Evaluation, Design and Application of Object Tracking Technologies for Vehicular Technology Applications

Che-Tsung Lin and Long-Tai Chen
Mechanical and Systems Research Laboratories
Industrial Technology Research institute
Chutung, Hsinchu, Taiwan 31040, R.O.C
{alexlin,ltchen}@itri.org.tw

Yuan-Fang Wang
Department of Computer Science
University of California
Santa Barbara, CA 93106
yfwang@cs.ucsb.edu

Abstract— Advances in video technology have enabled its wide adoption in the auto industry. Today, many vehicles are equipped with backup, front-looking, and side-looking cameras that allow the driver to easily monitor the traffic around the vehicle for enhanced safety. This paper reports our research on evaluating many existing object tracking techniques, and proposing a new tracker design and its application for 3D environmental mapping in vehicular technology applications. The contribution of our research is 4-fold: (1) We evaluate a large collection of state-of-the-art trackers using multiple criteria relevant to vehicular technology applications, (2) We show how to derive useful evaluation metrics from public-domain, real-world driving videos that do not come with ground-truth information on pixel tracking, (3) we propose a new tracker that is geared specifically for vehicular technology application and show that it achieves tracking accuracy that outperforms SIFT and is on-par with the state-of-the-art optical-flow tracking algorithm, which has the best accuracy in our evaluation. Furthermore, we show that our tracker is 600 times more efficient than optical flow and 7 times more efficient than SIFT, and (4) we validated our new tracker design for 3D environmental map building application and showed that the new tracker can obtain comparable results as SIFT but at a significant saving in runtime.

I. INTRODUCTION

Video cameras are becoming ubiquitous in the modern societies. They are increasingly being adopted by the auto industry for its falling price and improving capabilities.

This paper reports our research on using a vehicle's onboard video data for vehicular technology applications. More specifically, this paper is about evaluating existing object tracking techniques, and propose a new tracker design and its application for 3D environmental map building during driving. The contribution of our research is 4-fold: (1) We evaluate a large collection of state-of-the-art trackers, taking into consideration multiple criteria relevant to vehicular applications, (2) We show how to derive useful evaluation metrics from public-domain, real-world driving videos that do not come with ground-truth information to validate pixel tracking, (3) We propose a new tracker that is geared specifically for vehicular technology application and show that it achieves tracking accuracy that outperforms SIFT and is on-par with the state-of-the-art optical-flow tracking algorithm, which has the best accuracy in our evaluation. Furthermore, we show that our tracker is 600 times more efficient than optical flow and 7 times more efficient than SIFT, and (4) we validated our new tracker design for 3D environmental map building application and showed that the new tracker can obtain similar results as SIFT but at a significant saving in runtime.

The particular application of 3D environmental map building deserves some discussion. While Google's Self-Driving Car [1] has many sensors to enable building such an environmental map (e.g., GPS, gyroscope, shaft encoder, and 3D ladar), these sensors are often expensive and bulky; requiring complex circuity to link together and on-board computing and recording devices to function. In particular, 3D data come from 3D laser radar that can be many times more expensive than the host vehicle itself. *In contrast, our 3D map building pipeline has relied on a single, front-looking video camera, but no other sensor to function*. That is, we address the most difficult and data-deprived scenario. We demonstrate that the new tracker design can make video-based 3D environmental map building much more efficient.

II. TECHNICAL RATIONALE

Traditionally, object tracking can be accomplished by either a flow-based analysis or a feature-based analysis [2]. A flow-based approach may provide a dense and regular (i.e., defined on a pixel grid) motion field. However, (1) bland image regions lacking surface texture present a significant challenge for robust motion estimate everywhere [6], (2) it is arguable if a dense and regular motion field is required for vehicular technology applications. For example, if the goal is to track nearby objects for collision avoidance, the focus could be placed on regions in the center of the image and those exhibiting temporal changes, but not every pixel, and (3) while most, if not all, flow estimation algorithms were formulated based on the brightness constancy constraint [2,7], it is highly nontrivial to make such a framework robust and practical [6]. The formulation requires expensive hierarchical processing to handle large object displacement. To ensure robustness, many formulations were proposed that end up solving a very large non-linear optimization problem that can be time consuming and algorithmically complex as to prevent their adaptation onto a vehicle platform. For example, in a recent (2011) paper [6] over 20 flow-based algorithms were evaluated on some standard test data sets. The runtime on a short, 8-frame sequence ranged from a few minutes to a few hours, which makes them unsuitable for a real-time application.

Instead, feature-based tracking does not usually produce a regular flow field – as the feature detection mechanism is

designed to report only on image patches with significant local intensity gradients. Such a method can produce a fairly dense flow filed for textured regions though, if certain detection sensitivity setting is properly adjusted. Furthermore, a large variety of feature detectors and descriptors have been developed in the recent past with varying complexity and efficiency that can potentially be employed here. We have evaluated a fairly comprehensive collection of modern-day feature detectors and descriptors for vehicular technology applications, including well-known designs such as SIFT [8], SURF [9], USURF [9], FAST [10], and BRIEF [11]. The selection intentionally includes some sophisticated designs, such as SIFT and SURF, which employ clever mechanisms to guard against incidental environmental changes in lighting, scale, and pose, but can be slow, and some simpler ones, such as FAST and BRIEF, which are without extensive safeguard but are easy to implement and computationally efficient.

While the performance of SIFT and SURF has been studied before [12], the focus was on accuracy and correctness, not efficiency and complexity. Even when special implementation, e.g., using GPU, can speed up SIFT [13] and SURF [14], such special hardware is often unavailable on a vehicle's onboard computing system. It can be argued that many factors, such as accuracy, efficiency, complexity, all play a role for a vehicular application. It is therefore important to understand the trade-off among these factors for the particular application scenario.

One important point to note is that a complete feature-based tracker must encompass two components: a detection mechanism and a description mechanism. The detection stage answers the question of "*where* it is" and the description stage answers the question of "*what* it is." SIFT and SURF comprise both a detector and a descriptor. On the other hand, FAST is, strictly speaking, a detection mechanism while BRIEF a description mechanism only. Certain enhancement need be employed to make them a complete framework for tracking.

FAST corner detector uses a circle of 16 pixels (radius 3), around a point *p* to classify if *p* is actually a corner. Each pixel on the circle is given an integer label from 1 to 16 clockwise starting from the pixel at the 12 o'clock position. If a set of *n* contiguous pixels on the circle are all brighter than the center pixel *p* plus a threshold value *t* or are all darker than pixel *p* minus the threshold value *t*, then *p* is classified as a corner.

As the basic FAST formulation comprises only a detection mechanism, we have augmented it with a description scheme. The FAST descriptor follows the spirit of SIFT and SURF where local histograms of gradients (HOG) are used for describing the surface pattern. We use a neighborhood of size 16x16, which is partitioned into 2x2 non-overlapping blocks of size 8x8 each. A gradient histogram is computed for each 8x8 block with the gradient directions quantized into four bins. Each pixel in the block contributes its gradient direction, properly weighed by its distance to the center of the block and the strength of its gradient response. Concatenating these gradient histograms produces a descriptor of length 2x2x4=16. Comparing with SURF's descriptor of length 64 and SIFT's length 128, the FAST descriptor is significantly shorter. The tradeoff is then obviously in efficiency over sophistication.

BRIEF is a very simple descriptor, and its premise is that image patches can be effectively classified on the basis of a relatively small number of pairwise intensity comparisons. The feature vector is a bit vector out of the test responses. For a local image patch of size *S* by *S*, a test is performed as

$$\tau(p,x,y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & otherwise \end{cases} \quad (1.)$$

where *p(x)* is the pixel intensity in a smoothed version of *p* at pixel $x = (u, v)^T$. Choosing a set of *n* pixel location pairs uniquely defines a set of binary tests. BRIEF descriptor is then defined to be the *n*-dimensional bit string

$$f_n(p) = \sum_{1 \le i \le n} 2^{i-1} \tau(p, x_i, y_i) \quad (2.)$$

BRIEF can be used to describe any image patch, or it is a descriptor but not a detector. We have augmented the BRIEF descriptor with a detector (FAST) to make a complete combo. Again, the choice is aimed at deriving an efficient mechanism for feature tracking. Hence, a goal of the evaluation is to see if an efficient detector and descriptor combo can match the performance of SIFT and SURF but at a much reduced computation load to enable efficient tracking onboard a vehicle.

A. Evaluation Metrics from Real-World Data with no Ground-Truth Pixel Movement Information

To make the evaluation relevant, we used videos collected from real-world driving conditions. One such public-domain data set is KITTI Vision Benchmark Suite [1,4,5]. KITTI Suite contains many video sequences of driving in city streets, residential neighborhoods, rural areas, and highways. KITTI Suite does provide ground-truthed optical flow data, but only for small number of image pairs. Furthermore, different vendors will want to evaluate trackers on their own data sets, collected using their own onboard cameras. To enable a more general evaluation using KITTI and other video data sets, we observe that in between two successive video frames separated by less than, say, a few seconds during driving, the movement of the vehicle can often be reasonably approximated as a simple translation along the *z* (road) direction. If the scene is static, such a *z* translation induces an image flow field that exhibits a 2D zoom pattern. This can be easily verified by considering a 3D point (*X,Y,Z*) and its location ($X,Y,Z+\Delta Z$) sometime (Δt) later, where $\Delta Z = V\Delta t$, and *V* is the speed of the vehicle. Projecting the 3D point onto the image plane using the pinhole model [2,21], we arrive at (*f* is the focal length):

$$x' = f\frac{X}{Z+\Delta Z} = f\frac{X}{Z\left(1+\frac{\Delta Z}{Z}\right)} \approx f\frac{X}{Z}\left(1-\frac{\Delta Z}{Z}\right) = x - \frac{\Delta Z}{Z}x$$

$$y' = f\frac{Y}{Z+\Delta Z} = f\frac{Y}{Z\left(1+\frac{\Delta Z}{Z}\right)} \approx f\frac{Y}{Z}\left(1-\frac{\Delta Z}{Z}\right) = y - \frac{\Delta Z}{Z}y$$

$$v_x = x' - x = -\frac{\Delta Z}{Z}x$$
$$v_y = y' - y = -\frac{\Delta Z}{Z}y \quad (3.)$$

This zoom model applies to both front-mounted and rear-mounted cameras. While the magnitude of the flow vector cannot be ascertained without knowing the speed of the vehicle ($V=\Delta Z/\Delta t$) and the distance to the road object (*Z*), the direction

of the flow vector is nonetheless completely determined by the pixel location, or

$$\theta = \tan^{-1}(v_y/v_x) = \tan^{-1}(y/x) \qquad (4.)$$

This observation then allows us to use for evaluation a large number of video frames from KITTI Suite (and other datasets as well) that depict a static scene, as we can examine the directional error in the flow field even without ground-truth. The evaluation results will be presented in the next section, but the conclusion is that simple trackers such as BRIEF and FAST can perform comparably well as the more elaborate trackers such as SIFT, SURF and optical flow in these applications, but at a fraction of the costs.

B. A New Tracker Design for Vehicular Safety

Note that FAST detector and BRIEF descriptor have little protection against incidental scene changes. A generalization of the BRIEF descriptor to tolerate in-plane rotation was proposed recently and was shown to achieve comparable results with SIFT in certain applications [19]. The idea is to record the pair test results multiple times, each time on a patch that is a rotated version of the original patch. By incorporating multiple descriptors for different amounts of rotation, the BRIEF scheme becomes more robust in matching.

For vehicular safety application, we do not foresee in-plane rotation as a common motion pattern. As argued before, 2D zoom is prevalent. Hence, we propose here a variation of the BRIEF descriptor, or ZBRIEF, that allows robust matching with image zoom. To maintain the simplicity and efficiency of BRIEF, ZBRIEF computes and records multiple BRIEF descriptors at different zoom scales. More specifically, a test at a zoom scale (z) is represent as:

$$\tau_z(p,x,y) = \begin{cases} 1 & p(zx) < p(zy) \\ 0 & otherwise \end{cases} \qquad (5.)$$

where $zx = (zu, zv)^T$. ZBRIEF descriptor is then defined to be the $|z|$ by n-dimensional bit string ($|z|$ denotes the number of zoom levels and n the length of the BRIEF bit vector)

$$f_{zn}(p) = \sum_{1\le j\le |z|} 2^{(j-1)n} \sum_{1\le i\le n} 2^{i-1}\tau_j(p, x_i, y_i) \qquad (6.)$$

Again, the evaluation results of ZBRIEF will be presented in the next section. We summarize here that ZBRIEF is almost as efficient as BRIEF and FAST but performs better than SIFT, and close to the best optical-flow scheme we tested.

C. 3D Environment Building Using a Single Forward-Looking Camera with ZBRIEF

As mentioned before, 3D environmental mapping is an important advance to enhance 2D maps that are widely available these days. However, projects such as Google's Self-Driving Car [1] gather 3D data from 3D laser radar that can be many times more expensive than the host vehicle itself. *In contrast, our 3D map building pipeline, which we called PhotoModel3D, has relied on a single, front-looking video camera, but no other sensor to function.*

PhotoModel3D employs a photo- and video-based analysis paradigm known as structure from motion (SfM) [2,21]. The principles are to exploit the motion parallax effect exhibited in multiple images taken by a moving camera to infer the 3D scene structures and the camera poses. PhotoModel3D (1) works with both discrete images and continuous videos taken by a consumer-market digital camera, camcorder, or camera phone, (2) uses no special equipment (e.g., lens and tripod), active projection, artificial lighting, prior camera calibration, and man-made registration markers, (3) functions both indoor and outdoor and over long range, unlike many commercial RGBD devices that are mostly for indoor, short-range applications, (4) requires no user training (just point and shoot), (5) is fully automated and end-to-end (from photographs to fully colored and textured 3D models) without manual intervention or data-specific parameter tuning, (6) is a software-based solution that runs on commodity Linux and Windows servers without the need of special hardware (GPU, DSP, etc.) acceleration.

A video-based 3D pipeline like PhotoModel3D often involves detecting and matching features across the video frames. These detection and correspondence processes must be carried out with care as they form the basis of all ensuing analysis. Our experience has been that the feature analysis process can take anywhere from 15% to 40% of the total processing time in SfM. Exploiting a fast feature tracker like ZBRIEF can be advantageous in reducing computational power on a mobile device and allowing appropriate client-server partition of the 3D mapping pipeline.

III. EXPERIMENTAL RESULTS

To evaluate the feature detector and descriptor for tracking, we manually selected from KITTI Suite 10 sequences, each with 10 frames, where the vehicle was advancing steadily (no turning) with no independently moving vehicles or pedestrians on the road. For these 100 frames, we reasonably expected that feature-based tracking should associate pixels in adjacent frames in such a way that the resulting flow vectors obey the zoom model in Eq. 3. We then compared the runtime and accuracy of various feature detectors and descriptors and identified the best parametric setting for them.

As SIFT, SURF and USURF are well-tuned algorithms, we used the publicly available implementations without modification [15,16,17]. The drawback for real-time use is their complexity and inefficiency, which is difficult to remedy without special hardware acceleration (e.g., GPU) that is often lacking onboard a vehicle.

BRIEF: There are just a few parameters in BRIEF that need be specified to complete the framework; namely, the smoothing filter used, the patch size (S), and the length of the bit string (n). While [11] suggested some reasonable default values for these parameters, we have also conducted our own experiments to determine the proper values for vehicular safety applications, which turned out to be somewhat different from those suggested in [11].

We have experimented with three levels of smoothing: (1) no smoothing, (2) Gaussian smoothing with a filter of size 5x5 and σ=1, and (3) Gaussian smoothing with a filter size of 9x9 and σ=2. Of these, (3) was the setting recommended by [11]. We have tried patch size from 21x21, 25x25, and all the way to 55x55 with the bit-string length at 128, 256 and 512, as recommended by [11].

Two more parameters: a matching neighborhood size and a matching threshold, were used for improved efficiency and

robustness. The same parameter values were used for all trackers so as not to disadvantage any scheme unfairly. In matching features detected in two frames, we used a matching neighborhood size of 50, i.e., a feature in one frame was matched only to a feature in the other frame that was less than 50 pixels apart. This was a reasonable domain constraint for driving videos and was realized by building a KD tree of the detected features using the ANN package [18] for efficient nearest neighbor search.

However, within a 50-pixel radius to a feature there could exist many candidate features for pairing in the other frame. Instead of matching a feature to whichever feature in the other frame that came close and had the most similar descriptor, we insisted that the best match had to also "beat out" other potential matches significantly, or the pairing had to be un-ambiguous. This constraint was captured by the matching threshold. We have used a matching threshold of 0.7 consistently for all schemes, which implied that the Euclidean distance between the best pairing candidate must be better than that of the second best candidate even when the second best was shrunk 70%.

For these parameter combinations, we ran the BRIEF scheme and recorded the runtime, pairing density (i.e., how many detected features were matched) and the alignment error (in degree) with respect to that predicted by the 2D zoom flow. Due to the page limit, we only state our finding here without presenting the data:

- The run time was proportional to both the detection and matching density, but it did not vary significantly for different patch sizes or feature lengths. Unlike the parameter suggested by [11], we found 5x5 Gaussian kernel with σ=1 (instead of 9x9 Gaussian kernel with σ=2) seemed to achieve the best balance of a reasonable runtime and high feature density. A 9x9 Gaussian kernel with σ=2 tended to over-smooth the images and caused the detected feature density to drop precipitously.
- Both pairing density and direction error were influenced by the patch size. The pairing density peaked around 25x25 to 31x31 for all levels of smoothing, while at the same time the direction error dipped to the lowest point. This observation implied that both small and large patches performed worse than patches of a median size.
- Feature lengths of 256 and 512 were both better than 128. This was again in terms of both pairing density and direction error. Performance difference between feature lengths of 256 and 512 was quite small.

Based on these observations, we narrowed our choices to the following two configurations: (1) smooth the input images using a Gaussian kernel of size 5x5 and σ=1, and (2) use either a patch size of 31x31 and a feature length of 256 (brief_31_256) or a patch size of 25x25 and a feature length of 512 (brief_25_512).

ZBRIEF: We conducted experiments to see if adding the zoom dimension improved the tracking performance. Note that we have chosen $z \geq 1$ (i.e., zoom in only) for ZBRIREF. If the camera is front-mounted, we expect that stationary road objects to move closer (zoom in) over time. Hence, the pairing operation compares the feature at scale z=1 (no zoom) in a frame with features at all recorded zoom levels ($z \geq 1$) in a later frame. For a rear-mounted camera, we expect that stationary road objects to move further away (zoom out) over time. Hence, the pairing operation compares the feature at scale z=1 (no zoom) in a frame with features at all recorded zoom levels ($z \geq 1$) in a previous frame. Hence, only features of $z \geq 1$ need be considered.

We tested on the best two configurations (brief_31_256 and brief_25_512) obtained from the previous experiments and used zoom levels from 0% (z=1) all the way to 30% (z=1.3), in 5% increment. Our observation was that the runtime increased as more zoom levels were used, and this was to be expected with a longer bit string. However, the matching density increased and direction error dropped by using zoom. The best zoom level seemed to at 20%. Based on the analysis, we have chosen the final tracker configuration to be: *(1)* smooth the input images using a Gaussian kernel of size 5x5 and σ=1, and (2) use a patch size of 31x31 and a bit string length of 256 (brief_31_256).

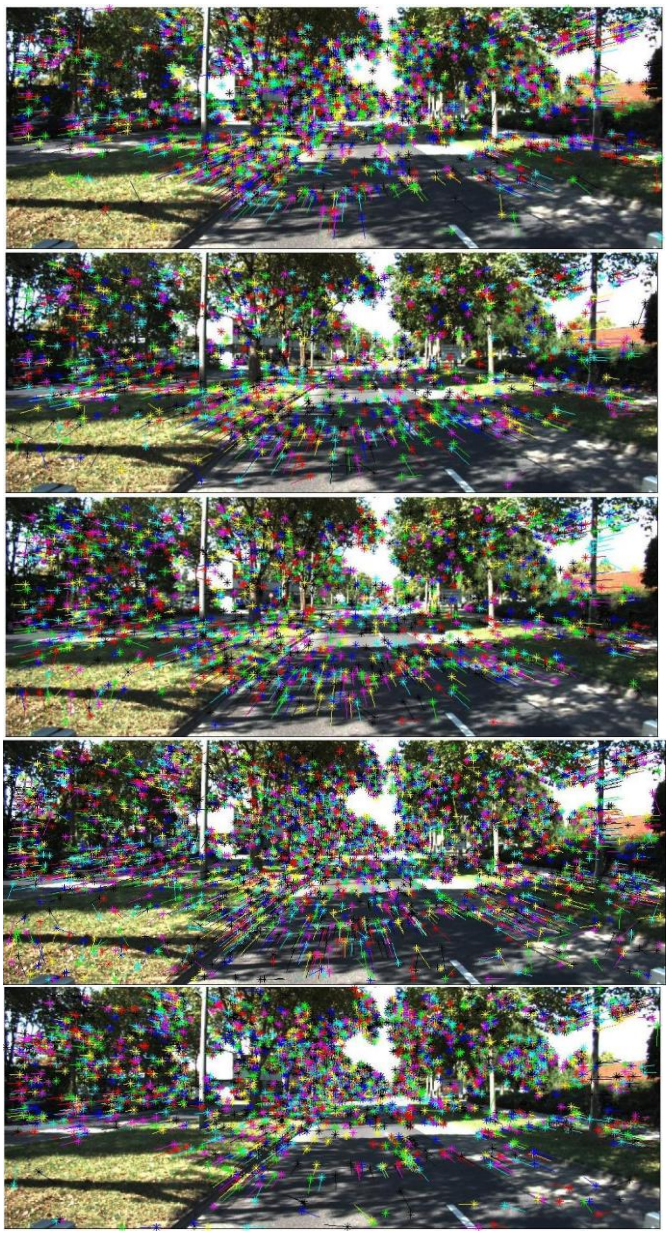
Comparison with Other Feature-based Methods: While ZBRIEF improved on BRIEF, we need to compare our best design - ZBRIEF with five zoom levels up to 20%, on a 31x31 patch with a bit string of length 256 - against SIFT, SURF, USURF, and FAST. The results are shown in Table 1. SIFT, SURF and USURF had built-in mechanism for image smoothing, and hence, they were run on the original images. FAST and ZBRIEF used images smoothed with a Gaussian kernel of 5x5 and σ=1. We tried to keep the feature detection density roughly the same, subject to what the public-domain codes allowed [15,16,17].

As can be seen, ZBRIEF had the highest pairing efficiency, in that it converted the highest percentage of detected features (37%) into pairs. ZBRIEF was the second fastest, about 40% slower than FAST, but was twice as fast as USURF, four times as fast as SURF, and seven times faster than SIFT. Furthermore, ZBRIEF achieved the said speed-up while processing 58% more featuresthan SURF and USURF and generating significantly more feature correspondences than SIFT (30% more), SURF (105% more) and USURF (80% more). FAST, while efficient, was the least accurate, probably due to its short descriptor.

What was surprising was that ZBRIEF outperformed SIFT. As seen from *Table 1***,** SIFT and ZBRIEF detected roughly the same number of features, but ZBRIEF was able to pair 30% more of them seven times faster and with a lower directional error. The good performance might be attributed to ZBRIEF's design focusing on translation and zoom motion that matched well with the driving video characteristics. While SURF and USURF had smallest directional errors, it might be because they had the lowest detection density (63% lower than ZBRIEF and SIFT) that allowed them to focus on fewer, high-quality features (the public-domain program [9,17] did not allow the feature density to increase beyond what we used). A sample detection/matching result is shown in *Figure 1*. While no single design outperformed all others in all aspects, ZBRIEF appears to be a good compromise in terms of speed and accuracy for vehicular technology applications.

Table 1 Comparison of best ZBRIEF against other feature detectors/descriptors

	detection density	Run time	pairing density	direction error	pairing efficiency
SIFT	6712	6:08	1940	9.42	29%
SURF	4271	3:31	1221	7.73	29%
USURF	4271	1:55	1396	7.51	32%
FAST	6523	0:38	1999	10.45	30%
ZBRIEF	6776	0:52	2505	8.47	37%



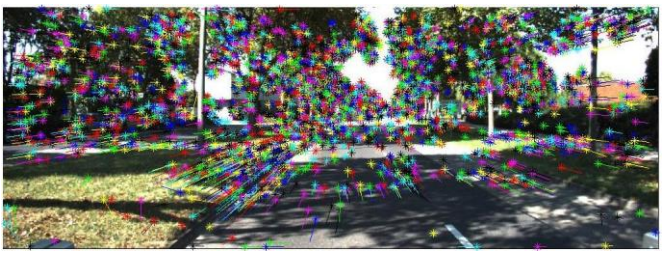


Figure 1 From top to bottom: SIFT, SURF, USURF, FAST, BRIEF, ZBRIEF

Comparison with Flow-based Methods: While we do not advocate a flow-based approach, as a reality check we also tested a state-of-the-art flow-based method [20] on the test images. We used the codes provided by [20] without changes. The average direction error over 90 pairs in KITTI data set came out to be 6.21 degrees, about 2 degrees better than ZBRIEF. Unfortunately, the codes were considerably slower. We estimated that the runtime of the optical-flow method would be about 600 times slower than ZBRIEF. A graphic example using the same pair of images as in *Figure 1* is shown in *Figure 2* (Seq3 in Error! Reference source not found.). To avoid cluttering the display, only one flow vector in a 30x30 neighborhood is shown. As can be seen that the flow field is indeed dense and regular.

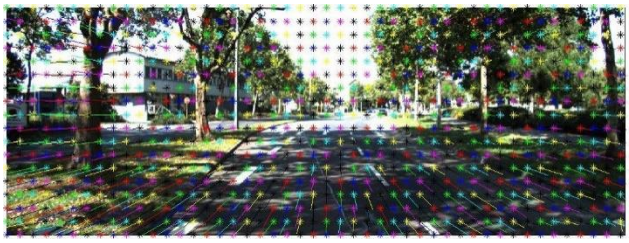


Figure 2 Optical-flow-based motion estimation example

While results presented in Table 1 look promising, it doesn't tell the whole story. A more convincing illustration is to replace SIFT and SURF with ZBRIEF in our 3D pipeline, while keeping all other components the same, and then compare the resulting 3D models. Two such examples are shown in Figure 3. The left column shows a sample image in the video sequence of length 35 (top) and 45 (bottom), respectively. The middle column shows the 3D models based on SIFT feature analysis while the right column the corresponding 3D models using ZBRIEF analysis. Note that these models differ only in the 2D feature used; all other processing components, parameter settings, and input photos are exactly the same.

IV. Concluding Remarks

This paper summarizes our research on feature tracking for vehicular technology applications. We have conducted an extensive comparison of the state-of-the-art trackers, with an eye to design an efficient tracker performing on par with sophisticated feature-based trackers like SIFT and SURF and the best optical-flow-based trackers. We have realized such a design and application with a comparable performance with the

state-of-the-art, but our design can be implemented efficiently, achieving fast update a commodity PC without hardware acceleration. Certainly, a robust and 3D environmental mapping system may encompass many components; such an efficient and robust tracker should hopefully prove useful.

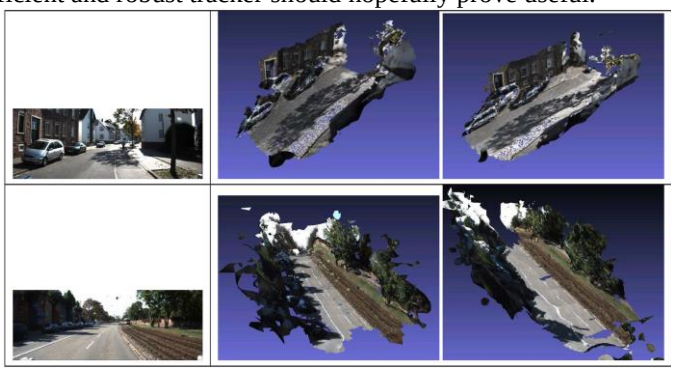


Figure 3 Left: a sample image in the video sequence, middle: 3D models using SIFT features and right: the corresponding models using ZBRIEF features.

REFERENCES

1. E. Guizzo, "How Google's Self-Driving Car Works," IEEE Spectrum, Feb 2013.
2. D. Forsyth and J. Ponce, Computer Vision, A Modern Approach, Prentice Hall, NJ 2003.
3. Andreas Geiger, Philip Lenz and Raquel Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
4. Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun, Vision meets Robotics: The KITTI Dataset, International Journal of Robotics Research (IJRR), 2013.
5. Jannik Fritsch, Tobias Kuehnl and Andreas Geiger, A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms, International Conference on Intelligent Transportation Systems (ITSC), 2013.
6. Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael Black, Richard Szeliski, A Database and Evaluation Methodology for Optical Flow, International Journal of Computer Vision, 92(1):1-31, March 2011.
7. B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121-130, 1981.
8. Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.
9. Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359, 200.
10. E. Rosten and T. Drummond, "Machine learning for high-speed corner detection". European Conference on Computer Vision. Springer. pp. 430–443, 2006.
11. M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "{BRIEF}: Computing a Local Binary Descriptor Very Fast," IEEE Transactions on Pattern Analysis and Machine Intelligence 2012.
12. Mikolajczyk, K., and Schmid, C., "A Performance Evaluation of Local Descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp 1615--1630, 2005.
13. Wu C. C., "SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT), http://cs.unc.edu/~ccwu/siftgpu/
14. T. B. Terriberry, L. M. French, J. Helmsen, "GPU Accelerating Speeded-up Robust Features," Proceedings of 3DPTV 2008.
15. OpenCV http://opencv.org/
16. VLfeat Library, http://www.vlfeat.org/
17. SURF Library, http://www.vision.ee.ethz.ch/~surf/
18. David M. Mount and Sunil Arya, "ANN: A Library for Approximate Nearest Neighbor Searching", http://www.cs.umd.edu/~mount/ANN/, 2010.
19. Ethan Rublee Vincent Rabaud Kurt Konolige and Gary Bradski, "ORB: an efficient alternative to SIFT or SURF", International Conference on Computer Vision, 2011.
20. Sun, D.; Roth, S. & Black, M. J. "Secrets of Optical Flow Estimation and Their Principles" IEEE Int. Conf. on Comp. Vision & Pattern Recognition, 2010.
21. R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, Cambridge, MA, 2003.