An Analysis of Errors in Graph-Based Keypoint Matching and Proposed Solutions

Toby Collins, Pablo Mesejo, and Adrien Bartoli

ALCoV-ISIT, UMR 6284 CNRS/Université d'Auvergne, Clermont-Ferrand, France

Abstract. An error occurs in graph-based keypoint matching when keypoints in two different images are matched by an algorithm but do not correspond to the same physical point. Most previous methods acquire keypoints in a black-box manner, and focus on developing better algorithms to match the provided points. However to study the complete performance of a matching system one has to study errors through the whole matching pipeline, from keypoint detection, candidate selection to graph optimisation. We show that in the full pipeline there are six different types of errors that cause mismatches. We then present a matching framework designed to reduce these errors. We achieve this by adapting keypoint detectors to better suit the needs of graph-based matching, and achieve better graph constraints by exploiting more information from their keypoints. Our framework is applicable in general images and can handle clutter and motion discontinuities. We also propose a method to identify many mismatches a posteriori based on Left-Right Consistency inspired by stereo matching due to the asymmetric way we detect keypoints and define the graph.

1 Introduction

Nonrigid keypoint-based image matching is the task of finding correspondences between keypoints detected in two images that are related by an unknown nonrigid transformation. This lies at the heart of several important computer vision problems and applications including nonrigid registration, object recognition and nonrigid 3D reconstruction. Graph-based methods solve this with a discrete optimisation on graphs whose edges encode geometric constraints between keypoints. This is NP hard in general and current research involves finding good approximate solutions [1–3] or better ways to learn the graph's parameters [4–6]. Most methods tend to treat the underlying keypoint detector as a black box, however to improve the overall performance of a graph matching system one should design or adapt the keypoint detector to also reduce errors and ease the matching problem. We show that there are six types of errors that cause mismatches which occur throughout the whole matching pipeline from keypoint detection to graph optimisation. We then give a general framework for reducing these errors and show this greatly improves the end performance. This includes a method for detecting mismatches a posteriori that is based on the Left-Right Consistency (LRC) test [7, 8] originating in stereo matching. Although simple, this has not

D. Fleet et al. (Eds.): ECCV 2014, Part VII, LNCS 8695, pp. 138–153, 2014.
© Springer International Publishing Switzerland 2014

been used in graph-matching before because many prior formulations are symmetric (*i.e.* if the roles of the two images are reversed the solution remains the same). We argue that, to reduce errors, an asymmetric approach (where keypoints sets in either image are significantly different) has many advantages, and it also permits using the LRC[1].

Previous Work. Graph-based keypoint matching is typically formulated as a quadratic binary assignment problem [2, 6, 9–16]. This can represent constraints on individual matches (*i.e.* unary terms) and pairs of matches (*i.e.* pairwise terms). Some works have incorporated higher-order constraints [13, 14] to handle the fact that pairwise constraints, such as preserving 2D Euclidean distances between points [6, 15] are not invariant to deformation and viewpoint change. However increasing the order to third and beyond increases exponentially the cost to compute and store the graph's terms. Thus second-order graphs remain the dominant approach. A limitation of many previous works is that they enforce all nodes in the graph to have a match. In some works this is done explicitly by using a permutation assignment matrix [9, 17]. In the others this is done implicitly because their cost function is always reduced by making a match [10–14, 16, 4]. Works that deal with unmatchable keypoints include [6, 18, 19, 15]. [6] allows keypoints to be assigned to an unmatchable state in a Markov Random Field (MRF)-based energy function. However reported results required ground-truth training images of the scene to learn the right weighting of the energy terms, which limits applicability. Unmatchable keypoints are found in [18, 15] by detecting those which have poor geometric compatibility with high-confidence matches. [15] requires a fully-connected graph and is very computationally demanding for large keypointsets. [18] iteratively computes correspondence subspace from high-confident matches and uses this as a spatial prior for other points. This was shown to work well for smooth deformable surfaces but may have difficulty with motion discontinuous or cluttered scenes. Other approaches are the *Hard Matching with Outlier Detection* (HMOD) methods [20–23]. These match keypoints using only their texture descriptors, and because many of these matches will be false, use a secondary stage of detecting mismatches by computing their agreement with a deformation model fitted from the matches that are considered correct.

2 Types of Errors in Graph-Based Keypoint Matching

We define $\mathcal{R} = \{r_1, ..., r_m\}$ and $\mathcal{T} = \{t_1, ..., t_n\}$ to be the keypoint-sets for the two images, which we refer to as the *reference* and *target* images respectively. Without loss of generality let $m \geq n$. The goal of keypoint matching is to find, if it exists, a correct correspondence for each member of $\mathcal{T}$ in $\mathcal{R}$ (Fig. 1). Because keypoint detectors localise keypoints with some margin of error, a soft criteria is required to distinguish correct from incorrect correspondences.This means that for any member of $\mathcal{T}$ there may exist multiple correct matches in $\mathcal{R}$. We define

[1] Source code is available at `http://isit.u-clermont1.fr/~ab/Research/`

$\mathcal{S}_i \subset \mathcal{R}$ to be all correct matches for a keypoint t_i. We define t_i to be a *matchable keypoint* if $\mathcal{S}_i$ is non-empty. We define t_i to be an *unmatchable keypoint* if $\mathcal{S}_i$ is empty. We define the *visible keypoint-set* $\mathcal{V} \subseteq \mathcal{T}$ to be all keypoints in $\mathcal{T}$ that are visible in the reference image (but are not necessarily in $\mathcal{R}$). In this paper we tackle problems where $\mathcal{T}$ and $\mathcal{R}$ are large unfiltered keypoint-sets generated by a keypoint detector such as SIFT [24] or SURF [25]. For typical images m and n will be $O(10^2 \rightarrow 10^5)$. To keep the costs of graph-based matching manageable in terms of *(i)* storage, *(ii)* time to compute the graph's constraints and *(iii)* time to find the solution, the possible members of $\mathcal{R}$ that t_i can be matched to should be pruned significantly. This has been done previously by taking the p members in $\mathcal{R}$ that have the most similar descriptors, with p being small (*e.g.* $O(10^1)$). We define a keypoint's *candidate match set* $\mathcal{C}(t_i) \subset \mathcal{R}$ to be the pruned set for t_i. We define the output of a matching system by the assignment vector $\mathbf{s} \in \{0, 1, ...m\}^n$, where $\mathbf{s}(i) \neq 0$ means that t_i is matched to $r_{\mathbf{s}(i)} \in \mathcal{C}(t_i)$, and $\mathbf{s}(i) = 0$ means that t_i is unmatched and in the *unmatchable state.* In all prior works in graph-based keypoint matching performance is evaluated by measuring the number of correct matches made between $\mathcal{T}$ and $\mathcal{R}$. This does not tell us the whole story for the complete performance of the system, nor does it provide a breakdown of where the error occurred. For a visible keypoint $t_i \in \mathcal{V}$ a matching system will fail to establish a correct correspondence in the reference image according to four types of errors (Fig. 1). These are as follows:

- *Detection Error*: $E_d(t_i \in \mathcal{V}) \stackrel{\text{def}}{=} (\mathcal{S}_i = \emptyset)$. t_i is visible in the reference image but there was no correct correspondence detected in the reference image.
- *Candidate Pruning Error*: $E_{cp}(t_i \in \mathcal{V}) \stackrel{\text{def}}{=} (\mathcal{S}_i \neq \emptyset, \mathcal{C}_i \cap \mathcal{S}_i = \emptyset)$. There is no detection error but the keypoint's candidates do not contain a correct correspondence.
- *Candidate Selection Error*: $E_{cs}(t_i \in \mathcal{V}) \stackrel{\text{def}}{=} (\mathcal{S}_i \neq \emptyset, \mathcal{C}_i \cap \mathcal{S}_i \neq \emptyset, \mathbf{s}_i \neq 0, \mathbf{s}_i \notin \mathcal{S}_i)$. There is neither a detection nor candidate pruning error, but $\mathbf{s}_i$ selects a wrong correspondence in the candidate-set.
- *Unmatched Error*: $E_u(t_i \in \mathcal{V}) \stackrel{\text{def}}{=} (\mathcal{S}_i \neq \emptyset, \mathcal{C}_i \cap \mathcal{S}_i \neq \emptyset, \mathbf{s}_i = 0)$. There is neither a detection nor candidate pruning error, but the keypoint is unmatched.

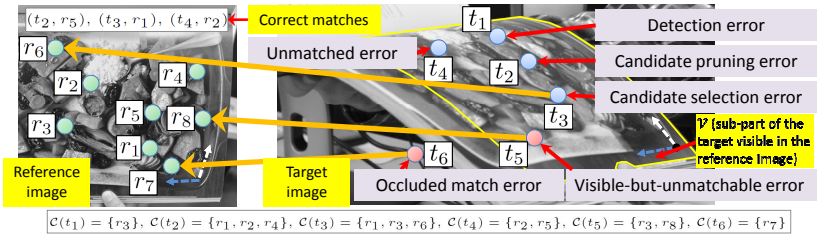


Fig. 1. The six types of errors in graph-based matching. Reference and target keypoint-sets are illustrated by circles and candidate matches are given at the bottom. Orange arrows illustrate matches made by a matching algorithm. Best viewed in colour.

We can aggregate these errors into what we call a *visible-keypoint match error*: $E_v(t_i \in \mathcal{V}) = (E_d(t_i) \vee E_{cp}(t_i) \vee E_{cs}(t_i) \vee E_u(t_i))$. A matching system can make two other types of errors which involve matching keypoints that have no match in $\mathcal{R}$. The first is an *occluded match error*: $E_{oc}(t_i \notin \mathcal{V}) = (\mathbf{s}_i \neq 0)$. The second is what we call a *visible-but-unmatchable error*: $E_{vu}(t_i \in \mathcal{V}) = (E_d \vee E_{cp})$, $\mathbf{s}_i \neq 0$. We can also combine E_{oc} and E_{vu} into what we call an *unmatchable-keypoint error*: $E_u(t_i) = E_{oc}(t_i) \vee E_{vu}(t_i)$. Developing a good graph-based matching system is very challenging because it must simultaneously reduce unmatchable keypoint errors yet try to match as many visible keypoints as possible. For visible keypoint matching errors, we will show that errors in detection and candidate pruning tend to have a compounding effect: not only does a visible keypoint become unmatchable, but also geometric constraints are lost between it and its neighbours in the graph. When this occurs too frequently the graph is weakened, which can then lead to more candidate selection errors.

How might one reduce visible-keypoint match errors? Current research in keypoint detection involves reducing the so-called *repeatability error* [26]. A repeatability error occurs when a keypoint detector fails to find the same two keypoints in both images. Thus perfect repeatability implies $\mathcal{V} = \mathcal{V}'$, but this is different from perfect detection error (which is what we want), which implies $\mathcal{V} \subseteq \mathcal{V}'$. *To improve matching performance, we should design or adapt keypoint detectors to reduce detection and candidate pruning errors.* The challenge here is how to incorporate this into graph matching efficiently. For example, one could reduce detection errors by having keypoints positioned densely in scale-space in the reference image. This requires an expensive dense computation of keypoint descriptors, and heavy candidate pruning. Reducing candidate selection errors has been the main objective in prior graph matching papers, and this has mainly been used as the evaluation criteria. This ignores the other types of errors we have listed. We now propose the first matching framework designed to reduce *all* types of errors. We call this framework Graph-based Affine Invariant Matching (GAIM), and its main contributions are:

1. To reduce detection and candidate pruning errors by constructing $\mathcal{R}$ using a redundant set keypoints. We do this so that even if there is large deformation between the two images, there is a high likelihood of a correct correspondence in $\mathcal{R}$ with a similar descriptor (§3.1).
2. To reduce candidate selection errors by constructing *second-order* graph-constraints that are fast to evaluate and *deformation invariant* (§3.3). To achieve this we show that one can obtain automatically an estimate of the local affine transform for each candidate match from our solution to point 1. These can then be used to enforce piecewise smooth matches. We show that this reduces assignment errors compared with the most common second-order constraint, which is to preserve Euclidean distance [6, 2].
3. To handle unmatchable keypoints with a robust graph cost function, and show that many can be effectively detected *a posteriori* using an LRC procedure adapted to graph matching (§3.4).

3 Proposed Framework: GAIM

3.1 Reference Keypoint-Set Construction to Reduce Detection and Candidate Pruning Errors

We use the fact that keypoint detectors trigger at different image positions in scale-space depending to a large part on deformation and viewpoint change. Thus, rather than attempting to make a keypoint's descriptor affine-invariant (*e.g.* [26]) we construct $\mathcal{R}$ by simulating many deformations of the reference image, and for each simulation we harvest keypoints and add them into $\mathcal{R}$. Although the deformation space is enormous, most deformations can be well-approximated locally by an affine transformation. Thus we only need to simulate global affine deformations of the reference image (Fig. 2). We do not simulate all of its six DoFs because keypoint methods such as SIFT are reasonably invariant to 2D translation, isotropic scale and rotation. Thus we simulate deformations by varying the remaining two DoFs (shear and aspect ratio). A similar strategy was proposed in [27] to make SIFT affine invariant. However, our goal is different because we want to have asymmetric keypoints (where all keypoints in $\mathcal{V}$ have a correspondence in $\mathcal{R}$, but not necessarily all keypoints in $\mathcal{R}$ have a correspondence in $\mathcal{V}$). We also want to obtain an estimate of the affine transform between t_i and its candidate matches, as we will include these in the graph's cost. In [27] the affine transforms were not inferred. We uniformly quantise shear in the range $[-1.0 : 1.0]$ and anisotopic scale in the range $[-0.25 : 4.0]$. We use x intervals that we experimentally vary from 3 to 8. We denote the simulated image set by $\mathcal{I}_s$, $s \in [1...S]$, and its simulated affine transform by $\mathbf{A}_s$. We represent each keypoint $r_k \in \mathcal{R}$ by $r_k = (y_k, \mathbf{q}_k, \mathbf{d}_k, \sigma_k, \theta_k, \mathbf{p}_k)$. We use y_k to denote the index of the simulated reference image from which r_k was harvested. We use $\mathbf{q}_k \in \mathbb{R}^2$ to denote the 2D position of the keypoint in $\mathcal{I}_{y_k}$ and $\mathbf{d}_k \in \mathbb{R}^d$ denotes its descriptor (using SIFT we have $d = 128$). We use $\mathbf{p}_k \in \mathbb{R}^2$ to denote its 2D position in the reference image. This is given by $\mathbf{p}_k = w(\mathbf{A}_{\sigma_k}^{-1}, \mathbf{q}_k)$ where $w(\mathbf{A}, \cdot) : \mathbb{R}^2 \to \mathbb{R}^2$ transforms a 2D point by an affine transform $\mathbf{A}$. We use $\sigma_k \in \mathbb{R}^+$ and $\theta_k \in [0; 2\pi]$ to denote the scale and orientation of the keypoint in $\mathcal{I}_{y_k}$ respectively.

3.2 Target Keypoint-Set, Candidate-Sets, and Graph Construction

We construct $\mathcal{T}$ by running the keypoint detector on the target image with the same parameters as used to construct $\mathcal{R}$. Unless otherwise stated we rescale the images to a default image diagonal resolution of 1,000 pixels before running the detector. For any keypoint that have multiple descriptors due to multiple orientation estimates, we use the descriptor that corresponds to the strongest orientation. However in $\mathcal{R}$ multiple descriptors are kept. We denote each keypoint $t_i \in \mathcal{T}$ by $t_i = (\tilde{\mathbf{p}}_i, \tilde{\mathbf{d}}_i, \tilde{\sigma}_i, \tilde{\theta}_i)$, which holds its 2D position, descriptor, scale and orientation respectively. We construct $\mathcal{C}(t_i)$ by running an Approximate Nearest Neighbour (ANN) search over $\mathcal{R}$. Currently we use FLANN [28] with default parameters with a forest of 8 KD trees. We modify the search to account for the fact that due to the redundancy in $\mathcal{R}$, several keypoints are often

returned as nearest neighbours but they are approximately at the same position in the reference image. If this occurs then slots in the candidate-set are wasted. We deal with this in a simple manner by sequentially selecting as a candidate the keypoint in $\mathcal{R}$ with the closest descriptor from the ANN method, then eliminating keypoints which are approximately at the same image location (we use a default threshold of 5 pixels). The graph is then constructed that connects spatial neighbours in $\mathcal{T}$. In our experiments this is done with a simple Delaunay triangulation. We handle the fact that edges may cross motion discontinuities by including robustness into the graph's cost function. Because the geometric constraints are defined locally a natural way to express matching constraints is with a MRF. We use a second-order MRF of the form:

$$E(\mathbf{s}) = \sum_{i \in [1 \ldots |\mathcal{T}|]} U_i(\mathbf{s}(i)) + \lambda \sum_{(i,j) \in \mathcal{E}} P_{ij}(\mathbf{s}(i), \mathbf{s}(j))) \tag{1}$$

The first-order term $U_i \in \mathbb{R}$ is used to penalise matches with dissimilar descriptors. We use the standard unary term $U_i(\mathbf{s}(i) > 0) \stackrel{\text{def}}{=} \|\mathbf{d}_{\mathbf{s}(i)} - \tilde{\mathbf{d}}_i\|_1$. The 2^{nd}-order term $P_{ij} \in \mathbb{R}$ enforces geometric consistency. The term λ balances the influence of P_{ij}, and can be set by hand or learned from training images.

3.3 Second-Order Deformation-Invariant Graph Constraints

We now show how the full-affine transform $\mathbf{A}_i^k$ that maps the local region in the target image about a keypoint t_i to the local region about a candidate match r_k in the reference image can be computed very cheaply (Fig. 2). Unlike affine normalisation [26] this requires no optimisation. We achieve this by making the following assumption. Suppose t_i and r_k is a correct correspondence. Because their descriptors are close (since r_k was selected as a candidate match), the local transform between the simulated image $\mathcal{I}_{y_k}$ and the target image at these points is likely to not have undergone much anisotropic scaling or shearing, and can be approximated by a similarity transform $\mathbf{S}_i^k$. We obtain $\mathbf{S}_i^k$ from the keypoint detector: $\mathbf{S}_i^k \approx \begin{bmatrix} \sigma_k \mathbf{R}(\theta_k) & \mathbf{q}_k \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \tilde{\sigma}_i \mathbf{R}(\tilde{\theta}_i) & \tilde{\mathbf{p}}_i \\ \mathbf{0}^\top & 1 \end{bmatrix}^{-1}$ where $\mathbf{R}(\theta)$ is a 2D rotation by an angle θ. $\mathbf{A}_i^k$ is then given by composing this transform with $\mathbf{A}_{y_i}^{-1}$ as $\mathbf{A}_i^k \approx \mathbf{A}_{y_i}^{-1} \mathbf{S}_i^k$ (Fig. 2). There are two points to note. The first is that we are exploiting the fact that the keypoint is *not* fully affine invariant to give us $\mathbf{A}_i^k$. Two of its DoFs come from the simulated deformation transform $\mathbf{A}_{y_i}$ (shear and aspect ratio) and four come from the keypoint's built-in invariance to give $\mathbf{S}_i^k$. The second is that computing it is virtually free given any candidate match. We then use these affine transforms to construct P_{ij} in a way which makes the graph invariant to local affine deformation. By enforcing this robustly we can encourage matches that can be explained by a piecewise smooth deformation. This writes as follows:

$$P_{ij}(\mathbf{s}(i) > 0, \mathbf{s}(j) > 0) \stackrel{\text{def}}{=} \min \left[\left\| w(\mathbf{A}_i^{\mathbf{s}(i)}, \tilde{\mathbf{p}}_j) - \mathbf{p}_{\mathbf{s}(j)} \right\|_2, \left\| w(\mathbf{A}_j^{\mathbf{s}(j)}, \tilde{\mathbf{p}}_i) - \mathbf{p}_{\mathbf{s}(i)} \right\|_2, \tau \right] \tag{2}$$

The first term in Eq. (2) penalises a pair of matches if $\mathbf{A}_i^{\mathbf{s}(i)}$ (the affine transform between t_i and $r_{\mathbf{s}(i)}$) cannot predict well $\mathbf{p}_{\mathbf{s}(j)}$ (the 2D position of $r_{\mathbf{s}(j)}$). Because

we also have $\mathbf{A}_j^{\mathbf{s}(j)}$, the second term is similar but made by switching the roles of i and j. We take the minimum of these two terms to improve robustness, which allows tolerance if either $\mathbf{A}_i^{\mathbf{s}(i)}$ or $\mathbf{A}_j^{\mathbf{s}(j)}$ is estimated wrongly. The term τ is a constant which truncates the pairwise term. This is used to provide robustness. Currently we set τ manually using a small dataset of training images (see §4).

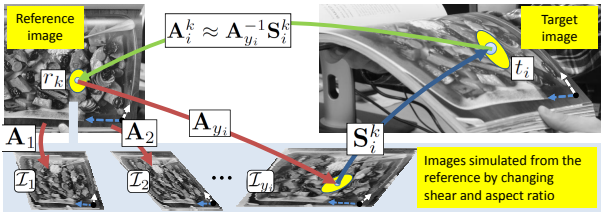


Fig. 2. Full affine transform $\mathbf{A}_i^k$ for a candidate match using simulation (for shear and aspect ratio) and keypoint-invariance (scale, rotation and translation)

3.4 Detecting Unmatchable Keypoints *a posteriori*

The unmatchable state is difficult to handle as a graph constraint. To define the unary term $U_i(\mathbf{s}(i) = 0)$ we would require to associate some cost (otherwise the graph would prefer a solution with all nodes in the unmatchable state [29]). Balancing this cost ideally is non-trivial because $\mathcal{V}$ can vary significantly between image pairs depending on clutter, different backgrounds and surface self-occlusions. We also do not usually know *a priori* the expected number of detection and candidate pruning errors, as these vary between scenes and imaging conditions. We face the same difficulty with defining the pairwise terms involving the unmatchable state. To address this problem we note that our matching framework *is not a symmetric process.* When searching for matches, only the detected keypoints in the target image are matched. If the role of the reference and target images is reversed, a second set of matches would be found, and this should be consistent with the first set. This is the so-called LRC test [7, 8] applied to graph-based keypoint matching. We note that the LRC is also equivalent to the uniqueness constraint which ensures each point in one image can match at most one point in the other image [7].

Our solution to handle the unmatchable state and the uniqueness constraint is to apply the LRC constraint (Fig. 3). First the MRF is defined in the target image *without* utilising the unmatchable state (we call this the *target MRF*). The robustness of the graph's terms are used to prevent unmatchable keypoints adversely affecting the solution. The target MRF is then solved and we denote its solution by $\hat{\mathbf{s}}$ and the matched pairs by $\mathcal{M}_T = \{(t_i, r_{\hat{\mathbf{s}}(i)})\}$, $i \in \{1...n\}$ (Fig. 3, top right). We then form the set $\mathcal{R}' \stackrel{\text{def}}{=} \{r_{\hat{\mathbf{s}}(i)}\} \subset \mathcal{R}$, $i \in \{1...n\}$, and define a second MRF called the *reference MRF* using $\mathcal{R}'$ as its nodes. Duplicates or

near-duplicates may exist in $\mathcal{R}'$ because two keypoints in $\mathcal{T}$ may have been matched to approximately the same location in the reference image. We detect these using agglomerative clustering (we use a cluster threshold of 5 pixels), and collapse reference MRF nodes to have one node per cluster. A new MRF is then constructed using the cluster centres and their neighbours. Then the target image is treated as the reference image and the reverse match is computed (Fig. 3, bottom left). We denote its solution by $\mathcal{M}_T = \{(r_{\hat{\mathbf{s}}(i)}, t'_i)\}$. The LRC is then applied by ensuring that t_i and t'_i are close, and we assign t_i to the unmatchable state if $\|\tilde{\mathbf{p}}'_i - \tilde{\mathbf{p}}_i\|_2 \geq \tau_c$. We use a default threshold of $\tau_c = 15$ pixels.

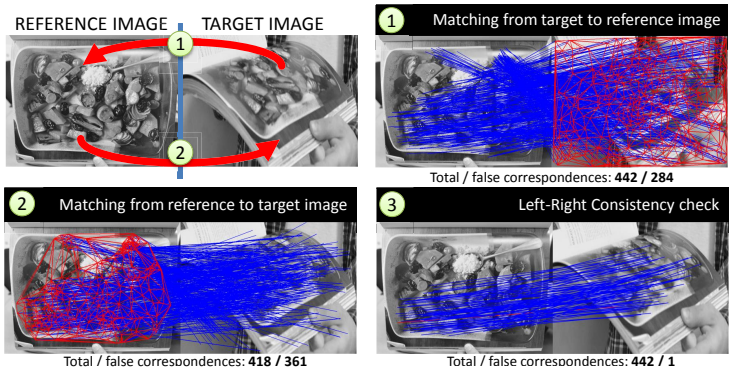


Fig. 3. Proposed Left-Right Consistency test for keypoint-based graph-matching

4 Experimental Evaluation

We divide the experimental evaluation into three parts. The first part evaluates GAIM for reducing detection and candidate pruning errors. The second part evaluates GAIM for reducing assignment errors. The third part evaluates our method for detecting incorrect matches after graph optimisation. We compare against two state-of-the-art HMOD methods: PB12 [21] and TCCBS12 [22]. Because we deal with such large keypoint-sets ($O(n) \approx 10^3$ and $O(m) \approx 10^4$), most factorisation-based methods are unsuitable to tackle the problem. As a baseline graph-based method we use [6] which can handle large m and n via candidate match pruning. We use the naming scheme F/G/O/M to describe a graph matching method configuration. This denotes a method that computes keypoints with a method F, computes graph constraints with a method G, optimises the graph with an MRF optimiser O, and performs mismatch detection *a posterior* with a method M. In our experiments F is either SIFT, AC-SIFT (Affine-Covariant SIFT), or Gx-SIFT (our proposed asymmetric method by adapting SIFT in §3.1, where x denotes the number of synthesised views plus the original image). SIFT and AC-SIFT are computed with VLFeat's implementation [30]. We use SIFT because it is still very popular, has matching accuracy that is competitive with state-of-the-art, and currently supports switching between standard

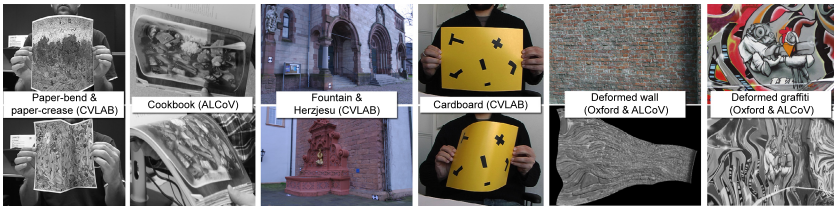
and affine-Covariant versions of the descriptor. G is either GAIM (our proposed graph constraint) or RE (which stands for Robust Euclidean). RE is the Euclidean distance-preserving constraint from [6] but made robust by introducing the truncation term τ. This allows a fair comparison with our constraint, and we manually tune τ using the same training images. For O we use fast methods known to work well on correspondence problems. These are AE (α-expansion [31] with QPBO [32, 33]) or BP (loopy Belief Propagation). M is either PB12 [21], TCCBS12 [22] or LRC+ (our proposed Left-Right Consistency test). We use GAIM to be shorthand for G26-SIFT/GAIM/AE/LRC+.

4.1 Test Datasets

We evaluate using public datasets from the deformable surface registration literature, and some new challenging datasets that we have created (Fig. 4). We divide the datasets into 6 groups (Fig. 4). Group 1 involves 8 representative images from CVLAB's paper-bend and paper-crease sequences [34]. The first frame is the reference image and five other frames were used as target images. We also swapped the roles of the reference and target images, giving a total of 20 reference/target pairs. Group 2 involves 8 reference/target pairs of an open deforming book with strong viewpoint change and optic blur that we shot with a 720p smartphone. Group 3 involves 8 reference/target pairs taken from CVLAB's multiview 3D reconstruction dataset with GT computed by [35]. This is a rigid scene, but it was used since GT optic flow is available and the scene has motion and surface discontinuities. Although the epipolar constraint is available due to rigidity, we did not allow methods to use this constraint. Group 4 involves 16 reference/target pairs from CVLAB's cardboard dataset [36] that we constructed in a similar manner to Group 1. This is challenging due to texture sparseness and local texture ambiguity. Groups 5 and 6 involve 20 reference/target pairs taken from Oxford's wall and graffiti datasets respectively. We used the first frame as reference image, and the target images were generated from the first and third frames by applying randomised synthetic deformations to them using perspective and TPS transformations. In total there are 92 reference/target pairs. We trained λ and τ by hand on a training set comprising CVLAB's bedsheet and cushion datasets [34]. GT optic-flow is not provided for Groups 1,2 and 4. We computed this carefully using an interactive warping tool. All images were scaled to a resolution with diagonal 1,000 pixels and a match was considered correct if it agreed to the optic flow by less than 12 pixels.

4.2 Experiment 1: Reduction of Candidate-Set Errors

The first source of errors in graph-based keypoint matching is a candidate-set error, which is when either a detection error or a candidate pruning error occurs (see §2). In Fig. 5 we plot the mean candidate-set error rate as a function of the candidate-set size p for SIFT, AC-SIFT and Gx-SIFT, with p varying from 1 to 200 and x varying from 10 to 65. In solid lines we plot the error rates when using the default detection parameters for building $\mathcal{R}$ and $\mathcal{T}$. Across all datasets we



		SIFT				AC-SIFT				G26-SIFT			
	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{R}\rvert$	Index $\mathcal{R}$	ANN 10	ANN 100	$\lvert\mathcal{R}\rvert$	Index $\mathcal{R}$	ANN 10	ANN 100	$\lvert\mathcal{R}\rvert$	Index $\mathcal{R}$	ANN 10	ANN 100
Paper-bend & Paper-crease	485	942	20.19	2.82	16.89	997	21.53	2.83	16.06	54279	1248.24	9.30	34.57
Cookbook	635	1352	28.67	3.83	22.57	1482	32.84	3.86	22.29	45137	1051.12	10.91	41.54
Fountain & Herzjesu	602	1047	25.81	3.71	22.10	1123	25.81	3.76	23.02	50461	1199.47	9.87	37.37
Cardboard	65	88	1.93	0.94	8.17	96	2.66	0.87	9.30	4891	116.15	1.19	6.15
Wall with deformation	748	1688	37.84	6.04	28.34	1778	43.82	6.29	28.73	81587	1882.41	14.25	50.77
Graffiti with deformation	848.03	1235	26.54	5.43	31.22	1325	27.94	6.07	31.40	41490	974.01	22.61	61.90

Fig. 4. Top: The six groups of test images used in evaluation. In total we test 92 reference/target image pairs. Bottom: average size of $\mathcal{R}$ and $\mathcal{T}$ and time (in ms) to construct, indexing and ANN querying $\mathcal{R}$ on an i7-3820 PC with FLANN [37] with a default maximum search depth of 15. Although the time to index G26-SIFT is considerably lager (taking a second or more) the time to query is only approximately a factor of two/three slower than SIFT and AC-SIFT. In tracking tasks where indexing only needs to be done once the benefits of using Gx-SIFT are very strong.

find a clear advantage in using Gx-SIFT. The benefits in a larger x is stronger for smaller p. For $p > 100$ we see no real benefit in G8-SIFT over G26-SIFT in any testset. For SIFT and AC-SIFT one might expect lower error rates by keeping in $\mathcal{R}$ all detections without filtering those with low edge and scale-space responses (and so to potentially reduce detection errors). In dashed we show the error rates when $\mathcal{R}$ was constructed without the post-filtering. However there is no clear error reduction in general, and in some cases this actually increased the error. The large improvement in using Gx-SIFT tells us that a major cause for candidate-set errors are the lack of viewpoint and local deformation invariance. Despite AC-SIFT being designed to handle this, the results show that it lags quite far behind Gx-SIFT.

4.3 Experiment 2: Reduction of Candidate Selection Errors

After detection and candidate pruning the next source of errors are candidate selection errors (we refer the reader to §2 for the formal definition). We compared 10 matching configurations. These are listed in the first row of Fig. 6. For our proposed keypoint detection method we use G26-SIFT. Our proposed matching configurations in full is G26-SIFT/GAIM/BP & AE. Because AC-SIFT also provides local affine transforms between candidate matches we also test the configuration AC-SIFT/GAIM by using the affine transforms from AC-SIFT in Eq. (2). In all experiments we use a default value of $p = 90$. The results

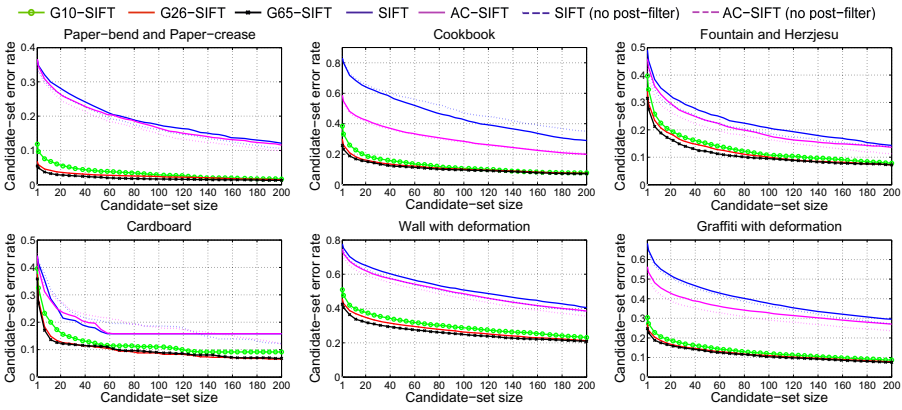


Fig. 5. Candidate-set error rate versus candidate-set size across the six test-sets. For a fair comparison all methods use the same detected keypoints in the target image.

on the test sets are shown in rows 1 and 2 of Fig. 6. The best results are obtained by the four configurations on the left (*i.e.* the ones where the reference keypoint-set was built using G26-SIFT keypoints). This result is important because it tells us that when we use keypoints with higher candidate-set errors (*i.e.* SIFT and AC-SIFT), this weakens the graph and causes keypoints that do not have candidate-set errors to be matched incorrectly. With the exception of Cardboard, there is a clear performance improvement with using GAIM constraints over RE. We believe the reason is because in Cardboard there is *(i)* little scale change between the views and *(ii)* the texture is much sparser than other datasets, which means that the local affine model used by GAIM has to be valid between distant surface points. There is little difference in the performance of G26-SIFT/GAIM/BP and G26-SIFT/GAIM/AE, which indicates both graph optimisers tend to find the same solution (although more tests are required). With respect to AC-SIFT, although this performs significantly worse than G26-SIFT we do see a performance improvement with using GAIM graph constraints over RE. We also measure the visible keypoint match error rates for each configuration. Recall from §2 that a visible keypoint match error occurs when a keypoint is visible in the target image, but the graph's solution does provide a correct correspondence. This is a combination of both candidate-set and assignment errors. The results are shown in rows 3 and 4 of Fig. 6.

4.4 Experiment 3: Complete Performance Evaluation

We now evaluate the complete performance of our approach, and show that unmatchable keypoints can be successfully detected *a posteriori* using the LRC. Space limitations prevent a full performance breakdown, so we present here the complete system recall/precision performance. This is given by the proportion of matches that a method computes correctly versus the proportion of visible

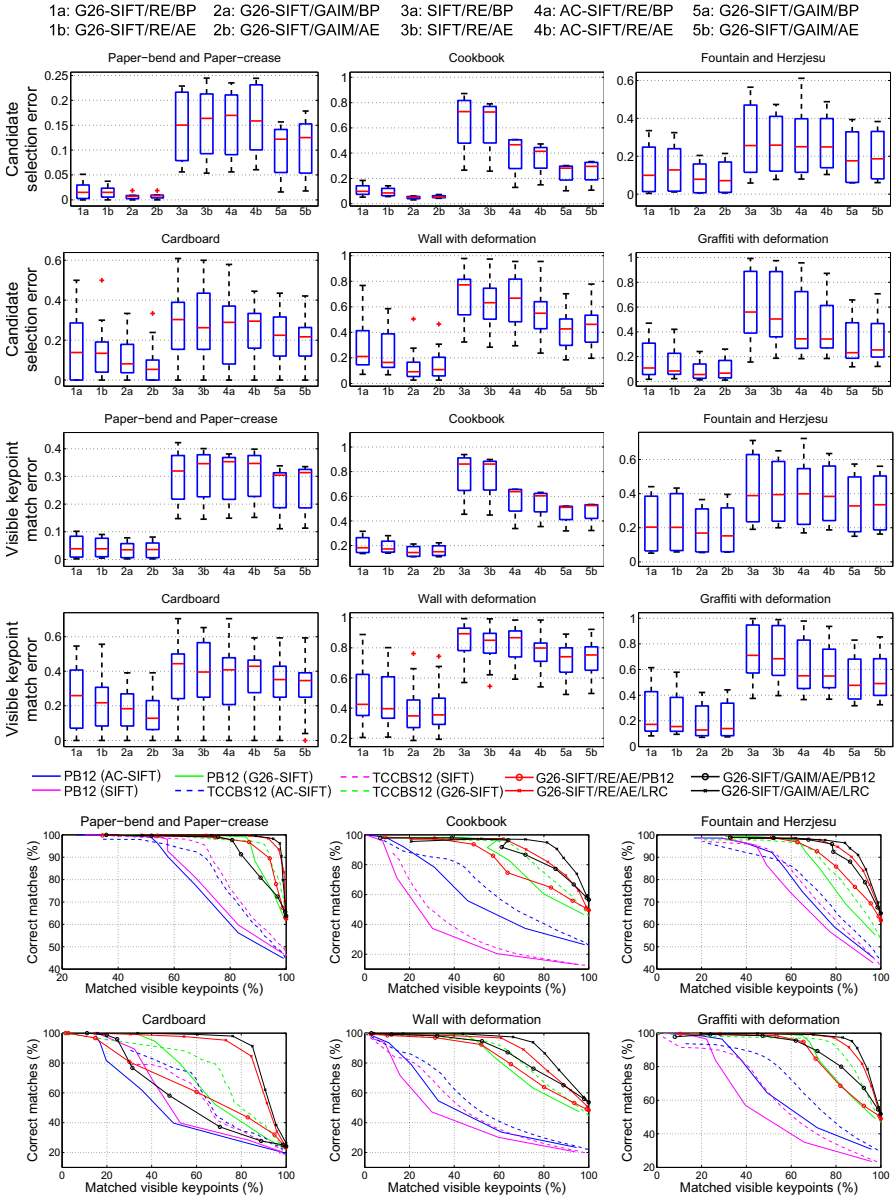


Fig. 6. Results of experiments 2 and 3. Rows 1 to 4: Assignment and visible keypoint match errors for 10 graph matching configurations. Rows 5, 6: Complete matching performance (including mis-match detection) of best-performing configurations and HMOD methods.

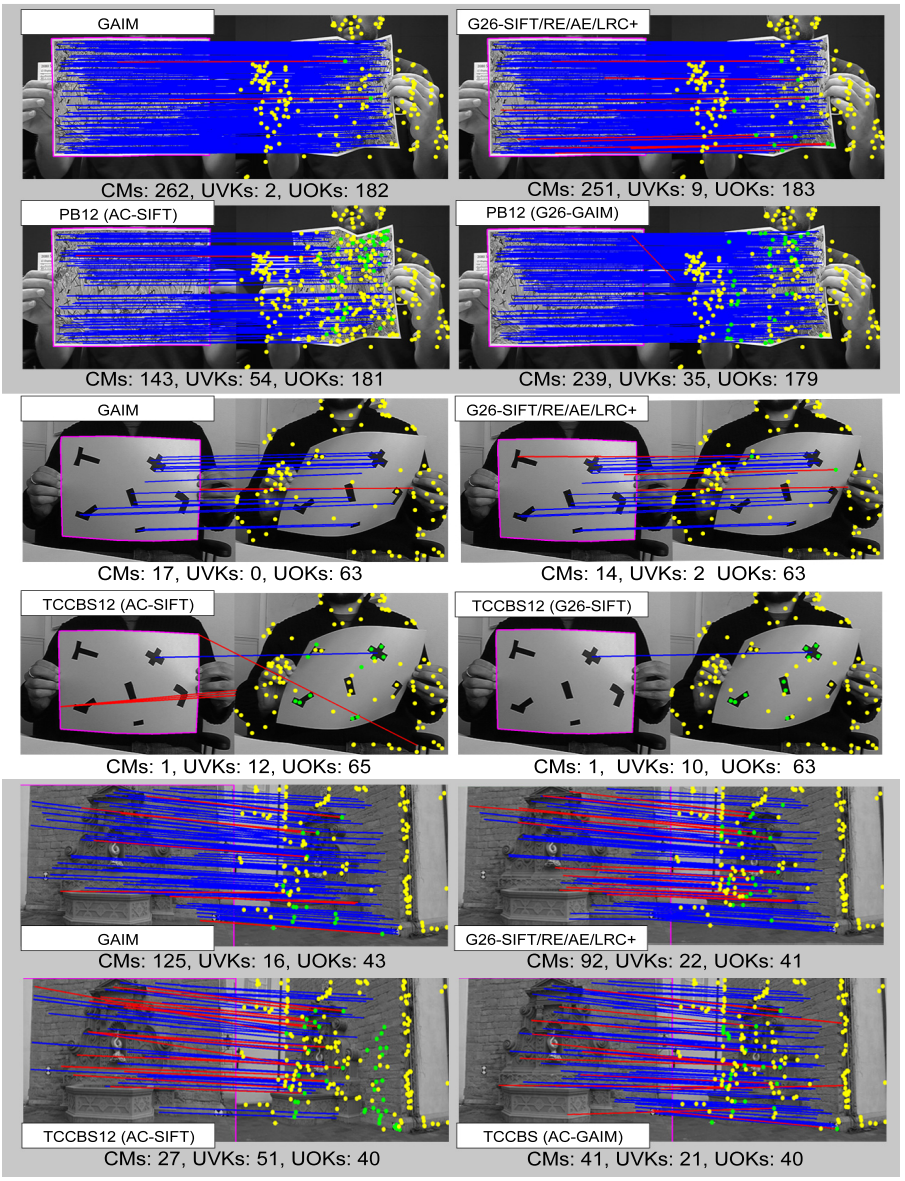


Fig. 7. Sample results from test groups showing keypoint matching with mismatch detection. CMs stands for the number of Correct matches (higher is better, in blue), UVKs stands for the number of Unmatched Visible Keypoints (lower is better, in green) and UOKs stands for the number of unmatched occluded keypoints (higher is better, in yellow). In red are false matches. In each target image we show the Region-Of-Interest (ROI) within which we had GT optic flow. To ease evaluation we restricted $\mathcal{R}$ to be keypoints within the ROI. Therefore any keypoint in the target image that does not have a correct match in the ROI is an occluded keypoint. No ROI was used to filter keypoints in the target images.

keypoints that have been matched. We compare against PB12 and TCCBS12 using hard matches from G26-SIFT, AC-SIFT and SIFT. We also investigate if PB12 and TCCBS12 can detect incorrect matches given the solution of a graph-based method. This is an interesting test and has not been done in the literature. We plot these results in rows 5 and 6 in Fig. 6. The trend we see is that the HMOD methods perform significantly better with G26-SIFT, and the reason is that many more hard matches are correct with G26-SIFT than with SIFT or AC-SIFT. For both G26-SIFT/RE/AE and G26-SIFT/GAIM/AE the performance when using PB12 to detect incorrect matches in their outputs is not significantly greater than hard matching using G26-SIFT, and in some instances is worse. The reason for this is because incorrect matches outputted by a graph method tend spatially correlated, and this makes them hard to distinguish from correct matches. The best performing method across all test sets is G26-SIFT/GAIM/AE/LRC. In Fig. 7 we give some representative visual results of the methods.

5 Conclusions

We have given a comprehensive breakdown of errors in graph-based keypoint matching into six different types. These errors occur at various stages of matching; from keypoint detection, candidate selection to final graph optimisation. In previous works keypoint detectors have been used in a rather black-box style, however there is a deep interplay between the keypoint detector and graph-based matching that should not be ignored. We hope the results of this paper will stimulate the design of new keypoint methods that specifically reduce candidate-set errors in graph-matching rather than the commonly used repeatability metric. We have presented the first matching system that has been designed to reduce all six error types. Candidate-set errors have been considerably reduced by Gx-SIFT features. These also provide automatic information about a keypoint's local affine transform that can be used as a second-order deformation invariant matching constraint. This produces lower candidate-selection errors than the commonly-used euclidean distance-preserving constraint. We have provided a method to detect mismatches *a posteriori* using a Left-Right Consistency procedure adapted to asymmetric deformable graph matching, and shown that the full framework outperforms state-of-the-art HMOD methods.

Acknowledgments. This research has received funding from the EU's FP7 through the ERC research grant 307483 FLEXABLE.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of Graph Matching in Pattern Recognition. Int. J. Pattern Recogn., 265–298 (2004)
2. Leordeanu, M., Hebert, M., Sukthankar, R.: An integer projected fixed point method for graph matching and MAP inference. In: Neural Information Processing Systems (NIPS), pp. 1114–1122 (2009)

3. Cho, M., Lee, J., Lee, K.M.: Reweighted Random Walks for Graph Matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 492–505. Springer, Heidelberg (2010)
4. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE T. Pattern Anal. 31, 1048–1058 (2009)
5. Leordeanu, M., Sukthankar, R., Hebert, M.: Unsupervised learning for graph matching. Int. J. Comput. Vision 96, 28–45 (2012)
6. Torresani, L., Kolmogorov, V., Rother, C.: Feature Correspondence Via Graph Matching: Models and Global Optimization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 596–609. Springer, Heidelberg (2008)
7. Faugeras, O., Hotz, B., Mathieu, H., Viville, T., Zhang, Z., Fua, P., Thron, E., Robotvis, P.: Real time correlation-based stereo: Algorithm, implementations and applications (1996)
8. Kowdle, A., Gallagher, A., Chen, T.: Combining monocular geometric cues with traditional stereo cues for consumer camera stereo. In: Fusiello, A., Murino, V., Cucchiara, R. (eds.) ECCV 2012 Ws/Demos, Part II. LNCS, vol. 7584, pp. 103–113. Springer, Heidelberg (2012)
9. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Trans. Pattern Anal. Mach. Intell. 10, 695–703 (1988)
10. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: International Conference on Computer Vision (ICCV) (2013)
11. Zhou, F., la Torre, F.D.: Deformable graph matching. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2922–2929 (2013)
12. Zaslavskiy, M., Bach, F., Vert, J.-P.: A path following algorithm for graph matching. In: Elmoataz, A., Lezoray, O., Nouboud, F., Mammass, D. (eds.) ICISP 2008 2008. LNCS, vol. 5099, pp. 329–337. Springer, Heidelberg (2008)
13. Chertok, M., Keller, Y.: Efficient high order matching. IEEE Trans. Pattern Anal. Mach. Intell. 32, 2205–2215 (2010)
14. Duchenne, O., Bach, F., Kweon, I.S., Ponce, J.: A tensor-based algorithm for high-order graph matching. IEEE Trans. Pattern Anal. Mach. Intell. 33, 2383–2395 (2011)
15. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: International Conference on Computer Vision (ICCV), pp. 1482–1489 (2005)
16. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Trans. Pattern Anal. Mach. Intell. 18, 377–388 (1996)
17. Scott, G.L., Longuet-Higgins, H.C.: An Algorithm for Associating the Features of Two Images. Royal Society of London Proceedings Series B 244, 21–26 (1991)
18. Hamid, R., DeCoste, D., Lin, C.J.: Dense non-rigid point-matching using random projections. In: CVPR, pp. 2914–2921. IEEE (2013)
19. Albarelli, A., Rodolà, E., Torsello, A.: Imposing semi-local geometric constraints for accurate correspondences selection in structure from motion: A game-theoretic perspective. Int. J. Comput. Vision 97, 36–53 (2012)
20. Pilet, J., Lepetit, V., Fua, P.: Fast non-rigid surface detection, registration and realistic augmentation. Int. J. Comput. Vision 76, 109–122 (2008)
21. Pizarro, D., Bartoli, A.: Feature-Based Deformable Surface Detection with Self-Occlusion Reasoning. Int. J. Comput. Vision 97, 54–70 (2012)

22. Tran, Q.-H., Chin, T.-J., Carneiro, G., Brown, M.S., Suter, D.: In defence of RANSAC for outlier rejection in deformable registration. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part IV. LNCS, vol. 7575, pp. 274–287. Springer, Heidelberg (2012)
23. Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. Comput. Vis. Image Underst. 89, 114–141 (2003)
24. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60, 91–110 (2004)
25. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. 110, 346–359 (2008)
26. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. Int. J. Comput. Vision 60, 63–86 (2004)
27. Morel, J.M., Yu, G.: ASIFT: A New Framework for Fully Affine Invariant Image Comparison. SIAM J. Imaging Sci. 2, 438–469 (2009)
28. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2014)
29. Torresani, L., Hertzmann, A., Bregler, C.: Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. IEEE Trans. Pattern Anal. Mach. Intell. 30(5), 878–892 (2008)
30. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008), `http://www.vlfeat.org/`
31. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. 23, 1222–1239 (2001)
32. Rother, C., Kolmogorov, V., Lempitsky, V.S., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
33. Hammer, P., Hansen, P., Simeone, B.: Roof duality, complementation and persistency in quadratic optimization. Mathematical Programming 28, 121–155 (1984)
34. Salzmann, M., Hartley, R., Fua, P.: Convex optimization for deformable surface 3-d tracking. In: International Conference on Computer Vision (ICCV), pp. 1–8 (2007)
35. Strecha, C., Bronstein, A.M., Bronstein, M.M., Fua, P.: LDAHash: Improved matching with smaller descriptors. In: EPFL-REPORT-152487 (2010)
36. Salzmann, M., Urtasun, R., Fua, P.: Local deformation models for monocular 3d shape recovery. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
37. Bartoli, A.: Maximizing the predictivity of smooth deformable image warps through cross-validation. Journal of Mathematical Imaging and Vision 31(2-3), 133–145 (2008)