

# Robust Multi-body Feature Tracker: A Segmentation-free Approach

Pan Ji<sup>1</sup>, Hongdong Li<sup>1</sup>, Mathieu Salzmann<sup>2</sup>, and Yiran Zhong<sup>1</sup>

<sup>1</sup>ANU, Canberra; <sup>2</sup>EPFL, Switzerland

## Abstract

*Feature tracking is a fundamental problem in computer vision, with applications in many computer vision tasks, such as visual SLAM and action recognition. This paper introduces a novel multi-body feature tracker that exploits a multi-body rigidity assumption to improve tracking robustness under a general perspective camera model. A conventional approach to addressing this problem would consist of alternating between solving two subtasks: motion segmentation and feature tracking under rigidity constraints for each segment. This approach, however, requires knowing the number of motions, as well as assigning points to motion groups, which is typically sensitive to the motion estimates. By contrast, here, we introduce a segmentation-free solution to multi-body feature tracking that bypasses the motion assignment step and reduces to solving a series of subproblems with closed-form solutions. Our experiments demonstrate the benefits of our approach in terms of tracking accuracy and robustness to noise.*

## 1. Introduction

Feature tracking is a prerequisite for many computer vision tasks, such as visual SLAM and action recognition. Among all the feature tracking methods, the Kanade-Lucas-Tomasi (KLT) tracker [18, 23, 21], although developed 30 years ago, still remains one of the most widely used techniques. One of the reasons for this popularity is its computational efficiency; the KLT tracker is local, in the sense that it treats each local region independently of the others, which makes it highly parallelizable. This locality, however, comes at a cost in tracking robustness: the tracking of each feature cannot benefit from intrinsic scene constraints, and thus often suffers from drift.

The real-world scenes, however, are often strongly constrained. For example, in autonomous driving, most of the moving objects (cars, vehicles, pedestrian) are rigid, or quasi-rigid if seen from afar. Several methods have therefore been proposed to exploit this scene rigidity to improve feature tracking [24, 3, 20]. Unfortunately, these methods all assume an affine camera model and are thus

ill-suited to handle strong perspective effects. More importantly, they work either as a post-processing step on an entire sequence [24], which is sensitive to initial tracking results and does not apply to online feature tracking, or within a temporal sliding window [3, 20], which is sensitive to initialization in the first few frames.

By contrast, in this paper, we introduce a novel feature tracker that takes advantage of *multi-body scene rigidity* to improve tracking robustness under a general *perspective camera model*. A conventional approach to addressing this problem would consist of alternating between two subtasks: motion segmentation and feature tracking under rigidity constraints for each segment. This, however, suffers from the following drawbacks: First, it requires knowing the number of observed motions; and, second, it relies on assigning points to individual motions, which is very sensitive to the initial motion estimates.

Here, we introduce a *segmentation-free* multi-body feature tracker that overcomes these drawbacks. Specifically, our approach bypasses the motion assignment step by making use of subspace constraints derived directly from the epipolar constraints of multiple motions. As a result, our algorithm does not require prior knowledge of the number of motions. Furthermore, this allows us to formulate tracking as an optimization problem whose subproblems all have closed-form solutions.

We demonstrate the effectiveness of our method on both feature point tracking and frame-by-frame motion segmentation on real world sequences. Our experiments show that, by incorporating multi-motion constraints, our tracker yields better accuracies and is more robust to noise than the standard KLT tracker and the state-of-the-art tracking algorithm of [20].

## 2. Related Work

The KLT tracker [23, 21] was derived from the Lucas-Kanade algorithm for image alignment [18]. Feature tracking was achieved by optimizing the sum of squared differences between a template patch and an image patch with the Gauss-Newton method. It was later extended to handle relatively large displacements by the use of image pyramids [1].

Global rigidity constraints have been incorporated in feature point tracking to improve robustness. For instance, Torresani and Bregler [24] proposed to regularize tracking with a global low-rank constraint on the trajectory matrix of the whole sequence. They relied on the original KLT tracker to get a set of reliable tracks, and explicitly factorized the reliable trajectory matrix into two low-rank matrices with the rank given *a priori*. One of the low-rank matrices, called the motion parameter matrix, was then used to rectify the unreliable tracks. In short, this method can be viewed as a post-processing step on the results of the KLT tracker, and is therefore not suitable for online frame-to-frame tracking.

Instead of using the whole sequence, low-rank constraints [3] and similar subspace priors [20] were applied within a temporal sliding window. Specifically, Buchanan and Fitzgibbon [3] exploited the low-rank constraints within a Bayesian tracking framework, making predictions of the new location of a particular point using a low rank approximation obtained from the previous frames. Recently, Poling *et al.* [20] proposed a better feature tracker by adding soft subspace constraints to the original KLT tracker and jointly solving for the displacement vectors of all feature points. These methods, however, assume an affine camera model within a temporal window, and are therefore ill-suited to handle strong perspective effects. Moreover, since the low-rank constraints are enforced in a temporal sliding window, these methods are sensitive to initialization in the first few frames.

By contrast, [19] exploits perspective projection by making use of epipolar constraints to track edgels in two consecutive frames. This method, however, was specifically designed to model a single motion, and thus does not easily extend to the multi-body case.

In the closely related optical flow literature, several methods have been devoted to improving robustness via rigidity constraints. For instance, Valgaerts *et al.* [26] introduced a variational model to jointly recover the fundamental matrix and the optical flow; Wedel *et al.* [30, 29] leveraged the fundamental matrix prior as an additional weak prior within a variational framework. These methods, however, assume that the scene is mostly stationary (and thus a single fundamental matrix is estimated), and treat the dynamic parts as outliers [29]. Garg *et al.* [7, 8] proposed to make use of subspace constraints to regularize the multi-frame optical flow within a variational approach. This approach, however, assumes an affine camera model and works over entire sequences.

While, to the best of our knowledge, explicitly modeling multi-body motion has not been investigated in the context of feature tracking and optical flow estimation, a large body of work [5, 27, 31, 15, 16, 28, 6, 12, 14, 13] has been devoted to multi-body motion segmentation given good point trajectories in relatively long sequences. Typically,

these tracks are first obtained with the KLT tracker, and then manually cleaned up, *e.g.*, the Hopkins155 dataset [25]. In a sense, the lack of better tracking algorithms that can incorporate the intrinsic constraints of dynamic scenes prevents the practical use of these motion segmentation algorithms.

In this paper, we seek to track feature points in dynamic scenes where multiple motions are present. In this scenario, a single fundamental matrix is not sufficient to express the epipolar constraints any more. While one could think of alternating between estimating multiple fundamental matrices, motion assignments and displacement vectors, the resulting algorithm would typically be very sensitive to initialization, since the motion assignments strongly depend on the motion estimates. By contrast, we introduce a segmentation-free approach that bypasses the motion assignment problem by exploiting subspace constraints derived from epipolar geometry. This yields a robust multi-body tracking algorithm that, as demonstrated by our experiments, opens up the possibility to perform motion segmentation in realistic scenarios.

### 3. Multi-body Feature Tracker

We now introduce our approach to multi-body feature tracking. Formally, let  $I(\mathbf{x})$  denote the current image,  $T(\mathbf{x})$  the previous image (or template image), and  $\mathbf{x}_{ij} = [x_{ij}, y_{ij}]^T$  the  $j^{\text{th}}$  image point in the  $i^{\text{th}}$  patch  $\Omega_i$  of the template image. Our goal is to estimate the displacement vector  $\mathbf{u} = [\mathbf{u}_1^T, \dots, \mathbf{u}_N^T]^T \in \mathbb{R}^{2 \times N}$  for all  $N$  tracked feature points. To this end, we rely on the standard brightness constancy assumption [22], which lets us derive the data term

$$\mathcal{D}(\mathbf{u}) = \sum_{i=1}^N \sum_{\mathbf{x}_{ij} \in \Omega_i} \psi(I(\mathbf{x}_{ij} + \mathbf{u}_i) - T(\mathbf{x}_{ij})) , \quad (1)$$

where, typically,  $\psi(x) = x^2$  or  $\psi(x) = |x|$ . In particular, we use the  $\ell_1$  norm, which provides robustness to outliers.

Estimating the displacements from this data term only is typically sensitive to noise and may be subject to drift. A general approach to making the process more robust consists of introducing a regularizer  $\mathcal{R}(\mathbf{u})$  to form an energy function of the form

$$\mathcal{F}(\mathbf{u}) = \gamma \mathcal{D}(\mathbf{u}) + \mathcal{R}(\mathbf{u}) . \quad (2)$$

As mentioned above, several attempts at designing such a regularizer have been proposed. For example, under an affine camera model,  $\mathcal{R}(\mathbf{u})$  can encode a low-rank prior [24, 20]; with a general projective camera model,  $\mathcal{R}(\mathbf{u})$  can represent epipolar constraints (*i.e.*, a fundamental matrix prior) [30, 26, 19]. In the latter case, the fundamental matrix can be either pre-computed via an existing feature matching method [19], or re-computed iteratively.

When multiple motions are present, however, a single epipolar constraint is not sufficient. Instead, multiple fundamental matrices should be estimated so as to respect the assignments of the tracked points to individual motions. A straightforward way to addressing this problem consists of adding a motion segmentation step in the tracking algorithm, so that the fundamental matrices can be iteratively re-estimated. This leads to the simple *segmentation-based* approach to multi-body feature tracking as described below.

### 3.1. A First Attempt: Segmentation-based Tracking

To derive a segmentation-based approach, we rely on epipolar constraints. Recall that, in epipolar geometry [10], the homogeneous coordinates  $\bar{\mathbf{x}}'_i = (x'_i, y'_i, 1)^T$  and  $\bar{\mathbf{x}}_i = (x_i, y_i, 1)^T$  of two corresponding image points in two frames are related by a fundamental matrix  $\mathbf{F}$ , such that

$$\bar{\mathbf{x}}'^T_i \mathbf{F} \bar{\mathbf{x}}_i = 0. \quad (3)$$

It is therefore natural to exploit these constraints to regularize tracking according to the motion assignments of the different points.

More specifically, in the segmentation-based approach, three types of variables must be estimated: the displacement vector  $\mathbf{u}$ , the fundamental matrices  $\{\mathbf{F}^k\}_{k=1, \dots, K}$  (where  $K$  is the number of motions), and the motion label of each tracked point. Let us denote by  $\bar{\mathbf{x}}^k_i$  the homogeneous coordinate of the  $i^{\text{th}}$  feature point (*i.e.*, the center of the patch  $\Omega_i$ ) assigned to motion  $k$ . We can define a multi-body regularization term as

$$\mathcal{R}_1(\mathbf{u}, \mathbf{F}^k) = \sum_k \sum_i [(\bar{\mathbf{x}}^k_i + \bar{\mathbf{u}}_i)^T \mathbf{F}^k \bar{\mathbf{x}}^k_i]^2, \quad (4)$$

where  $\bar{\mathbf{u}}_i = [\mathbf{u}_i^T, 0]^T$ .

The energy function can then be approximately minimized by iterating over the three following steps:

1. Update  $\mathbf{u}$  by first-order gradient descent [20];
2. Estimate  $\mathbf{F}^k$  for each motion given the current point assignments;
3. Re-assign the motion labels of the feature points to the nearest  $\mathbf{F}^k$ .

This segmentation-based approach suffers from several drawbacks. First, the number of motions needs to be known *a priori*, which is typical hard for general-purpose tracking. Second, and more importantly, the quality of solution obtained with this approach will strongly depend on the initializations of  $\mathbf{F}^k$  and of the motion labels. This, in a sense, is a chicken-and-egg problem, since good initialization for these variables could be obtained from good motion estimates. Instead, in the remainder of this section, we introduce a new segmentation-free approach that bypasses the need to explicitly compute the fundamental matrices and the motion assignments.

## 3.2. Our Segmentation-free Approach

In this section, we introduce our segmentation-free multi-body feature tracker, which is the key contribution of this paper. We first show how the epipolar constraints can be converted to subspace constraints, and incorporated into our tracking formalism. We then derive the solution to the resulting optimization problem by decomposing it into several convex subproblems all with closed-form solutions.

### 3.2.1 Epipolar Subspace Constraints

As in the segmentation-based approach, we seek to rely on epipolar geometry. To this end, we make use of the constraint expressed in Eq. 3. We first note that this constraint can be re-written as

$$\mathbf{f}^T \text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}^T_i) = 0, \quad (5)$$

where  $\mathbf{f} \in \mathbb{R}^9$  is the vectorized fundamental matrix  $\mathbf{F}$ , and

$$\text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}^T_i) = (x_i x'_i, x_i y'_i, x_i, y_i x'_i, y_i y'_i, y_i, x'_i, y'_i, 1)^T. \quad (6)$$

Let us define  $\mathbf{w}_i = \text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}^T_i)$ . Then,  $\mathbf{w}_i$  lies in the orthogonal complement of  $\mathbf{f}^T$ , which is a subspace of dimension up to eight<sup>1</sup>, and which we call the *epipolar subspace*. Since image points undergoing the same motion share the same fundamental matrix, all  $\mathbf{w}_i$ s corresponding to points belonging to the same rigid motion lie on the same subspace [16].

Therefore, in our multi-body feature tracking scenario, if the feature points are correctly tracked, the data vectors defined as

$$\mathbf{w}_i = \text{vec}((\bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i) \bar{\mathbf{x}}^T_i), \quad \forall 1 \leq i \leq N, \quad (7)$$

should lie in a union of linear subspaces. This subspace constraint can be characterized by the self-expressiveness property [6, 12], *i.e.*, a data point drawn from one subspace in a union of subspaces can be represented as a linear combination of the points lying in the same subspace.

In our case, this self-expressiveness property can be expressed as

$$\mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C}, \quad (8)$$

where  $\mathbf{W}_{(\mathbf{u})} = [\mathbf{w}_1 \cdots \mathbf{w}_N]^T$ , and  $\mathbf{C}$  is the coefficient matrix encoding the linear combinations. On its own, this term has a trivial solution for  $\mathbf{C}$  (*i.e.*, the identity matrix). To avoid this solution,  $\mathbf{C}$  needs to be regularized. In the subspace clustering literature,  $\mathbf{C}$  is encouraged to be either

<sup>1</sup>Note that, in practice, this dimension is typically smaller than 8, since, in real scenes, the motion of objects, such as cars or people, is not arbitrary, and thus corresponds to degenerate (*i.e.*, low-rank) motion [16].

<sup>2</sup>In the following, we make use of subscript  $(\mathbf{u})$ , *i.e.*,  $\mathbf{W}_{(\mathbf{u})}$ , to indicate that  $\mathbf{W}$  depends on the variable  $\mathbf{u}$ . For compactness, and without causing confusion, we drop this explicit dependency in Section 3.2.3.

sparse [6] by minimizing  $\|\mathbf{C}\|_1$ , low rank [17] by minimizing  $\|\mathbf{C}\|_*$ , or dense block diagonal [12] by minimizing  $\|\mathbf{C}\|_F^2$ . Here, we choose the Frobenius norm, which has proven effective and is easy to optimize. Furthermore, we explicitly model noise and outliers, which are inevitable in real-world sequences.

More specifically, we write our regularization term for multi-body tracking as

$$\mathcal{R}_2(\mathbf{u}, \mathbf{C}) = \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1, \text{ s.t. } \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E}, \quad (9)$$

where  $\mathbf{E}$  accounts for noise and outliers, and is thus encouraged to be sparse. Note that, for a given displacement  $\mathbf{u}$ , and ignoring noise, the optimal value of this regularizer depends on the intrinsic dimension of the motion [12]. Since here we optimize  $\mathbf{u}$ , this regularizer therefore tends to favor degenerate rigid motions over purely arbitrary rigid motions. This actually reflects reality, since, in real scenes, cars, people and other objects typically move in a well-constrained manner.

Importantly, this regularization term requires explicitly computing neither the fundamental matrices, nor the motion assignments. As such, it therefore yields a *segmentation-free* approach.

Altogether, the energy function of our multi-body tracking framework can be written as

$$\mathcal{F}(\mathbf{u}, \mathbf{C}) = \gamma \mathcal{D}(\mathbf{u}) + \mathcal{R}_2(\mathbf{u}, \mathbf{C}). \quad (10)$$

Our goal is to minimize  $\mathcal{F}(\mathbf{u}, \mathbf{C})$  w.r.t.  $\mathbf{u}$  and  $\mathbf{C}$ . We next show how to solve this optimization problem.

### 3.2.2 Approximation and Problem Reformulation

To optimize Eq. 10, we first approximate the data term in the same manner as the original KLT. In other words, given an initial displacement  $\mathbf{u}_i^0$  for patch  $i$ , we approximate the intensity values  $I(\mathbf{x}_{ij} + \mathbf{u}_i)$  with their first-order Taylor expansion at  $\mathbf{x}_{ij} + \mathbf{u}_i^0$ . This can be written as

$$I(\mathbf{x}_{ij} + \mathbf{u}_i) \approx I(\mathbf{x}_{ij} + \mathbf{u}_i^0) + \nabla I(\mathbf{x}_{ij} + \mathbf{u}_i^0)(\mathbf{u}_i - \mathbf{u}_i^0). \quad (11)$$

For notational convenience, let  $\nabla I_{ij} = \nabla I(\mathbf{x}_{ij} + \mathbf{u}_i^0)$ , and  $\tau_{ij} = \nabla I_{ij} \mathbf{u}_i^0 + T(\mathbf{x}_{ij}) - I(\mathbf{x}_{ij} + \mathbf{u}_i^0)$ . Then, the data term can be expressed as

$$\mathcal{D}(\mathbf{u}) = \sum_{i,j} |\nabla I_{ij} \mathbf{u}_i - \tau_{ij}|. \quad (12)$$

By combining this data term with our regularizer, we get the optimization problem

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}} \quad & \gamma \|\mathbf{A}_{(\mathbf{u})}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E}, \end{aligned} \quad (13)$$

where  $\mathbf{A}_{ij} = \nabla I_{ij} \mathbf{u}_i - \tau_{ij}$ .

For convenience of optimization, we introduce an auxiliary variable  $\mathbf{Z} = \mathbf{A}_{(\mathbf{u})}$ . Then, (13) can be equivalently written as

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}} \quad & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_{(\mathbf{u})}, \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E}. \end{aligned} \quad (14)$$

The main hurdle in optimizing (14) now lies in the term with  $\mathbf{W}_{(\mathbf{u})}$  due to its seemingly complicated dependency on  $\mathbf{u}$ . However, we show below that this term can be simplified by a few matrix derivations.

First, note that, by definition, we have

$$\text{vec}(\mathbf{W}_{(\mathbf{u})}) = \underbrace{\begin{bmatrix} \bar{\mathbf{x}}_1 \otimes \mathbf{I}_{3 \times 3} & & \\ & \ddots & \\ & & \bar{\mathbf{x}}_N \otimes \mathbf{I}_{3 \times 3} \end{bmatrix}}_{\bar{\mathbf{P}}} (\bar{\mathbf{x}} + \bar{\mathbf{u}}), \quad (15)$$

where  $\bar{\mathbf{x}} = [\bar{\mathbf{x}}_1^T \cdots \bar{\mathbf{x}}_N^T]^T$ ,  $\bar{\mathbf{u}} = [\bar{\mathbf{u}}_1^T \cdots \bar{\mathbf{u}}_N^T]^T$ ,  $\mathbf{I}_{3 \times 3}$  is the 3-by-3 identity matrix and  $\otimes$  denotes the Kronecker product. Let us define  $\mathbf{b} = \bar{\mathbf{P}} \bar{\mathbf{x}}$  (or equivalently  $\mathbf{b}_i = \text{vec}(\bar{\mathbf{x}}_i \bar{\mathbf{x}}_i^T)$ ) and introduce another auxiliary variable  $\mathbf{m} = \bar{\mathbf{P}} \mathbf{u}$  (where  $\bar{\mathbf{P}}$  is obtained by removing every  $3i^{\text{th}}$  column of  $\bar{\mathbf{P}}$ )<sup>3</sup>. Our optimization problem then becomes

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}, \mathbf{m}} \quad & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_{(\mathbf{u})}, \mathbf{W}_{(\mathbf{m})} = \mathbf{W}_{(\mathbf{m})} \mathbf{C} + \mathbf{E}, \mathbf{m} = \bar{\mathbf{P}} \mathbf{u}, \end{aligned} \quad (16)$$

where now  $\text{vec}(\mathbf{W}_{(\mathbf{m})}) = \mathbf{b} + \mathbf{m}$ .

The above optimization problem involves a large number of variables. We propose to solve it via the Alternating Direction Method of Multipliers (ADMM) [2], which decomposes a big optimization problem into several small subproblems. Below, we show how this can be achieved for our problem.

### 3.2.3 ADMM Solution

To apply the ADMM, we first need to derive the augmented Lagrangian of (16), which can be expressed as

$$\begin{aligned} \mathcal{L}_\rho = & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 + \mathbf{y}^T (\mathbf{m} - \bar{\mathbf{P}} \mathbf{u}) + \\ & \langle \mathbf{Y}_1, \mathbf{W} - \mathbf{W} \mathbf{C} - \mathbf{E} \rangle + \langle \mathbf{Y}_2, \mathbf{Z} - \mathbf{A}_{(\mathbf{u})} \rangle + \\ & \frac{\rho}{2} (\|\mathbf{W} - \mathbf{W} \mathbf{C} - \mathbf{E}\|_F^2 + \|\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}\|_F^2 + \|\mathbf{m} - \bar{\mathbf{P}} \mathbf{u}\|_2^2), \end{aligned} \quad (17)$$

where  $\langle \cdot, \cdot \rangle$  denotes the matrix inner product,  $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{y}$  are Lagrange multipliers, and  $\rho$  is the penalty parameter. The ADMM then works by alternatively minimizing  $\mathcal{L}_\rho$  w.r.t.

<sup>3</sup>Note that  $\bar{\mathbf{P}} \mathbf{u} = \bar{\mathbf{P}} \bar{\mathbf{u}}$ , since  $\bar{\mathbf{u}}_i = [\mathbf{u}_i^T, 0]^T$ .



---

**Algorithm 1** Solving (16) via the ADMM

---

**Input:**

Image  $I$  and template  $T$ , positions of the feature points  $\mathbf{x}$  in  $T$ , initial displacement vector  $\mathbf{u}^0$ , parameters  $\gamma, \lambda$ .

**Initialize:**  $\mathbf{C} = 0, \mathbf{Y}_1 = 0, \mathbf{Y}_2 = 0, \mathbf{y} = 0, \mathbf{A}_{(\mathbf{u}^0)}, \mathbf{W}_{(\mathbf{u}^0)}, \rho_0, \rho_m, \eta, \epsilon$

**while** not converged **do**

1. Update  $\mathbf{Z}, \mathbf{E}, \mathbf{C}, \mathbf{u}$  and  $\mathbf{m}$  in close-form via Eqs. 18-22, respectively;
2. Update  $\mathbf{A}_{(\mathbf{u})}$  and  $\mathbf{W}_{(\mathbf{m})}$  with updated  $\mathbf{u}$  and  $\mathbf{m}$ ;
3. Update the Lagrange multipliers and penalty parameter via Eqs. 23- 26;
4. Check the convergence conditions  $\|\mathbf{m} - \mathbf{P}\mathbf{u}\|_\infty \leq \epsilon, \|\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{E}\|_\infty \leq \epsilon$ , and  $\|\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}\|_\infty \leq \epsilon$ ;

**end while**

**Output:** Displacement vector  $\mathbf{u}$ , coefficient matrix  $\mathbf{C}$ .

---

one of the five variables  $\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}, \mathbf{m}$  while keeping the remaining four fixed.

As shown in appendix, the five subproblems derived from the augmented Lagrangian are all convex problems that can be solved efficiently in closed-form. These closed-form solutions can be written as

$$\mathbf{Z} = \mathcal{T}_{\frac{\gamma}{\rho}}[\mathbf{A}_{(\mathbf{u})} - \mathbf{Y}_2/\rho], \quad (18)$$

$$\mathbf{E} = \mathcal{T}_{\frac{\lambda}{\rho}}[\mathbf{W} - \mathbf{W}\mathbf{C} + \mathbf{Y}_1/\rho], \quad (19)$$

$$\mathbf{C} = (\mathbf{I} + \rho\mathbf{W}^T\mathbf{W})^{-1}[\rho\mathbf{W}^T(\mathbf{W} - \mathbf{E} + \mathbf{Y}_1/\rho)], \quad (20)$$

$$\mathbf{u} = (\rho\mathbf{P}^T\mathbf{P} + \rho\mathbf{H})^{-1}(\mathbf{g} + \mathbf{P}^T\mathbf{y} + \rho\mathbf{P}^T\mathbf{m}), \quad (21)$$

$$\mathbf{M} = -(\rho\mathbf{G} + \mathbf{B}\mathbf{Q} + \mathbf{T})(\lambda\mathbf{Q} + \rho\mathbf{I})^{-1}, \quad (22)$$

where  $\mathbf{m} = \text{vec}(\mathbf{M})$  is the vectorized form of  $\mathbf{M}$ ,  $\mathcal{T}_\alpha[x] = \text{sign}(x) \cdot \max(|x| - \alpha, 0)$  is the soft-thresholding operator, and the definitions of  $\mathbf{g}, \mathbf{H}, \mathbf{Q}, \mathbf{T}$  are given in appendix.

Finally, the Lagrange multipliers and penalty parameter can be updated as

$$\mathbf{Y}_1 = \mathbf{Y}_1 + \rho(\mathbf{W} - \mathbf{W}\mathbf{C} - \mathbf{E}), \quad (23)$$

$$\mathbf{Y}_2 = \mathbf{Y}_2 + \rho(\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}), \quad (24)$$

$$\mathbf{y} = \mathbf{y} + \rho(\mathbf{m} - \mathbf{P}\mathbf{u}) \quad (25)$$

$$\rho = \min(\eta\rho, \rho_m), \quad (26)$$

where  $\eta > 1$ , and  $\rho_m$  is the predefined maximum of  $\rho$ .

Our approach to solving (16) is outlined in Algorithm 1. Note that the problem we are trying to solve is non-convex in that i) the intensity function  $I(\mathbf{x}; \mathbf{u})$  is non-convex w.r.t.  $\mathbf{u}$ ; ii) the optimization problem 16 involves a bilinear term in an equality constraint. While the ADMM does not guarantee convergence to the global optimum, it has proven effective in practice [11].

---

**Algorithm 2** Our Multi-body Feature Tracker

---

**Input:**

Image  $I$  and template  $T$ , positions of the feature points  $\mathbf{x}$  in  $T$ , initial displacement vector  $\mathbf{u}^0$ , number of pyramid levels  $L$ , parameters  $\gamma, \lambda, \rho, \rho_m, \max_i, \epsilon$

**for**  $\ell = L - 1 : 0$  **do**

Update  $\mathbf{u}^0 \leftarrow \mathbf{u}^0/2^\ell, \mathbf{x} \leftarrow \mathbf{x}^0/2^\ell$  and compute  $\nabla I$  at current image pyramid level;

**for**  $i = 1 : \max_i$  **do**

1. Approximate the image intensities with Eq. 11, and compute  $\tau, \mathbf{P}, \mathbf{H}$  according to their definitions;
2. Update  $\mathbf{u}$  with Algorithm 1;
3. Check the convergence condition  $\|\mathbf{u} - \mathbf{u}^0\| < \epsilon$ ;
4. If not converged, update  $\mathbf{u}^0 = \mathbf{u}$ .

**end for**

Update  $\mathbf{u} \leftarrow 2^\ell \mathbf{u}, \mathbf{u}^0 \leftarrow 2^\ell \mathbf{u}^0$ , and  $\mathbf{x} \leftarrow 2^\ell \mathbf{x}$ .

**end for**

**Output:** Displacement vector  $\mathbf{u}$ , coefficient matrix  $\mathbf{C}$ .

---

### 3.2.4 Our Complete Multi-body Feature Tracker

In the same spirits as [1], we make use of an image pyramid to handle large displacements and avoid local optima. The results obtained at a coarser level  $\ell$  of the pyramid are used as initialization for the next ( $\ell - 1$ , finer) level. Within each pyramid level, the initial displacement  $\mathbf{u}^0$ , where the first-order Taylor approximation is performed, is updated with the displacement vector of the previous iteration. We iterate over successive Taylor approximations until the displacement vector does not change significantly. Our complete segmentation-free multi-body feature tracker is outlined in Algorithm 2.

## 4. Experiments

To show the benefits of our multi-body feature tracker, we performed extensive experiments on different sequences. In the remainder of this section, we present both qualitative and quantitative results.

In these experiments, we compare our approach with the following baselines: the original KLT tracker (KLT), the L1-norm KLT tracker (L1-KLT), and the more recent Better Feature Tracker (BFT) through Subspace Constraints [20]. For the original KLT, we used the Matlab built-in vision toolbox *vision.PointTracker*; we implemented the L1-norm KLT tracker using the same framework as our method by just disabling the regularization term; and for BFT, we used the code released by the authors.

Due to the lack of benchmark datasets for feature tracking, we make use of motion segmentation datasets where both the ground-truth tracks and the original videos are



Figure 1: **Performance of different trackers on the 1RT2TC checkerboard sequence:** The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. Best viewed on screen with zoom-in.

available. Since those videos are typically only provided for illustration purpose, they are generally highly compressed and not ideal for reliable feature tracking. This, however, is not really a problem when one seeks to evaluate feature tracking methods, since (i) it essentially represents a challenging scenario; and (ii) all algorithms are evaluated on the same data. In particular, here, we employed 10 checkerboard (indoor) sequences and 12 cars-and-people (outdoor) sequences from the well-known Hopkins155 dataset [25]. Moreover, we used another 8 outdoor sequences from the more recent MTPV dataset [16]. To test the robustness of the different methods, we added different levels of Gaussian noise (with variance  $\sigma^2 = 0.01, 0.02, 0.03$ , or  $0.04$ )<sup>4</sup> to the images. Altogether, this results in 150 evaluation sequences. The values of the parameters ( $\gamma = 1.8 \times 10^4$  and  $\lambda = 1.0 \times 10^4$ ) were tuned on a separate validation set and kept unchanged for all our experiments.

To compare the algorithms, we measure the number of tracking errors, *i.e.*, the number of points that drift from the ground-truth by more than a certain error tolerance  $\varepsilon$ . Note that, in the sequences that we use, the ground-truth was obtained by the standard KLT tracker and then manually cleaned up, so the ground-truth itself contains some noise whose level depends on the scene itself. In particular, we observed that the ground-truth of the indoor checkerboard sequences generally has more noise than that of the outdoor ones. Therefore, we set a larger error tolerance for the checkerboard sequences ( $\varepsilon = 10$ ) than for the outdoor ones ( $\varepsilon = 5$ ). For every sequence, we compute the average number of incorrectly tracked feature points over all the frames, and then average this number over the sequences.

#### 4.1. Hopkins Checkerboard Sequences

We first evaluated our method and the baselines on the Hopkins checkerboard sequences, which depict controlled indoor scenes with multiple rigidly moving objects. The average number of tracks in this dataset is 202.9. Generally, the repetitive texture in these sequences makes feature

<sup>4</sup>Note that the intensities of the images are normalized to  $[0, 1]$ . So the Gaussian noise with  $\sigma^2 = 0.04$  is already big noise and more noise may never occur in practice.

Table 1: Average number of tracking errors ( $\varepsilon = 10$ ) on the Hopkins checkerboard sequences with noise of different variances  $\sigma^2$ . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	47.63	34.69	39.68	<b>27.77</b>
$\sigma^2 = 0.01$	46.92	30.86	39.30	<b>27.32</b>
$\sigma^2 = 0.02$	45.95	29.69	38.84	<b>27.13</b>
$\sigma^2 = 0.03$	46.59	30.16	39.16	<b>28.18</b>
$\sigma^2 = 0.04$	47.19	31.16	39.35	<b>27.21</b>

tracking more ambiguous and thus harder. However, in this experiment, we show that our multi-body feature tracker is more robust to this ambiguity. To provide a fair comparison, we used the same patch size ( $7 \times 7$ ) and the same number of image pyramid levels (4) for all the methods. Furthermore, we initialized all the tracking methods with the ground-truth locations of the feature points in the first frame.

From Table 1, we can see that the L1-KLT tracker consistently achieves better results than the original KLT tracker and than BFT. Our algorithm, however, consistently outperforms L1-KLT, which clearly evidences the benefits of incorporating our multi-body prior. We observed that BFT generally fails to track moving objects, as illustrated in Fig. 1. This is mainly because BFT heavily relies on a good estimate of the global motion, obtained by registering the entire current image to the previous one. For scenes with multiple motions, however, global motion estimation becomes unreliable, thus causing BFT to fail to track the moving objects. Note that the performance of all the trackers remain relatively unaffected as the noise level increases. This is mainly due to the fact that the corners in the checkerboard, while resembling each other, are very strong features that are robust to noise.

#### 4.2. Hopkins Car-and-People Sequences

We then evaluated the algorithms on the Hopkins Car-and-People sequences, depicting real-world outdoor scenes with multiple rigid motions. The number of tracks provided by the ground-truth ranges from 147 to 548 with an average of 369. Here, for all the methods, we used the same patch size and image pyramid levels as in the previous experi-

Table 2: Average number of tracking errors ( $\varepsilon = 5$ ) on the Hopkins Car-and-People sequences with noise of different variances  $\sigma^2$ . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	21.71	24.28	49.13	<b>16.14</b>
$\sigma^2 = 0.01$	34.59	29.31	51.69	<b>18.82</b>
$\sigma^2 = 0.02$	54.95	36.32	54.63	<b>26.56</b>
$\sigma^2 = 0.03$	76.02	46.49	57.57	<b>33.80</b>
$\sigma^2 = 0.04$	95.17	56.92	58.36	<b>42.43</b>

ment, and initialized the feature points with their ground-truth locations in the first frame. The average number of tracking errors for the different methods under different image noise level is reported in Table 2. Again, our multi-body feature tracker achieves the lowest tracking error compared to the baselines, which confirms the robustness of our method.

### 4.3. MTPV Sequences

We further tested our method on the MTPV sequences, which provide images of higher quality and resolution<sup>5</sup> than the Hopkins dataset and contains sequences with strong perspective effects. By contrast, however, this dataset contains some outliers and missing data. For evaluation purpose, *i.e.*, to create a complete and accurate ground-truth, we discarded the outliers and missing data. Since the image resolution is higher in this dataset, we used a larger patch size of  $13 \times 13$  for all the methods. The results of all the algorithms are provided in Table 3. Note that we still outperform all the baselines for most noise levels, with the exception of BFT for  $\sigma^2 = 0.04$ . We believe that the slightly less impressive gap between our approach and the baselines, in particular BFT, is due to the fact that the feature points in this dataset are often dominated by the background. See Fig. 2 for typical examples of this dataset.

Table 3: Average number of tracking errors ( $\varepsilon = 5$ ) on the MTPV sequences with noise of different variances  $\sigma^2$ . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	3.07	13.34	6.83	<b>2.34</b>
$\sigma^2 = 0.01$	17.76	22.12	8.84	<b>3.87</b>
$\sigma^2 = 0.02$	28.39	27.26	11.17	<b>6.94</b>
$\sigma^2 = 0.03$	40.61	35.53	11.26	<b>9.92</b>
$\sigma^2 = 0.04$	47.69	38.93	<b>12.34</b>	13.22

### 4.4. KITTI Sequence

To evaluate the algorithms on realistic, high-quality images, we employed four sequences<sup>6</sup> from KITTI [9],

<sup>5</sup>Note, however, that they are still highly compressed and not well-suited for tracking, as pointed out in the readme file of the dataset.

<sup>6</sup>2011\_09\_26\_drive\_0018, 2011\_09\_26\_drive\_0051, 2011\_09\_26\_drive\_0056, and 2011\_09\_28\_drive\_0016.



Figure 2: **The MAN and MONK sequences of the MTPV dataset:** The feature points are marked as red. Note that the number of points on the walking man and monk is much smaller than on the background.

depicting a street/traffic scene with multiple motions. Since no ground-truth trajectories are provided with this data, to obtain quantitative results, we took 10 consecutive frames from each sequence, applied the KLT tracker to them, and manually cleaned up the results to get ground-truth trajectories with an average 177 points per sequence. The results of this experiments for different levels of noise added to the input are reported in Table 4, and Fig. 3 shows a qualitative comparison of the algorithms. Note that our method also outperforms the baselines on this data.

Table 4: Average number of tracking errors ( $\varepsilon = 5$ ) on the KITTI sequences with different noise variances  $\sigma^2$ . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.01$	21.43	22.05	27.48	<b>14.18</b>
$\sigma^2 = 0.02$	24.35	22.85	27.80	<b>16.70</b>
$\sigma^2 = 0.03$	31.15	26.88	27.85	<b>17.70</b>
$\sigma^2 = 0.04$	34.43	29.23	27.75	<b>20.33</b>

### 4.5. Frame-by-Frame Motion Segmentation

In our formulation, we optimize our energy function w.r.t. two variables: the displacement vector  $\mathbf{u}$  and the self-expressiveness coefficients  $\mathbf{C}$ . While the vector  $\mathbf{u}$  provides the tracking results, the matrix  $\mathbf{C}$ , as in the subspace clustering literature, can be used to build an affinity matrix for spectral clustering, and thus, if we assume that the number of motions is known *a priori*, lets us perform motion segmentation. In other words, our method can also be interpreted as simultaneous feature tracking and frame-by-frame motion segmentation. In this experiment, we therefore aim to evaluate the frame-by-frame motion segmentation accuracy of our method. Since, To the best of our knowledge, no existing motion segmentation methods perform feature tracking and frame-by-frame motion segmentation jointly, we compare our results with the following two-steps baselines: first, we find the tracks by KLT or L1-KLT and form the epipolar subspaces as in Eq. 7; second, we apply a subspace clustering method, *i.e.*, Sparse Subspace Cluster-



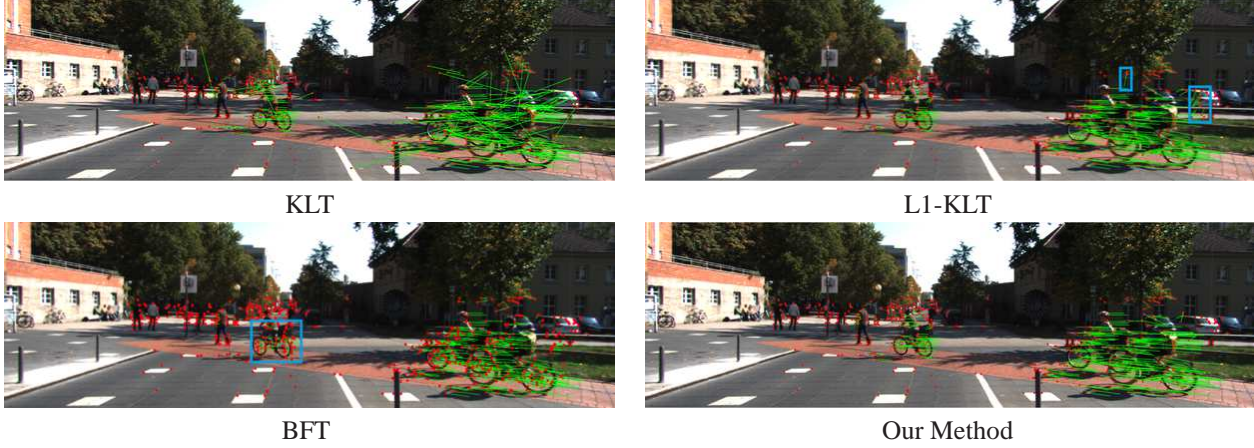


Figure 3: **Performance of different trackers on the KITTI sequence:** The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. As evidenced by the regions highlighted with a blue rectangle, L1-KLT and BFT make more tracking errors than our approach. Best viewed on screen with zoom-in.

Table 5: Average error rate (in %) of two-frame motion segmentation on the 22 Hopkins sequences with noise of different variances  $\sigma^2$ . The lower, the better.

Methods	KLT+SSC	KLT+EDSC	L1+SSC	L1+EDSC	Ours
$\sigma^2 = 0.00$	19.76	20.57	18.71	19.11	<b>8.97</b>
$\sigma^2 = 0.01$	19.76	20.61	19.61	20.41	<b>9.35</b>
$\sigma^2 = 0.02$	19.21	20.99	21.02	21.92	<b>9.33</b>
$\sigma^2 = 0.03$	20.63	20.69	22.21	20.48	<b>9.89</b>
$\sigma^2 = 0.04$	20.38	19.82	21.35	20.80	<b>11.26</b>

ing (SSC) or Efficient Dense Subspace Clustering (EDSC), to perform motion segmentation. This results in four baselines denoted by KLT+SSC, KLT+EDSC, L1+SSC [16] and L1+EDSC. The results of motion segmentation on the 22 Hopkins sequences used previously are shown in Table 5. These results clearly evidence that our method outperforms the baselines significantly in terms of motion segmentation.

## 5. Conclusion and Future Work

In this paper, we have introduced a novel feature tracker that incorporates a multi-body rigidity prior into feature tracking. To this end, we have derived epipolar subspace constraints that prevent us from having to compute fundamental matrices and motion assignments explicitly. Our formulation only involves a series of convex subproblems, all of which have closed-form solutions. We have demonstrated the effectiveness of our method via extensive experiments on indoor and outdoor sequences.

While adding global rigidity constraints (be it the low-rank or the epipolar subspace constraints) to the local KLT tracker improves robustness, it comes with some computational overhead. The current Matlab implementation of our method runs at about 1 frame per second for 200 points on a single core CPU (3.4GHZ), which is on par with

BFT [20], but slower than the original KLT tracker. In the future, we will therefore study how to speed up our approach, for instance by exploiting the GPU. Furthermore, our current model assumes that each patch undergoes only translation between consecutive frames. We therefore plan to investigate the use of more accurate models, such as affine transformations.

## Appendix: ADMM Derivations

Given the augmented Lagrangian in Eq. 18, the ADMM subproblems can be derived as follows:

(1) Computing  $\mathbf{Z}$  can be expressed as the convex program

$$\min_{\mathbf{Z}} \frac{\gamma}{\rho} \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{Z} - (\mathbf{A}_{(\mathbf{u})} - \mathbf{Y}_2/\rho)\|_F^2, \quad (27)$$

which can be solved in closed-form by element-wise thresholding [4], which directly yields Eq. 18.

(2) Similarly, computing  $\mathbf{E}$  translates to

$$\min_{\mathbf{E}} \frac{\lambda}{\rho} \|\mathbf{E}\|_1 + \frac{1}{2} \|\mathbf{E} - (\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} + \mathbf{Y}_1/\rho)\|_F^2, \quad (28)$$

which again can be solved by element-wise thresholding, thus yielding Eq. 19.

(3) To compute  $\mathbf{C}$ , we have the least-squares problem

$$\min_{\mathbf{C}} \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\rho}{2} \|\mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{W}_{(\mathbf{m})} + \mathbf{E} - \mathbf{Y}_1/\rho\|_F^2, \quad (29)$$

which can easily be solved in closed-form as in Eq. 20.

(4) Computing  $\mathbf{u}$  requires solving the problem

$$\min_{\mathbf{u}} \frac{\rho}{2} \|\mathbf{P}\mathbf{u} - \mathbf{m} - \mathbf{y}/\rho\|_2^2 - \mathbf{g}^T \mathbf{u} + \frac{\rho}{2} \mathbf{u}^T \mathbf{H} \mathbf{u}, \quad (30)$$

where  $\mathbf{g}$  is a column vector defined as

$$\mathbf{g} = \left( \cdots, \sum_j (\mathbf{Y}_{2ij} + \rho(\tau_{ij} + \mathbf{Z}_{ij})) \nabla I_{ij}, \cdots \right)^T \in \mathbb{R}^{2N},$$



and  $\mathbf{H}$  is a sparse block-diagonal matrix expressed as

$$\mathbf{H} = \begin{bmatrix} \ddots & & \\ & \sum_j \nabla I_{ij}^T \nabla I_{ij} & \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{2N \times 2N}.$$

This subproblem has again a closed-form solution given by Eq. 21. Note that  $\mathbf{P}$  and  $\mathbf{H}$  are sparse matrices, so  $\mathbf{u}$  can be computed efficiently by sparse matrix techniques.

(5) While solving for  $\mathbf{m}$  may not seem straightforward, we show below that it is nothing but a least-squares problem. The subproblem w.r.t.  $\mathbf{m}$  can be written as

$$\min_{\mathbf{m}} \frac{\lambda}{2} \|\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})} \mathbf{C}\|_F^2 + \frac{\rho}{2} \|\mathbf{m} - \mathbf{P}\mathbf{u} + \mathbf{y}/\rho\|_2^2. \quad (31)$$

Let  $\mathbf{M}, \mathbf{B}, \mathbf{G} \in \mathbb{R}^{9 \times N}$  be the matrix forms of  $\mathbf{m}, \mathbf{b}, \mathbf{y}/\rho - \mathbf{P}\mathbf{u}$ , respectively. Then, (31) can be equivalently written as

$$\min_{\mathbf{M}} \frac{\lambda}{2} \|\mathbf{M}(\mathbf{I} - \mathbf{C}) + \mathbf{B}(\mathbf{I} - \mathbf{C})\|_F^2 + \frac{\rho}{2} \|\mathbf{M} + \mathbf{G}\|_F^2. \quad (32)$$

This again leads to a closed-form solution for  $\mathbf{M}$  given by Eq. 22, where  $\mathbf{Q} = (\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C})^T$  and  $\mathbf{T} = (\mathbf{Y}_1/\rho - \mathbf{E})(\mathbf{I} - \mathbf{C}^T)$ .

## References

- [1] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Technical Report, Intel Microprocessor Research Labs*, 2001. 1, 5
- [2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011. 4
- [3] A. Buchanan and A. Fitzgibbon. Combining local and global motion models for feature point tracking. In *CVPR*, 2007. 1, 2
- [4] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. 8
- [5] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998. 2
- [6] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *PAMI*, 35(11):2765–2781, 2013. 2, 3, 4
- [7] R. Garg, L. Pizarro, D. Rueckert, and L. Agapito. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In *ACCV*, 2010. 2
- [8] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *IJCV*, 104(3):286–314, 2013. 2
- [9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 7
- [10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 3
- [11] P. Ji, H. Li, M. Salzmann, and Y. Dai. Robust motion segmentation with unknown correspondences. In *ECCV*, 2014. 5
- [12] P. Ji, M. Salzmann, and H. Li. Efficient dense subspace clustering. In *WACV*, 2014. 2, 3, 4
- [13] P. Ji, M. Salzmann, and H. Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *ICCV*, 2015. 2
- [14] P. Ji, Y. Zhong, H. Li, and M. Salzmann. Null space clustering with applications to motion segmentation and face clustering. In *ICIP*, 2014. 2
- [15] H. Li. Two-view motion segmentation from linear programming relaxation. In *CVPR*, 2007. 2
- [16] Z. Li, J. Guo, L.-F. Cheong, and S. Z. Zhou. Perspective motion segmentation via collaborative clustering. In *ICCV*, 2013. 2, 3, 6, 8
- [17] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *PAMI*, 35(1):171–184, 2013. 4
- [18] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981. 1
- [19] T. Piccini, M. Persson, K. Nordberg, M. Felsberg, and R. Mester. Good edgels to track: Beating the aperture problem with epipolar geometry. In *ECCV Workshops*, 2014. 2
- [20] B. Poling, G. Lerman, and A. Szlam. Better feature tracking through subspace constraints. In *CVPR*, 2014. 1, 2, 3, 5, 8
- [21] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 1
- [22] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 2
- [23] C. Tomasi and T. Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991. 1
- [24] L. Torresani and C. Bregler. Space-time tracking. In *ECCV*, 2002. 1, 2
- [25] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *CVPR*, 2007. 2, 6
- [26] L. Valgaerts, A. Bruhn, and J. Weickert. A variational model for the joint recovery of the fundamental matrix and the optical flow. In *Pattern Recognition*, pages 314–324, 2008. 2
- [27] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *PAMI*, 27(12):1945–1959, 2005. 2
- [28] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. Segmentation of dynamic scenes from the multibody fundamental matrix. In *CVPR*, 2001. 2
- [29] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *ICCV*, 2009. 2
- [30] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality TV-L1 flow with fundamental matrix prior. In *IVCNZ*, 2008. 2

- [31] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV*, 2006. [2](#)