Real-time Feature Acquisition and Integration for Vision-based Mobile Robots

Thomas Hübner and Renato Pajarola

Visualization and MultiMedia Lab, University of Zurich
Binzmühlestr. 14, 8050 Zurich, Switzerland
http://vmml.ifi.uzh.ch

Abstract. In this paper we propose a new system for real-time feature acquisition and integration based on high-resolution stereo images that is suitable for mobile robot platforms with limited resources. We combine a fast feature detection stage with a stable scale-invariant feature description method utilizing optimized spatial matching. Putative image feature matches are used to determine 3D coordinates of feature points and to estimate corresponding view transformations. Experimental results show the advantages of our system in terms of performance and accuracy compared to the standard methods used in the area.

Key words: Feature acquisition, feature integration, vision, mobile robot

1 Introduction

Reliable feature localization and mapping from multiple images is an important capability for mobile robots. A major fraction of robotic systems utilizes laser scanners as the primary input sensor to obtain information about the environment, in particular distance values. However, vision-based approaches using single- or multiple camera configurations are able to provide much more information relatively cheap, but also lead to a vast amount of image data that needs to be processed within the speed and latency constraints of the robot platform.

In vision-based approaches natural visual features are exploited as landmarks. These landmarks need to be detected and described by a feature vector to enable tracking and unique view-independent matching. The main computational effort is spent on feature tracking and description.

We address the problem of real-time acquisition and integration of visual features in natural indoor environments based on a rotation sequence of high-resolution stereo images. Our optimized feature detection and matching methods are suitable for online processing on vision-based robot platforms with limited computational resources. Despite focusing mainly on system performance, we achieve excellent accuracy in feature integration even in the presence of outliers or sparse landmarks. Our contributions include:

- Adaptive feature detection based on the *FAST* corner detector
- Improved *SIFT* based feature description for real-time processing

– Fast feature matching exploiting spatial and temporal coherence
– Stable and accurate feature integration from a non-stationary stereo camera
– Experimental comparisons to standard methods used in the area

2 Related work

The acquisition of landmarks is generally carried out in two steps: (1) Detection of suitable visual features that can be used as landmarks. (2) Description of the features with a feature vector that uses local image neighborhood information. A number of methods for both steps have been proposed in the past. We focus here on the ones primarily used in natural feature acquisition.

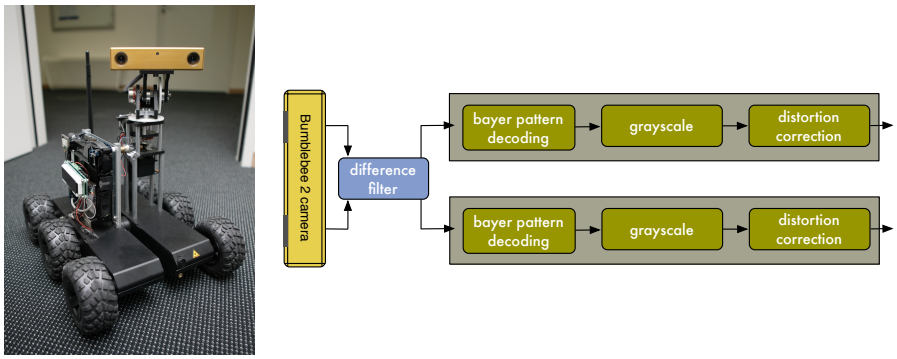
KLT developed by Kanade, Lucas and Tomasi [1] extracts image features that are adequate for tracking. *Normalized cross correlation* (NCC) tracks these features in subsequent images. Tracking has the advantage that features can still be followed even after the feature detector ceases to detect them. Thus, feature localization is very accurate, but if the search area is not sufficiently limited NCC exposes a poor performance with increasing image size (see also Section 4). The most popular feature detection and description approach is *SIFT* (Scale-Invariant-Feature-Transform) [2]. Visual features are detected as local extrema of the *Difference of Gaussian* (DoG) over scale space. The SIFT descriptor is rotation and scale invariant. Computationally, SIFT is one of the most expensive descriptors though achieving an excellent invariant feature description. More recently *SURF* (Speeded up Robust Features) [3] has been proposed. In SURF, feature detection is based on the *Hessian matrix* while the descriptor uses sums of 2D Haar wavelet responses. SURF is exposing a similar description quality as SIFT at a better performance [4].

Acquired landmarks are matched by calculating the Euclidian of their descriptors. To avoid a brute-force comparison of the descriptors, k-d trees, spatial hashing, and epipolar matching can be employed. Once putative matches are found, 3D coordinates of the landmarks can be calculated by triangulation. Features are spatially integrated estimating the view transformation from corresponding 2D image points [5, 6] or corresponding 3D world points [7, 8]. Estimating the view transformation based on 3D point correspondences is inferior as triangulations are much more uncertain in the depth direction. Therefore, the estimation based on 2D image points could give a more precise view transformation.

Our system for the online acquisition and integration of image features avoids expensive computational feature detection and tracking by using the *FAST* corner detector proposed by Rosten et al. [9] combined with a modified reduced SIFT descriptor [2]. Registration and matching of features in real-time is achieved by exploiting optimized spatial hashing [10]. Features are spatially integrated by estimating the view transformation on corresponding 2D image points directly, using a rather simple but stable algorithm. Our system provides very accurate results at comparatively low computational cost that no other previously proposed method is capable of.

3 System Overview

Our mobile robot platform as shown in Figure 1(a) uses a Bumblebee 2 stereo camera from Point Grey Research as the optical imaging sensor that is capable of capturing stereo image frames with a resolution of 1024×768 pixels at 20fps. The camera is attached to a pan-and-tilt device permitting an absolute rotation of 130°. Mechanically, the rotation angle can be controlled from $-65°$ to $+65°$, however, no exact feedback or control is possible for accurate setting of intermediate angles.



(a) (b)

Fig. 1. (a) Mobile robotic platform with Bumblebee 2 stereo camera. (b) Pre-processing pipeline.

Feature points are acquired and integrated in real-time from a continuous stream of images. Feature acquisition includes detection, description and matching of features while during integration the transformations between the images are estimated. The consecutive processing steps of our system are outlined in Figure 2 and will be explained in detail below.

Fig. 2. Consecutive steps of real-time feature acquisition and integration.

3.1 Pre-processing

Raw stereo image data acquired with the camera needs to be pre-processed before actual feature acquisition can take place. Figure 1(b) shows the pre-processing pipeline.

Captured stereo frames are run initially through a difference filter that limits the number of frames to be processed. For this purpose the filter subtracts two consecutive stereo frames and calculates the mean value of the resulting difference frame. If the mean value is under a certain threshold, the stereo frame is discarded. Image noise defines the threshold for this filter. Subsequently, the frame rate varies between 0-20fps depending on the image differences and thus the robot and camera motion.

Frames that reach the pre-processing stage are decoded from their raw Bayer pattern format to 8-bit grayscale. This is the image format which is used by practically all feature detectors. Camera lens distortions are corrected by a lookup-table and bi-linear interpolation.

The whole pre-processing pipeline is multi-threaded so that both stereo frames are processed in parallel. Use of SSE (Streaming SIMD Extensions) ensures that pre-processing consumes a minimal amount of available computational time before the actual feature detection is executed.

3.2 Feature Detection

Our feature detection is based on the *FAST* corner detector with non-maximum suppression for feature detection [9]. FAST is a high quality corner detector that significantly outperforms other existing algorithms. The principle of FAST is to examine a small patch around a candidate image point to see if it looks like a corner. This approach is efficiently implemented by a segment test algorithm improved through machine learning. We use the 9-point variant of this corner detector for our feature detection stage, as it provides optimal performance.

Regardless of being optimized for performance, the FAST corner detector is invariant to substantial view transformations and independent of the feature type. Its major disadvantage lies in the dependance on a user defined threshold. Nevertheless, this feature detector exposes such a great performance so that it can be used for adaptive feature detection. Instead of defining an image-based threshold we can define a desired feature count. The optimal threshold can then be found in a few iterations using the algorithm shown in Figure 3.

First we pre-define threshold step sizes that proved to be appropriate for fast threshold determination (1). We then iterate until a user defined limit (2), run the feature detector (3), and return the features if they lie within a 10% threshold of the desired feature count (4, 5). If the feature number differs to a greater extent the threshold is adjusted by a step value in the direction of the desired feature count (7, 8). Should we pass the target features we start reducing the step size and repeat steps (2-9). If no appropriate threshold can be found after exceeding the iteration limit, the feature search with the closest count is returned. This way we aim for a constant feature count by adaptive thresholding.

Section 4.1 shows that adaptive feature detection has only a marginal impact on the overall performance, whereas keeping a constant feature count is an important component for feature description and integration. Too sparse features result in an uncertainty in the estimated view transformation, while an excessive

number of features increases significantly the time spent on feature description and thus breaking the real-time constraint of the robot platform.

```
1   threshold step = {10, 5, 2, 1};
2   while iteration limit not reached
3       run feature detector;
4       if feature count within 10% of target features
5         return features;
6       (* adjust feature detection threshold *);
7         determine threshold step sign;
8         threshold ± = threshold step;
9         if passed target features
10          begin reducing threshold step;
```

Fig. 3. Adaptive feature detection procedure.

3.3 Feature Tracking

Detected features could be tracked between image frames by cross correlation or sum-of-squared differences (SSD) matching. We avoid actual feature tracking because the involved search in high resolution images would be much more time-consuming than our fast feature detection, description and matching. Tracking is eventually achieved by matching features in multiple images as outlined below in Section 3.5.

3.4 Feature Description

Feature detection is directly followed by feature description. Feature description is a fundamental part as it is used to associate landmarks from different views. Wrong associations between landmarks will result in inaccurate feature registration and integration, thus each detected feature needs to be assigned a unique invariant descriptor.

As noted in Section 2, SIFT achieves an excellent invariant feature description at the expense of decreased performance. Though quite fast implementations exist that employ SSE and OpenMP (Open Multi-Processing) claiming speed improvements of a factor of 6 over Lowe's standard approach [11], they are still not sufficient for real-time usage on high-resolution images.

Referring to SIFT, it is always considered in its most expensive variant with a 128-element feature vector. A smaller variant of the descriptor exists that performs only 8% worse in terms of feature association than the full version [2]. This variant uses a 3×3 descriptor array with only 4 gradient orientations. The resulting 36-element feature vector is much faster to compute and suits the real-time constraints of our system.

In detail, we choose a subregion around the feature that is 15×15 pixels wide. This subregion is divided into 5×5 pixel wide regions. Keeping close to the original implementation we calculate the gradient orientation and magnitude for each pixel in the subregion. Weighted by the distance from the region center and the magnitude, gradient orientations are accumulated into 4 gradient orientation histograms (Figure 4). The desriptor is normalized to unit length to reduce effects of illumination changes. To reduce the influence of large gradient magnitudes, descriptor values are clamped to 0.2 and re-normalized as proposed in [2].

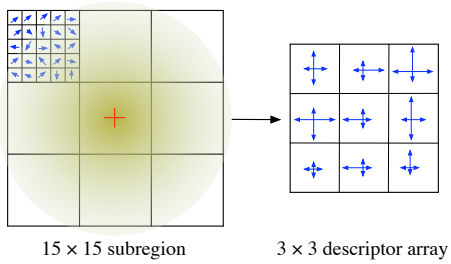


Fig. 4. 36-element reduced SIFT descriptor creation.

We implemented the descriptor calculation without any specific multi-processing or SSE extensions. Nevertheless, we achieve real-time performance on high resolution images (see Section 4.1). Figure 5 shows the outline of our *real-time SIFT* (RTSIFT) method.

During the initialization of RTSIFT we pre-calculate square root and arc tangent lookup tables (2, 3). Before considering individual descriptors we calculate the x- and y-image gradients once (6, 7). This can be performed more efficiently on the entire image than on separate subimages as feature regions tend to overlap. To describe an image feature we lookup the gradient magnitudes and orientations for each pixel in the feature's subregion (9, 10) and accumulate them after Gaussian weighting into the descriptor array (11). A list of descriptors is eventually returned (12).

3.5 Feature Matching and Outlier Removal

Feature matching associates landmarks from different views that correspond to the same feature. The similarity of two features is defined by the Euclidian distance of their feature descriptors. Comparing feature descriptors using a brute-force approach leads to a matching time that is linearly dependent on the number of features. Using spatial and temporal coherence we can avoid the linear dependency and considerably decrease the time spent on feature matching and reduce outliers at the same time.

To limit the number of descriptor comparisons we use *spatial hashing* [10]. Spatial hashing divides the image space into grid cells (Figure 6(a)). Each of

```
1   (* Initialization *)
2     pre-calculate square-root lookup table [256, 256];
3     pre-calculate arc-tangent lookup table [512, 512];
4
5   (* Descriptor calculation *)
6     calculate image x - gradients;
7     calculate image y - gradients;
8     for each detected feature do
9       lookup gradient magnitudes in square-root table;
10      lookup gradient orientations in arc-tangent table;
11      accumulate weighted orientations to descriptor array;
12    return descriptors;
```

Fig. 5. Feature description process.

these grid cells is assigned with a unique hash value. The grid cell size influences the number of landmarks that will reside in a cell. For matching, only landmarks within the same and neighboring grid cells are considered. The optimal grid cell size depends on the matching type. In stereo matching the grid size is set according to the expected depth range and thus the corresponding disparities. To estimate view transformations we use knowledge about the pixel displacement given by a certain robot or camera movement. We set the grid cell size to the expected maximum disparity or pixel displacement.

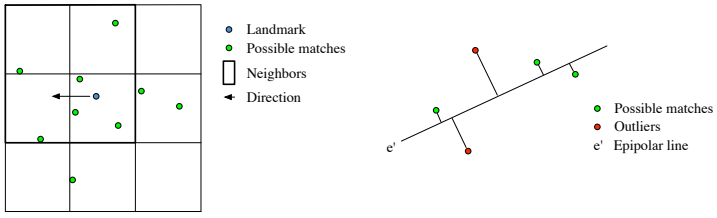


Fig. 6. (a) Spatial hashing to limit feature search. (b) Epipolar matching.

Our matching method is outlined in Figure 7. After setting the grid size according to the matching type the hash table is initialized with landmarks. We use a hash function (4) with the prime numbers $p1 = 73856093$, $p2 = 19349663$ and the hash table size n set to the landmark count. For each landmark (3) its grid position and the corresponding hash value are generated (4). The landmark's Euclidian distance is calculated to all landmarks corresponding to the same hash value (5). If a match is not found (6), neighboring grid cells are searched (7). These neighbor grid cells depend on the landmark position and the direction of

the pixel displacement. Features are only associated if they are closer than 50% of the second closest match (9).

Spatial hashing contributes greatly to the reduction of outlier matches as the spatial coherence constraint generates fewer mismatches.

```
1  set grid size according matching type;
2  initialize hash table;
3  for each described feature do
4     generate hash = (x · p1 ⊕ y · p2)  mod n;
5     compare feature descriptors inside grid cell;
6     if feature not found
7        search neighboring grid cells;
8     if match 50% closer than second match
9        associate features;
```

Fig. 7. Feature matching algorithm.

Feature matching can be improved further by considering temporal coherence. Robot and camera motion are nearly constant over a short time, thus it is possible to predict a landmark position based on previous matches. This is done by *epipolar matching*. The fundamental matrix F can be determined based on 2D image point correspondances [6]. Having F, we can estimate the epipolar lines along which landmarks should move. Matching is hence reduced to features that are near an epipolar line e' as illustrated in Figure 6(b). But uncertainty in the estimation of the fundamental matrix leads to wrong predictions.

As the camera movement of our robotic platform at a single position is limited to rotations, landmarks move either horizontal or vertical in image space, and we can reduce the problem to a simple 1D lookup table. Mismatches between similar features that remain after feature matching are handled during the following feature integration stage.

3.6 Feature Integration

Assuming a calibrated camera with known intrinsic parameter matrix (Equation 1), f being the focal length and X_c, Y_c the camera image center, we can easily triangulate the 3D position of associated landmarks (Equation 2).

$$\begin{pmatrix} f & 0 & X_c \\ 0 & f & Y_c \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

The parameter b represents the camera baseline and d the stereo disparity.

$$z = \frac{b \cdot f}{d}, \quad x = \frac{(x_i - X_c) \cdot z}{f}, \quad y = \frac{(y_i - Y_c) \cdot z}{f} \tag{2}$$

As noted in Section 2, estimating the view transformation from 3D points is unreliable due to depth uncertainties. Hence we estimate the camera movement based on 2D points from monocular images at continuous time steps.

Standard algorithms for estimating the camera movement based on the fundamental matrix F [6] showed to be inappropriate for feature integration on our robotic system. The angle and axis of rotation recovered from the fundamental matrix varied strongly with the standard algorithms and in most cases misleadingly a translation was found instead of a proper rotation. Furthermore, the fundamental matrix is very sensitive to slight changes in the landmarks' 2D positions and to outliers even when using RANSAC.

The camera on our robotic platform is rotating in angular steps $\leq 1°$, we therefore need a more robust feature integration approach that is able to reliably estimate even small angles. For the sake of simplicity we consider in the following a camera rotation around the y-axis, as a x-axis rotation can easily be derived from the given equations.

For each pair of associated landmarks we calculate the rotation angle ϕ using the cameras intrinsic parameters and trigonometric relations as indicated in Figure 8.

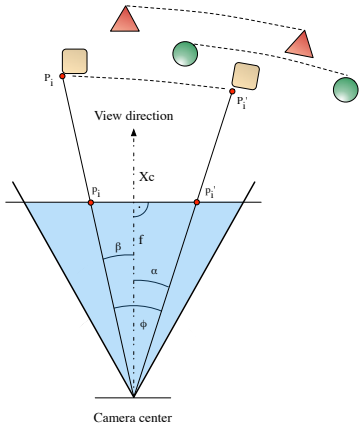


Fig. 8. Relation of camera rotation angles to disparities.

The rotation angle ϕ is found based on the projected image positions (p_i, p_i') of the landmark (P_i, P_i'). According to the projection of p_i and p_i' relative to the camera's image center X_c we determine the intermediate angles α and β as:

$$\alpha = atan(\frac{p_i' - X_c}{f}), \quad \beta = atan(\frac{X_c - p_i}{f}). \tag{3}$$

The angle ϕ is then easily found from

$$\phi = \alpha + \beta. \tag{4}$$

Given a set of individual rotation angles ϕ we need to find a common angle that agrees with most landmarks and takes outliers into account which were not eliminated during the previous feature matching stage.

Our solution to this is the following: First, we find the angle ϕ that corresponds to the majority of angles inside the set within a 10% threshold. This is done by simply testing each angle ϕ against all others. Second, landmarks that are not consistent with ϕ are considered to be outliers and excluded from estimating the rotation angle. The angle ϕ is finally used as starting point for iterative, non-linear least squares Levenberg-Marquardt optimization under the condition that the error $E(\phi)$ becomes minimal.

$$E(\phi) = \sum_{n=0}^{N} [p'_i - f(p_i, \phi)]^2. \tag{5}$$

The angle obtained through this optimization is used for 3D feature integration according to triangulation and the estimation of the cumulative rotation angle (see Section 4.2).

4 Experimental Results

We tested our system on a mobile robot platform that uses a Apple Mac Mini Core 2 Duo (2GHz CPU) for vision processing. The results presented are using two parallel threads unless stated otherwise.

4.1 Performance

To compare the performance of our RTSIFT implementation with adaptive feature detection to different standard methods, a real-world indoor sequence of 100 frames was recorded. On this pre-recorded sequence we compared RTSIFT to KLT [12], Fast SIFT [11], and SURF [13]. Timings are given in Figure 9(a) for the feature detection and description and for three different image resolutions. RTSIFT clearly outperforms any standard method for the feature detection and description on high-resolution images. We achieve 33fps at a resolution of 1024×768, while SURF (4.8fps), Fast SIFT (1.2fps) and KLT (0.8fps) are significantly slower.

We additionally evaluated the influence of the adaptive feature detection (AFD) on the performance of the feature detection, as well as on the number of detected features and the resulting description time. For the adaptive detection the number of target features was set to 500 and the iteration limit to 6. The initial threshold for FAST feature detection was set to 15. Figure 9(b) shows the comparison between the original FAST corner detector and our implementation with adaptive feature detection. With increasing image size the number of

	1024 × 768	512 × 384	256 × 192
RTSIFT	2.98	1.57	1.27
KLT	125.11	23.43	5.5
SIFT	84.31	25.65	7.54
SURF	20.77	7.34	1.99

(a)

	3648 × 2736	1920 × 1440	1024 × 768
FAST	0.21	0.06	0.02
# Features	21817	6587	2582
Description	1.40	0.32	0.18
FAST AFD	0.63	0.21	0.07
# Features	521	460	501
Description	0.49	0.15	0.06

(b)

	130°	Error	65°	Error	10°	Error
RTSIFT	128.88	1.12	63.93	1.07	8.56	1.44
KLT	135.36	5.36	66.35	1.35	9.01	0.99
SIFT	128.92	1.08	63.70	1.3	8.59	1.41
SURF	128.59	1.41	62.69	2.31	8.13	1.87

(c)

Fig. 9. (a) Feature detection and description applied to 100 frames. (b) Adaptive feature detection (single-threaded). (c) Accuracy of estimating the cumulative rotation angle. All timings are given in seconds.

detected features increases proportionally when using the original FAST corner detector. While the excessive number of features is not necessarily beneficial, this reduces the performance of the subsequent feature description stage. Our adaptive feature detection keeps the number of features near a given constant and thus guarantees fast computation for feature description.

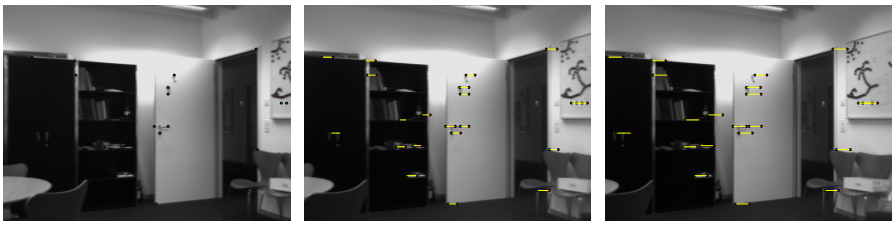
4.2 Accuracy

The accuracy of the system is an important factor for the feature integration. We tested our method with different pre-defined rotation angles. While rotating the camera, landmarks are continuously acquired and intermediate rotation angles are estimated. The resulting accumulated angle should ideally correspond to the real camera rotation. Figure 9(c) shows the error in estimating the cumulative rotation angle for the different methods. While all methods achieve a rather low error for estimating the rotation angle, using our feature integration, RTSIFT is significantly faster.

In Figure 10 we show an example of the real-world environment with matched features in the left-right stereo images as well as matched features in subsequent frames over time.

5 Conclusion

In this paper we presented an efficient system for real-time feature acquisition and integration for vision-based mobile robots. A fast adaptive feature detection stage is combined with a SIFT-based stable, scale-invariant and real-time feature description while utilizing spatial hashing and epipolar matching for feature association and outlier removal. The proposed feature integration method showed to be robust in real-world indoor environments with low texture information as well as fast, and thus can be successfully applied for real-time robot navigation and 3D reconstruction.



(a) (b) (c)

Fig. 10. (a) Left-view image with detected landmarks in green. (b) Right-view with matched stereo correspondences. (c) Right image at $t+1$ with matched correspondences over time.

References

1. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle (1994)
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60** (2004) 91–110
3. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded-up robust features. In: 9th European Conference on Computer Vision, Graz, Austria (2006)
4. Bauer, J., Sünderhauf, N., Protzel, P.: Comparing several implementations of two recently published feature detectors. Proc. of the International Conference on Intelligent and Autonomous Systems (2007)
5. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, ISBN: 0521540518 (2004)
6. Zhang, Z., Kanade, T.: Determining the epipolar geometry and its uncertainty: A review. International Journal of Computer Vision **27** (1998) 161–195
7. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14** (1992) 239–256
8. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. Pattern Anal. Mach. Intell. **13** (1991) 376–380
9. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: European Conference on Computer Vision. Volume 1. (2006) 430–443
10. Teschner, M., Heidelberger, B., Mueller, M., Pomeranets, D., Gross, M.: Optimized spatial hashing for collision detection of deformable objects. Proceedings of Vision, Modeling, Visualization (VMV 2003) (2003) 47–54
11. OpenSource: Fast sift image features library. Website (2008) Available online at http://libsift.sourceforge.net/; visited on July 14th 2009.
12. Birchfield, S.: Klt: An implementation of the kanade-lucas-tomasi feature tracker. Website (2007) Available online at http://www.ces.clemson.edu/ stb/klt/; visited on July 14th 2009.
13. Bay, H.: Surf: Speeded up robust features. Website (2006) Available online at http://www.vision.ee.ethz.ch/ surf/; visited on July 14th 2009.