Motion from Structure (MfS): Searching for 3D Objects in Cluttered Point Trajectories

Jayakorn Vongkulbhisal[†,‡], Ricardo Cabral[†,‡], Fernando De la Torre[‡], João P. Costeira[†]
[†]ISR - Instituto Superior Técnico, Lisboa, Portugal
[‡]Carnegie Mellon University, Pittsburgh, PA, USA
jvongkul@andrew.cmu.edu, rcabral@cmu.edu, ftorre@cs.cmu.edu, jpc@isr.ist.utl.pt

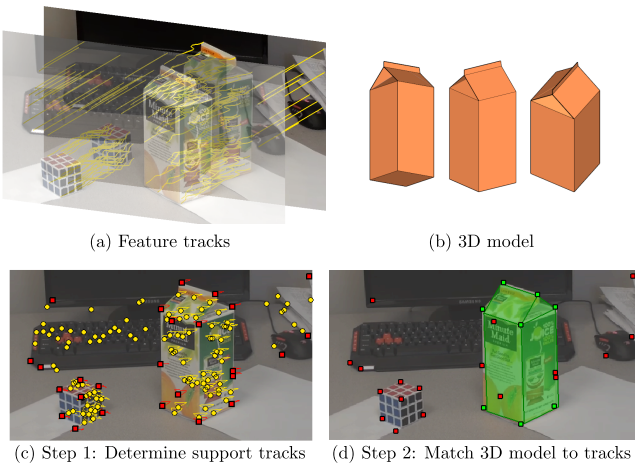
Abstract

Object detection has been a long standing problem in computer vision, and state-of-the-art approaches rely on the use of sophisticated features and/or classifiers. However, these learning-based approaches heavily depend on the quality and quantity of labeled data, and do not generalize well to extreme poses or textureless objects.

*In this work, we explore the use of 3D shape models to detect objects in videos in an unsupervised manner. We call this problem Motion from Structure (MfS): given a set of point trajectories and a 3D model of the object of interest, find a subset of trajectories that correspond to the 3D model and estimate its alignment (*i.e.*, compute the motion matrix). MfS is related to Structure from Motion (SfM) and motion segmentation problems: unlike SfM, the structure of the object is known but the correspondence between the trajectories and the object is unknown; unlike motion segmentation, the MfS problem incorporates 3D structure, providing robustness to tracking mismatches and outliers. Experiments illustrate how our MfS algorithm outperforms alternative approaches in both synthetic data and real videos extracted from YouTube.*

1. Introduction

Object detection is one of the most common tasks that humans perform. We are constantly looking and detecting people, roads, chairs or automobiles. Yet, much of how we perceive objects so accurately and with so little apparent effort remains a mystery. Although much progress has been done in the last few years, we are still far away from developing algorithms that can match human performance. Most previous efforts in the area of object detection have emphasized 2D approaches on different types of features and classifiers. Features have included shape (*e.g.* contours) and/or appearance descriptors (*e.g.*, SIFT, deep learning features). Classifiers have included Support Vector Machines (SVM), Boosting or Gaussian Processes. While 2D approaches are fast and have been shown to scale well to a large number of objects, they typically depend heavily on texture regions, lack robustness to viewpoints and incorporate video information in a heuristic manner. Surprisingly, little attention has been paid to the problem of 3D object detection, especially since acquiring 3D models has become more accessible due to the ubiquity of RGBD cameras and crowd-sourced repositories such as 3D Warehouse [21, 29].



(a) Feature tracks (b) 3D model

(c) Step 1: Determine support tracks (d) Step 2: Match 3D model to tracks

Figure 1. Motion from Structure (MfS) problem: Given a set of point trajectories (a) and a 3D model of an object (b), find a subset of trajectories that are aligned with the model. Our approach splits this combinatorial problem in two parts. First, we reduce potential candidate matching points by finding trajectories that compose the convex hull of objects (red squares in (c)). Second, a subset of the remaining trajectories are aligned with the 3D model (d).

To address the limitations of 2D approaches for detection, this paper proposes a new problem that we call Motion from Structure (MfS). Given a set of point trajectories containing multiple objects and a 3D model of an object (*i.e.*, a point cloud), MfS finds a subset of trajectories that are well-aligned (*i.e.*, via motion matrix) with the 3D model. Con-

sider Fig. 1; Fig. 1a shows the point trajectories and Fig. 1b plots several 3D views of the object of interest to detect. The goal of MfS is to solve the alignment between the 3D model and its trajectories (see Fig. 1d). MfS differs from 2D approaches to object detection in several aspects. First, MfS provides a natural framework to incorporate geometry into object detection in video. It computes the motion using a 3D model, which makes it more robust to strong viewpoint changes. Second, the MfS problem is unsupervised and there is no need for expensive and error-prone labeling process. Finally, in contrast to image-based approaches which rely on similar appearance, MfS allows 3D models to be aligned to objects with the same shape while having no similarity in textures.

The main challenge of MfS lies in the lack of discriminative features (*i.e.*, it is an unsupervised method without any appearance features) to solve for correspondence between trajectories and the queried model. Recall that most detection problems [7, 36] use supervised detectors for parts of the objects. Since the number of trajectories can be significantly larger than the number of vertices representing the object, the problem is NP-hard in general. To alleviate the combinatorial problem, we proposed a two-step approach to the MfS problem: (1) Reducing the number of trajectories by considering only the tracked points in the convex hull of an object (see red squares in Fig. 1c). This provides a compact representation of objects in the scene. (2) Aligning the remaining tracks with the 3D model via a guided sampling approach (see Fig. 1d).

The MfS problem is related to two well-known problems in computer vision: Structure from Motion (SfM) and motion segmentation (MSeg). Similar to SfM, the MfS problem starts from point trajectories, but, unlike SfM, in MfS a 3D model of the object of interest is given. However, in the MfS problem the point trajectories that correspond to the 3D object are unknown. MfS is able to find not only the subset of trajectories corresponding to the object, but also its alignment (*i.e.*, the motion matrix). MfS also relates to the MSeg problem that performs grouping of trajectories corresponding to the same rigid object. MSeg may seem to be an intuitive approach to solve the MfS problem: we could use MSeg methods to group trajectories into objects, and then fit the 3D model to each set of points, and select the one with less error. However, each of these steps is prone to errors by itself, and as we will show in experimental validation, MSeg-based algorithms may lead to worse solutions for MfS than our approach. Moreover, in MSeg the number of clusters needs to be known a priori and they are susceptoble to outliers which are typically present due to background motion, camera blur, and noise. On the other hand, our approach does not perform clustering or 3D reconstruction.

2. Related Work

Our work has a direct connection to Structure from Motion (SfM) and Motion Segmentation (MSeg). In this section, we briefly review these two problems and other works that perform partial matching and pose estimation.

Structure from Motion (SfM) Let $\mathbf{W} \in \mathbb{R}^{2F\times n}$ (see notation[1]) be a matrix containing n feature tracks of an object in F image frames. Tomasi and Kanade [25] showed that under an orthographic camera model $\mathbf{W}$ lies in a four dimensional subspace, leading to the factorization

$$\mathbf{W} = \mathbf{MS}, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{2F\times 4}$ is the F-frame motion matrix and $\mathbf{S} \in \mathbb{R}^{4\times n}$ contains 3D homogeneous coordinates of the vertices. This factorization relates feature tracks in a sequence of 2D images to its 3D reconstruction. Since this factorization was first proposed, it has been extended to handle different camera models, articulated and non-rigid objects, outlier handling, missing data, and other feature types [1, 12, 13, 20, 28].

Motion Segmentation (MSeg) Given a matrix $\mathbf{W}$ containing tracks of multiple objects, the goal of MSeg is to group the tracks of different objects. Since the tracks of each object lie in a low dimensional subspace [25], subspace clustering techniques have been proposed as a solution to MSeg. Previous works in this direction include the use of shape interaction matrix [4], generalized principal component analysis [30], and principal subspace angles [31], with extensions to handle articulated and non-rigid objects, and also segmentation of dense trajectories [33]. Recent approaches such as sparse subspace clustering [6] and low rank representation [15] impose sparsity or low rank priors to recover the affinity matrix in a convex formulation. However, MSeg only clusters tracks into groups with low dimensional subspaces, and it is not clear how to incorporate 3D models of objects as additional information.

Track-based information processing Beyond reconstruction and MSeg, feature tracks have also been used for other higher level computer vision tasks, such as foreground object segmentation [24], action recognition [17], object detection [35], and object recognition [19]. However, most of the approaches to detection and recognition still require classifiers to be learned from training data, which have to be manually labeled and can be laborous to obtain.

Correspondence and pose estimation Instead of using labeled data, 3D models provide compact representations of

[1]Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ represents the i^{th} column of the matrix $\mathbf{X}$. x_{ij} denotes the scalar in the i^{th} row and j^{th} column of the matrix $\mathbf{X}$. All non-bold letters represent scalars. $\mathbf{1}_{m\times n}$, $\mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$ are matrices of ones and zeros. $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is an identity matrix. diag($\mathbf{X}$) and tr($\mathbf{X}$) denotes the vector of diagonal entries and the trace of matrix $\mathbf{X}$, respectively.

an object's shapes. To incorporate 3D models for object detection or recognition, pose estimation must be performed to align the models to the images. In the case of single images, several works utilize both appearance features (*e.g.* SIFT) and 3D shape as input [3, 21, 22, 37]. These approaches require the correspondence between appearance features of 3D models and the images to be established prior to estimating pose. However, without appearance features, there is much less cue for obtaining the correspondence, and hence both correspondence and pose must be solved simultaneously [5, 18].

In the case of videos, motion can be used to provide information to solve for pose and correspondence. Toshev *et al*. [27] proposed to synthesize different views from 3D models and match their silhouettes to a moving object segmented from a video. On the other hand, we aim to directly align 3D point clouds to feature tracks from videos using the subspace relation as in (1). When the number of tracks is the same as that of the model vertices, the correspondence can be obtained by permuting the nullspace of the shape matrix to match with the track matrix [11, 16]. However, there is no obvious way to extend these approaches to deal with extra tracks from videos. To deal with such cases, [34] included texture features as input and sought the correspondence that minimizes the nuclear norm of the selected tracks and features.

Recently, Zhou and De la Torre [36] proposed a method to select a subset of trajectories that aligns body configurations with a motion capture data. However, the selection of trajectories uses supervised information. Our work tackles MfS by reducing the candidate for matching to the convex hull of objects. Although the idea of using convex hulls has been used in image registration [8, 32], their focus was on registering two sets of 2D point clouds whereas we register the 3D structure by observing 2D tracks from videos.

Despite the name resemblance, we note that our problem is different from [14], which proposed a large-scale 3D reconstruction approach for rolling-shutter cameras, while our goal is to register 3D models onto 2D tracks.

3. The Motion from Structure Problem

The motion from structure (MfS) problem can be stated as follows. Given a measurement matrix

$$\mathbf{W} = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \cdots & \mathbf{x}_n^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \cdots & \mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1^F & \mathbf{x}_2^F & \cdots & \mathbf{x}_n^F \end{bmatrix} \in \mathbb{R}^{2F \times n}, \qquad (2)$$

where $\mathbf{x}_i^f \in \mathbb{R}^2$ is the 2D coordinate of the i^{th} feature track in frame f, and a shape matrix $\mathbf{S} \in \mathbb{R}^{4 \times m}$ containing m 3D vertices of an object of interest, find the subset of tracks in $\mathbf{W}$ that belongs to the object. MfS is different from MSeg as it does not require tracks to be clustered into groups beforehand, but rather finds a partial permutation matrix $\mathbf{P} \in \{0,1\}^{n \times m}$ and a motion matrix $\mathbf{M} \in \mathbb{R}^{2F \times 4}$ that aligns a given 3D object to a subset of tracks. Assuming no noise and no missing data, the relationship between $\mathbf{W}$, $\mathbf{P}$, and $\mathbf{S}$ is given by

$$\mathbf{WP} = \mathbf{MS}. \qquad (3)$$

Assuming that the motion is scaled orthographic, we can expand $\mathbf{M}$ as:

$$\mathbf{M} = \begin{bmatrix} \alpha^1 \overline{\mathbf{M}}^1 & \mathbf{b}^1 \\ \vdots & \vdots \\ \alpha^F \overline{\mathbf{M}}^F & \mathbf{b}^F \end{bmatrix}, \qquad (4)$$

where $\overline{\mathbf{M}}^f \in \mathbb{R}^{2\times 3}$, $\mathbf{b}^f \in \mathbb{R}^2$, and $\alpha^f \in \mathbb{R}$ are rotation, translation, and scaling components in frame f.

3.1. Problem Analysis

In this section, we analyze the challenges of the MfS problem.

Complexity due to large number of tracks: The objects in general videos typically contain complex texture resulting in large number of additional tracks (besides the tracks belonging to the 3D object of interest). Since we only use the shape of the object, MfS problem is formulated as a combinatorial problem, with increasing complexity with the number of trajectories.

Lack of appearance features: The lack of appearance features (*e.g.* SIFT) causes each track to be indistinct from others, and prevents us from obtaining putative matches between them and the 3D model. This difference sets MfS apart from general pose estimation problems where distinctive appearance features play a significant role as an initialization to obtain pose. Without these putative matching, pose estimation is in general an ill-posed problem as it is not clear how to obtain the alignment.

Self-occlusion of 3D model: Generally, a 3D object in a real scene is subject to self-occlusion. This means not all vertices of $\mathbf{S}$ in (3) should be matched to the tracks. Since spurious matches can easily lead to a bad alignments, how to handle self-occlusion is an important issue that needs to be addressed.

Coherent motion trajectories: Although the tracks are indistinct, coherent motion of tracks from the same objects allows their tracks to be clustered into groups (*e.g.* by MSeg). In addition, they also encode information about the shape of each object (*e.g.* as in (1)). These two properties provide enough structure for us to solve the MfS problem.

4. Solving MfS

In order to tackle MfS, we make two assumptions: (1) the target objects have independent motion from other ob-

jects in the scene, and (2) the camera model is scaled orthographic. These assumptions can well approximate typical scenes and are generally adopted in solving MSeg [6, 15]. However, MfS is different as it does not require all tracks to be clustered into groups, but rather selecting tracks belonging to a given shape. To this end, we propose a two-step approach. In the first step, we solve a convex optimization problem to select a subset of special tracks constituting the convex hulls of moving objects. We call these tracks Support Tracks (STs). Since STs constitute convex hull of objects, they are more likely to be matched to the target shape. Selecting STs before matching results in the reduction of both false matches and complexity of the problem. In the second step, we align the 3D model to STs using a guided sampling procedure, where the selected STs are more likely to come from the same object. The following section provides details on the cost function and the algorithms.

4.1. What is a good alignment?

Defining a good metric for alignment is critical for MfS, especially since the vertices in the given 3D model may not exactly match the tracks of the target object. Rather than solving for both $\mathbf{P}$ and $\mathbf{M}$ in (3), we say that an alignment is good when there exists a matrix $\mathbf{M}$ such that the convex hulls[2] of back-projection of $\mathbf{S}$ and all tracks that undergo the same motion[3] are similar (see Fig. 2):

$$\pi^f(\mathbf{W_M}) \sim \pi^f(\mathbf{MS}), \tag{5}$$

where $\pi^f(\cdot)$ returns the convex hull in frame f of the input track matrix, and $\mathbf{W_M}$ is a matrix containing columns of $\mathbf{W}$ that lie in span($\mathbf{M}$). Specifically, we use the intersection-over-union (IoU) between $\pi^f(\mathbf{W_M})$ and $\pi^f(\mathbf{MS})$ to measure alignment. This leads to the following optimization problem:

$$\max_{\mathbf{M}} \sum_{f=1}^{F} \frac{\text{Area}(\pi^f(\mathbf{W_M}) \cap \pi^f(\mathbf{MS}))}{\text{Area}(\text{conv}(\pi^f(\mathbf{W_M}) \cup \pi^f(\mathbf{MS})))}, \tag{6}$$
$$\text{subject to } \overline{\mathbf{M}}^f \, \overline{\mathbf{M}}^{f\top} = \mathbf{I}_2, f = 1, \ldots, F,$$

where the relation between $\overline{\mathbf{M}}^f$ and $\mathbf{M}$ is given in (4), conv$(\cdot)$ returns convex hull of the input set, and Area$(\cdot)$ returns the area of the input convex hull. The cost function in (6) possesses many desirable properties, namely it (1) uses motion and shape to measure alignment, (2) is insensitive to self-occlusion of the 3D model since alignment is measured based on regions, and (3) takes into account coherent motion of tracks. Note that $\mathbf{P}$ is not included in (6), but it can be recovered after obtaining $\mathbf{M}$.

[2]In general, other types of region can be used. However, since we are given point clouds without any relation between the vertices, convex hull provides the most detailed description of the region.

[3]We say a group of tracks undergoes the same motion when there is a motion matrix $\mathbf{M}$ that can well explain them (see (1)).

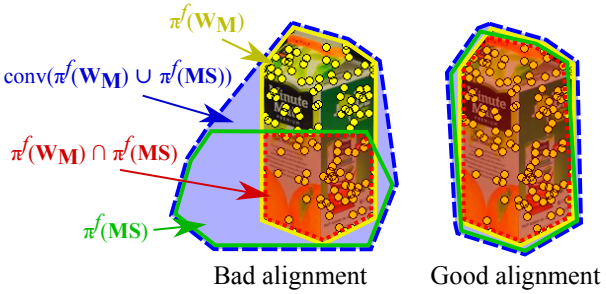


Figure 2. Example of good and bad alignments from a frame f. The yellow dots are coherent tracks. The convex hulls shown are of $\mathbf{W_M}$ (yellow), $\mathbf{MS}$ (green), their intersection (red), and union (blue). An alignment is good when the ratio between intersection and union is high. (Best viewed in color.)

Solving (6) requires estimating motion $\mathbf{M}$, which involves sampling 4 matches between tracks from $\mathbf{W}$ and vertices of $\mathbf{S}$. However, the total number of candidate pairs is in the order of $\mathcal{O}(n^4k^4)$, while the number of correct pairs can be significantly smaller. In the next section, we propose an approach to identify special tracks constituting visible part of objects' convex hull by using convex optimization. We refer to these tracks as support tracks (STs). Selecting STs as candidates for matching helps reduce problem complexity, while also retain useful tracks that are more likely to match to a given shape.

4.2. Identifying support tracks (STs)

To identify STs in $\mathbf{W}$, we exploit the idea that such tracks cannot be represented by convex combinations of tracks of the same object. Inspired by sparse subspace clustering (SSC) [6], we formulate the identification of STs and non-STs as

$$\min_{\mathbf{C},\mathbf{E}} \text{tr}(\mathbf{C}) + \mu f(\mathbf{E}) \tag{7}$$
$$\text{subject to } \mathbf{E} = \mathbf{W} - \mathbf{WC},$$
$$\mathbf{1}_n^\top \mathbf{C} = \mathbf{1}_n,$$
$$\mathbf{C} \geq \mathbf{0}_{n\times n},$$

where $\mathbf{C} \in \mathbb{R}^{n\times n}$ is the matrix containing coefficients for representing each track as a convex combination of others, and $\mathbf{E} \in \mathbb{R}^{2F\times n}$ allows for errors. The error function $f(\cdot)$ can be ℓ_1 norm, $\ell_{2,1}$ norm, or squared Frobenius norm. The intuition of (7) is akin to the *self-expressiveness* property in [6], which states that each vector in a subspace can be represented as a linear combination of others in the same subspace. By using *convex* combination rather than *linear*, STs can be identified as tracks that cannot be represented by other tracks in the same subspace. Fig. 3 illustrates the idea behind (7).

There are three subtle but non-trivial differences between (7) and SSC, in both the tasks they try to accomplish and the

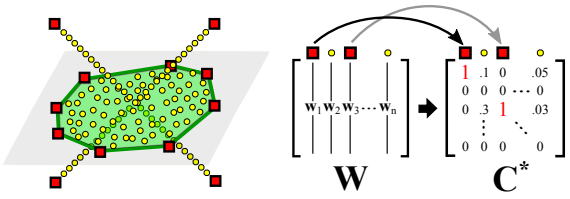


Figure 3. (Left) Since tracks (yellow circles) of independently moving objects lie in independent subspaces, any tracks that cannot be expressed as a convex combination of other tracks are considered STs (red squares) that constitute the convex hull of objects (see Fig. 1c). (Right) By solving (7), these STs induce 1's in the diagonal of $\mathbf{C}$.

variables involved. First, SSC was proposed to perform subspace clustering, where spectral clustering is performed on $\mathbf{C}$ to obtain the clusters. On the contrary, our formulation focuses on both $\mathbf{C}$ and $\mathbf{E}$, which encode the information for distinguishing STs from non-STs. Second, $\mathbf{C}$ is nonnegative and its columns sum to 1, which makes them *convex* combination coefficients instead of *linear* ones. As $\mathbf{C}$ is nonnegative, $\|\mathbf{C}\|_1$ in SSC's objective function becomes a constant value n, and thus can removed. Lastly, we penalize $\text{tr}(\mathbf{C})$ as opposed to restricting $\text{diag}(\mathbf{C}) = \mathbf{0}_n$. We show in Theorem 1 that at the global optimum $\text{diag}(\mathbf{C})$ can be evaluated as a binary vector and provides information on whether a track is a ST or not.

Theorem 1. *Let $\mathbf{C}^*$ and $\mathbf{E}^*$ be the optimal solutions of (7), and $\hat{\mathbf{E}}$ be the optimal solution of variable $\mathbf{E}$ of (7) with $\text{diag}(\mathbf{C}) = \mathbf{0}_n$ as an additional constraint. The relation between c^*_{ii}, μ, and $f(\hat{\mathbf{e}}_i)$ for any i can be stated as follows:*

$$c^*_{ii} \in \begin{cases} \{0\} & f(\hat{\mathbf{e}}_i) < \mu^{-1}, \\ [0,1] & f(\hat{\mathbf{e}}_i) = \mu^{-1}, \\ \{1\} & \textit{otherwise}. \end{cases} \tag{8}$$

Proof. It can be seen that (7) can be column-wise separated. For each column i, the objective is a tradeoff between c_{ii} and $\mu f(\hat{\mathbf{e}}_i)$. If $\mu f(\hat{\mathbf{e}}_i) > 1$ then it would cost less to set $c^*_{ii} = 1$ with $\mathbf{e}^*_i = \mathbf{0}_{2F}$. On the other hand, if $\mu f(\hat{\mathbf{e}}_i) < 1$ then setting $c^*_{ii} = 0$ and $\mathbf{e}^*_i = \hat{\mathbf{e}}_i$ would cost less. Finally, if $\mu f(\hat{\mathbf{e}}_i) = 1$, then the cost can be kept constant at 1 by letting c^*_{ii} be any values in $[0, 1]$ while $f(\mathbf{e}_i) = (1 - c^*_{ii}) f(\hat{\mathbf{e}}_i)$. □

Corollary 2. *For any μ, there exists an optimal solution of (7) where $\text{diag}(\mathbf{C}^*)$ is a binary vector.*

In Theorem 1, $\hat{\mathbf{e}}_i$ is the error of representing $\mathbf{w}_i$ with a convex combination of *strictly* other columns than $\mathbf{w}_i$. We can see that μ acts as a threshold for determining the value of c^*_{ii}. Specifically, if the error $f(\hat{\mathbf{e}}_i)$ is larger than μ^{-1}, it would cost less to represent $\mathbf{w}_i$ by itself, resulting in $c^*_{ii} = 1$. On the other hand, if the error is smaller than μ^{-1}, then it costs less for $\mathbf{w}_i$ to be represented by other columns, resulting in $c^*_{ii} = 0$. For our problem, this implies that $\mathbf{w}_i$ with $c^*_{ii} = 1$ is a ST, while $\mathbf{w}_i$ with $c^*_{ii} = 0$ is not.

By noting that μ is simply a threshold value, we can solve (7) for all values of μ in a single optimization run. This is done by solving (7) with $\text{diag}(\mathbf{C}) = \mathbf{0}_n$ as an additional constraint to obtain $\hat{\mathbf{E}}$. Since $c^*_{ii} = 1$ and $\mathbf{e}^*_i = \mathbf{0}_{2F}$ for all i with $f(\hat{\mathbf{e}}_i) > \mu^{-1}$, determining the order of tracks that gradually become STs as μ increases is equivalent to sorting $f(\hat{\mathbf{e}}_i)$ in a descending order. Thus, instead of setting the value μ, we can specify the number of desired STs p by selecting p tracks with highest $f(\hat{\mathbf{e}}_i)$.

After obtaining STs, we proceed to estimate motion matrix. To do so, we need to sample 4 STs and match them to 4 vertices from $\mathbf{S}$. Although the number of tracks is reduced from n to p, the total number of matches can still be very large. To increase the likelihood of selecting good matches, it would be beneficial to sample 4 STs that are more likely to come from the same object instead of uniform sampling.

4.3. Guided Sampling

We propose to sequentially sample STs using random walk strategy, where the transition probability of sampling ST i after ST j depends on how likely they are from the same object. In this work, we measure this value by representing each non-ST as a convex combination of only STs[4], and counting the number of non-STs that each pair of STs mutually represent.

To represent all non-STs by only STs, we leverage on the structure of $\mathbf{C}^*$ obtained from previous section; the constraints of (7) enforces $\mathbf{C}^*$ to be a stochastic matrix with p ones in its diagonal. In essence, $\mathbf{C}^*$ is a transition matrix of an absorbing Markov chain with p absorbing nodes corresponding to the p STs. From the Markov chain theory [9], the limiting matrix of $\mathbf{C}^*$:

$$\hat{\mathbf{C}} = \lim_{t \to \infty} (\mathbf{C}^*)^t, \tag{9}$$

can be interpreted as convex coefficients of representing non-STs by only STs. Let $d_{ik} = I(\hat{c}_{i'k} > 0)$ where i' is the index of ST i in the original $\mathbf{W}$, and $I(\cdot)$ is the indicator function which returns 1 if the argument is true and 0 otherwise. We calculate transition probability of sampling ST j after ST i using Jaccard similarity:

$$\Pr(i \to j) = \begin{cases} \frac{1}{z_i} \frac{\sum_{k=1}^{n} \min(d_{ik}, d_{jk})}{\sum_{k=1}^{n} \max(d_{ik}, d_{jk})} & ; i \neq j, \\ 0 & ; i = j, \end{cases} \tag{10}$$

where z_i is a normalization factor. Note that $\Pr(i \to j)$ is not equal to $\Pr(j \to i)$. For the 3D model, 4 vertices are

[4] By independent motion assumption, STs should represent only non-STs in the same subspace.

uniformly sampled to match the 4 sampled STs without any special procedure.

Given 4 matches between tracks and 3D vertices, we first estimate affine transformation between them, then project the rotation part of each frame to the Stiefel manifold (see supplementary material for more detail). The $\mathbf{M}$ that maximizes (6) can be used to recover $\mathbf{P}$ in (3) by selecting closest tracks in $\mathbf{W}$ to $\mathbf{MS}$.

The algorithm for MfS is summarized in Alg. (1).

Algorithm 1 Motion from Structure

Input: $\mathbf{W}, \mathbf{S}, \lambda_0, \mu, nIter$
Output: $\mathbf{P}, \mathbf{M}$

Step 1: Identifying STs
Compute $\mathbf{C}$ and $\mathbf{E}$ with (7)
Select columns of $\mathbf{W}$ indicated by $\text{diag}(\mathbf{C}) = 1$ as STs

Step 2: Guided Sampling
Compute $\hat{\mathbf{C}}$ with (9)
Compute transition matrix with (10)
for $i = 1$ **to** $nIter$ **do**
  Uniformly sample a ST
  Use guided sampling to obtain another 3 STs
  Uniformly sample 4 vertices from $\mathbf{S}$
  Estimate motion matrix $\mathbf{M}$
  Keep $\mathbf{M}$ if it has highest value in (6)
end for
Compute $\mathbf{P}$ by selecting tracks closest to $\mathbf{MS}$

4.4. Implementation details

This section provides the implementation details for each step of our algorithm.

Identifying STs: We use ℓ_1 norm as the error function $f(\cdot)$ in (7). The optimization problem is solved using Gurobi optimizer [10].

Guided sampling: In (9), rather than using eigendecomposition to compute $\hat{\mathbf{C}}$, we perform 1000 rounds of power iterations on $\mathbf{C}^*$, which results in a similar $\hat{\mathbf{C}}$ while being faster than eigendecomposition. To calculate $d_{i,k}$, it is likely that most $\hat{c}_{i'k}$ will not be zero. Instead, we set the threshold in $I(\cdot)$ to $2/p$, where p is the number of STs. During the sampling, we perform two steps of random walk before selecting a ST, and allow the walker to restart at any sampled STs with equal probability [26]. We prevent the sampled STs from being resampled by setting $\Pr(i \to j)$ to 0 for all sampled STs j.

5. Experiments

We evaluated our algorithm on synthetic data and real videos downloaded from YouTube. Recall that MfS is a new problem, and there is no existing baseline algorithm. Hence, we constructed two baselines to compare our algorithm against. (a) *All-R*: This baseline approach directly samples 4 tracks and 3D vertices uniformly to form the matches required for estimating $\mathbf{M}$. (b) *ST-R*: For this approach, we use (7) to first identify STs, then uniformly sample 4 track-vertex pairs. We refer to our ST selection and guided sampling as *STGS-R*.

5.1. Synthetic data

In this section, we used synthetic data to quantitatively evaluate the performance of our approach (see Fig. 4a). We generated a set of 15-frame tracks with three moving shapes: a cube (8 corners), a double pyramid (6 corners), and a cuboid (8 corners). A number of internal vertices (IVs) were generated randomly inside each shape to imitate non-STs. For each shape, we generated a motion matrix $\mathbf{M}_{gt}$ comprising rotation, translation, and scaling. 200 additional tracks are generated as background tracks. We added a single random translation to all tracks to simulate camera motion, and perturbed each track with Gaussian noise. To model outliers, we set the background tracks to follow the first shape that comes close to them. This is to imitate real scenarios when parts of background got occluded by moving objects, causing background tracks to follow foreground objects instead.

We evaluated our algorithm in two situations. First, we tested the robustness of our MfS against the density of tracks for each object with varying IVs for each object from 20 to 100 (corners inclusive). Second, we randomly removed tracks belonging to the corner of each shape with probability from 0 to 100 percent to imitate the situations where corner vertices may not be tracked. In the second case, we set the number of IVs to 100. The simulation was repeated 100 rounds for each setting. In all experiments, we selected 10% and 20% of tracks as STs for ST-R and STGS-R and sampled track-vectex pairs 5×10^4 time on each set of STs, while we sampled 1×10^5 times for All-R. The sample that yielded the highest cost in (6) was returned as the result of each algorithm.

We used three metrics to measure performance. The first metric is IoU between the projections due to $\mathbf{M}$ and $\mathbf{M}_{gt}$, defined as:

$$\text{IoU} = \sum_{f=1}^{F} \frac{\text{Area}(\pi^f(\mathbf{M}_{gt}\mathbf{S}) \cap \pi^f(\mathbf{MS}))}{\text{Area}(\text{conv}(\pi^f(\mathbf{M}_{gt}\mathbf{S}) \cup \pi^f(\mathbf{MS})))}. \quad (11)$$

The second metric is segmentation precision, defined as the number of correctly selected tracks against the total number of selected tracks (*i.e.* number of rows of $\mathbf{W}_\mathbf{M}$). The last metric is segmentation recall, defined as the number of correctly selected tracks against the number of tracks of the correct objects.

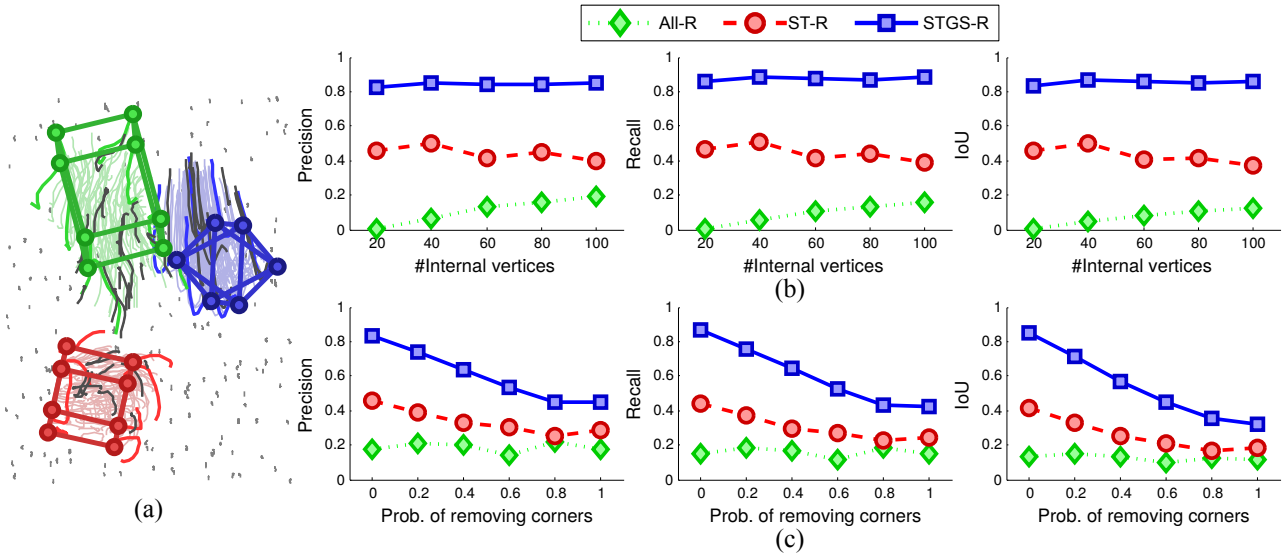


Figure 4. Synthetic experiment. (a) A synthetic scene comprising tracks from a cube (red), a double pyramid (blue), a cuboid (green) and background (gray). Outlier tracks are shown in dark gray. The performance was measured by varying (b) number of internal vertices, and (c) probability of having each corner track removed.

Fig. 4 shows the result of the synthetic experiment. The performance of All-R is very low, implying that randomly selecting matches is not a good approach to MfS. This is due to the challenging nature of the MfS that the ratio of good matches against all matches is very low. Improvements due to ST selection and guided sampling can be seen by the performance gaps between the three algorithms. STGS-R performs very well regardless of the number of IVs. On the other hand, since the shapes are similar, removing corners caused the performance to drop. In real cases, we would expect at least 50% of corners to be present, in which the performance is still in the acceptable range.

5.2. Real videos

In this section, we provide qualitative evaluation of our approach using videos collected by us and videos downloaded from YouTube (see Fig. 5). The videos are 10 to 97 frames long, and are extremely challenging since they contain multiple moving objects, camera motion, zoom changes, and perspective effects. These geometric changes cause the scenes to be very dynamic and difficult to track features over long periods of time. We detected the feature points using Shi-Tomasi feature detector [23] and used Lucas-Kanade tracker [2] to track, resulting in 490 to 1014 tracks for each video (see Col. 3, Fig. 5). Note that there are multiple outlier tracks caused by features corresponding to the occlusion boundary. We downloaded 3D models that approximate target objects from [29], and generated the shape matrix by manually selecting 8 to 18 vertices that represent well the convex hull of each 3D model (see Col. 2, Fig. 5). For the algorithm settings, we selected 10% of tracks as STs, and sampled 1×10^6 track-vertex pairs for all algorithms.

Col. 4 and 5 of Fig. 5, respectively, show the selected STs and alignment results for the real videos experiment. We encourage the readers to see the results in the supplementary material to appreciate the difficulty of the data. As can be seen in Col. 4 of Fig. 5, our ST selection can reliably select tracks on the convex hull of objects, which significantly reduced the complexity of the problem. This strategy combined with guided sampling and the appropriate cost function in (6) effectively allows to solve for the correspondence between the 3D models and trajectories. With only a few vertices representing convex hulls of objects, our MfS algorithm can correctly find an alignment under self occlusion without knowing which vertices are occluded. It also can handle different camera motion effects, and imprecise 3D models (Fig. 5, different models of harrier (row 2) and cars (row 5)). Due to space restriction, we include in the supplementary document additional results with comparison to baselines, and several examples where MSeg-based approaches cannot solve this problem. It is important to remind the reader that MfS is a different problem from MSeg, but we use MSeg as baseline since the problems are related.

While the method has worked well in most of the sequences that we have tried, the last row of Fig. 5 shows a failure case of our algorithm. In this case, the camera was panning from left to right, allowing only a portion of the background to be fully tracked. Rather than aligning to one of the cars, the 3D model is aligned to the background tracks because they coincidentally form a rectangular shape similar to the top view of the car's 3D model. In such cases, only tracks may not provide enough information to obtain the correct result. Additional information, such as expected

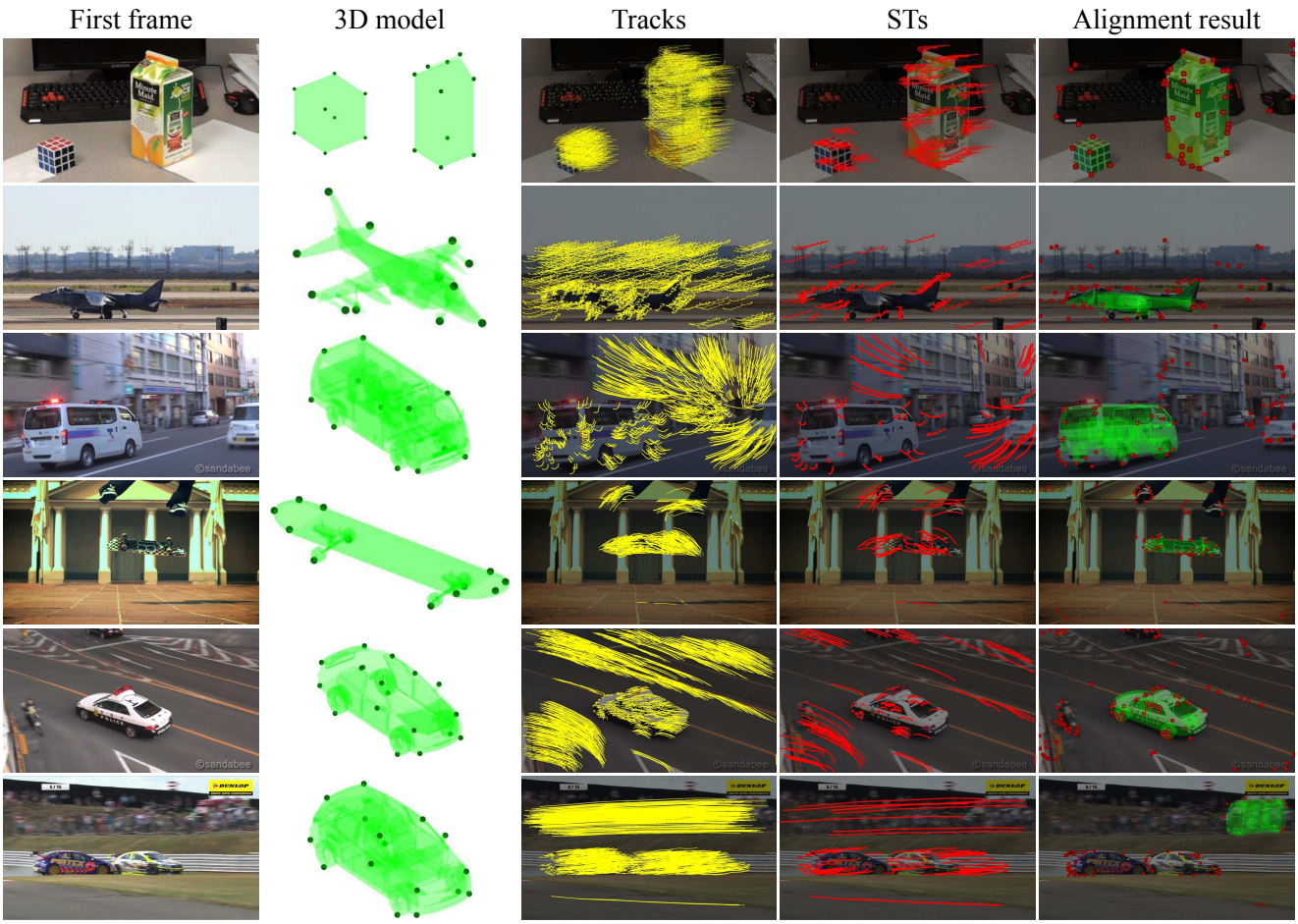


Figure 5. Results from a real video experiment. Column 1: First frame of the videos. Column 2: 3D models with the vertices represented by $\mathbf{S}$ shown in dark green dots. Column 3: Tracks are shown with yellow lines. Recall that we only use tracks and 3D models as input, not the images. Column 4: Selected STs. Column 5: Results of alignment by backprojecting 3D models to the video. The STs in this frame are shown in red points. We reduce the image intensity of columns 3 to 5 for visualization purpose. (Best viewed in color.)

orientation (*e.g*. wheels should point down), can be incorporated to reject wrong solutions during estimation of $\mathbf{M}$.

6. Conclusions

This paper proposes a new problem, Motion from Structure (MfS), where a given 3D model is aligned in an unsupervised manner to tracks in video. A two-step approach is proposed where convex hulls of objects are discovered in the first step, then guided sampling is performed for alignment in the second step. Our approach does not require the segmentation and 3D reconstruction of objects, and thus allows for bypassing their drawbacks. We tested our approach on synthetic data and real videos, and showed that it outperformed baseline approaches.

One limitation of using convex hulls for alignment is that the convex hull of some objects may be symmetric, which may lead to incorrect alignment due to oversimplification of shapes. We also notice that some corners were not identified as STs. This occurs due to (1) close proximity between tracks allowing them to well represent each other, and (2) violation of independent subspace assumption. Further improvements also include incorporating other information (*e.g*. orientation), and handling incomplete tracks (*i.e*. missing data). We will address these issues in future works.

Acknowledgments

This research was partially supported by Fundação para a Ciência e a Tecnologia (project FCT [UID/EEA/50009/2013] and a PhD grant from the Carnegie Mellon-Portugal program).

References

[1] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *TPAMI*, 33(7):1442–1456, 2011. 2

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004. 7

[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *ICRA*, 2009. 3

[4] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998. 2

[5] P. David, D. Dementhon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *IJCV*, 59(3):259 – 284, 2004. 3

[6] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *TPAMI*, 35(11):2765–2781, 2013. 2, 4

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 2

[8] A. Goshtasby and G. C. Stockman. Point pattern matching using convex hull edges. *IEEE Trans. Systems, Man, and Cybernatics*, 15(5):631–637, 1985. 3

[9] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 1997. 5

[10] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2015. 6

[11] Y. Igarashi and K. Fukui. 3D object recognition based on canonical angles between shape subspaces. In *ACCV*, 2010. 3

[12] D. W. Jacobs. Linear fitting with missing data for structure-from-motion. *CVIU*, 82:206–2012, 1997. 2

[13] Q. Ke and T. Kanade. Robust L1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR*, 2005. 2

[14] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *ICCV*, 2013. 3

[15] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *TPAMI*, 35(1):171–184, 2013. 2, 4

[16] M. Marques, M. Stošić, and J. Costeira. Subspace matching: Unique solution to point matching with geometric constraints. In *ICCV*, 2009. 3

[17] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV*, 2009. 2

[18] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose priors for simultaneously solving alignment and correspondence. In *ECCV*, 2008. 3

[19] B. Ommer, T. Mader, and J. M. Buhmann. Seeing the objects behind the dots: Recognition in videos from a moving camera. *IJCV*, 83(1):57–71, 2009. 2

[20] L. Quan and T. Kanade. A factorization method for affine structure from line correspondences. In *CVPR*, 1996. 2

[21] S. Satkin, J. Lin, and M. Hebert. Data-driven scene understanding from 3D models. In *BMVC*, 2012. 1, 3

[22] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *ICCV*, 2007. 3

[23] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 7

[24] S. W. Sun, Y. C. F. Wang, F. Huang, and H. Y. M. Liao. Moving foreground object detection via robust sift trajectories. *Journal of Visual Communication and Image Representation*, 24(3):232–243, 2013. 2

[25] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992. 2

[26] H. Tong, C. Faloutsos, and J. Y. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006. 6

[27] A. Toshev, A. Makadia, and K. Daniilidis. Shape-based object recognition in videos using 3d synthetic object models. In *CVPR*, 2009. 3

[28] P. Tresadern and I. Reid. Articulated structure from motion by factorization. In *CVPR*, 2005. 2

[29] Trimble Navigation Ltd. 3D warehouse. http://3dwarehouse.sketchup.com/. 1, 7

[30] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *TPAMI*, 27(12):1945–1959, 2005. 2

[31] J. Yan and M. Pollefeys. A factorization-based approach for articulated non-rigid shape, motion and kinematic chain recovery from video. *TPAMI*, 30(5):865–877, 2008. 2

[32] X. Y. Yu, H. Sun, and J. Chen. Points matching via iterative convex hull vertices pairing. In *Int. Conf. Machine Learning and Cybernatics*, 2005. 3

[33] L. Zelnik-Manor, M. Machline, and M. Irani. Multibody factorization with uncertainty: Revisiting motion consistency. *IJCV*, 68(1):27–41, 2006. 2

[34] Z. Zeng, T. H. Chan, K. Jia, and D. Xu. Finding correspondence from multiple images via sparse and low-rank decomposition. In *ECCV*, 2012. 3

[35] M. Zhai, L. Chen, J. Li, M. Khodabandeh, and G. Mori. Object detection in surveillance video from dense trajectories. In *MVA*, 2015. 2

[36] F. Zhou and F. De la Torre. Spatio-temporal matching for human detection in video. In *ECCV*, 2014. 2, 3

[37] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3D representations for object recognition and modeling. *TPAMI*, 35:2608–2623, 2013. 3