Evaluation of Point Matching Methods for Wide-baseline Stereo Correspondence on Mobile Platforms

Endre Juhász, Attila Tanács, and Zoltan Kato
Department of Image Processing and Computer Graphics, University of Szeged, Hungary
Email: {jendre,tanacs,kato}@inf.u-szeged.hu

Abstract—Wide-baseline stereo matching is a common problem of computer vision. By the explosion of smartphones equipped with camera modules, many classical computer vision solutions have been adapted to such platforms. The widespread use of various networking options for mobile phones, one can consider a set of smartphones as an ad-hoc camera network, where each camera is equipped with a more and more powerful computing engine in addition to a limited bandwidth communication with other devices. Therefore, the performance of classical vision algorithms in a collaborative mobile environment is of particular interest. In such a scenario we expect that the images are taken almost simultaneously but from different viewpoints, implying that the camera poses are significantly different but lighting conditions are the same. In this work, we provide a qualitative comparison of the most important keypoint detectors and descriptors in the context of wide-baseline stereo matching. We found that for resolution of 2 megapixels images the current mobile hardware is capable of providing results efficiently.

I. INTRODUCTION

The explosion of camera-equipped consumer smartphones opened new possibilities in computer vision applications. These devices have varying quality of image acquisition capabilities, dramatically and continuously increasing computing power and network connectivity — and billions of potential users. Smartphones in a given local area can be organized into an ad-hoc camera network to create e.g. panoramic or panoramic images or 3D reconstruction of a given scene, among others.

Finding correspondences between images of the same scene taken from different viewpoints is a fundamental problem of wide-baseline stereo matching. The classical approach is based on keypoint matching, which typically consists of the following steps: 1) detection of keypoints, 2) extracting descriptors in the neighborhood of keypoints and 3) finding corresponding point pairs using the similarities between these descriptors. The last step often involves outlier detection.

The sensor of a typical mobile camera is usually cheap and small sized. Although their megapixel resolution can be quite high (even 8-13 megapixels) they usually have problems in low light situations and introduce large amount of noise, blurriness and color distortions. Most of them are capable of providing images in JPEG format only. In a collaborative mobile environment, however, matching algorithms are not only challenged by the sparse representation of images but also by the limited mobile network bandwidth. Therefore the exchange of complete images between mobile devices is not feasible, which makes point-wise matching particularly appealing, even though the precision of keypoint localization is inherently compromised by lossy compression. On the positive side, built-in mobile cameras have fixed focus optics which makes it possible to calibrate the camera and use this calibration data in subsequent processing.

There is a vast literature on keypoint detectors and descriptors including surveys [1], [2] and performance evaluations [3]. In our study, we will evaluate state of the art detectors and descriptors that are also available in the OpenCV open source package for Android platform.

II. METHODS

In this section, we present each of the evaluated detectors and descriptors. Then the algorithm used for matching and outlier detection is described and finally we define the back-projection error used as a quantitative error measure in the evaluation.

A. Detectors

1) SIFT: is one of the most popular detectors and descriptors in the literature [4]. It is rotational and scale invariant. Its detector part provides a list of keypoints based on the Difference-of-Gaussians (DoG) scale-space representation of the image. A multiresolution pyramid consisting of octaves are built where each octave contains a given number of layers. Octaves and layers are formed applying different scale of Gaussian blur. Potential keypoints are local extrema that can be found in more scales. Keypoints along edges and from low contrast regions are suppressed. In the OpenCV implementation we can set the number of layers (default is 3), the contrast threshold (value of 0.04), edge threshold (10.0) and the blur level σ. The number of octaves are computed from the size of the image as round\((\log\min(width, height)/\log 2)) - 2\).

2) SURF: is a more efficient modification of SIFT and it includes both a detector and a descriptor [5]. Instead of Gaussians, the image is convolved by box filters at different sizes using the integral image representation for faster computation. Thus, the calculation time is independent of the filter size. Also, the normalized Laplace operator is approximated by the Difference-of-Hessian (DoH) representation. The scale space is analysed by up-scaling the filter size rather than iteratively reducing the image size. Filter sizes of 9 × 9, 15 × 15, 21 × 21 and 27 × 27 are used. SURF detects blob-like
structures at locations where the approximated determinant of the Hessian matrix is maximum. The responses are stored in a blob response map over different scales, and local maxima are detected. A non-maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied. The OpenCV implementation allows for setting a threshold value for the Hessian, features whose value is larger than this are kept. Also, the number of octaves (default value of 4) and layers per octave (value of 2) can be set. Other parameters influence the descriptor and those will be described there (see subsection II-B).

c) STAR: feature detector is derived from the CenSurE (Center Surrounded Extrema [6]) detector. CenSurE tries to maintain accuracy by computing features at all scales at every pixel in the original image. The operator takes the extrema of the Laplacian across scale. While CenSurE uses polygons such as square, hexagon and octagon as a more computable alternative to the circle, STAR mimics the circle with 2 overlapping squares (1 upright and 1 45-degree rotated) for convolution. In OpenCV we can set the maximal feature size (default value is 16), the threshold for the approximated Laplacian (default value of 5), two parameters to filter out features along lines and the size of the non-maximal supression (value of 5).

d) FAST: (Features from Accelerated Segment Test [8]) is a specialized version of the SUSAN corner criterion. 16 points lying on the Bresenham circle (essentially a discrete raster circle) of radius $r = 3$ around a central pixel are checked. If at least $n = 9$ pixels are all brighter or darker by at least $t$ intensity levels, then the pixel is taken as a feature. FAST is computationally very efficient and produces stable features. Its OpenCV implementation allows for setting the value of $t$ and whether we want to apply non-maximal supression.

e) GFTT: (Good Features to Track [7]) or the Shi-Tomasi corner detector is a very popular method in computer vision. Since it is primarily used for analyzing video sequences, it is computationally very efficient and assumes that the movement between the images is small. Although this latter condition does not hold for wide-baseline stereo image pairs, this detector combined with the SIFT descriptor produced the most efficient and stable results in our evaluation tests (see Section IV). The detector is based on the $2 \times 2$ Harris matrix containing averages of the partial derivatives in a local neighborhood of the image pixels. The smallest eigenvalue of the Harris matrix is taken as the cornerness measure. A high cornerness value indicate that the point is in the intersection of two nearly perpendicular lines. The parameters of the OpenCV implementation contain the maximal number of corner points (set to 1000), the block size for gradient averaging (set to 3). Alternatively the use of the Harris cornerness measure can also be selected to fine tune the results (it is turned off by default).

f) MSER: A recent alternative approach to detect keypoints is to find maximally stable extremal regions (MSER) in the images [9]. Instead of gradient based point detection, it tries to identify homogeneous regions in the images by gradually changing the threshold level and building a component tree. Components which satisfy a stability criterion are marked. By computing the centroid of the detected regions and fitting an ellipse to each region, orientation and scale parameters can also be produced. The original MSER method has a few parameters including the $\delta$ step size of the threshold value, and the minimal and maximal sizes of the blobs. The OpenCV implementation handles the maximal blob size parameter differently. Here, size is given in pixels, while the original algorithm defines it relatively to the image size. According to the original, we set this value 1% of the image size. Value of $\delta$ is set to 10. Besides, the OpenCV implementation introduces some new parameters related to the component tree creation. Since these are not part of the original method, we accepted their default values.

B. Descriptors

An extracted keypoint can be characterized by various descriptors. These are basically vectors representing the local structural properties of the images around the keypoint. Herein, we evaluated two popular descriptors: SIFT and SURF. Note that in a collaborative processing scenario, the keypoints are detected by each camera independently of the others. However for matching, the descriptors of the extracted keypoints should be exchanged by neighboring cameras, hence the necessary communication bandwidth is determined by the size of the descriptor vector as well as by the number of extracted keypoints.

g) SIFT: descriptor computes a feature vector of 128 dimensions based on the histograms of gradients around each keypoint. Scale and orientation parameters define a coordinate system for each keypoint in which the computations are taken. These might be provided by the keypoint detector or computed by the descriptor directly. The scale parameter is the extrema in the DoG pyramid (as for the SIFT detector, see II-A). In the given scale a dominant gradient direction is detected and used as orientation. Gradients are examined in a $16 \times 16$ window grouped in $4 \times 4$ blocks. In each block, gradient histograms are computed in 8 directions. This yields $8 \times 4 \times 4 = 128$ elements.

h) SURF: descriptor describes the distribution of the intensity content within the interest point neighborhood, instead of the gradient information extracted by SIFT. Distribution of first order Haar wavelet responses in $x$ and $y$ direction are taken into account. The region is split up regularly into smaller $4 \times 4$ square sub-regions. For each sub-region, the Haar wavelet responses are computed in $5 \times 5$ regularly spaced sample points. The dimensionality of the feature vector originally is 64, which makes the use of it more efficient than SIFT. However, a variant of SURF offers feature vectors of 128 elements equivalent of SIFT. The integral image representation is exploited here also. For each keypoint, orientation information is assigned by a sliding orientation window which detects the dominant orientation of the Gaussian weighted Haar wavelet responses within a circular neighborhood. For certain image processing tasks – such as image stitching – where the orientation of the keypoints is within 15 degrees this step can be omitted which significantly speed up the procedure. In OpenCV, one can set the size of the feature vector and whether orientation information should be calculated.

III. EVALUATION PIPELINE

The detectors and descriptors presented in the previous section provide the input of the evaluation pipeline. The components of this pipeline are independent of the particular
choice of detector/descriptor pairs, hence providing an unbiased method to measure the quality of the feature points in terms of the stereo reprojection error.

A. Matching and outlier filtering

For finding matches between the keypoint lists we used the FLANN (Fast Approximate Nearest Neighbor Search) kd-tree implementation of OpenCV optimized for fast nearest neighbor search in large datasets and for high dimensional features [10] using its default settings.

RANSAC (RANdom SAmple Consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers [11]. We use a modified RANSAC to eliminate incorrect matchings [12]. Here, the function that fits a model to data is the 8-point fundamental matrix computation algorithm. For each model, the points that are within distance \( t = 0.001 \) of the model are taken as inliers. RANSAC randomly samples keypoint pairs from the matched lists and returns a model which provides the most inliers.

B. Reprojection error

The mobile cameras with fixed focal length can be calibrated easily [12]. We make use of this calibration information in our evaluation scheme. Once the potential corresponding point pairs are available, we can compute a 3D position of the given point by triangulation and project it back to the imaging planes. The distance between the keypoint and the reprojected point of their corresponding point can be taken as the measure of correspondence correctness.

Epipolar geometry provides the theoretical background for stereo vision. Here we only give a short overview of the definitions necessary for our framework, for a more detailed description see e.g. [12]. The epipolar geometry between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as axis (the baseline is the line joining the camera centres). This geometry is usually motivated by considering the search for corresponding points in stereo matching.

The fundamental matrix \( F \) is the algebraic representation of epipolar geometry. Every point \( x \) in one image has one corresponding epipolar line \( l' \) on the other camera image. Thus, there is a map \( x \rightarrow l' \) from a point in one image to its corresponding epipolar line in the other image, expressed by the fundamental matrix as \( l' = Fx \). \( F \) is a \( 3 \times 3 \) homogeneous matrix of rank 2 and it has 7 degrees of freedom.

The essential matrix is the specialization of the fundamental matrix to the case of calibrated cameras. Thus the essential matrix has fewer degrees of freedom, and additional properties, compared to the fundamental matrix. Consider a camera matrix decomposed as \( P = K[R|t] \). It can be shown that the relationship between the fundamental and essential matrices is \( E = K^T F K \). To compute essential matrix from corresponding point pairs, we can use an 8 point algorithm as in [12].

Given the camera matrices \( P \) and \( P' \), let \( x \) and \( x' \) be two points in the two images that satisfy the epipolar constraint, i.e. \( x'^T F x = 0 \). In particular, it means that \( x' \) lies on the epipolar line \( Fx \). In turn this means that the two rays back-projected from image points \( x \) and \( x' \) lie in a common epipolar plane. Since the two rays lie in a plane, they will intersect in some point. This point \( X \) projects via the two cameras to the points \( x \) and \( x' \) in the two images.

Therefore from a matched point pair \( x \) and \( x' \), \( X \) is determined via triangulation, which is then projected back by the cameras \( P \) and \( P' \) onto the images. Let us denote these backprojected points by \( y \) in the first image and \( y' \) in the second image. Now, we can calculate a reprojection error as

\[
\frac{1}{2} (\| x - y \| + \| x' - y' \|)
\]  

C. Evaluation steps

We can summarize the steps of the evaluation as follows:

- Calibrate all cameras in advance.
- Detect keypoints in stereo images.
- Construct a descriptor for each keypoint.
- Using the descriptors, identify potential point pairs.
- Use RANSAC to eliminate outliers using the 8-point algorithm.
- Compute the essential matrix from point pairs.
- Compute rotation \( R \) and translation \( t \).
- Compute 3D point coordinates using triangulation.
- Use backprojection on both images to calculate the reprojection error in (1)

IV. TEST ENVIRONMENT AND RESULTS

We made two experiments. Images of city buildings were taken in both cases from varying distances, positions and orientations, even including complex, partly overlapping, symmetric objects. In the first one we used a Samsung Galaxy S smartphone. For performance reasons we took images of VGA resolution (640x480 pixels). We formed 40 stereo pairs of 19 different scenes (for certain scenes we had more than two images). Some examples are shown in Fig. 1.

In the second case we used four different smartphone cameras: Samsung Galaxy S, Sony Ericsson Ray, HTC EVO 3D (only one camera) and Samsung Galaxy Note 10.1 tablet and took images of 2 megapixel resolution. 39 stereo image pairs were formed from 9 scenes in a way that the image pairs were from different cameras. This makes the problem harder since the smartphones have different sensors. Noise level, color saturation, white balance JPEG artifacts varied among the images.

Camera calibration was carried out in Matlab using the Camera Calibration Toolbox [13]. The tests were performed offline in batch mode using different smartphone/tablet hardware (see Table I). In the second experiment only the Samsung Galaxy Note tablet was used as it had no problem with available RAM and processing power.

All smartphones run Android operating system. The evaluation method was implemented in the Android version of
Fig. 1. Some examples from the test image database. Images were taken nearly at the same time but from different viewpoints from forefrounds of buildings using different mobile cameras. Most of the image pairs are only partly overlapping.

<table>
<thead>
<tr>
<th>Model</th>
<th>CPU</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung Galaxy S</td>
<td>1 Ghz single-core</td>
<td>512 MB</td>
</tr>
<tr>
<td>HTC EVO 3D</td>
<td>1.2 Ghz dual-core</td>
<td>1 GB</td>
</tr>
<tr>
<td>Samsung Galaxy Note 10.1</td>
<td>1.4 Ghz quad-core</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

OpenCV 2.4.1. For convenience, the backprojection errors were computed offline using Matlab on a PC, based on the detected pairs.

We used some combinations of SIFT, SURF, GFTT, FAST, MSER and STAR detectors, and SIFT and SURF descriptors implemented in OpenCV. SIFT and SURF detectors work best with their respective descriptor. We found that their combination is either incredibly slow (SURF-SIFT) or many times gives no matching point pairs (SIFT-SURF). Same holds for the GFTT-SURF combination. The FAST detector provides very large number of keypoints so that the SIFT descriptor is not feasible.

In OpenCV, for descriptor matching there are two approaches, and both has the same three methods. One is Brute-Force, where for each descriptor in the first descriptor set the matcher finds the closest descriptor in the second set by trying each possibility. The other is the FLANN-based, where the matcher trains indices on a train descriptor collection and calls its nearest search methods to find best matches. Available methods for both approaches are k-nn, 1-nn, and radius-matching. For descriptor matching we used the FLANN-based 1-nn method. The Android version sets the maximal number of points to 100 which we kept. This matcher and the RANSAC method for outlier detection introduces randomness to the approach. Besides one single run, we also repeated the procedure five times and selected the pairing with the minimal reprojection error. Figure 2 shows the reprojection errors for detector/descriptor pairs of a single and five runs. Computing time is also shown for the five run case1. Figure 3 visualizes the reprojection errors of two stereo image pairs. We indicated the median error values in the diagrams since unsuccessful pairings might produce arbitrarily high reprojection errors making average values useless.

1We note that the computation time of the reprojection error is not included.
Fig. 2. Reprojection errors of the different combinations of point detectors and descriptors for VGA and 2 megapixel (MP) image sizes. Average computing times are also shown for the VGA tests.

V. DISCUSSION

Although the SURF descriptor-based methods perform slightly faster, it can be clearly seen that the SIFT descriptor-based ones outperform them. STAR, GFTT and SIFT detectors combined with SIFT descriptor provide stable results even for VGA and 2 MP images. Among these, GFTT is the computationally most effective.

The case of five runs is necessary. By keeping the computing time below 1 second most of the time the results become more stable. Since the reprojection error is measured in pixels, it is natural that the error values are higher for the 2 megapixel dataset (there is a scale value of 2.5 between the image sizes).

The performance of the MSER detector seems to be OK but it is very computing intensive compared to others. Besides, when combined with the SIFT descriptor, it fails to execute 45% of the cases on Samsung Galaxy S, and around 10-15% of the cases for higher end devices which are not indicated in the diagrams. This percentage rises to 55% in the case of the 2 megapixel images. This might be related to memory allocation errors in the used OpenCV implementation.

As expected, more powerful hardware can provide the results almost an order of magnitude faster than older ones. For the 2 megapixel images the current hardware solves the problem efficiently. Computing times are 3–5× slower than for the VGA sizes. Since the hardware evolution in the mobile space is incredibly fast, we can assume that in the coming years lower end consumer smartphones will have comparable specifications to today’s top of the line ones.

Figure 3(B) gives insight into the meaning of the reprojection error. The cathedral has two very similar towers and the stereo images show either of them. Based on their descriptors, keypoints detected in the different towers are strong candidates. Interestingly even the RANSAC method keeps some of them. Four such points are highlighted by ellipses in the images. These outliers cause problems when $R$ and $t$ are computed inducing higher reprojection errors.

VI. CONCLUSIONS

In this paper we evaluated some state of the art keypoint detector and descriptor methods that are available in the OpenCV package in the context of wide baseline stereo matching in mobile environment. Images taken by different smartphone and tablet cameras were used. The SIFT descriptor combined together with SIFT, GFTT or STAR detector methods proved to be successful in solving real world problems efficiently on current mobile hardware. The results suggest applicability of the methods in collaborative mobile image processing tasks.

ACKNOWLEDGMENT

This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KOV-2012-0013).
Fig. 3. Visualization of the reprojection error. Black crosses are the detected and matched points, yellow X shapes are the reprojected points. Yellow rectangles delineate the cutout boxes. (A) Almost perfect matches for a complex scene (median of reprojection error is 0.88 pixels). (B) Ellipses denote the non-corresponding points which produce rotational error in pose estimation (median of reprojection error is 20.1 pixels).

REFERENCES