DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo

Engin Tola, Vincent Lepetit, and Pascal Fua, *Member, IEEE*

Abstract

In this paper, we introduce a local image descriptor, DAISY, which is very efficient to compute densely. We also present an EM based algorithm to compute dense depth and occlusion maps from wide baseline image pairs using this descriptor. This yields much better results in wide baseline situations than the pixel and correlation based algorithms that are commonly used in narrow baseline stereo. Also, using a descriptor makes our algorithm robust against many photometric and geometric transformations.

Our descriptor is inspired from earlier ones such as SIFT and GLOH but can be computed much faster for our purposes. Unlike SURF which can also be computed efficiently at every pixel, it does not introduce artifacts that degrade the matching performance when used densely.

It is important to note that our approach is the first algorithm that attempts to estimate dense depth maps from *wide baseline image pairs* and we show that it is a good one at that with many experiments for depth estimation accuracy, occlusion detection, and comparing it against other descriptors on laser scanned ground truth scenes. We also tested our approach on a variety of indoor and outdoor scenes with different photometric and geometric transformations and our experiments support our claim to being robust against these.

Index Terms

Image processing and computer vision, Dense Depthmap Estimation, Local Descriptors

I. Introduction

Though dense short-baseline stereo matching is well understood [9], [25], its wide baseline counterpart is, by contrast, much more challenging due to large perspective distortions and increased occluded areas.

The authors are with the Ecole Polytechnique Fédérale de Lausanne (EPFL), Computer Vision Laboratory, CH-1015 Lausanne, Switzerland.

E-mail: {Engin.Tola, Vincent.Lepetit, Pascal.Fua}@epfl.ch.

It is nevertheless worth addressing because it can yield more accurate depth estimates while requiring fewer images to reconstruct a complete scene. Also, it may be necessary to compute depth from two widely separated cameras such as a surveillance application in which installing cameras side-by-side is not feasible.

Large correlation windows are not appropriate for wide-baseline matching because they are not robust to perspective distortions and tend to straddle areas of different depths or partial occlusions. Thus, most researchers favor simple pixel differencing [6], [16], [24] or correlation over very small windows [26]. They then rely on optimization techniques such as graph-cuts [16] or PDE based diffusion operators [27] to enforce spatial consistency. The drawback of using small image patches is that reliable image information can only be obtained where the image texture is of sufficient quality. Furthermore, the matching becomes very sensitive to illumination changes and repetitive patterns.

An alternative to performing dense wide-baseline matching is to first match a few feature points, triangulate them, and then locally rectify the images. This approach, however, potentially is not without problems. If some matches are wrong and are not detected as such, gross reconstruction errors will occur. Furthermore, image rectification in the triangles may not be sufficient if the scene within cannot be treated as locally planar.

We, instead, advocate replacing correlation windows with local region descriptors, which lets us take advantage of powerful global optimization schemes such as graph-cuts to force spatial consistency. Existing local region descriptors such as SIFT [19] or GLOH [21] have been designed for robustness to perspective and lighting changes and have proved successful for sparse wide-baseline matching. However, they are much more computationally demanding than simple correlation. Thus, for dense wide-baseline matching purposes, local region descriptors have so far only been used to match a few seed points [33] or to provide constraints on the reconstruction [27].

We, therefore, introduce a new descriptor that retains the robustness of SIFT and GLOH and can be computed quickly at every single image pixel. Its shape is closely related to that of [32], which has been shown to be optimal for sparse matching but is not designed for efficiency. We use our descriptor for dense matching and view-based synthesis using stereo-pairs having various image transforms or for pairs with too large a baseline for standard correlation-based techniques to work, as shown in Figs. 1, 2, 3 and 4. For example, on a standard laptop, it takes less than 4 seconds to perform the computations using our descriptor over all the pixels of an 800×600 image, whereas it takes over 250 seconds using SIFT. Furthermore, it gives better results than SIFT, SURF, NCC and pixel differencing as will be shown by comparing the resulting depth maps to laser scanned data.

Figure 1. **Contrast Change.** First two images are used as input. We manually increased the contrast of the first image and tried to estimate the depth from the 2^{nd} image's point-of-view. We used NCC, SIFT and DAISY and their reconstructions are displayed in the second row respectively. We also resynthesized the second image using the depth map of DAISY and the first image's intensities and show it at the end of the first row.

To be specific, SIFT and GLOH owe much of their strength to the use of gradient orientation histograms, which are relatively robust to distortions. The more recent SURF descriptor [4] approximates them by using integral images to compute the histograms bins. This method is computationally effective with respect to computing the descriptor's value at every pixel but does away with SIFT's spatial weighting scheme. All gradients contribute equally to their respective bins, which results in damaging artifacts when used for dense computation. The key insight of this paper is that computational efficiency can be achieved without performance loss by convolving orientation maps to compute the bin values using Gaussian kernels. This lets us match relatively large patches — 31×31 — at an acceptable computational cost and improve robustness in un-occluded areas over techniques that use smaller patches. Using large areas requires to handle occlusion boundaries properly though and we address this issue by using different masks at each location and select the best one by using an EM framework. This is inspired by the earlier works of [13], [14], [15] where multiple or adaptive correlation windows are used.

After discussing related work in Section II, we introduce our new local descriptor and present an efficient way to compute it in Section III. In Section IV we detail our EM based occlusion handling framework. Finally, in Section V, we present results and compare our descriptor to SIFT, SURF, NCC

and pixel differencing.

II. Related Work

Even though multi-view 3–D surface reconstruction has been investigated for many decades [9], [25], it is still far from being completely solved due to many sources of errors such as perspective distortion, occlusions, and textureless areas. Most state-of-the-art methods rely on first using local measures to estimate the similarity of pixels across images and then on imposing global shape constraints using dynamic programming [3], level sets [11], space carving [17], graph-cuts [16], [24], [8], PDE [1], [27], or EM [26]. In this paper, we do not focus on the method used to impose the global constraints and use a standard one [8]. Instead, we concentrate on the similarity measure all these algorithms rely on.

In a short baseline setup, reconstructed surfaces are often assumed near fronto-parallel, so the similarity between pixels can be measured by cross-correlating square windows. This is less prone to errors compared to pixel differencing and allows normalization against illumination changes.

In a wide-baseline setup, however, large correlation windows are especially affected by perspective distortions and occlusions. Thus, wide-baseline methods [1], [16], [26], [27] tend to rely on very small correlation windows or revert to point-wise similarity measures, which loose the discriminative power larger windows could provide. This loss can be compensated by using multiple [2], [27] or high-resolution [27] images. The latter is particularly effective because areas that appear uniform at a small scale are often quite textured when imaged at a larger one. However, even then, lighting changes remain difficult to handle. For example, [27] shows results either for wide baseline without light changes, or with light changes but under a shorter baseline.

As we shall see, our feature descriptor reduces the need for higher-resolution images and achieve comparable results using fewer number of images. It does so by considering large image patches while remaining stable under perspective distortions. Earlier approaches to this problem relied on warping the correlation windows [10]. However the warps were estimated from a first reconstruction obtained using classical windows, which is usually not practical in wide baseline situations. By contrast, our method does not require an initial reconstruction. Additionally, in a recent publication [32], a descriptor, which is very similar to ours in shape, has been shown to outperform many state-of-the-art feature descriptors for sparse point matching. However, unlike this descriptor, ours is designed for fast and efficient computation at every pixel in the image.

Local image descriptors have already been used in dense matching, though in a more traditional way, to match only sparse pixels that are feature points [31], [19]. In [27], [33], these matched points are used

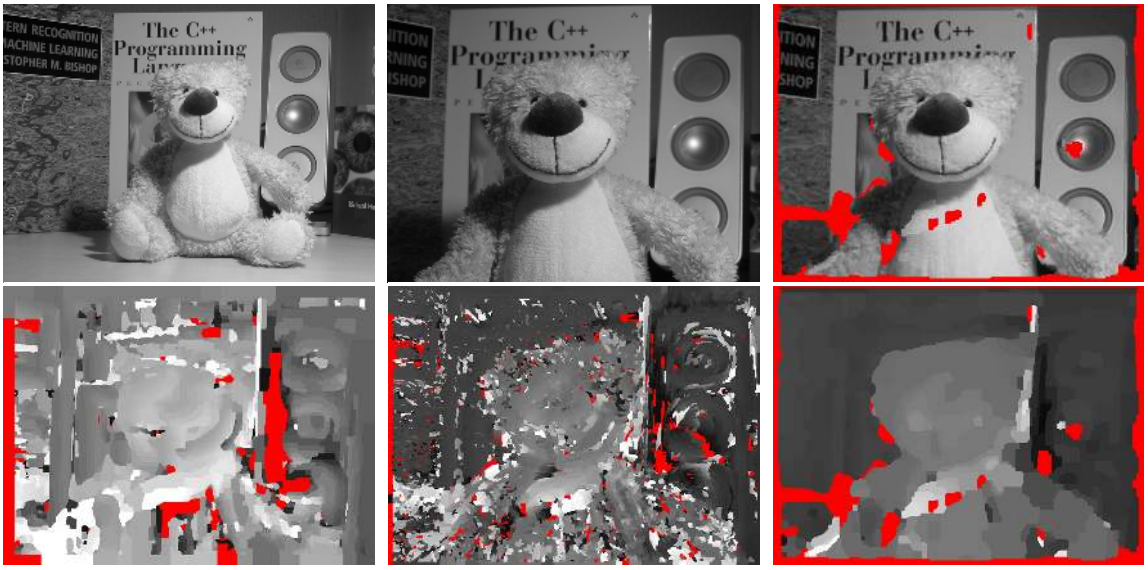


Figure 2. **Scale Change.** We used the first 2 images of the upper row for computing the depth map from the 2^{nd} image's point of view. The depth maps are computed using NCC, SIFT and DAISY and they're displayed in the lower row in that order. The last image in the first row shows the resynthesized image using the DAISY's depth estimate. Although scale change is not explicitly addressed in any way and we used the same parameters for the descriptors of two images, we obtain a very acceptable depth map.

as anchors for computing the full reconstruction. [33] propagates the disparities of the matched feature points to their neighbors, while [27] uses them to initialize an iterative estimation of the depth maps.

To summarize, local descriptors have already proved their worth for dense wide baseline matching, but only in a limited way. This is due in part to their high computational cost and in past their sensitivity to occlusions. The technique we propose addresses both issues.

III. Our Local Descriptor

In this section, we briefly describe SIFT [19] and GLOH [21] and then introduce our DAISY descriptor. We discuss both its relationship with them and its greater effectiveness for dense computations.

The SIFT and GLOH descriptors involve 3–D histograms in which two dimensions correspond to image spatial dimensions and the additional dimension to the image gradient direction. They are computed over local regions, usually centered on feature points but sometimes also densely sampled for object recognition tasks [12], [18]. Each pixel belonging to the local region contributes to the histogram depending on its location in the local region, and on the orientation and the norm of the image gradient at its location. As depicted by Fig. 5-a, when an image gradient vector computed at a pixel location is integrated to

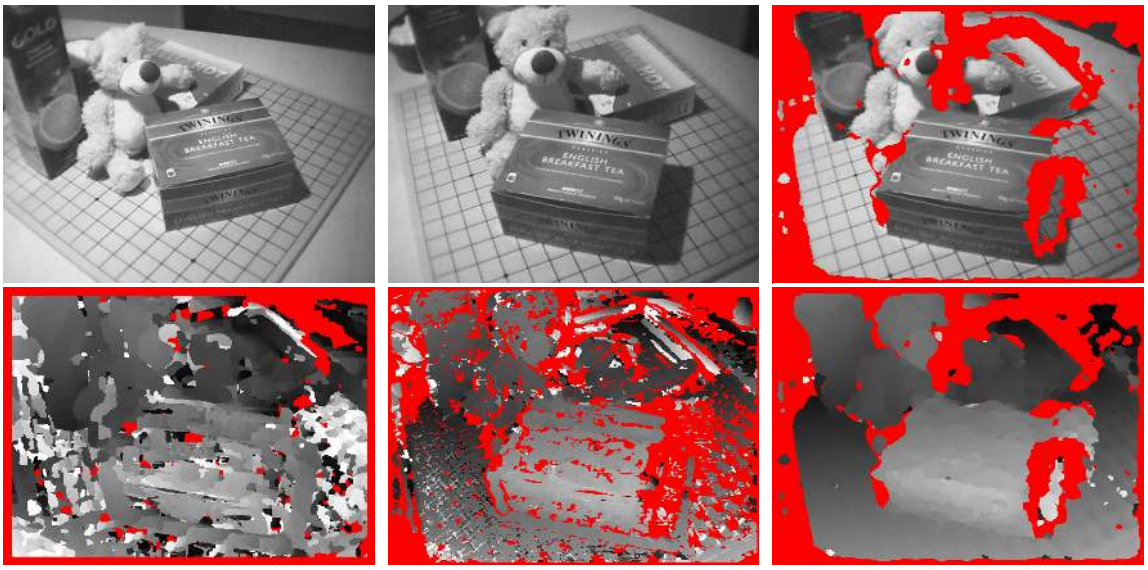


Figure 3. **Image Quality.** We used the first 2 images of the upper row, which are obtained by a webcam, for computing the depth map from the 2^{nd} image's point of view. The depth maps are computed using NCC, SIFT and DAISY and they are displayed in the lower row in that order. The last image in the first row shows the resynthesized image using the DAISY's depth estimate. Despite the somewhat blurry, low-quality nature of the images, we can still compute a good depth map.

the 3–D histogram, its contribution is spread over $2 \times 2 \times 2 = 8$ bins to avoid boundary effects. More precisely, each bin is incremented by the value of the gradient norm multiplied by a weight inversely related to the distances (i.e. as the distance increases, weight decreases) between the pixel location and the bin boundaries, and also to the distance between the pixel location and the one of the keypoint. As a result, each bin contains a weighted sum of the norms of the image gradients around its center, where the weights roughly depend on the distance to the bin center.

In this work, our goal is to reformulate these descriptors so that they can be efficiently computed at every pixel location. Intuitively, this means computing the histograms only once per region and reusing them for all neighbouring pixels.

To this end, we replace the weighted sums of gradient norms by convolutions of the gradients in specific directions with several Gaussian filters. We will see that this gives the same kind of invariance as the SIFT and GLOH histogram building, but is much faster for dense-matching purposes and allows the computation of the descriptors in all directions with little overhead.

Even though SIFT, GLOH and DAISY involve different weighting schemes for the orientation gradients, the computed histograms can be expected to be very similar which gives a new insight on what makes

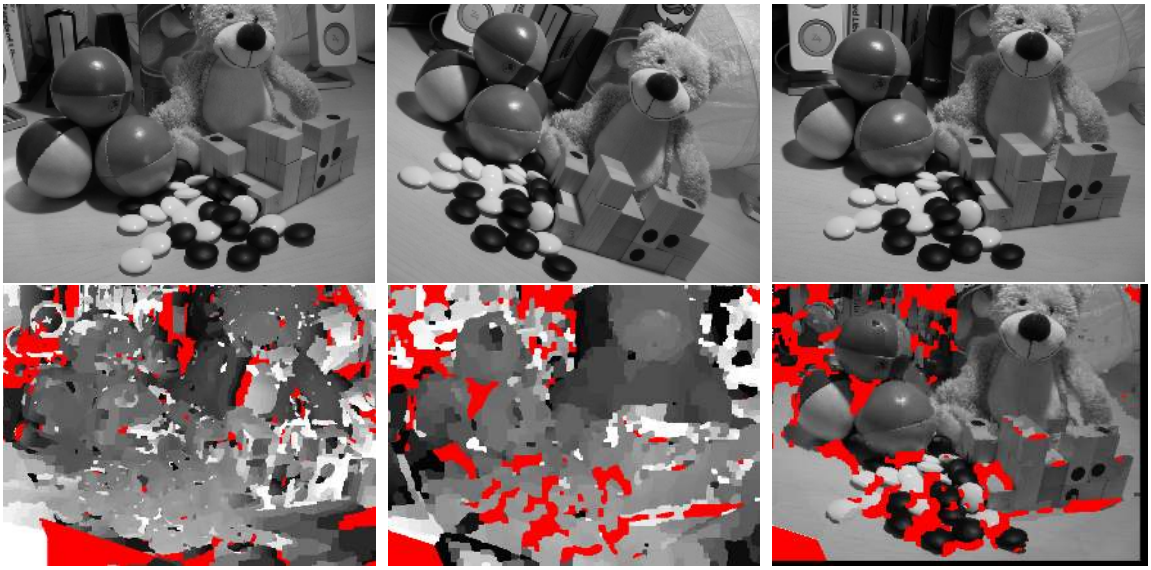


Figure 4. **Rotation around a point.** We used the first 2 images of the upper row for computing the depth map from the 3^{rd} image's point of view. The depth maps are computed using NCC, and DAISY and they're displayed in the lower row in that order. The last image in the second row shows the resynthesized image using the DAISY's depth estimate.

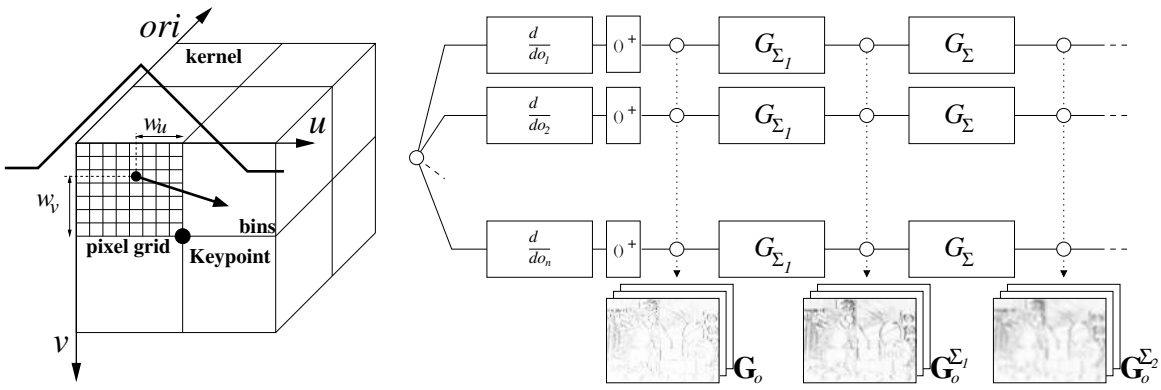
SIFT work: convolving with a kernel simultaneously dampens the noise and gives a measure of invariance to translation. This is also better than integral image-like computations of histograms [22] in which all gradient vectors have the same contribution. We can very efficiently reduce the influence of gradient norms from distant locations.

Fig. 6 depicts the resulting descriptor. Note that its shape resembles that of a descriptor [32] that has been shown to outperform many state-of-the-art ones. However, unlike that descriptor, DAISY is also designed for effective dense computation. The parameters that control its shape are listed in Table I. We will discuss in Section V how they should be chosen.

There is a strong connection between DAISY and geometric blur [5]. In this work, the authors recommended using smaller blur kernels near the center and larger away from it and reported successful results using oriented edge filter responses. DAISY follows this recommendation by using larger Gaussian kernels in its outer rings but replaces the edge filters by simple convolutions for the sake of efficiency.

A. The DAISY Descriptor

We now give a more formal definition of our *DAISY* descriptor. For a given input image, we first compute H number of *orientation maps*, $\mathbf{G}_i, 1 \leq i \leq H$, one for each quantized direction, where



(a) (b)

Figure 5. **Relationship between SIFT and DAISY.** (a) SIFT is a 3–D histogram computed over a local area where each pixel location contributes to bins depending on its location and the orientation of its image gradient, the importance of the contribution being proportional to the norm of the gradient. Each gradient vector is spread over $2 \times 2 \times 2$ bins to avoid boundary effects, and its contribution to each bin is weighted by the distances between the pixel location and the bin boundaries. (b) DAISY computes similar values but in a dense way. Each gradient vector also contributes to several of the elements of the description vector, but the sum of the weighted contributions is computed by convolution for better computation times. We first compute orientation maps from the original images, which are then convolved to obtain the convolved orientation maps $\mathbf{G}_o^{\Sigma_i}$. The values of the $\mathbf{G}_o^{\Sigma_i}$ correspond to the values in the SIFT bins, and will be used to build DAISY. By chaining the convolutions, the $\mathbf{G}_o^{\Sigma_i}$ can be obtained very efficiently.

$\mathbf{G}_o(u, v)$ equals the image gradient norm at location (u, v) for direction o if it is bigger than zero, else it is equal to zero. This preserves the polarity of the intensity changes. Formally, orientation maps are written as $\mathbf{G}_o = \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+$ where $\mathbf{I}$ is the input image, o is the orientation of the derivative, and $(.)^+$ is the operator such that $(a)^+ = max(a, 0)$.

Each orientation map is then convolved several times with Gaussian kernels of different Σ values to obtain *convolved orientation map*s for different sized regions as $\mathbf{G}_o^{\Sigma} = G_{\Sigma} * \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+$ with G_{Σ} a Gaussian kernel. Different Σ's are used to control the size of the region.

Our primary motivation here is to reduce the computational requirements and convolutions can be implemented very efficiently especially when using Gaussian filters, which are separable. Moreover, we can compute the orientation maps for different sizes at low cost because convolutions with a large Gaussian kernel can be obtained from several consecutive convolutions with smaller kernels. More specifically, given $\mathbf{G}_o^{\Sigma_1}$, we can efficiently compute $\mathbf{G}_o^{\Sigma_2}$ with $\Sigma_2 > \Sigma_1$ as

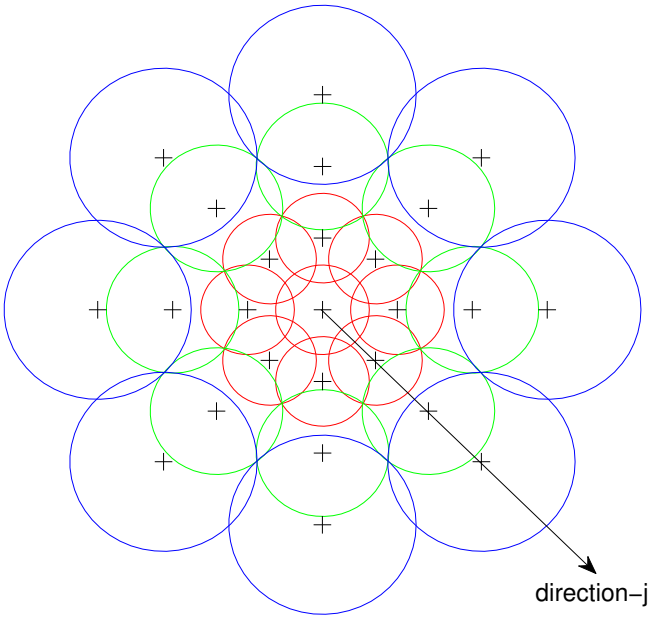


Figure 6. **The DAISY descriptor**. Each circle represents a region where the radius is proportional to the standard deviations of the Gaussian kernels and the '+' sign represents the locations where we sample the convolved orientation maps center being a pixel location where we compute the descriptor. By overlapping the regions we achieve smooth transitions between the regions and a degree of rotational robustness. The radii of the outer regions are increased to have an equal sampling of the rotational axis which is necessary for robustness against rotation.

Parameter Name	Symbol	Description and Default Value
Radius	R	Distance from the center pixel to the outer most grid point. (15)
Radius Quantization No.	Q	Number of convolved orientations layers with different Σ's. (3)
Angular Quantization No.	T	Number of histograms at a single layer. (8)
Histogram Quantization No.	H	Number of bins in the histogram. (8)
Grid Point No.	S	Number of histograms used in the descriptor = $Q * T + 1$.
Descriptor Size	$\mathcal{D}_s$	The total size of the descriptor vector = $S * H$

Table I

DAISY PARAMETERS

$$\mathbf{G}_o^{\Sigma_2} = G_{\Sigma_2} * \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+ = G_{\Sigma} * G_{\Sigma_1} * \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+ = G_{\Sigma} * \mathbf{G}_o^{\Sigma_1} \,,$$

with $\Sigma = \sqrt{\Sigma_2^2 - \Sigma_1^2}$. This computational flow, the incremental computation of the *convolved orientation*

maps from an input image, is summarized in Fig. 5-b.

To make the link with SIFT and GLOH, note that each pixel location of the convolved orientation maps contains a value very similar to the value of a bin in SIFT or GLOH that is a weighted sum of gradient norms computed over a small neighbourhood. We use a Gaussian kernel whereas SIFT and GLOH rely on a triangular shaped kernel. It can also be linked to tensor voting in [20] by thinking of each location in our orientation maps as a voting component and of our aggregation kernel as the voting weights.

As depicted by Fig. 6, at each pixel location, DAISY consists of a vector made of values from the convolved orientation maps located on concentric circles centered on the location, and where the amount of Gaussian smoothing is proportional to the radii of the circles. As can be seen from the figure, this gives the descriptor the appearance of a flower, hence its name.

Let $\mathbf{h}_\Sigma(u, v)$ represent the vector made of the values at location (u, v) in the orientation maps after convolution by a Gaussian kernel of standard deviation Σ.

$$\mathbf{h}_\Sigma(u, v) = \left[\mathbf{G}_1^\Sigma(u, v), \ldots, \mathbf{G}_H^\Sigma(u, v)\right]^\top , \tag{1}$$

where $\mathbf{G}_1^\Sigma$, $\mathbf{G}_2^\Sigma$, and $\mathbf{G}_H^\Sigma$ denote the Σ-convolved orientation maps in different directions. We normalize these vectors to unit norm, and denote the normalized vectors by $\widetilde{\mathbf{h}}_\Sigma(u, v)$. The normalization is performed in each histogram independently to be able to represent the pixels near occlusions as correct as possible. If we were to normalize the descriptor as a whole, then the descriptors of the same point that is close to an occlusion would be very different when imaged from different viewpoints.

Problems might arise in homogeneous regions since we normalize each histogram independently. However, consider that we are designing this descriptor for a stereo application. In a worst case scenario, DAISY will not perform any worse than a standard region based metric like NCC. However, this will not happen as often because we use a relatively large descriptor. Furthermore, the global optimization algorithm discussed in Section IV will often fix the resulting errors. If one truly wants the robustness of large regions, one solution to this might be computing unnormalized descriptors and normalizing the visible parts of the descriptor globally before dissimilarity computation. This, however, will increase the computation time of the matching stage with two additional normalization operations for each possible depth per pixel. For applications other than stereo, the normalization should probably be changed depending on the specifics of the application, but this is beyond the scope of this paper.

If Q represents the number of different circular layers, then the full DAISY descriptor $\mathcal{D}(u_0, v_0)$ for location (u_0, v_0) is defined as the concatenation of $\widetilde{\mathbf{h}}$ vectors :

Image Size	DAISY	SIFT
800x600	3.8	252
1024x768	6.5	432
1280x960	9.8	651

Table II

COMPUTATION TIME IN SECONDS ON AN IBM T60 LAPTOP

$$
\begin{aligned}
&\mathcal{D}(u_0, v_0) = \\
\Big[\;&\widetilde{\mathbf{h}}_{\Sigma_1}^\top(u_0, v_0), \\
&\widetilde{\mathbf{h}}_{\Sigma_1}^\top(\mathbf{l}_1(u_0, v_0, R_1)), \cdots, \widetilde{\mathbf{h}}_{\Sigma_1}^\top(\mathbf{l}_T(u_0, v_0, R_1)), \\
&\widetilde{\mathbf{h}}_{\Sigma_2}^\top(\mathbf{l}_1(u_0, v_0, R_2)), \cdots, \widetilde{\mathbf{h}}_{\Sigma_2}^\top(\mathbf{l}_T(u_0, v_0, R_2)), \\
&\cdots \\
&\widetilde{\mathbf{h}}_{\Sigma_Q}^\top(\mathbf{l}_1(u_0, v_0, R_Q)), \cdots, \widetilde{\mathbf{h}}_{\Sigma_Q}^\top(\mathbf{l}_T(u_0, v_0, R_Q)) \quad \Big]^\top,
\end{aligned}
$$

where $\mathbf{l}_j(u, v, R)$ is the location with distance R from (u, v) in the direction given by j when the directions are quantized into the T values of Table I.

We use a circular grid instead of SIFT's regular one since it has been shown to have better localization properties [21]. In that sense, our descriptor is closer to GLOH without PCA than to SIFT. Combining an isotropic Gaussian kernel with a circular grid also makes our descriptor naturally resistant to rotational perturbations. The overlapping regions ensure a smoothly changing descriptor along the rotation axis and by increasing the overlap we can make it more robust up to the point where the descriptor starts losing its discriminative power.

As mentioned earlier, one important advantage of the circular design and using isotropic kernels is that when we want to compute the descriptor in a different orientation, there is no need to re-compute the convolved orientation maps; they are still valid and we can recompute the descriptor by simply rotating the sampling grid. The histograms will then also need to be shifted circularly to account for the change in relative gradient orientations but this only represents a very small overhead.

We mentioned earlier that the variance of the Gaussian kernels is chosen to be proportional to the size of the regions in the descriptor. Specifically, they are taken to be

$$
\sigma_i = \frac{R(i+1)}{2Q}\,, \tag{2}
$$

where i represents the i^{th} layer in the circular grid (see Fig. 6). Histogram locations are expressed in polar coordinates as

$$r_i = \frac{R(i+1)}{Q}$$
$$\theta_j = \frac{2\pi j}{T} \ .$$

B. Computational Complexity

The efficiency of DAISY comes from the fact that most computations are separable convolutions and that we avoid computing more than once the histograms common to nearby descriptors. In this section we give a formal complexity analysis of both DAISY and SIFT and, then, compare them.

III-B.1 Computing DAISY: Recall from Table I that DAISY is parameterized with its radius R, number of rings Q, number of histograms in a ring T, and the number of bins in each histogram H. Assuming that the image has P pixels, we begin by computing the orientation layers. In practice, we do not compute gradient norms for each direction separately since they can be computed from the horizontal and vertical ones as

$$\mathbf{G}_\theta = \left(\cos\theta \frac{\partial \mathbf{I}}{\partial x} + \sin\theta \frac{\partial \mathbf{I}}{\partial y} \right)^+ . \tag{3}$$

Therefore, for horizontal and vertical gradients, we perform two $1D$ convolutions with kernels $[1, \ -1]$ and $[1, \ -1]^T$ respectively requiring $2P$ additions to calculate in both directions. Orientation layers are, then, computed from these according to Equation 3 with $2P$ multiplications and P additions for each layer. Then for each radius quantization level, Q, we perform H convolutions. This is done again as two successive $1D$ convolutions instead of a single $2D$ one thanks to the separability of Gaussian kernels.

Given those gradients, we sample the convolved orientation layers at $Q \times T + 1$ locations for every pixel. For orientations other than 0, an additional shifting operation is required to account for it.

Sampling can be performed by either interpolation or by rounding point locations to the nearest integer. We found that using either method returns roughly equivalent results. Nevertheless, we include both options in the source code we supply [30].

To summarize, computing all the descriptors of an image requires $2H \times Q + 1$ $1D$ convolutions, $P \times (Q \times T + 1)$ samplings, $2P \times H$ multiplications and $P \times H$ additions.

III-B.2 Computing SIFT: To compute a SIFT descriptor, the image gradient magnitudes and orientations are sampled around the point location at an appropriate scale. Because of the non-circularly symmetric support region and the kernel employed in SIFT, the sampling has to be done at all the

sample locations within the descriptor region. Let us, therefore, consider the computation of a single descriptor at a single scale where the descriptor is computed over a W_s sample array with S_s histograms of H_s bins.

As DAISY, SIFT requires image gradients that are computed in the same way and then sampled within the descriptor region at W_s locations. The gradients are then Gaussian smoothed and histograms are formed using trilinear interpolation, that is, each bin is multiplied by a weight of $1-d$ where d is the distance of the sample from the central value of the bin. The smoothing and interpolation takes $4W_s$ multiplications. Finally, each sample is assigned to a bin and accumulated which requires W_s multiplications and $S_s \times (\frac{2W_s}{S_s}) = 2W_s$ summations.

To summarize, the computation of one SIFT descriptor requires W_s samplings, $5W_s$ multiplications, and $2W_s$ summations plus the initial convolution required for gradient computation.

III-B.3 Comparing DAISY and SIFT: For comparison purposes, assume that convolving of a $1D$ kernel of length N can be done with N multiplications and $N-1$ summations per pixel. Then, DAISY requires $2H \times Q \times N + 2$ multiplications, $2H \times Q \times N - 1$ summations and S samplings per pixel.

If we insert the parameters $W_s = 16 \times 16$, $S_s = 4 \times 4$ and $H_s = 8$ for SIFT as reported in [19] and the parameters we used in this paper for DAISY, $H = 8$, $Q = 3$, and $S = 25$ with an average $N = 5$, we see that SIFT requires 1280 multiplications, 512 summations and 256 samplings per pixel whereas DAISY requires 122 multiplications, 119 summations and 25 samplings.

Note that, computing the descriptor in a different orientation also requires an additional shifting operation in DAISY with $S \times H$ shifts per pixel whereas this is handled in SIFT during the sampling phase of the gradients with W_s additions.

In any event, a direct comparison of these numbers might be somewhat misleading as one can approximate some of the operations in SIFT in a dense implementation to increase computational efficiency. One might also consider a more efficient implementation of the convolution operation using FFT, which would boost DAISY's performance. The most important speed-up difference, however, is due to two facts. Firstly, DAISY descriptors share histograms so that once a histogram is computed for one pixel, it is not computed again for the T other descriptors surrounding this histogram location. Secondly, the computation pipeline enables a very efficient memory access pattern and the early separation of the histogram layers greatly improves efficiency. This also allows DAISY to be parallelized very easily and our current implementation [30], allows the use of multiple cores using the OpenMP^{TM} library. In Fig. 7 we show the time required to compute DAISY descriptors with varying number of cores on different image sizes. Computation time falls almost linearly with the number of the cores used. The algorithm is

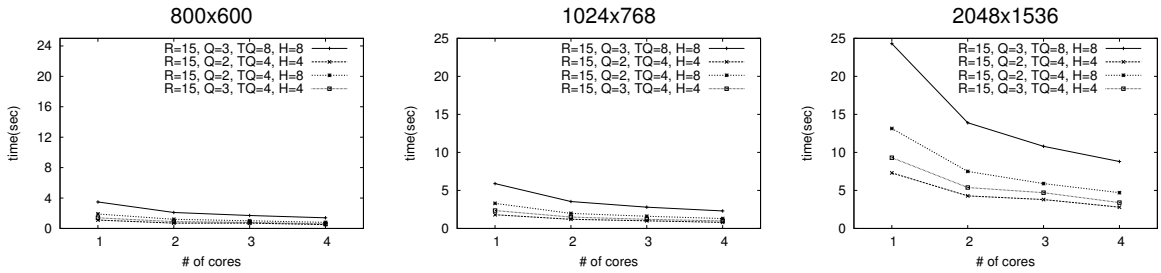


Figure 7. **DAISY Computation Times.** Computation times of the DAISY descriptor for all the pixels of an image with various settings on 3 different sized images. We present the change of the computation time with respect to the number of cores used in parallel.

also quite suitable for GPU programming which we will pursue in future.

In terms of memory, the precomputed convolved orientation layers require $4Q \times H \times P$ bits and if we also want to precompute all the descriptors of an image, this will require $4\mathcal{D}_s \times P$ bits. In example, for an image of size 1024×1024, these equal to $96MB$'s and $800MB$'s respectively for the standard parameter set of $R = 15$, $Q = 3$, $T = 8$ and $H = 8$.

IV. Depth Map Estimation

To perform dense matching, we use DAISY to measure similarities across images as shown by Fig. 8. We then feed these measures to a standard graph-cut-based reconstruction algorithm [8]. To properly handle occlusions, we incorporate an occlusion map, which is the counterpart of the visibility maps in other reconstruction algorithms [16]. We do this by introducing an occlusion node with a constant cost in the graph structure. The value of this cost has a significant impact on the proportion of pixels that are labeled as occluded. As will be discussed in Section V, we use one image data set to set it to a reasonable value and retain this value for all other experiments shown in this paper. The depth and occlusion maps are estimated by Expectation Maximization (EM), which we formalize below.

We compute the descriptor of every point from its neighbourhood. However, for pixels that are close to an occluding boundary, part of the neighbourhoods, thereby part of the descriptors, will be different when captured from different viewpoints. To handle this, we exploit the occlusion map and define *binary masks* over our descriptors. We introduce predefined masks that enforce the spatial coherence of the occlusion map, and show that they allow for proper handling of occlusions.

In practice, we assume that we are given at least 2 calibrated gray-scale images and we compute the dense depth map of the scene with respect to a particular viewpoint which can either be equal to

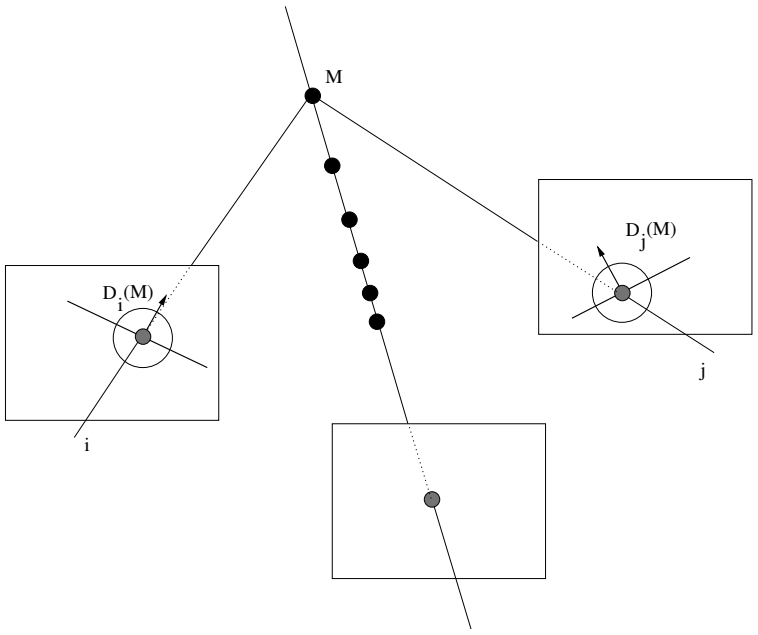


Figure 8. **Problem Definition.** The inverse depth is discretized uniformly and the matching score is computed using neighbourhoods centered around the projected locations. These neighbourhoods are rotated to conform to the orientation of the epipolar lines.

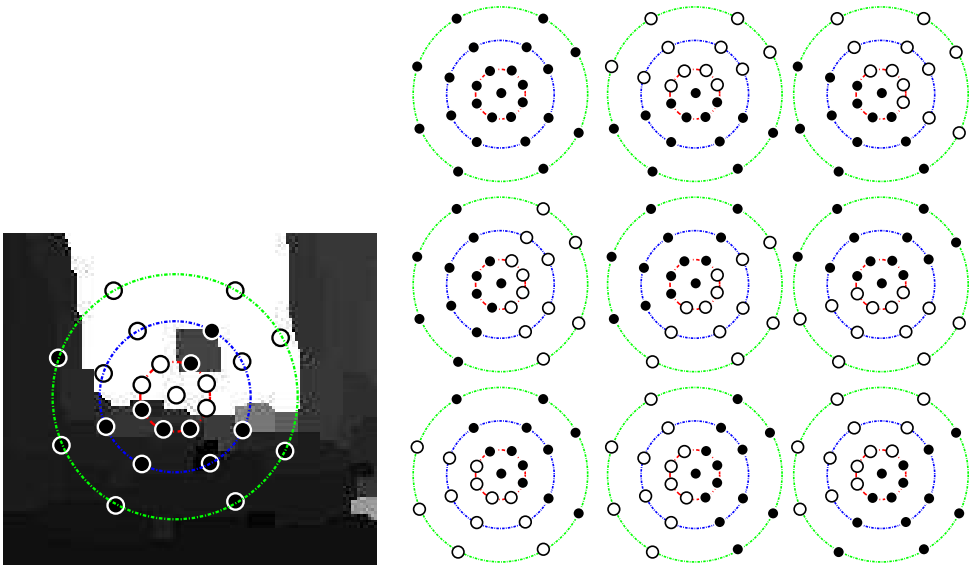
one of the input view points or it can be a completely different virtual position. We use the calibration information to discretize the 3–D space and compute descriptors that take into account the orientation of the epipolar lines, as shown in Fig. 8. In this way, we do not require a rotationally invariant descriptor and take advantage of the fact that DAISY descriptor is very easy to rotate, as described in the previous section.

A. Formalization

Given a set of N calibrated images of the scene, we denote their descriptors by $\mathbf{D}_{1:N}$. We estimate the dense depth map $\mathbf{Z}$ for a given viewpoint by maximizing:

$$p(\mathbf{Z}, \mathbf{O} \mid \mathbf{D}_{1:N}) \propto p(\mathbf{D}_{1:N} \mid \mathbf{Z}, \mathbf{O}) p(\mathbf{Z}, \mathbf{O}) \tag{4}$$

where we also introduced an occlusion map term $\mathbf{O}$ that will be exploited below to estimate the similarities between image locations. As in [8], we enforce piecewise smoothness on the depth map as well as our occlusion map by penalizing nearby different labels using the Potts model, i.e. $V = \delta(q_i \neq q_j)$ with q_i and q_j are labels of nearby pixels.



(a) (b)

Figure 9. **Binary masks for occlusion handling.** We use binary masks over the descriptors to estimate location similarities even near occlusion boundaries. In this figure, a black disk with a white circumference corresponds to "on" and a white disk to "off". (a) We use the occlusion map to define the masks; and in (b) predefined masks make it easy to enforce spatial coherence and to speed-up the convergence of EM estimation.

For the data driven posterior, we also assume independence between pixel locations:

$$p(\mathbf{D}_{1:N} \mid \mathbf{Z}, \mathbf{O}) = \prod_{\mathbf{x}} p\left(\mathbf{D}_{1:N}(\mathbf{x}) \;\middle|\; \mathbf{Z}, \mathbf{O}\right) . \tag{5}$$

Each term $p\left(\mathbf{D}_{1:N}(\mathbf{x}) \mid \mathbf{Z}, \mathbf{O}\right)$ of Eq. 5 is estimated using our descriptor. Because the descriptor considers relatively large regions, we introduce binary masks computed from the occlusion map $\mathbf{O}$, as explained in the next section, to avoid including occluded parts into our similarity score.

B. Using Masks over the Descriptor

Given the descriptors, we can take the $p\left(\mathbf{D}_{1:N}(\mathbf{x}) \mid \mathbf{Z}, \mathbf{O}\right)$ probability to be inversely related to the dissimilarity function

$$\mathfrak{D}'\Big(\mathbf{D}_i(\mathbf{X}), \mathbf{D}_j(\mathbf{X})\Big) = \frac{1}{S}\sum_{k=1}^{S}\left\|\mathbf{D}_i^{[k]}(\mathbf{x}) - \mathbf{D}_j^{[k]}(\mathbf{x})\right\|_2 , \tag{6}$$

where $\mathbf{D}_i(\mathbf{X})$ and $\mathbf{D}_j(\mathbf{X})$ are the descriptors at locations obtained by projecting the 3–D point $\mathbf{X}$ (defined by location $\mathbf{x}$ and its depth $\mathbf{Z}(\mathbf{x})$ in the virtual view) onto image i and j, $\mathbf{D}_i^{[k]}(\mathbf{x})$ is the k^{th} histogram $\widetilde{\mathbf{h}}$ in $\mathbf{D}_i(\mathbf{x})$ and S is the number of histograms used in the descriptor.

However, as discussed above, we should account for occlusions. The descriptor is computed over image patches and the formulation of Eq. 6 is not robust to partial occlusions. Even for a good match, if the pixel is close to an occlusion boundary, the histograms of the occluded parts have no reason to resemble each other.

We, therefore, introduce binary masks $\{\mathcal{M}_m(\mathbf{x})\}$ such as the ones depicted in Fig. 9, which allow DAISY to take into account only the visible parts when computing the distances between descriptors. The mask length is equal to the number of histograms used in the descriptor, the S of Table I. We use these masks to rewrite the $\mathfrak{D}'$ of Eq. 6 as

$$\mathfrak{D} = \frac{1}{\sum_{q=1}^{S} \mathcal{M}^{[q]}} \sum_{k=1}^{S} \mathcal{M}^{[k]} \left\| \mathbf{D}_i^{[k]}(\mathbf{x}) - \mathbf{D}_j^{[k]}(\mathbf{x}) \right\|_2 , \tag{7}$$

where $\mathcal{M}^{[k]}$ is the k^{th} element of the binary mask $\mathcal{M}$. Following [8], we define the $p\left(\mathbf{D}_{1:N}(\mathbf{x}) \mid \mathbf{Z}, \mathbf{O}\right)$ term of Eq. 5 as a Laplacian distribution $Lap\left(\mathfrak{D}(\mathbf{D}_{1:N}(\mathbf{x}) \mid \mathbf{Z}, \mathbf{O}); 0, \lambda_m\right)$ with our occlusion handling dissimilarity function.

To select the most likely mask at each pixel location, we rely on an EM algorithm and tried three different strategies:

- The simplest one depicted by Fig. 9-a involves disabling the histograms that are marked as occluded in the current estimate of the occlusion map $\mathbf{O}$ and obtaining a single binary mask $\mathcal{M}_m(\mathbf{x})$.
- A more sophisticated one is to use the predefined masks depicted by Figure 9-b which have a high special coherence. The probability of each mask is computed such that the masks that have large visible areas with similar depth values are favored. We write

  $$p(\mathcal{M}_m(\mathbf{x})|\mathbf{Z}, \mathbf{O}) = \frac{1}{Y}\Big(\overline{v}_m + \frac{1}{\sigma_m^2(\mathbf{Z}) + 1}\Big), \tag{8}$$

  where $\overline{v}_m$ is the average visible pixel number, $\sigma_m(\mathbf{Z})$ is the depth variance within the mask region, and Y is the normalization term that is equal to the sum of all mask probabilities.

  Then, we take the data posterior to be the weighted sum of individual mask responses as

  $$p\left(\mathbf{D}_{1:N}(\mathbf{x})|\mathbf{Z}, \mathbf{O}\right) = \sum_m p\left(\mathbf{D}_{1:N}(\mathbf{x})|\mathbf{Z}, \mathbf{O}, \mathcal{M}_m(\mathbf{x})\right) p\left(\mathcal{M}_m(\mathbf{x})|\mathbf{Z}, \mathbf{O}\right) . \tag{9}$$

- The third strategy is a simplified version of the second one, where we only use the result of the most probable mask instead of a mixture.

In the first and third strategy, we use only one mask. By contrast, the second strategy involves a mixture computed from several masks. Note that, the mask probabilities are re-estimated at each step of

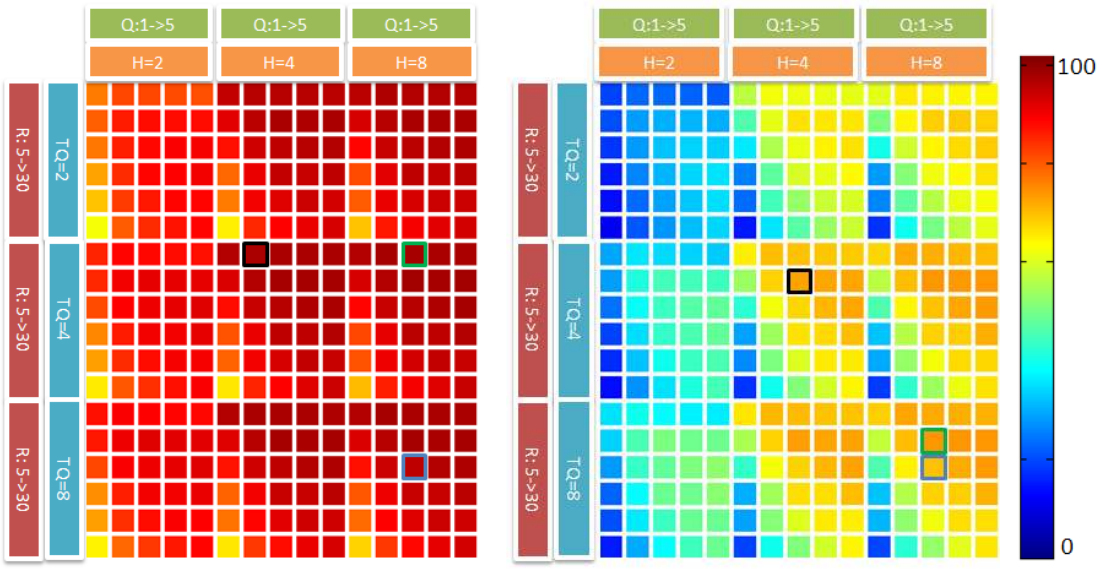


Figure 10. **Parameter sweep test for narrow and wide baseline cases.** As described by Table I, there are 4 parameters that specify the shape and size of DAISY: radius(R), radius quantization(Q), angular quantization(T), and number of bins of the histogram (H). The above figures depict the results of a 4D sweep of these parameters and the color of each square represents the percentage of depths that are correctly estimated. The value associated to a color is given by the color scale on the right. To assess the correctness of an estimated depth, we used laser scanned depth maps and assumed an estimate as correct if the estimate error is within 1% of the scene's depth range. **Left:** The averaged result for 5 narrow baseline image pairs of the Fountain sequence of Fig. 11. The green rectangle denotes the best parameter set for this configuration which is $R = 5, Q = 3, T = 4, H = 8$, resulting in a descriptor of size 104 with a 81.2% correct depth estimates. However, upon closer inspection, we see that many other configurations produce similar results ($80^{+}\%$). Among these, the configuration $R = 5, Q = 2, T = 4, H = 4$ produces the shortest descriptor length of 36. It is denoted by the black rectangle. **Right:** The averaged result for 3 wide baseline image pairs of the Fountain sequence. The best result, again denoted by a green rectangle, 73% correct depth estimate is achieved with $R = 10, Q = 3, T = 8, H = 8$ which yields a 200 length descriptor. However, as in the narrow baseline case, there are many other configurations that produce a very similar performance ($71^{+}\%$) with shorter descriptor sizes. The shortest one (black rectangle) is $R = 10, Q = 3, T = 4, H = 4$ with 52 length. Having multiple configurations that result in similarly high performance shows that we don't really need to change our descriptor parameters depending on the baseline to improve performance, but we can change to meet speed or memory requirements. The configuration we used ($R = 15, Q = 3, T = 8, H = 8$) in all the other experiments presented in this paper is outlined in blue.

the EM algorithm. In our experiments, the second and third strategies always performed better than the first, mainly because they enforce spatial smoothness. The second strategy, however, is computationally more expensive than the third without any perceptible improvement in performance. Therefore, we use only the third strategy in the remainder of the paper. We generally run our EM algorithm for only 2-3 iterations, which results in better occlusion estimates around occlusion boundaries.

V. Experiments and Results

In this section, we present various experiments we performed in order to measure the performance of DAISY. In Section V-A, we present results of a parameter sweep experiment we performed to understand and optimize the DAISY parameters with respect to the baseline. We, then, compare DAISY against other descriptors for depth estimation purposes. In Section V-C, we pushed the baseline to very large values to explore the range within which DAISY yields acceptable depth accuracy and the quality of our occlusion estimates. Then finally in Section V-D, we test our approach on image pairs with various photometric and geometric transformations and show that it is robust to these and compared our reconstructions with that of a state-of-the-art *multi-view* algorithm [26].

A. Parameter Selection

To understand the influence of the DAISY parameters of Table I, we performed 2 parameter sweep experiments, one in the narrow baseline case and the other in the wide baseline case. We used the data set depicted by Fig. 11, which includes laser scanned ground truth depth and occlusion maps as discussed in [28], [29]. For the narrow baseline case, we used the image pairs $\{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}\}$. For the wide baseline, we used $\{\{1,4\},\{2,5\},\{3,6\}\}$. This guarantees similar baselines within each one of the two groups.

Fig. 10 depicts the results. Correct depth estimates of $80^{+}\%$ can be achieved using a less complex descriptor for short baseline. However, as the baseline increases, Fig. 10 suggests that, it becomes necessary to use a more complex descriptor at the expense of increased computation and matching time.

Most of the time devoted to descriptor computation is spent on convolutions. It can be reduced by using a smaller number of bins in the histogram (H) or by using a smaller number of layers (Q). We can use $H = 4$ with a little performance loss but the layer number Q should be chosen carefully depending on the baseline. It appears that 2 or 3 layers give similar responses. As far as T, the discretization of the angular space, is concerned, 4 or 8 levels perform similarly for both narrow and wide baseline cases. However, when increasing the baseline, the radius R should also be increased, but only up to a point. Going beyond this point causes a loss of discriminative power and a performance drop, especially in the wide baseline case.

One might argue that there is no need to use more than 4 bins in the histograms as one could generate the in-between responses of a gradient from the horizontal and vertical directions only. However, this is not the case when summing over a group of pixels, because aggregating the low resolution responses will

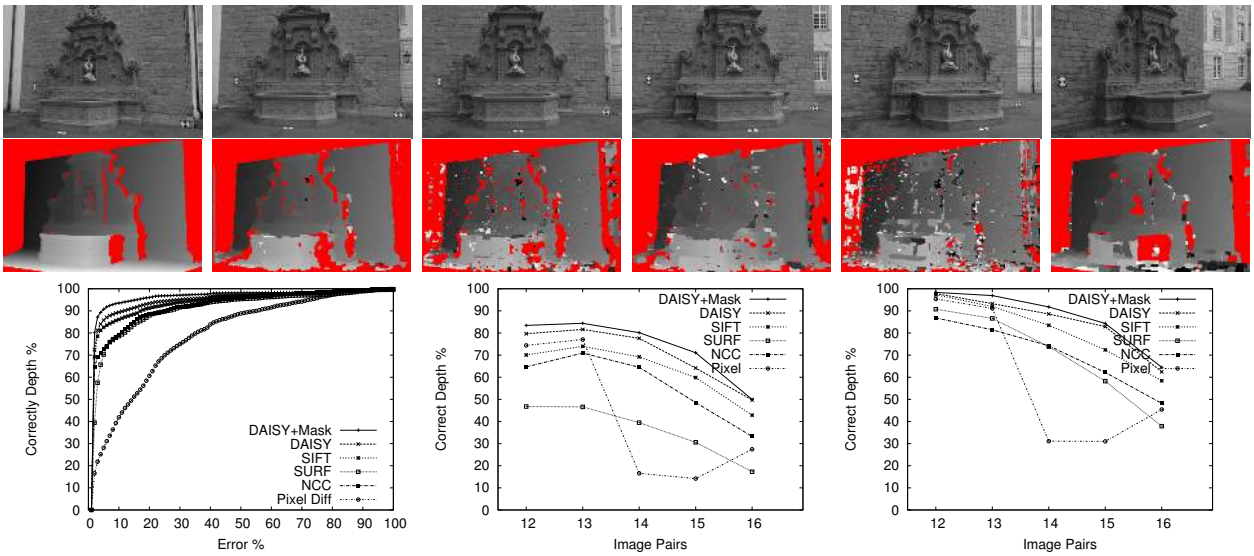


Figure 11. **Comparing different descriptors: Fountain Sequence [28] First row.** In our tests, we match the left-most image against each one of the other five. **Second row.** The laser-scan depth-map we use as a reference and five depth-maps computed from the first and third images. From left to right, we used DAISY, SIFT, SURF, NCC and Pixel Difference. **Third row.** The leftmost plot shows the corresponding distributions of deviations from the laser-scan data, expressed as a fraction of the scene's depth-range. The other plots summarize these distributions for the five stereo pairs of increasing baseline with discrete error thresholds set to be 1% and 5% of the scene's depth range, respectively. Each data point represents a pair where the baseline increases gradually from left to right and individual curves correspond to DAISY with masks, DAISY without masks, SIFT, SURF, NCC, and Pixel Difference. In all cases DAISY does better than the others and using masks further improves the results.

lose the gradient distribution information of individual pixels and computing a higher resolution version of the histogram from the low resolution one will not be equal to summing individual high resolution responses. This is why increasing the histogram resolution makes the descriptor more distinctive at the cost of some computational overhead.

Although descriptor parameters could be adapted depending on the baseline, scene complexity and texturedness, it is difficult to automate this process. The purpose of this experiment was to see whether there exists a set of parameters that clearly outperform other parameter sets. However, the experimental results suggest that the descriptor is relatively insensitive to parameter choice for an extended range for both narrow and wide baseline image pairs: different parameter sets produce similar results. Looking at this experiment, we can conclude about 3 of the 4 parameters of the descriptor, namely the radius quantization (Q), angular quantization (T) and histogram quantization (H). However, the effect of the size of the descriptor radius (R) is not so clear since a relatively textured scene is used in the experiment and

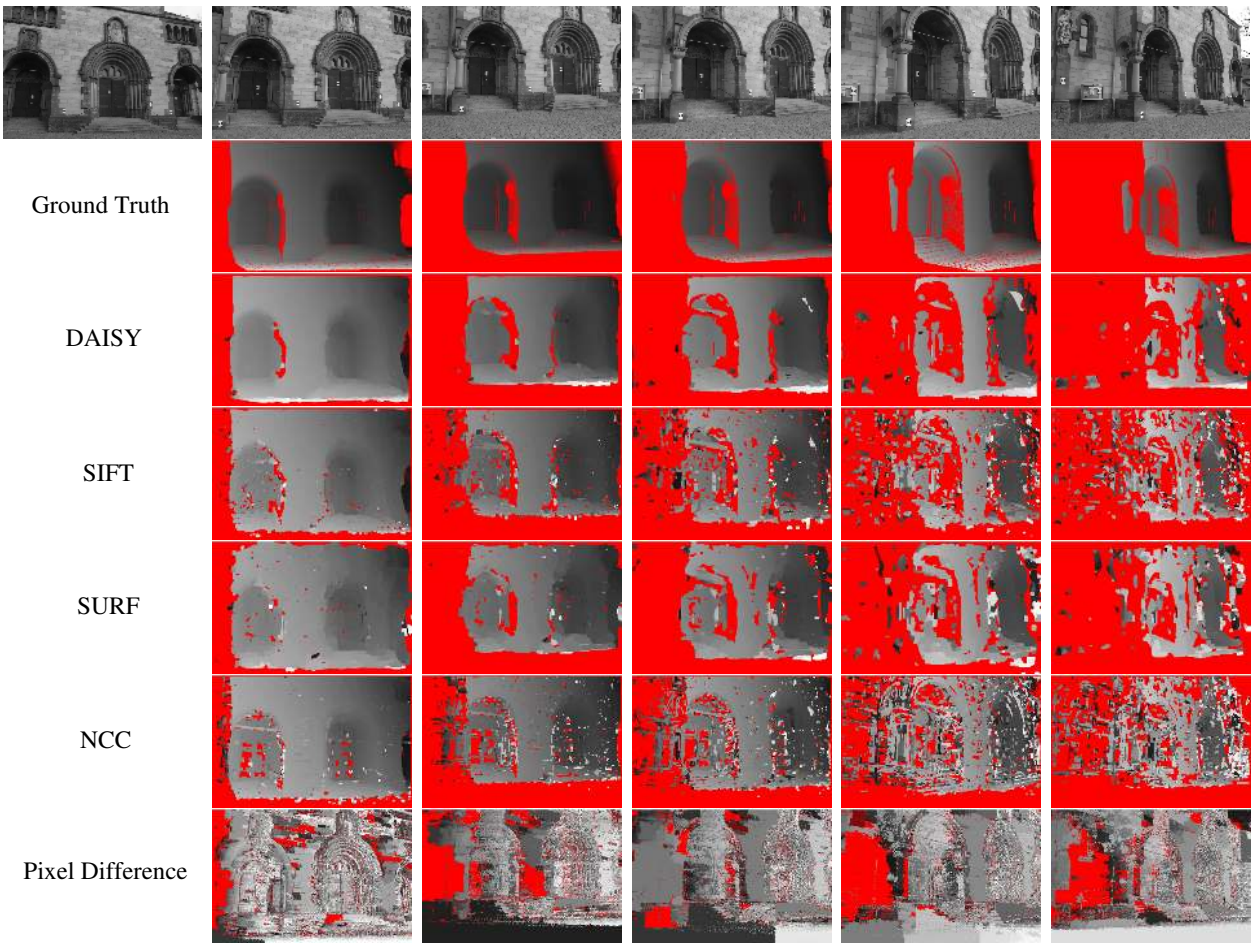


Figure 12. **Comparing different descriptors: HerzJesu Sequence [28].** As in the Fountain sequence of Fig. 11, we use different descriptors to match the left-most images with each one of the other images. We compute the depth map in the reference frame of these. **Second row**: Ground truth depth maps with overlaid occlusion masks. **Remaining rows:** Depth maps computed using DAISY, SIFT, SURF, NCC and Pixel differencing in that order.

we believe the effect of R will be more apparent for less textured scenes. Although we do not have a data set with a ground truth depth map of such a scene and therefore can not accurately quantify this effect, we have observed that using a larger R is beneficial for less textured scenes from other data sets used in this paper. Hence, in practice, we use the most generic parameter set $R = 15, Q = 3, T = 8, H = 8$ for all the experiments presented in this paper. Admittedly, this produces a longer descriptor than strictly necessary but it performs well for both narrow and wide baselines. However, depending on the application DAISY is used for, the parameters can be set accordingly. For example, if it is known that input images have a short baseline and scene is more or less textured, $R = 10, Q = 3, T = 4, H = 4$ will produce

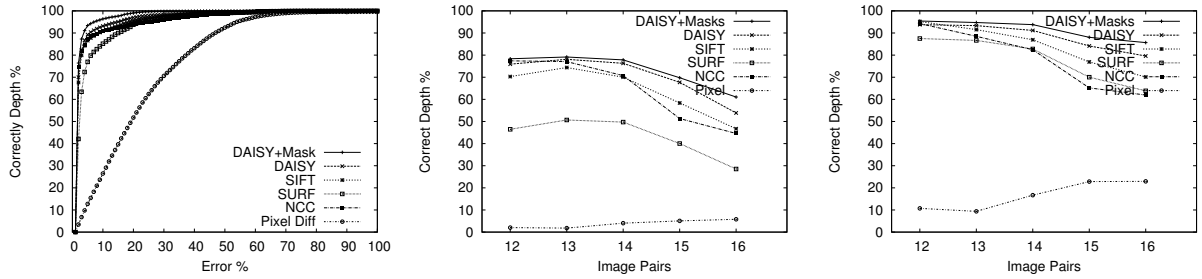


Figure 13. **Quantitative comparison results for Fig. 12.** The **leftmost** plot shows the corresponding distributions of deviations from the laser-scan data, expressed as a fraction of the scene's depth-range. The **other plots** summarize these distributions for the five stereo pairs of increasing baseline with discrete error thresholds set to be 1% and 5% of the scene's depth range, respectively. Each data point represents a pair where the baseline increases gradually from left to right and individual curves correspond to DAISY with masks, DAISY without masks, SIFT, SURF, NCC, and Pixel Difference. In all cases DAISY does better than the others and using masks further improves the result.

good results with a shorter footprint of length 52.

B. Comparison with other Descriptors

To compare DAISY's performance with that of the other descriptors we used again the data set of Fig. 11 and also the data set of Fig. 12, for which laser scanner data is available as well. Arguably our results on the data of Fig. 11 should be treated with caution since we have used these images to set our parameters. However we have done no such thing with the data of Fig. 12 and obtain very similar results.

We used DAISY with occlusion masks, DAISY, SIFT, SURF, NCC and Pixel differencing to densely compute matching scores. They are then all handled similarly, as described in Section IV, to produce depth maps. The only difference is that we do not use binary masks to modify matching scores for descriptors other than DAISY. All the region based descriptors are computed perpendicular to the epipolar lines, SURF and SIFT descriptors are 128-length vectors and NCC is 11×11.

For the data set of Fig. 11, the left-most image in the first row is matched against each one of the other five, which implies a wider and wider baseline. The second row depicts the laser scanner data on the left and the depth maps computed from the first and third images using DAISY, SIFT, SURF, NCC and pixel differencing. The third row summarizes the comparison of different descriptors against DAISY. The leftmost graph shows the result for the first and third image pairs by plotting the percentage of correctly estimated depths against the amount of allowed error which is represented as a fraction of scene's depth range and remaining graphs summarize these curves for all image pairs at discrete error levels. In one

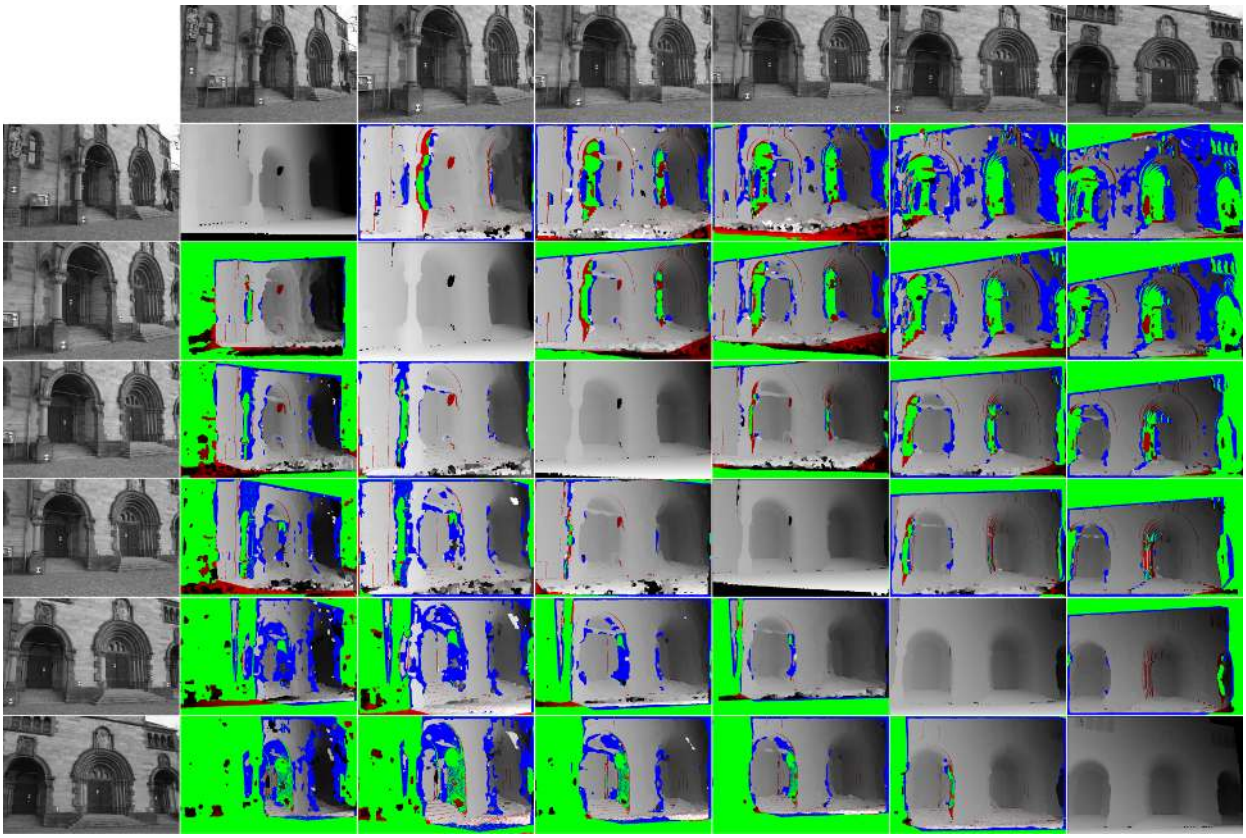


Figure 14. **Herz-Jesu Grid.** By using two images, one from the left-most column and one from the upper row, we compute depth and occlusion maps from the view point of the row image. In the diagonal, we display the ground truth depth maps. We marked the correctly detected occlusions with green, incorrectly detected ones with blue and the missed ones with red. From this figure, it is apparent that DAISY can handle quite large baselines without losing too much from its accuracy as can be seen from Table III.

case the depths are considered to be correct if they are within 1% of the scene's depth range and in the other within 5%. We present results using DAISY with and without using the occlusion masks. DAISY by itself outperforms the other descriptors and the masks provide a further boost.

For data set of Fig. 12, we match the left-most image with each one of the other first row images in turn and compute the depth map with respect to the latter image. We display the ground truth maps in the second row and show the estimated depth maps for different descriptors in the remaining rows. Fig. 13 depicts the quantitative results for this data set as in the previous data set.

Both of the data sets shows that DAISY performs better than all of the other descriptors. Note that although the results of SIFT and DAISY are close for the 5% threshold, DAISY finds substantially more

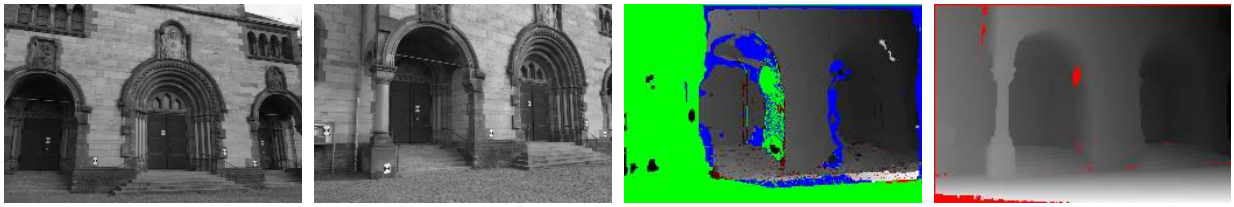


Figure 15. **Herzjesu Close-up.** Depth map is computed from the 2^{nd} image's point of view using DAISY. Correctly detected occlusions are shown in green, incorrectly detected ones with blue and the missed ones with red. The last image is the laser scanned ground truth depth map.

correct depths for the 1% threshold. This indicates that the depths found using DAISY are more accurate than those found using SIFT.

C. Occlusion Handling

We tested the performance of our occlusion detection scheme with the extended version of the HerzJesu sequence of Fig. 12 depicted by Fig. 14. The matching is done using two images, one from the first row and one from the first column, and the depth map is shown in the referential of the second image. The resulting depth map is displayed on the intersection of the respective column and row together with the ground truth on the diagonal. We use different colors to highlight correctly estimated occlusions, missed occlusions, and falsely labeled occlusions. An example pair of images from this sequence is shown in Fig. 15. Table III gives the percentage of the correctly estimated depths in visible areas where the correctness threshold is set to 5% of the scene's depth range.

As discussed in Section IV, the value of the occlusion cost in the graph structure has a direct influence on how many pixels are labeled as occluded. In Figure 16, we plot ROC curves obtained by using this for all the image pairs of Fig. 14. Here, each data point represents the result with a different occlusion cost, true positive rate shows how many percent of the visible areas we detect as visible and false positive rate shows the amount of missed occlusions. By using these plots, we picked a single value, 25% *of the maximum cost*, for the occlusion cost for all the results shown in this paper.

To show the effect of the baseline on the performance, we plot the area under the curve (AUC) of these ROC curves with respect to the angle between the cameras for all the image pairs of Figure 14. This curve shows that our approach works for a wide variety of camera configurations and is robust up to $30^\circ - 40^\circ$ changes. We also show two example ROC curves for narrow baseline and wide baseline cases in the same figure.

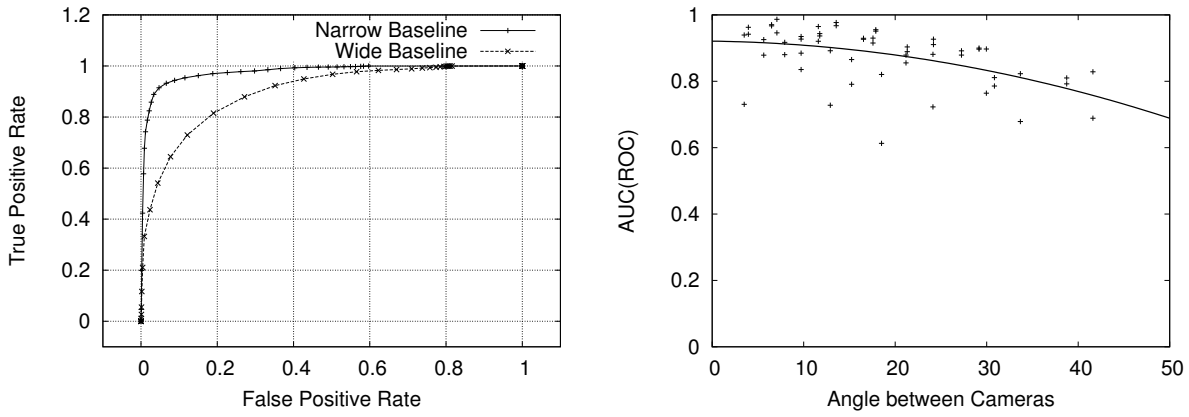


Figure 16. **ROC for occlusion threshold.** We plot ROC curves for the selection of the occlusion threshold for all the image pairs of Fig. 14. On the **left**, we show the ROC curves of the narrow baseline image pair $\{4, 5\}$ and the wide baseline image pair $\{1, 5\}$. On the **right**, we plot the area under the curve (AUC) of the ROC graphs of the image pairs with respect to the angle between the principal axes of the cameras. The result of each such pair is represented with a data point and the curve shows the fitted line to these.

Frame Number	1	2	3	4	5	6
1	-	90.8	88.7	86.9	87.5	90.1
2	86.9	-	94.3	92.4	91.4	92.9
3	85.5	90.3	-	93.2	93.9	96.4
4	83.6	87.1	93.0	-	95.4	96.8
5	83.9	86.3	91.7	93.3	-	97.7
6	86.4	89.2	94.5	95.6	96.7	-

Table III

CORRECTLY ESTIMATED DEPTH PERCENTAGE FOR FIG. 14

The progress of our EM based occlusion detection algorithm can be seen in Fig. 17. In this figure, we give an example for the evolution of the depth map with occlusion estimates at each iteration. The initial estimate is quickly improved in the next iteration with occlusions receding and new depths being estimated for these previously occluded-marked regions. We see that further iterations do not improve the result significantly and in practice we stop the process after one or two iterations.

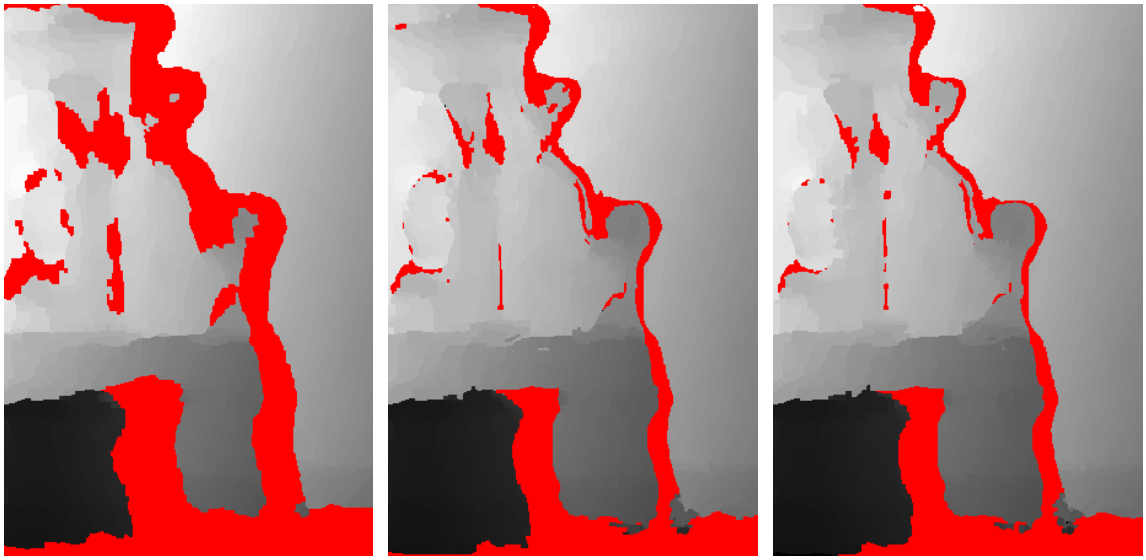


Figure 17. **Evolution of the occlusions during EM.** The three images show the evolution of the occlusion estimate during the iterations of the EM for Fig. 11 images. The initial solution (leftmost image) is quickly improved even after a single iteration (middle image) and does not change much thereafter (rightmost image).

D. Robustness to Image Transformations

Although we designed DAISY with only wide baseline conditions in mind, it exhibits the same robustness as histogram based descriptors to changes in contrast (Fig. 1), scale (Fig. 2), image quality (Fig. 3), viewpoint (Figs. 4 and 18) and brightness (Fig. 19). In these figures, we also present the depth maps obtained using NCC and SIFT which include more artifacts than ours. This result is noteworthy because, although it is a well known fact that histogram based descriptors are robust against these transforms at *feature point locations*, these experiments show that such robustness can also be found at many other point locations.

To compare our method to one of the best current techniques [26], we ran our algorithm on two sets of image pairs that were used in that paper, the Rathaus sequence of Fig. 20 and the Brussels sequence of Fig. 21. But instead of using the original 3072×2048 images, whose resolution is high enough for apparently blank areas to exhibit usable texture, we used 768×512 images in which this is not true. DAISY nevertheless achieved visually similar results.

Fig. 21 also highlights the effectiveness of our occlusion handling. When using only two images, the parts of the church that are hidden by people in one image and not in the other are correctly detected as occluded. When using three images, the algorithm returns an almost full depth map that lets us erase

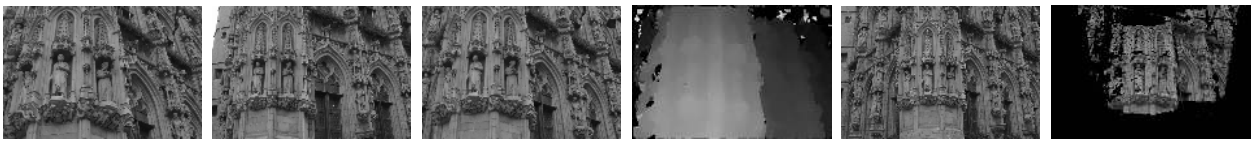
Figure 18. **Valencia Cathedral.** The reconstruction results of the exterior of the Valencia Cathedral from two very different viewpoints. Depth map is computed from the 2^{nd} image's point of view.

Figure 19. **Brightness Change.** We compute the depth map from the 2^{nd} image's point of view using DAISY. There is a brightness change between the 2 images.

the people in the synthetic images we produce.

VI. Conclusion

In this paper, we introduced DAISY, a new local descriptor, which is inspired from earlier ones such as SIFT and GLOH but can be computed much more efficiently for dense matching purposes. Speed increase comes from replacing weighted sums used by the earlier descriptors by sums of convolutions, which can be computed very quickly and from using a circularly symmetrical weighting kernel. The experiments suggest that although pixel differencing or correlation is good for short baseline stereo, wide



(a) (b) (c) (d) (e) (f)

Figure 20. **Results on low-resolution versions of the Rathaus images [27]**. (a,b,c) Three input images of size 768 $\times$ 512 instead of the 3072 $\times$ 2048 versions that were used in [26]. (d) Depth map computed using all three images. (e) A fourth image not used for reconstruction. (f) Image synthesized using the depth map and the image texture in (a) with respect to the view point of (e). Note how similar it is to (e). The holes are caused by the fact that a lot of the texture in (e) is not visible in (a).

(a) (b) (c) (d) (e) (f) (g)

Figure 21. **Low-resolution versions of the Brussels images [26].** (a,b,c) Three 768$\times$510 versions of the original 2048$\times$1360 images. (d,e) The depth-map computed using images (a) and (b) seen in the perspective of image (c) and the corresponding re-synthesized image. Note that the locations where there are people in one image and not in the other are correctly marked as occlusions. (f,g) The depth-map and synthetic image generated using all three images. Note that the previously occluded areas are now filled and that the people have been erased from the synthetic image.

baseline requires a more advanced measure for comparison. We showed DAISY to be very effective for this purpose.

Our method gives good results, even when using small images for stereo reconstruction. This means that we could use our algorithm to process video streams whose resolution is often lower than that of still images. When dealing with slanted surfaces and foreshortening, these results could be further improved by explicitly taking into account 3D surface orientation and warping the DAISY grid accordingly, which would not involve any significant computational overhead. This would fit naturally in a warp stereo approach [23] in which we would begin with unwarped detectors to compute a first surface estimate, use the corresponding orientations to warp the detectors, and iterate.

Computing our descriptor primarily involves performing Gaussian convolutions, which are amenable to hardware exportation or GPU implementation. This could lead to real-time, or even faster, computation

of the descriptor for all image pixels. This could have implications beyond stereo reconstruction because dense computation of image descriptors is fast becoming an important technique in other fields, such as object recognition [7], [18]. To encourage such developments, a C++ and MATLAB^{TM} implementation of DAISY is available for download from our webpage [30].

REFERENCES

[1] L. Alvarez, R. Deriche, J. Weickert, J., and Sanchez. Dense Disparity Map Estimation Respecting Image Discontinuities: A PDE and Scale-Space Based Approach. *Journal of Visual Communication and Image Representation*, 13(1/2):3–21, Mar 2002.

[2] N. Ayache and F. Lustman. Fast and Reliable Passive Trinocular Stereovision. In *International Conference on Computer Vision*, June 1987.

[3] H.H. Baker and T.O. Binford. Depth from edge and intensity based stereo. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 631–636, August 1981.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.

[5] A.C. Berg and J. Malik. Geometric Blur for Template Matching. In *Conference on Computer Vision and Pattern Recognition*, pages 607–614, 2001.

[6] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, April 1998.

[7] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *Proceedings of the European Conference on Computer Vision*, 2006.

[8] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 2001.

[9] M.Z. Brown, D. Burschka, and G.D. Hager. Advacnes in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008, August 2003.

[10] F. Devernay and O. D. Faugeras. Computing Differential Properties of 3–D Shapes from Stereoscopic Images without 3–D Models. In *Conference on Computer Vision and Pattern Recognition*, pages 208–213, Seattle, WA, June 1994.

[11] O.D. Faugeras and R. Keriven. Complete Dense Stereovision using Level Set Methods. In *European Conference on Computer Vision*, Freiburg, Germany, June 1998.

[12] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Conference on Computer Vision and Pattern Recognition*, 2005.

[13] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *International Journal of Computer Vision*, 14:211–226, 1995.

[14] S.S. Intille and A.F. Bobick. Disparity-space images and large occlusion stereo. In *European Conference on Computer Vision*, pages 179–186, May 1994.

[15] T. Kanade and M. Okutomi. A Stereo Matching Algorithm with an Adaptative Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, September 1994.

[16] V. Kolmogorov and R. Zabih. Multi-Camera Scene Reconstruction via Graph Cuts. In *European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[17] K.N. Kutulakos and S.M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3):197–216, July 2000.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[19] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[20] G. Medioni, C.K. Tang, and M.S. Lee. Tensor voting: Theory and applications. In *Reconnaissance des Formes et Intelligence in Artificielle*, 2000.

[21] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2004.

[22] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

[23] L.H. Quam. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, chapter Hierarchical Warp Stereo, pages 80–86. Morgan Kaufmann, 1987.

[24] S. Roy and I.J. Cox. A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem. In *International Conference on Computer Vision*, pages 492–499, Bombay, India, 1998.

[25] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002.

[26] C. Strecha, R. Fransens, and L. Van Gool. Combined Depth and Outlier Estimation in Multi-View Stereo. In *Conference on Computer Vision and Pattern Recognition*, 2006.

[27] C. Strecha, T. Tuytelaars, and L. Van Gool. Dense Matching of Multiple Wide-Baseline Views. In *International Conference on Computer Vision*, 2003.

[28] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In *Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, June 2008.

[29] C. Strecha. Multi-view evaluation-http://cvlab.epfl.ch/data, 2008.

[30] E. Tola. Daisy code. -http://cvlab.epfl.ch/software, 2008.

[31] T. Tuytelaars and L. Van Gool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *British Machine Vision Conference*, pages 412–422, 2000.

[32] S.A. Winder and M. Brown. Learning Local Image Descriptors. In *Conference on Computer Vision and Pattern Recognition*, Minneapolis, MI, June 2007.

[33] J. Yao and W.-K. Cham. 3–D Modeling and Rendering from Multiple Wide-Baseline Images. *Signal Processing: Image Communication*, 21:506–518, 2006.

PLACE
PHOTO
HERE

Engin Tola received his undergraduate degree in Electrical and Electronics Engineering from Middle East Technical University (METU) - Turkey in 2003. Afterwards, he completed his M.Sc. on Signal Processing in 2005 at METU and he is presently a PhD student at the Ecole Polytechnique Federale de Lausanne (EPFL) - Switzerland in the Computer Vision Laboratory. His research interests focus on point descriptors, scene reconstruction and virtual view synthesis.

PLACE
PHOTO
HERE

Vincent Lepetit received the engineering and master degrees in Computer Science from the ESIAL in 1996. He received the PhD degree in Computer Vision in 2001 from the University of Nancy, France, after working in the ISA INRIA team. He then joined the Virtual Reality Lab at EPFL (Swiss Federal Institute of Technology) as a post-doctoral fellow and became a founding member of the Computer Vision Laboratory. His research interests are feature point matching, 3D camera tracking, and object recognition.

PLACE
PHOTO
HERE

Pascal Fua received an engineering degree from Ecole Polytechnique, Paris, in 1984 and the Ph.D. degree in Computer Science from the University of Orsay in 1989. He joined EPFL (Swiss Federal Institute of Technology) in 1996 where he is now a Professor in the School of Computer and Communication Science. Before that, he worked at SRI International and at INRIA Sophia-Antipolis as a computer scientist. His research interests include shape modeling and motion recovery from images, human body modeling, and optimization-based techniques for image analysis and synthesis. He has (co)authored over 150 publications in refereed journals and conferences. He has been an associate editor of IEEE journal Transactions for Pattern Analysis and Machine Intelligence and has been a program committee member and area chair of several major vision conferences.