Online Image Collections. In *Proc. of ICCV*, 2007.

[22] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. In *EDGE 2006, workshop on Edge Computing Using New Commodity Architecture*, May 2006.

[23] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. of ICCV*, volume 2, pages 1470–1477, Oct. 2003.

[24] I. Skrypnyk and D. G. Lowe. Scene Modelling, Recognition and Tracking with Invariant Image Features. In *Proc. of International Symposium on Mixed and Augmented Reality (ISMAR)*, 2004.

[25] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *SIGGRAPH Conference Proceedings*, pages 835–846, 2006.

[26] R. Szeliski. Image alignment and stitching: a tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006.

[27] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceeding of ACM international conference on Multimedia Information Retrieval*, pages 427–434, 2008.

[28] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Vision Algorithms: Theory and Practice, number 1883 in LNCS*, pages 278–295. Springer-Verlag, 1999.

[29] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proc. ISMAR*, Cambridge, UK, 2008.

[30] Z. Zhang, R. Deriche, O. Faugeras, and Q. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.