

Better Feature Tracking Through Subspace Constraints

Bryan Poling
University Of Minnesota
poli0048@umn.edu

Gilad Lerman
University Of Minnesota
lerman@umn.edu

Arthur Szlam
City College of New York
aszlam@ccny.cuny.edu

Abstract

Feature tracking in video is a crucial task in computer vision. Usually, the tracking problem is handled one feature at a time, using a single-feature tracker like the Kanade-Lucas-Tomasi algorithm, or one of its derivatives. While this approach works quite well when dealing with high-quality video and “strong” features, it often falters when faced with dark and noisy video containing low-quality features. We present a framework for jointly tracking a set of features, which enables sharing information between the different features in the scene. We show that our method can be employed to track features for both rigid and non-rigid motions (possibly of few moving bodies) even when some features are occluded. Furthermore, it can be used to significantly improve tracking results in poorly-lit scenes (where there is a mix of good and bad features). Our approach does not require direct modeling of the structure or the motion of the scene, and runs in real time on a single CPU core.

1. Introduction

Feature tracking in video is an important computer vision task, often used as the first step in finding structure from motion or simultaneous location and mapping (SLAM). The celebrated Kanade-Lucas-Tomasi algorithm [20, 25, 24] tracks feature points by searching for matches between templates representing each feature and a frame of video. Despite many other alternatives and improvement, it is still one of the best video feature tracking algorithms [1].¹ However, there are several realistic scenarios when Lucas-Kanade and many of its

¹Feature tracking should be distinguished from object tracking, where there has been significant progress in the development of novel algorithms that significantly improve previous efforts.

Acknowledgements: This work was supported by NSF award DMS-09-56072, the Sloan foundation, the University of Minnesota Doctoral Dissertation Fellowship Program, and the Feinberg Foundation Visiting Faculty Program Fellowship of the Weizmann Institute of Science.

Supp. web page: <http://www.math.umn.edu/~lerman/RCTracking/>

alternatives do not perform well: poor lighting conditions, noisy video, and when there are transient occlusions that need to be ignored. In order to deal with such scenarios more robustly it would be useful to allow the feature points to communicate with each other to decide how they should move as a group, so as to respect the underlying three dimensional geometry of the scene.

This underlying geometry constrains the trajectories of the track points to have a low-rank structure; see [12, 18] for the case when tracking a single rigid object under an affine camera model, and [6, 28, 17, 13] for non-rigid motion and the perspective camera. In this work we will combine the low-rank geometry of the cohort of tracked features with the successful non-linear single feature tracking framework of Lucas and Kanade [20] by adding a low-rank regularization penalty in the tracking optimization problem. To accommodate dynamic scenes with non-trivial motion we apply our rank constraint over a sliding window, so that we only consider a small number of frames at a given time (this is a common idea for dealing with non-rigid motions [8, 23, 16]). We demonstrate very strong performance in rigid environments as well as in scenes with multiple and/or non-rigid motion (since the trajectories of all features are still low rank for short time intervals). We describe experiments with several choices of low-rank regularizers (which are local in time), using a unified optimization framework that allows real time regularized tracking on a single CPU core.

1.1. Relationship With Previous Work

Geometric structures (and low-rank structures in particular) have been effectively utilized for the problem of optical flow estimation. Irani [19] showed how 3-d constraints in the real world and various camera models imply the existence of a low-rank constraint on the flow problem. Brand extended Irani’s work to non-rigid motions, while developing a robust, subspace-estimating, flow-based tracker via an incremental singular value decomposition with missing values as well as by learning an object model [5, 4, 3, 6]. Torresani et al. [28] also extended Irani’s work to non-rigid motions by applying rank-bounds for recovering 3D non-rigid motions. More recently, Garg

et al. [15] introduced hard subspace constraints for long range optical flow estimation in a variational scheme. Garg et al. [16] improved the performance of this former work by making the constraint weak (as an energy regularizer), using a robust energy term and allowing more general basis terms. At last, Ricco and Tomasi [23] proposed a Lagrangian approach for long range motion estimation that allows more reliable detection of occlusion. It estimates a basis for a low-dimensional subspace of the trajectories (as in [16]) and employs a variational method to solve for the best-fit coefficients of the motion trajectories in this basis.

In optical flow estimation the goal is to find displacements of features between consecutive frames, while assuming that the flow field is locally nearly constant. Although the goal in the feature tracking problem is similar, it does not require estimating the flow by enforcing the brightness constancy constraint or a weaker version of it. The subspace constraints above were translated by Irani [19] to an image brightness constraint. However, small errors in the flow field in each frame from this approach lead to the accumulation of errors in the trajectories obtained by integrating the flow. These errors are unacceptable for tracking. Weak versions of this constraint for estimating flow along many frames (as in [6, 23, 16]) require rather dense trajectories, which represent continuous regions in the image frame. Indeed, they are based on either continuous variational methods [23, 16] (which often track all pixels in the image domain) or careful model estimation [6] (which requires sufficiently dense sampling from objects in the videos).

In tracking, one instead uses a formulation that allows for very precise feature registration (like the Lucas-Kanade tracker [20]), and there is no need to linearize the image to solve an approximation to the feature displacement problem. It is desirable to have a sparse set of features and track them only in local neighborhoods to allow real time implementation. There is not a canonical method for introducing an explicit low rank constraint as in [19]. We will argue below that any strict subspace constraint is not ideal in the tracking problem and will promote a soft constraint. This soft constraint is different than the ones advocated for flow estimation in both [23] and [16] since they carefully learn local basis elements and require dense feature sampling.

Torresani and Bregler [26] suggested the partial application of hard low-rank constraints to improve tracking (applying rank bounds as in [28]). They rely on initial Lucas-Kanade tracking [20] from which “reliable” features are identified and used to estimate a model for the scene. They used their constraint to re-track the “unreliable” features (the trajectories are now confined to a known subspace). Since they search in the space of trajectories, their minimization strategy is completely different than ours. Their tracker is also non-casual since it needs the full sequence to start tracking, so a real-time implementation is not possible.

This work was extended in [27] to develop a causal tracker in the same spirit that also does not rely on a set of “reliable” feature tracks. However, both methods require setting the rank of the constraint a-priori and they impose the constraint over very long time spans (up to the entire sequence), making the algorithms less applicable to dynamic scenes.

Buchanan and Fitzgibbon [8] continuously update a non-rigid motion model over a sliding temporal window. This motion model is used as a motion prior in a conventional Bayesian template tracker for a single feature. The local information is combined with weaker global low rank approximation for the set of initial local trajectories (in the spirit of [6, 23, 16], while different than the low rank constraint of this paper). Similarly to [6] this low rank constraint guides the tracking via Bayesian modeling.

Another line of work takes tracked feature points in videos, and then uses the underlying subspace structure of rigid bodies to segment different motions of such bodies [12, 30, 11, 31, 14]. This is related to the large body of work on recovering rigid or non-rigid structure from motion; see [18] or [13] and the references therein. However, these works are highly dependent on good tracking and it would be desirable to simultaneously track and segment motion, or exploit the subspace structure to improve tracking prior to finding structure from motion.

2. On Low-Rank Feature Trajectories

Under the affine camera model, the feature trajectories for a set of features from a rigid body should exist in an affine subspace of dimension 3, or a linear subspace of dimension 4 [12, 18]. However, subspaces corresponding to very degenerate motion are lower-dimensional than those corresponding to general motion [18].

Feature trajectories of non-rigid scenarios exhibit significant variety, but some low-rank models may still be successfully applied to them [6, 28, 17, 13, 16]. Similarly to [8, 16] (though in a different setting) we consider a sliding temporal window, where over short durations the motion is simple and the feature trajectories are of lower rank. The restriction on the length of feature trajectories can also help in satisfying an approximate local affine camera model in scenes which violate the affine camera model. In general, depth disparities give rise to low-dimensional manifolds [18] which are only locally approximated by linear spaces.

At last, even in the case of multiple moving rigid objects, the set of trajectories is still low rank (confined to the union of a few low rank subspaces). In all of these scenarios the low rank is unknown in general.

3. Feature Tracking

Notation: A *feature* at a location $z_1 \in \mathbb{R}^2$ in a given $N_1 \times N_2$ frame of an $N_1 \times N_2 \times N_3$ video is characterized

by a *template* T , which is an $n \times n$ sub-image of that frame centered at z_1 (n is a small integer, generally taken to be odd, so the template has a center pixel). If z_1 does not have integer coordinates, T is interpolated from the image. We denote $\Omega = \{1, \dots, n\} \times \{1, \dots, n\}$ and we parametrize T so that its pixel values are obtained by $\{T(\mathbf{u})\}_{\mathbf{u} \in \Omega}$.

A classical formulation of the single-feature tracking problem (see e.g., [20]) is to search for the translation \mathbf{x}_1 that minimizes some distance between a feature’s template T at a given frame and the next frame of video translated by \mathbf{x}_1 ; we denote this next frame by I . That is, we minimize the *single-feature energy function* $c(\mathbf{x}_1)$:

$$c(\mathbf{x}_1) = \frac{1}{n^2} \sum_{\mathbf{u} \in \Omega} \psi(T(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_1)), \quad (1)$$

where, for example, $\psi(x) = |x|$ or $\psi(x) = x^2$. To apply continuous optimization we view \mathbf{x}_1 as a continuous variable and we thus view T and I as functions over continuous domains (implemented with bi-linear interpolation).

3.1. Low Rank Regularization Framework

If we want to encourage a low rank structure in the trajectories, we cannot view the tracking of different features as separate problems. For $f \in \{1, 2, \dots, F\}$, let \mathbf{x}_f denote the position of feature f in the current frame (in image coordinates), and let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_F) \in \mathbb{R}^{2F}$ denote the joint state of all features in the scene. We define the total energy function as follows:

$$C(\mathbf{x}) = \frac{1}{Fn^2} \sum_{f=1}^F \sum_{\mathbf{u} \in \Omega} \psi(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f)), \quad (2)$$

where $T_f(\mathbf{u})$ is the template for feature f . Now, we can impose desired relationships between features in a scene by imposing constraints on the domain of optimization of (2).

Instead of enforcing a hard constraint, we add a *penalty term* to (2), which increases the cost of states which are inconsistent with low-rank motion. Specifically, we define:

$$\bar{C}(\mathbf{x}) = \alpha \sum_{f=1}^F \sum_{\mathbf{u} \in \Omega} \psi(T_f(\mathbf{u}) - I(\mathbf{u} + \mathbf{x}_f)) + P(\mathbf{x}), \quad (3)$$

where $P(\mathbf{x})$ is an estimate of, or proxy for, the dimensionality of the set of feature trajectories over the last several frames of video (past feature locations are treated as constants, so this is a function only of the current state, \mathbf{x}). Notice that we have replaced the scale factor $1/(Fn^2)$ from (2) with the constant α , as this coefficient is now also responsible for controlling the relative strength of the penalty term. We will give explicit examples for P in section 3.2.

This framework gives rise to two different solutions, characterized by the strength of the penalty term (definition

of α). Each has useful, real-world tracking applications. In the first case, we assume that most (but not necessarily all) features in the scene approximately obey a low rank model. This is appropriate if the scene contains non-rigid or multiple moving bodies. We can impose a *weak constraint* by making the penalty term small relative to the other terms. If a feature is strong, it will confidently track the imagery, ignoring the constraint (regardless of whether the motion is consistent with the other features in the scene). If a feature is weak in the sense that we cannot fully determine its true location by only looking at the imagery, then the penalty term will become significant and encourage the feature to agree with the motion of the other features in the scene.

In the second case, we assume that all features in the scene are supposed to agree with a low rank model (and deviations from that model are indicative of tracking errors). We can impose a strong constraint by making the penalty term large relative to the other terms. No small set of features can overpower the constraint, regardless of how strong the features are. This forces all features to move in a way that is consistent with a simple motion. Thus, a small number of features can even be occluded, and their positions will be predicted by the motion of the other features in the scene. We further explain these two scenarios and demonstrate them with figures in the supplementary material.

3.2. Specific Choices of the Low-Rank Regularizer

There is now a large body of work on low rank regularization, e.g., [10, 9, 21]. We will restrict ourselves to showing results using three choices for P described below. Each choice we present defines $P(\mathbf{x})$ in terms of a matrix \mathbf{M} . It is the $2(L+1) \times F$ matrix whose column f contains the feature trajectory for feature f within a sliding window of $L+1$ consecutive frames (current frame and L past frames). Specifically, $\mathbf{M} = [m_{i,j}]$, where $(m_{0,f}, m_{1,f})^T$ is the current (variable) position of feature f and $(m_{2l+1,f}, m_{2l+2,f})^T, l = 1, \dots, L$ contains the x and y pixel coordinates of feature f from l frames in the past (past feature locations are treated as known constants). One may alternatively center the columns of \mathbf{M} by subtracting from each column the average of all columns. Most constraints derived for trajectories (assuming, for instance, rigid motion) actually confine trajectories to a low rank *affine* subspace (as opposed to a *linear* subspace). Centering the columns of \mathbf{M} transforms an affine constraint into a linear one. Alternatively, one can forgo centering and view an affine constraint as a linear constraint in one dimension higher. We report results for both approaches.

Explicit Factorizations

A simple method for enforcing the structure constraint is to write $\mathbf{M} = \mathbf{BC}$, where \mathbf{B} is a $2(L+1) \times d$ matrix, and \mathbf{C} is

a $d \times F$ matrix. However, as mentioned in the previous section, because the feature tracks often do not lie exactly on a subspace due to deviations from the camera model or non-rigidity, an explicit constraint of this form is not suitable.

However, an explicit factorization can be used in a penalty term by measuring the deviation of M , in some norm, from its approximate low rank factorization. For example, if we let

$$M = U\Sigma V^T \quad (4)$$

denote the SVD of M , we can take $P(x)$ in (3) to be $\|BC - M\|_*$, where B is the first three or four columns of U , and C is the first three or four rows of ΣV^T . Then this P corresponds to penalizing M via $\sum_{i=d+1}^F \sigma_i$, where $\sigma_i = \Sigma_{ii}$ is the i 'th singular value of M . As above, since the history is fixed, U , Σ , and V^T are functions of x .

This approach is the closest analogue of [19] in the tracking setting, next to an explicit rank constraint. It assumes knowledge of the low-rank d . For simplicity, we assume a local rigid model and thus set $d = 3$ when centering M and $d = 4$ when not centering (following [12, 18]).

Nuclear Norm

A popular alternative to explicitly keeping track of the best fit low-dimensional subspace to M is to use the matrix nuclear norm and define

$$P(x) = \|M\|_* = \|\sigma\|_1. \quad (5)$$

This is a convex proxy for the rank of M (see e.g., [10, 9]). Here $\sigma = (\sigma_1 \ \sigma_2 \ \dots \ \sigma_{2(L+1) \wedge F})^T$ is the vector of singular values of M , and $\|\cdot\|_1$ is the l_1 norm. Unlike explicit factorization, where only energy outside the first d principal components of M is punished, the nuclear norm will favor lower-rank M over higher-rank M even when both matrices have rank $\leq d$. Thus, using this kind of penalty will favor simpler track point motions over more complex ones, even when both are technically permissible.

Empirical Dimension

Empirical Dimension [22] refers to a class of dimension estimators depending on a parameter $\epsilon \in (0, 1]$. The empirical dimension of M is defined to be:

$$\hat{d}_\epsilon(M) := \frac{\|\sigma\|_\epsilon}{\|\sigma\|_{\left(\frac{\epsilon}{1-\epsilon}\right)}}. \quad (6)$$

Notice that we use norm notation, although $\|\cdot\|_\epsilon$ is only a pseudo-norm. When $\epsilon = 1$, this is sometimes called the ‘‘effective rank’’ of the data matrix [29].

Empirical dimension satisfies a few important properties, which are verified in [22]. First, empirical dimension

is invariant under rotation and scaling of a data set. Additionally, in the absence of noise, empirical dimension never exceeds true dimension, but it approaches true dimension as the number of measurements goes to infinity for spherically symmetric distributions. Thus, d_ϵ is a true dimension estimator (whereas the nuclear norm is a proxy for dimension). To use empirical dimension as our regularizer, we define $P(x) = d_\epsilon(M)$.

Empirical dimension is governed by its parameter, ϵ . An ϵ near 0 results in a ‘‘strict’’ estimator, which is appropriate for estimating dimension in situations where you have little noise and you expect your data to live in true linear spaces. If ϵ is near 1 then d_ϵ is a lenient estimator. This makes it less sensitive to noise, and more tolerant of data sets that are only approximately linear. In all of the experiments we present, we use $\epsilon = 0.6$, although we found that other tested values also worked well.

3.3. Implementation Details

We fix $L = 10$ for the sliding window and let $\psi(x) = |x|$ in (3). We use this form for ψ so that all terms in the total energy function behave linearly in a known range of values. If our fit terms behaved quadratically, it would be more challenging to balance them against a penalty term. We also tested a Huber loss function for ψ and have concluded that such a regularization is not needed.

We fix a parameter m for each penalty form (selected empirically - see the supplementary material for our procedure), which determines the strength of the penalty. The weak and strong regularization parameters are set as follows:

$$\alpha_{weak} = \frac{1}{mn^2} \quad \text{and} \quad \alpha_{strong} = \frac{1}{mFn^2}. \quad (7)$$

The weak scaling implies that a perfectly-matched feature will contribute 0 to the total energy, and a poorly-matched feature will contribute an amount on the order of $1/m$ to the total energy. The penalty term will contribute on the order of 1 to the total energy. Since we do not divide the contributions of each feature by the number of features, the penalty terms contribution is comparable in magnitude to that of a single feature. The strong scaling implies that the penalty term is on the same scale as the sum of the contributions of all of the features in the scene.

Minimization Strategy

The total energy function we propose for constrained tracking is non-convex since the contributions from the template fit terms are not convex (even if P is convex); this is also the case with other feature tracking methods, including the Lucas-Kanade tracker. We employ a 1st-order descent approach for driving the energy to a local minimum.

To reduce the computational load of feature tracking, some trackers use 2nd-order methods for optimization (see

[1]). This works well when tracking strong features, but in our experience it can be unreliable when dealing with weak or ambiguous features. Since we are explicitly trying to improve tracking accuracy on poor features we opt for a 1st-order descent approach instead.

The simplest 1st-order descent method is (sub)gradient descent. Unfortunately, because there can be a very large difference in magnitude between the contributions of strong and weak features to our total energy, our problem is not well-conditioned. If we pursue standard gradient descent, the strong features dictate the step direction and the weak features have very little effect on it. Ideally, once the strong features are correctly positioned, they will no longer dominate the step direction. If we were able to perfectly measure the gradient of our objective function, this would be the case. In practice, the error in our numerical gradient estimate can be large enough to prevent the strong features from ever relinquishing control over the step direction. The result is that in a scene with both very strong and very weak features, the weak features may not be tracked.

To remedy this, we compute our step direction by blending the gradient of the energy function with a vector that corresponds to taking equal-sized gradient descent steps separately for each feature. We use a fast line search in each iteration to find the nearest local minimum in the step direction. This compromise approach allows for efficient descent while ensuring that each feature has some control over the step direction (regardless of feature strength).

Because the energy is not convex, it is important to choose a good initial state. We use a combination of two strategies to initialize the tracking: first, we generate our initial guess of \mathbf{x} by registering an entire frame of video with the previous (at lower resolution). Secondly, we use multi-resolution, or pyramidal tracking so that approximate motion on a large scale can help us get close to the minimum before we try tracking on finer resolution levels (see [2]).

We now explain the details of the algorithm. Let \mathbf{I} denote a full new frame of video and let \mathbf{x}^{prev} be the concatenation of feature positions in the previous frame. We form a pyramid for \mathbf{I} where level 0 is the full-resolution image and each higher level m (1 through 3) has half the vertical and half the horizontal resolution of level $m - 1$. To initialize the optimization, we take the full frame (at resolution level 3) and register it against the previous frame (also at resolution level 3) using gradient descent and an absolute value loss function. We initialize each features position in the current frame by taking its position in the previous frame and adding the offset between the frames, as found through this registration process). Once we have our initial \mathbf{x} , we begin optimization on the top pyramid level. When done on the top level, we use the result to initialize optimization on the level below it, and so on until we have found a local minimum on level 0. On any given pyramid level, we perform

optimization by iteratively computing a step direction and conducting a fast line search to find a local minimum in the search direction. We impose a minimum and maximum on the number of steps to be performed on each level (\min_i and \max_i , respectively). Our termination condition (on a given level) is when the magnitude of the derivative of \bar{C} is not significantly smaller than it was in the previous step. To compute our search direction in each step, we first compute the gradient of \bar{C} (which we will call $D\bar{C}$) and set $\mathbf{a} = -D\bar{C}$. We then compute a semi-normalized version of \mathbf{a} . This is done by breaking it into a collection of 2-vectors (elements 1 and 2 are together, elements 3 and 4 are together, and so on) and normalizing each of them. We then recombine the normalized 2-vectors to get \mathbf{b} . We blend \mathbf{a} with \mathbf{c} to compute our step direction. Algorithm 1 summarizes the full process. Source code for our implementation of this algorithm will be available on the first authors web page.

Algorithm 1 Optimization of rank-penalized energy

Input: $\mathbf{x}^{\text{prev}}, \mathbf{I}, T_f f \in \{1, 2, \dots, F\}, M, \alpha, \min_i, \max_i$

Output: \mathbf{x}

Initialize $\mathbf{x} = \mathbf{x}^{\text{prev}} + [\Delta\mathbf{x}, \Delta\mathbf{x}, \dots, \Delta\mathbf{x}]^T$ where $\Delta\mathbf{x}$ is the result of registering \mathbf{I} against the previous frame.

for $m = 3 : 0$ **do**

$\mathbf{x} \leftarrow (1/2)^m \mathbf{x}$

$\|D\bar{C}\|_{\text{old}} \leftarrow \infty$

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

Let $\mathbf{a} \leftarrow -D\bar{C}$

for $f = 1 : F$ **do**

$\mathbf{y}_f \leftarrow [\mathbf{a}(2f - 1), \mathbf{a}(2f)]^T$

$\mathbf{b}_f \leftarrow \mathbf{y}_f / |\mathbf{y}_f|$

$\mathbf{b} \leftarrow [\mathbf{b}_1^T, \mathbf{b}_2^T, \dots, \mathbf{b}_F^T]^T$

$\mathbf{c} \leftarrow 0.5\mathbf{a} + 0.5\mathbf{b}$

$\mathbf{x} = \mathbf{x} + \mathbf{c}d$ where d is output of line search

if $\|D\bar{C}\| > 0.99\|D\bar{C}\|_{\text{old}}$ and $i > \min_i$ **then**

Exit for loop early

else

Assign $\|D\bar{C}\|_{\text{old}} = \|D\bar{C}\|$

$\mathbf{x} \leftarrow 2^m \mathbf{x}$

Return \mathbf{x}

Efficiency and Complexity

We have found that our algorithm typically converges in about 20 iterations or less at each pyramid level (with fewer iterations on lower pyramid levels). In our experiments, we used a resolution of 640-by-480 (we have also done tests at 1000×562), and we found that 4 pyramid levels were sufficient for reliable tracking. Thus, on average, less than 80 iterations are required to track from one frame to the next. A single iteration requires one gradient evaluation and multiple evaluations of \bar{C} . The complexity of a gradient

evaluation is $k_1Fn^2 + k_2LF^2$, and the complexity of an energy evaluation is $k_3Fn^2 + k_4L^2F$ (details are given in the supplementary material). Our C++ implementation (which makes use of OpenCV) can run on 35 features of size 7-by-7 with a temporal window of 6 frames ($L = 5$)² on a 3rd-generation Intel i5 CPU at approximately 16 frames per second. SIMD instructions are used in places, but no multi-threading was used, so faster processing rates are possible. With a larger window of $L = 10$ our algorithm still runs at 2-5 frames per second.

4. Experiments

To evaluate our method, we conducted tests on several real video sequences in circumstances that are difficult for feature tracking. These included shaky footage in low-light environments. The resulting videos contained dark regions with few good features and the unsteady camera motion and poor lighting introduced time-varying motion blur.

In these video sequences it proved very difficult to hand-register features for ground-truth. In order to present a quantitative numerical comparison we also collected higher-quality video sequences and synthetically degraded their quality. We used a standard Lucas-Kanade tracker on the non-degraded videos to generate ground-truth (the output was human-verified and corrected). We therefore present qualitative results on real, low-quality video sequences, as well as quantitative results on a set of synthetically degraded videos.

4.1. Qualitative Experiments on Real Videos

In our tests on real video sequences containing low-quality features, single-feature tracking does not provide acceptable results. When following a non-distinctive feature, the single-feature energy function often flattens out in one or more directions. A tracker may move in any ambiguous direction without realizing a better or worse match with the features template. This results in the tracked location drifting away from a features true location (i.e. “wandering”). This is not a technical limitation of one particular tracking implementation. Rather, it is a fundamental problem due to the fact that the local imagery in a small neighborhood of a feature does not always contain enough information to deduce the features motion between frames. This claim can be verified by attempting to hand-register low-quality features by only looking at a small neighborhood of the features last known location.

In these situations, our method infers the global motion of the scene from the observable features and uses it to assist in locating the low-quality features. This yields better overall tracking results in hard-to-track videos. Fig. 1 shows

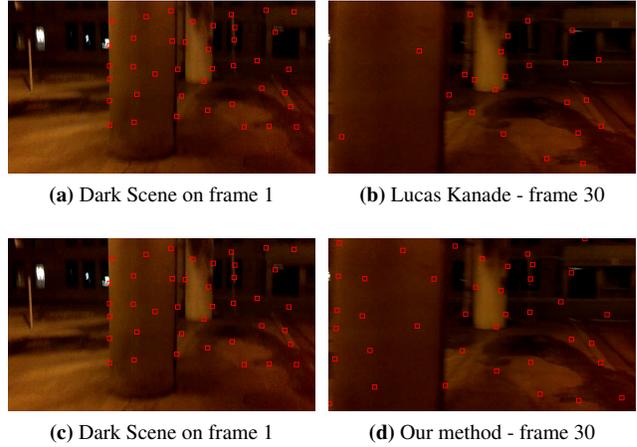


Figure 1: Results of tracking features in real low-light video. Most of the features wander significantly with the Lucas Kanade tracker. Our method provides better results on the low-quality features.

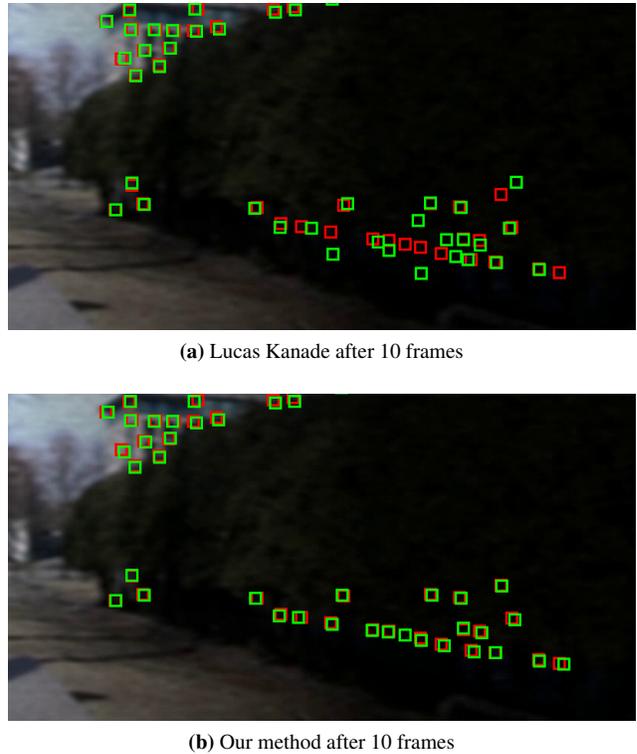


Figure 2: Characteristic results of the OpenCV Lucas-Kanade tracker vs our method in our synthetically degraded video experiment. The correct feature locations (according to the Lucas-Kanade tracker on the non-degraded video) are shown in red. Tracker-computed feature locations are shown in green.

²Accuracy for $L = 5$ is only slightly worse than for $L = 10$ and enables faster processing. See the supp. material for a brief comparison.

Table 1: Mean L1 trajectory error after 30 frames of tracking. Lower is better.

		Video Number								Average
		1	2	3	4	5	6	7	8	
Tracker	KLT	959.6	2484.0	958.3	1242.4	1630.2	1391.4	2105.0	4387.6	1894.8
	1st-Order Descent	92.5	137.8	159.5	273.2	87.5	198.4	70.6	685.7	213.2
	LDOF	508.0	408.6	898.5	385.2	104.9	122.3	256.1	721.3	425.6
	Multi-Tracker - Emp Dim - Uncentered	104.1	139.3	128.8	241.9	75.2	136.9	58.8	305.5	148.8
	Multi-Tracker - Emp Dim - Centered	102.9	115.3	108.3	226.8	69.7	128.6	54.1	292.5	137.3
	Multi-Tracker - Nuc Norm - Uncentered	106.8	134.0	131.7	243.9	73.4	132.6	58.4	293.9	146.8
	Multi-Tracker - Nuc Norm - Centered	103.5	137.7	141.5	243.9	73.4	135.5	60.3	341.0	154.6
	Multi-Tracker - Exp Fact - Uncentered	103.4	169.7	131.3	246.1	74.6	134.3	62.5	307.3	153.7
	Multi-Tracker - Exp Fact - Centered	102.9	167.0	129.1	245.0	73.2	133.4	58.9	302.5	151.5

Table 2: Average number of frames between feature re-initializations. Higher is better.

		Video Number								Average
		1	2	3	4	5	6	7	8	
Tracker	KLT	10.6	7.8	13.5	9.6	7.6	9.5	7.8	2.0	8.6
	1st-Order Descent	38.9	30.3	68.7	27.7	34.0	41.1	44.1	3.9	36.1
	LDOF	8.8	12.0	13.7	20.4	25.5	68.4	23.1	6.7	22.3
	Multi-Tracker - Emp Dim - Uncentered	73.4	35.3	111.5	55.6	70.2	102.3	63.7	14.0	65.8
	Multi-Tracker - Emp Dim - Centered	74.3	38.2	111.5	58.5	69.1	104.4	65.6	14.0	67.0
	Multi-Tracker - Nuc Norm - Uncentered	77.2	35.3	108.4	53.8	62.1	105.5	68.5	13.6	65.6
	Multi-Tracker - Nuc Norm - Centered	78.2	33.8	114.9	54.3	66.9	109.0	65.6	13.8	67.1
	Multi-Tracker - Exp Fact - Uncentered	76.2	34.3	114.9	54.3	66.9	98.3	68.5	13.6	65.9
	Multi-Tracker - Exp Fact - Centered	74.3	34.8	111.5	53.0	68.0	109.0	65.6	14.3	66.3

characteristic results from our tests. Several real videos are included in the supplemental material with results from the OpenCV Lucas Kanade tracker, and from our method.

4.2. Experiments on Synthetically Degraded Videos

For this experiment, we collected 8 video sequences of variable length in favorable lighting conditions. We used a Lucas Kanade tracker to track many features and manually verified and corrected the individual trajectories. Features come and go in these sequences (we do not assume all features persist through the entire sequence). These videos include 6 rigid environments as well as one video with multiple rigid bodies (video 7) and one video with a deformable body (video 8). The sequences range in length from 97 frames to 289 frames. On average, they are 210 frames each and contain over 6000 feature-frames each (this is the sum of each tracked features lifespan, measured in frames).

We degraded each video sequence by first darkening and adding noise to each frame, followed by applying a strong Gaussian blur to each frame. After this we added additional Gaussian noise. Adding noise before and after blurring gave the effect of noise at different scales (harder to deal with than per-pixel noise only). The test videos are included in the supplementary material.

For our comparison, we ran each tracker in two different modes. In the first mode we initialized each feature with its ground-truth location and re-initialized features when they wandered more than 10 pixels from ground truth. We recorded the average number of frames between feature re-initializations. In the second mode, we only tracked the features that were visible in frame 0, and features were never re-initialized. We looked at the mean L_1 difference between

the output trajectories and ground truth after 30 frames.

As a reference, we compared against the pyramidal Lucas Kanade tracker in the current OpenCV release (2.4.3). For a more recent comparison, we used LDOF (Large Displacement Optical Flow [7]) to generate dense flow fields for each sequence and we interpolated these flow fields to generate long-run trajectories for features. We also implemented our own single-feature gradient descent tracker (with an absolute value loss function). We present results for our rank-constrained tracker with the three previously introduced penalty functions. For each penalty function we present results with and without centering the history matrix M . In this experiment, whenever our algorithm is run with the penalty term, we use a weak constraint. All trackers were run on grayscale video. The results of this experiment are presented in Tables 1 and 2. An additional set of tests (on shorter video sequences) is included in the supplementary material.

4.3. Analysis of Results

From Tables 1 and 2, we can see that imposing our weak rank constraint significantly improves overall tracking ability, with all three rank regularizers that we tested showing improved tracking performance. Comparing the Lucas Kanade results to the results of our single-feature gradient descent tracker, we see a very large gap in performance. The core differences between these two algorithms are the definitions of ψ (squared error vs. absolute value) and the method of optimization employed. This performance difference supports our previous claim that the 2nd-order optimization technique used to accelerate convergence in the Lucas Kanade algorithm can be unreliable when

tracking poor-quality features.

5. Conclusion

Rank constraints have been successfully applied to several problems in computer vision, including motion segmentation and optical flow estimation. We have expanded on this previous work by developing a feature tracking framework which allows these constraints to be reliably used to assist in the tracking of features in rigid environments as well as in more general, non-rigid settings. The framework we presented permits these constraints to be imposed forcefully, allowing one to track features on a rigid object even if some features are occluded, or weakly, where the constraints are only used to help locate poor-quality features that cannot be tracked on their own. We showed that the weak constraint can yield significant gains in tracking performance, even in non-rigid scenes (with multiple or deformable objects) The framework we presented is completely causal and does not require explicitly modeling structure or motion in a scene. Furthermore, the algorithm we proposed is not prohibitively computationally expensive (real-time performance has been achieved). Our results provide evidence that when tracking features in low-quality video (especially in a rigid or semi-rigid scene), a 1st-order descent scheme is more robust than 2nd order methods used in standard Lucas-Kanade trackers, and applying rank regularizers to track a cohort of features results in better performance than classical single-feature tracking.

References

- [1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004. 1, 5
- [2] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, volume 588, pages 237–252. 1992. 5
- [3] M. Brand. Morphable 3d models from video. In *CVPR*, volume 2, pages II–456–II–463, 2001. 1
- [4] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV*, volume 2350, pages 707–720. 2002. 1
- [5] M. Brand. Subspace mappings for image sequences. In *Proc. Workshop Statistical Methods in Video Processing*, 2002. 1
- [6] M. Brand and R. Bhotika. Flexible flow for 3d nonrigid tracking and shape recovery. In *CVPR*, 2001. 1, 2
- [7] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE TPAMI*, 33(3):500–513, 2011. 7
- [8] A. Buchanan and A. W. Fitzgibbon. Combining local and global motion models for feature point tracking. In *CVPR*, 2007. 1, 2
- [9] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):11, 2011. 3, 4
- [10] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *FoCM*, 9:717–772, 2009. 3, 4
- [11] G. Chen and G. Lerman. Spectral curvature clustering (SCC). *IJCV*, 81(3):317–330, 2009. 2
- [12] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159–179, 1998. 1, 2, 4
- [13] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. In *CVPR*, pages 2018–2025, 2012. 1, 2
- [14] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. PAMI*, PP(99):1–1, 2013. 2
- [15] R. Garg, L. Pizarro, D. Rueckert, and L. de Agapito. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In *ACCV*, pages 460–473. 2010. 2
- [16] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *IJCV*, 104(3):286–314, 2013. 1, 2
- [17] R. Hartley and R. Vidal. Perspective nonrigid shape and motion recovery. In *ECCV (I)*, pages 276–289, 2008. 1, 2
- [18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 1, 2, 4
- [19] M. Irani. Multi-frame correspondence estimation using subspace constraints. *IJCV*, 48(3):173–194, 2002. 1, 2, 4
- [20] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 1, 2, 3
- [21] S. Negahban, P. D. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of m -estimators with decomposable regularizers. *Stat. Science*, 27(4):538–557, 2012. 3
- [22] B. Poling and G. Lerman. A new approach to two-view motion segmentation using global dimension minimization. arXiv:1304.2999, 2013. 4
- [23] S. Ricco and C. Tomasi. Dense lagrangian motion estimation with occlusions. *CVPR 2012*, 0:1800–1807, 2012. 1, 2
- [24] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600. IEEE, 1994. 1
- [25] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, 1991. 1
- [26] L. Torresani and C. Bregler. Space-time tracking. In *ECCV (I)*, pages 801–812, 2002. 2
- [27] L. Torresani, A. Hertzmann, and C. Bregler. Robust model-free tracking of non-rigid shape. Technical report, Technical Report TR2003-840, New York University, 2003. 2
- [28] L. Torresani, D. Yang, E. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR*, volume 1, pages I–493–I–500 vol.1, 2001. 1, 2
- [29] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. In *Compressed sensing*, pages 210–268. Cambridge Univ. Press, Cambridge, 2012. 4
- [30] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *ECCV*, volume 4, pages 94–106, 2006. 2
- [31] T. Zhang, A. Szlam, Y. Wang, and G. Lerman. Hybrid linear modeling via local best-fit flats. *IJCV*, 100:217–240, 2012. 2