Learning a Quality-Based Ranking for Feature Point Trajectories

Liangjing Ding*, Adrian Barbu**, Anke Meyer-Baese*

*Department of Scientific Computing, Florida State University
**Department of Statistics, Florida State University

Abstract. Long term motion analysis poses many standing challenges that need to be addressed for advancing this field. One of these challenges is finding algorithms that correctly handle occlusion and can detect when a pixel trajectory needs to be stopped. Very few optical algorithms provide an occlusion map and are appropriate for this task. Another challenge is finding a framework for the accurate evaluation of the motion field produced by an algorithm. This work makes two contributions in these directions. First, it presents a RMSE based error measure for evaluating feature tracking algorithms on sequences with rigid motion under the affine camera model. The proposed measure was observed to be consistent with the relative ranking of a number of optical flow algorithms on the Middlebury dataset. Second, it introduces a feature tracking algorithm based on RankBoost that automatically prunes bad trajectories obtained by an optical flow algorithm. The proposed feature tracking algorithm is observed to outperform many feature trackers based on optical flow using both the proposed measure and an indirect measure based on motion segmentation.

1 Introduction

Motion segmentation and estimation over tens or hundreds of frames is a difficult problem that still poses many challenges. Two of these challenges are finding good algorithms for tracking feature points over long sequences and a framework for accurately evaluating and comparing these algorithms.

While there exist frameworks and datasets for evaluating optical flow algorithms, such as the Middlebury dataset [1], the optical flow is evaluated only on two frames in these datasets. When estimating motion over long sequences, occlusion handling becomes very important because a large percentage of the image pixels will sooner or later be occluded in the sequence.

Many motion analysis algorithms based on tracking feature points [2] only track a subset of the image pixels, and these subsets can be different for different algorithms. Evaluation of the obtained motion fields would ideally require dense ground truth correspondences over the entire sequence, which is difficult to obtain.

There are efforts on evaluating the performance of a tracker [3] [4] objectively using a manually annotated dataset. For this reason, a number of metrics are defined in [3] [4]to evaluate motion trackers comprehensively. However, this raises the question: which metric is the best for ranking? Or should the metrics be combined in some way, such as by a weighted sum? A similar issue exists in the Middlebury optical flow dataset [1] where a number of error metrics are evaluated based on the ground truth.

Due to the complexity of reality scenes, no methods usually obtain the best performance on all sequences being tested. Also, when comparing the performance of trackers, people are always talking about the average performance. It is not strange that the best tracker would generate some bad trajectories, while a bad tracker would produce some good ones. Thus, in order to control the quality of the output trajectories, pruning out the bad trajectories seems to be a reasonable strategy.

In this paper we bring two contributions. First, we introduce an error measure for evaluating feature tracking algorithms on sequences containing objects undergoing rigid motion under the affine camera model. The proposed measure was observed to be consistent with the relative ranking of several selected optical flow algorithms on the Middlebury dataset. Moreover, the proposed error measure was observed to be more robust than an indirect measure based on evaluating the segmentation error of the generated trajectories.

Second, we introduce a feature tracking algorithm based on RankBoost that ranks the feature trajectories obtained by any optical flow algorithm from well tracked to badly tracked and removes a percentage of the badly tracked ones.

The proposed feature tracking algorithm is evaluated using two different error measures: the proposed error measure and an indirect measure based on motion segmentation accuracy. Both evaluations show that the proposed feature tracker outperforms other feature trackers based on optical flow on a number of publicly available image sequences.

2 A Method for Evaluating Feature Trackers

As mentioned in the introduction, it is difficult to evaluate feature tracking algorithms because dense ground truth motion fields over tens or hundreds of frames have not been obtained so far. There exist indirect error measures such as the segmentation error [5] returned by a predefined motion segmentation algorithm on the obtained feature trajectories. However, such a measure depends on the segmentation algorithm and tends to be quite unstable, as it will be seen in experiments.

In this section we investigate a direct error measure that can be used for sequences containing rigid motions under the affine camera model. The proposed measure is based on the observation that it is unlikely for a badly tracked feature point to accurately follow the rigid motion model of the object it belongs.

To use the proposed measure, the following are needed for each sequence being evaluated:

1. A set of full length *ground truth (GT) feature point trajectories* for each motion of the sequence. These trajectories have been verified to be correct and manually adjusted if necessary.
2. A dense (manual) segmentation of the first frame of the sequence (optional).

The proposed measure evaluates any given feature point trajectory by obtaining the motion label based on the dense segmentation of the first frame and the rigid motion model from the GT feature trajectories for that object.

Assume the set of full length GT trajectories for the rigid motion of interest is $t_i = (x_i^1, y_i^1, ..., x_i^T, y_i^T)', i = 1, ..., k$. Assume the sequence obeys the affine camera model [6], which generalizes orthographic, paraperspective and weak perspective projections. From the trajectories $t_1, ..., t_k$, we construct the *measurement matrix*

$$W = (t_1, t_2, ..., t_k) = \begin{pmatrix} x_1^1 & x_2^1 & ... & x_k^1 \\ y_1^1 & y_2^1 & ... & y_k^1 \\ .. & . & . & . \\ x_1^T & x_2^T & ... & x_k^T \\ y_1^T & y_2^T & ... & y_k^T \end{pmatrix}$$

The affine camera model [7, 6] factorization allows us to write

$$W = MS$$

where M is a $2T \times 4$ *motion matrix* and S is a $4 \times k$ matrix representing the 3D structure of the points. We will also refer to the matrix M as the motion model for the rigid object containing the trajectories $t_1, ..., t_k$.

There is an inherent ambiguity in M and S but we will show that it is irrelevant for our evaluation purpose.

2.1 RMSE Error for One Trajectory

Given a feature point trajectory $t = (x^b, y^b, ..., x^e, y^e)'$ that needs to be evaluated, assumed to belong to this motion, beginning at frame b and ending at frame e will have a corresponding 3D structure point P obtained by least squares:

$$P = \operatorname*{argmin}_P \|t - M_{be}P\|^2 = \operatorname*{argmin}_P (t - M_{be}P)'(t - M_{be}P) = (M_{be}'M_{be})^{-1}M_{be}'t \quad (1)$$

where M_{be} is the submatrix of M corresponding to the frames from b to e.

We can estimate P in a least square sense from the entire trajectory t and generate a most likely rigid motion trajectory as the projection $t^G = M_{be}P$ of P to the frames from b to e through the motion matrix M.

Finally define the SSE tracking error of the trajectory t as the sum of the squared errors between the trajectory points and the corresponding points of t^G:

$$\text{SSE}_W(t) = \min_P (t - M_{be}P)'(t - M_{be}P) = t't - t'M_{be}(M_{be}'M_{be})^{-1}M_{be}'t \quad (2)$$

Remark 1 *The $\text{SSE}_W(t)$ is invariant to the choice of M and S in the decomposition $W = MS$.*

Proof. Multiplying M by any 4×4 invertible matrix A results in multiplying M_{be} by A. It can be easily verified that $\text{SSE}_W(t)$ does not change when M_{be} is multiplied by an invertible matrix A. ■

We can now define the *RMSE error* for a trajectory t as:

$$\text{RMSE}_W(t) = \sqrt{\frac{\text{SSE}_W(t)}{e - b + 1}}, \quad (3)$$

which is measured in pixels and tells how well the trajectory fits the motion model of W.

The error measures $\text{SSE}_W(t), \text{RMSE}_W(t)$ can be computed for any trajectory t of a sequence that has the GT feature trajectories, against any measurement matrix W from the same sequence.

The $\mathrm{RMSE}_W(t)$ error can be used to evaluate the fitness of the trajectory t to the motion W, when the GT trajectories are available. If there is more than one motion in the ground truth, the error $\mathrm{RMSE}(t)$ of a trajectory can be obtained as follows:

1. If a dense motion segmentation is available, it can be used to obtain the motion label of the trajectory and the corresponding measurement matrix W can be used, so $\mathrm{RMSE}(t) = \mathrm{RMSE}_W(t)$.
2. If a dense segmentation is not available, the most likely motion is chosen as the motion that leads to the smallest RMSE error. Thus, the RMSE error is computed against all motions, and the smallest one is selected, so $\mathrm{RMSE}(t) = \min_W \mathrm{RMSE}_W(t)$.

In what follows we will use the second alternative since we don't have manual segmentations for the first frame of all 47 Hopkins sequences that will be used for evaluation.

We can now define the measure for evaluating the quality of the trajectories obtained by a feature tracking algorithm as the percentage of trajectories with RMSE larger than a threshold τ.

$$\mathrm{RMSE}_\tau = \frac{1}{N}|\{t_i, \mathrm{RMSE}(t_i) \geq \tau, i = \overline{1, N}\}| \tag{4}$$

2.2 Comparison with the Middlebury Dataset

The accuracy of the proposed error metric was evaluated on five standard optical flow algorithms: Kanade-Lucas-Tomasi (KLT) [2], Brox [8], Classic+NL [9], Black & Anandan (BA) [10] and Horn & Schunck (HS) [11].

The algorithms were evaluated on 47 image sequences from the Hopkins 155 dataset. The trajectories were obtained using the different algorithms starting from the same points in the first frame, as described in Section 4.1.

The relative ranking of these algorithms on the Middlebury dataset was compared with the ranking obtained from the proposed measure RMSE_τ with a range of thresholds τ.

In Figure 1 is shown the average RMSE_τ of the five algorithms for a range of values τ. The relative order based on the RMSE measure is consistent with the Middlebury relative ranking for a large range of values $\tau \geq 3$.

The only exception is the KLT algorithm that ranks better than in the Middlebury ranking.

We will see in the experimental section that the KLT performance changes with the strength of the feature points, so it would make sense that it ranks better than in the Middlebury dataset because the Hopkins sequences contain a majority of checkerboard images with strong feature points.

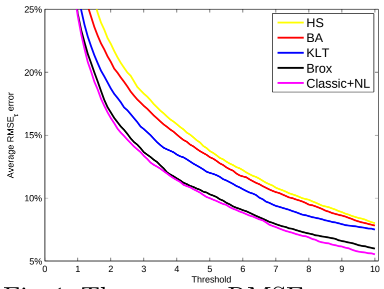

Fig. 1: The average RMSE_τ measure vs the threshold τ for five optical flow algorithms. The relative order of the algorithms is consistent with the Middlebury ranking for $\tau \geq 3$.

On the other hand, in Figure 6 are shown two indirect error measures based on motion segmentation. One can see from the graph that the measures are quite unstable and are not consistent with the Middlebury ranking.

3 A Trajectory Pruning Algorithm based on RankBoost

In this section we present a method for ordering feature point trajectories by predicting their relative quality using RankBoost [12]. This ranking is used to remove a percentage (e.g. 20%) of the trajectories predicted to have the worst quality.

3.1 The RankBoost Algorithm

We will use RankBoost [12] to learn an ordering of the feature point trajectories obtained by any feature tracker or optical flow algorithm.

RankBoost is a boosting algorithm that combines a number of ranking functions $h_i : \mathcal{X} \to \mathbb{R}, i = \overline{1, M}$ into a single ranking function $H : \mathcal{X} \to \mathbb{R}$ for instances $x \in \mathcal{X}$.

Given a set of training instances $S = \{x_i \in \mathcal{X}, i = \overline{1, n}\}$, we assume that a ground truth ranking is given on these instances as a function $\Phi : S \times S \to R$, where $\Phi(x_0, x_1) > 0$ means x_1 should be ranked above x_0 and vice versa.

RankBoost attempts to find a ranking that is similar to the given function Φ. In order to formalize this goal, construct the distribution D by $D(x_0, x_1) = c \cdot \max\{0, \Phi(x_0, x_1)\}$, where c is a constant to make $\sum_{x_0,x_1} D(x_0, x_1) = 1$. The learning algorithm tries to find a final ranking $H : \mathcal{X} \to \mathbb{R}$ that minimizes the weighted sum of wrong orderings:

$$\text{rloss}_D = \sum_{(x_0,x_1)\in S\times S} D(x_0, x_1)[\![H(x_1) \leq H(x_0)]\!]$$

where the notation $[\![\pi]\!]$ is defined to be 1 if predicate π holds and 0 otherwise.

Like other boosting algorithms, RankBoost operates in rounds. In each round t, RankBoost maintains a distribution D_t on $S \times S$ and selects the best weak ranker h_t along with its corresponding weight α_t from a large set of candidate weak rankers. The weight $D_t(x_0, x_1)$ will be decreased if $h_t(x_1) > h_t(x_0)$, and increased otherwise. Thus D_t will tend to concentrate on the pairs that are hard to rank. The final ranking H is a weighted sum of the weak rankers selected in each round. The algorithm is given in more detail in Figure 2.

```
Given: initial distribution $D$ over $\mathcal{X} \times \mathcal{X}$.
Initialize: $D_1 = D$.
for $t = 1, \dots T$ do
    1. Train weak learner using distribution $D_t$ to get weak ranking $h_t$.
    2. Choose $\alpha_t \in \mathbb{R}$.
    3. Update: $D_{t+1}(x_0, x_1) = \frac{D_t(x_0,x_1)exp(\alpha_t(h_t(x_0)-h_t(x_1))))}{Z_t}$ where $Z_t$ is a normaliza-
    tion factor (chosen so that $D_{t+1}$ will be a distribution).
end for
Output the final ranking: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.
```

Fig. 2: The original RankBoost algorithm [12].

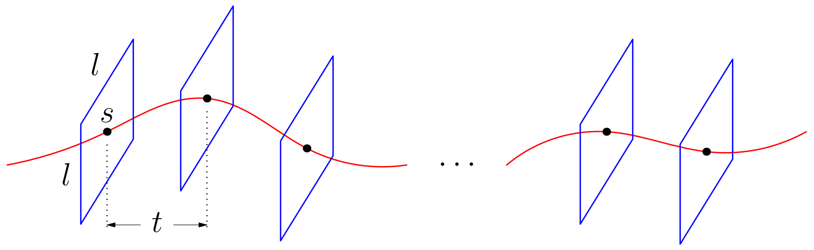

Fig. 3: The features used for ranking are trajectory shape features and intensity coherence features. By controlling the parameters s, t, and l, an over-complete feature representation of the trajectory can be obtained.

3.2 Features Used by the Weak Learners

The input to a weak learner $h : \mathcal{X} \to \mathbb{R}$ is a trajectory $x \in \mathcal{X}$. Based on the trajectory, shape and appearance features are generated. The process of obtaining the features is illustrated in figure 3.

The shape features are based on the position information along the trajectory. The features are parametrized by two attributes: the start time s, and the time gap t between two adjacent selected points. Based on these parameters we generate features that are supposed to measure constant velocity along the trajectory. For example, if we denote the positions of the points along a trajectory as $p_i, i = 1, \ldots, T$, the features are

$$f_{st} = \sum_{i=1}^{L-1} \|2 * p_{s+it} - p_{s+(i-1)t} - p_{s+(i+1)t}\|.$$

where L is the index of the last point that we could pick. The appearance features are intensity coherence features based on square image patches along the trajectory. The parameters of these features are s, t as in the geometric features and the side length of the square l. Different features are obtained using different brightness constancy measures such as SSD, normalized cross-correlation, etc. By changing the s, t, l and brightness constancy measure f, we get an over-complete feature representation of each trajectory. From the range of parameters used in our experiments, we obtained about 1200 features.

These features are easy to obtain and fast to compute. Each weak learner is based on one of these features, and RankBoost will select the ones that are most useful in obtaining an accurate ranking of the training set.

3.3 Training the Weak Learners

We want the weak rankers have range $[0, 1]$ rather than the actual values of the ranking features. For this reason, a threshold-based weak ranking h is adopted

$$h(x) = \begin{cases} 1 & \text{if } f_i(x) > \theta \\ 0 & \text{if } f_i(x) \le \theta \end{cases} \tag{5}$$

where $\theta \in \mathbb{R}$. A weak ranking is derived from a ranking feature f_i by comparing the score of f_i to a threshold θ.

A set of candidate thresholds $\{\theta_j\}_{j=1}^{J}, \theta_1 \ge \ldots \ge \theta_J$ are chosen for each weak learner. In this paper, we pick the thresholds evenly in the range of a feature in the training set. For a feature f_i, the maximum and minimum value in the training set is $\max(f_i)$ and $\min(f_i)$. The thresholds for it are set to

$$\theta_j = \max(f_i) - \frac{j-1}{J-1}(\max(f_i) - \min(f_i)), \qquad j = 1, \ldots, J. \tag{6}$$

Based on the discussion in [12], if a weak ranker has the form as equation (5), it should be trained to maximize $|r|$ where r is defined as

$$r = \sum_{x:f_i(x)>\theta} \pi(x)$$

where $\pi(x) = \sum_{x'}(D(x', x) - D(x, x'))$.

At each boosting iteration, we exhaust all weak rankers to find the one that maximizes $|r|$ along with its associated θ value. If $r_{\max}$ is the maximal value of $|r|$, then the corresponding weight α is calculated as

$$\alpha = \frac{1}{2}\ln(\frac{1+r_{\max}}{1-r_{\max}}).$$

3.4 Training the Ranking Algorithm

Training of the trajectory ranking algorithm requires a pool of weak rankers and ground truth ranking information in the form of a distribution D over all pairs of training examples.

Given a set of trajectories obtained by a feature tracking algorithm, the weak rankers are trained at each boosting iteration as described in Section 3.3, based on features extracted from the given trajectories and the image sequences.

The ground truth ranking of the trajectories is obtained from the RMSE error (3) based on the GT labeling of the trajectories into a number of rigid motions. All trajectories belonging to the same moving object from a video sequences can be ranked by their RMSE error. Obviously, a trajectory with a smaller error should be ranked above another one with a larger error. The initial distribution D is constructed based on the relative rank of the trajectories. Actually, the exact value of $D(i, j)$ is not important, so we simply put it 0 or 1 to indicate the rank between trajectories. Let the label and RMSE error of a trajectory x_i be l_i, e_i, respectively. The ranking $D(i, j)$ between trajectories x_i and x_j is calculated as

$$D(i, j) = \begin{cases} 1 & \text{if } l_i == l_j \text{ and } e_i > e_j \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

With the initial distribution D, the feature set and candidate weak rankers, the RankBoost training algorithm can be applied to get the boosted ranking algorithm. The process is explained in Algorithm 1.

Algorithm 1 Training the Trajectory Ranking Algorithm

```
Given: A set of trajectories with their labels and RMSE errors (3)
1. Compute features for each trajectory as described in section 3.2.
2. Calculate the candidate thresholds (equation (6)) for each feature.
3. Build the initial distribution $D$ by equation (7), and normalize it.
4. Perform $T$ iterations of RankBoost with weak rankers (5). In each iteration, obtain
a weak ranking $h_t$ with threshold $\theta_t$ and weight $\alpha_t$.
Output: The final ranking function: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.
```

3.5 Pruning Feature Trajectories with the Ranking Algorithm

The trained ranking function can be used to rank a set of trajectories produced by any feature tracker from a video sequence. Based on the final rank, a percentage of the worst trajectories can be discarded. The pruning algorithm is summarized as Algorithm 2.

In this way, one could obtain a more refined set of trajectories. It is worth noting that in this process no a priori information – such as segmentation, or RMSE error – is needed to obtain the rank. Thus, the algorithm could be widely used for videos with or without ground truth.

Algorithm 2 Feature Pruning using RankBoost

```
Given: a video sequence.
1. Detect interest points in the first frame.
2. Track interest points in all frames using a feature tracker,
3. Apply the trained ranker to sort the trajectories.
4. Discard the worst $p$% of the trajectories.
```

4 Experiments

In this section we present an evaluation of the proposed pruning-based ranking algorithm on the 47 video sequence of the Hopkins 155 Dataset [6] that contain ground truth trajectories for all motions that are present.

4.1 Trajectory Generation

For the comparison purpose, five optical flow algorithms were employed to generate trajectories. They are, the Kanade-Lucas-Tomasi (KLT) algorithm [2], the Brox algorithm [8], the Classic+NL algorithm [9], the Black & Anandan (BA) algorithm [10] and the Horn & Schunck (HS) algorithm [11]. When generating trajectories, Shi-Tomasi features [13] were detected on the first frame of the video, and then the corresponding feature points in the following frames were extracted by the five optical flow algorithms. Because many motion segmentation algorithms [14, 15] require the input trajectories to have the same length, the incomplete trajectories that were stopped before the last frame were discarded. Moreover, in the spirit of fairness, if any of the five algorithms stopped the trajectory from a feature point early, the trajectories of all five algorithms starting at that point were discarded. So in the end, the trajectories produced by all algorithms for one video sequence will share the same set of starting points.

4.2 Dataset and Evaluation Methods

We evaluated the proposed algorithm and error measure on the Hopkins 155 Dataset. The Hopkins 155 Dataset [6] has been created with the goal of providing an extensive benchmark for testing feature-based motion segmentation algorithms. It contains video sequences along with some feature points extracted and tracked in all the frames. The ground-truth segmentation is also provided for evaluation purposes. The 155 sequences are actually based on a dataset of 50 video sequences. Among them, there are three sequences missing the ground truth for the background motion: articulated, three-cars and arm. Since we need ground truth trajectories for all motions in the video, these three sequences were not included in the evaluation, remaining with 47 video sequences.

RMSE Error Measure. The ground truth trajectories provided by the Hopkins dataset enable us to calculate the RMSE_τ error (4) for each algorithm averaged over the 47 sequences. It should be mentioned that the number of motions in the ground truth may be smaller than that in the video, but most of the motions that are not in the ground truth are not present in all frames, and they always cover small regions in the frame, so it is reasonable to ignore their impact and simply set the number of motions as given in the ground truth.

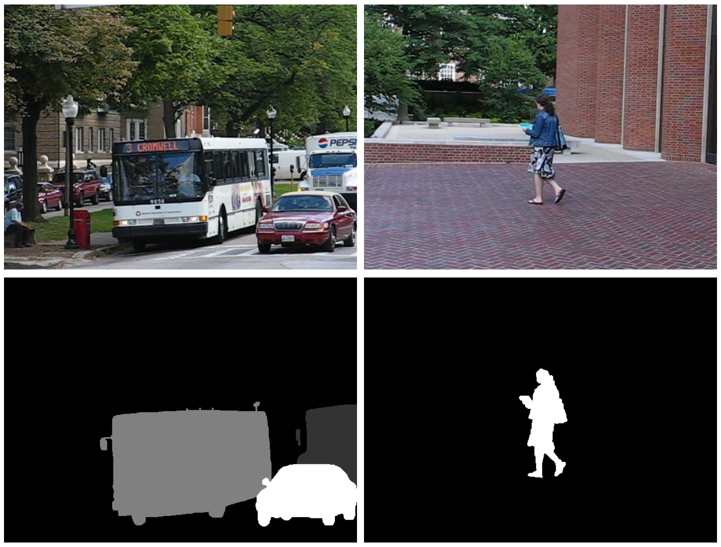

Fig. 4: Sample frames and their corresponding dense ground truth motion segmentation from the 12 sequences annotated by Brox et. al [5] in the Hopkins 155 dataset.

Segmentation Error Measure. Of the 50 video sequences of the Hopkins 155 dataset, Brox et.al. [5] annotated 10 car and 2 people sequences. Their annotation is dense in space and sparse in time, as shown in Figure 4. The first frame is always annotated to allow the evaluation of segmentation methods which could not deal with long trajectories. Based on this ground truth, any segmentation result obtained on any set of trajectories can be evaluated. This facilitates us to use the segmentation as an indirect measure of the performance of different trackers. There are five measure about segmentation proposed in [5]:

density The density of the tracked points.

overall clustering error The number of bad trajectory labels divided by the total number of labeled trajectories.

average clustering error Similar to the overall clustering error, the average clustering error is the average of the clustering error for each region separately. It gives more weight to small objects.

over-segmentation error The number of clusters that need to be merged to obtain the ground truth segmentation. This error is used to prevent obtaining a small error by producing a severe over-segmentation.

number of extracted objects The number of regions covered with a small error.

Among the five measures, the density is important for dense motion segmentation. In our experiments, the density is fixed at the very beginning since we use the Shi-Tomasi features as the starting points and we use the same starting points for all algorithms being compared. Moreover, the sparse motion segmentation methods [14] [15] that will be used for indirect evaluation ask for the number of motions in advance, so it has no meaning to measure the number of extracted objects here. In this case, the over-segmentation is hard to happen, so we do not measure the over-segmentation error either. As a result, two significant measures are adopted, the overall clustering error and the average clustering error. These two error measures will be evaluated on the 12 sequences that have a dense GT motion segmentation of the first frame.

4.3 Results

The RankBoost algorithm was trained on trajectories generated by the Classic+NL method from four image sequences and tested on all 47 sequences. From the pool of more than 1200 features, 150 were selected by RankBoost during training. The number of candidate thresholds J was set to 65. The ground truth ranking was obtained using the RMSE errors and trajectory labels based on the GT trajectories, as described in Section 3.4.

We evaluate the performance of the proposed algorithm using the two error measures described above. The first error measure is the proposed RMSE_τ error (4) that measures directly the quality of the trajectories based on rigid motion models. The second error measure is an indirect measure in which the obtained trajectories are segmented using a motion segmentation algorithm and the segmentation error is evaluated on the dense GT motion segmentation of the first frame. The performance using these two error measures is discussed in more details below.

RMSE Error. First, we evaluate the average RMSE_τ error (4) of all the trajectories. Starting with a large pool of trajectories, the pruning rate $p\%$ is changed for the RankBoost algorithm, to obtain different numbers of trajectories. For the other algorithms, the trajectory pruning can be controlled based on the strength of the Shi-Tomasi feature points. For different values of the pruning rate, the corresponding average RMSE error can be computed for the sets of trajectories obtained by the different algorithms. The average RMSE_5 error measure vs. the pruning rate from 0% to 80% is shown in figure 5, left.

From figure 5, left, one could find that the RMSE_5 error of BA, HS, Classic+NL and Brox algorithm is almost unchanged with respect to the pruning rate. This is understandable because these methods are independent of the selection of feature points (up to a point). However, the RMSE error of the KLT algorithm decreases moderately as the pruning rate is increased. This is indication that the Shi-Tomasi features have an impact on the performance of the KLT

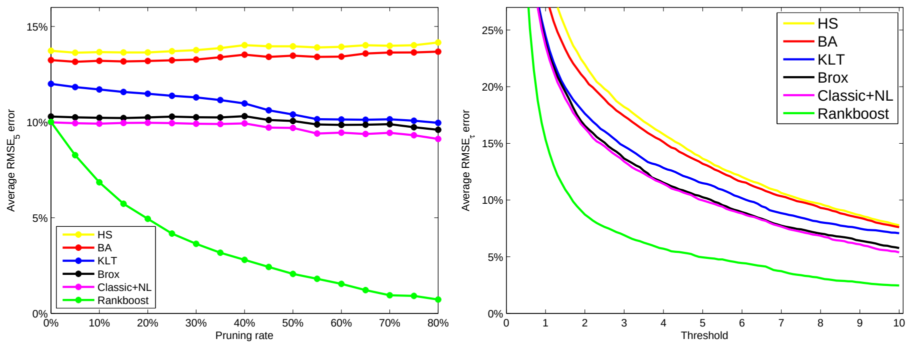

Fig. 5: Left: the RMSE$_5$ error (4) of the trajectories sets generated by different algorithms for different pruning rates. Right: the RMSE$_\tau$ error vs. the threshold τ when the pruning rate is set to 20%. The relative rank of the algorithms is consistent with the Middlebury ranking for a large range of thresholds.

tracker. Better Shi-Tomasi features will lead to better performance of the KLT algorithm. There is no wonder that the two are always used together. Among all the algorithms, the RankBoost algorithm performs the best according to the RMSE$_5$ measure. When the pruning rate is 80%, its average RMSE$_5$ error is less than 1%, while that of the second best (the Classic+NL algorithm) is more than 9%. Also, as the pruning rate increases, the error decreases much faster than the KLT algorithm. In particular, one could see the large difference in RMSE error between the RankBoost and Classic+NL algorithm, and keep in mind that the trajectories evaluated in the RankBoost algorithm and the Classic+NL algorithm were actually the same before pruning. The reason that the RankBoost algorithm could get better results is that the trained algorithm can predict very well which trajectories might not have been tracked properly.

In figure 5, right, is shown the RMSE$_\tau$ error vs the threshold τ for the pruning rate of 20%. From the figure, one could find that the RankBoost method with 20% pruning outperforms the other algorithms according to the RMSE$_\tau$ measure for any value of $\tau \in [2, 10]$. Moreover, we find that the relative performance of the tracking algorithms could be measured by the order of the curves for a given threshold τ. In figure 5, right, the performance order is Classic+NL $>$ Brox $>$ KLT $>$ BA $>$ HS for all values of the threshold $\tau \in [2, 10]$. This order is very similar to the order of the average endpoint error and average angle error in the Middlebury dataset, which is Classic+NL $>$ Brox $>$ BA $>$ HS $>$ KLT. As mentioned above, the Shi-Tomasi corner features could boost the performance of the KLT algorithms, it is better to take the KLT out of the list, then the orders are exactly the same. This study validates the power of the proposed RMSE error. We could pick other pruning rates than 20%, but the results will be almost the same.

Figure 7 shows the generated trajectories from different algorithms on a sample sequence in the Hopkins 155 dataset. The number of trajectories is kept the same and the pruning rate is 20%. It is clear the Rankboost algorithm prunes the trajectories in the path of the car and obtains better trajectories.

Segmentation Error. Another way to compare the performance of different feature tracking algorithms is by an indirect measure such as the segmentation error. The trajectories obtained by different tracking algorithms are segmented using a motion segmentation method and a measure of segmentation error is reported. Observe that this is an indirect measure since it depends on an additional step, that could introduce additional noise in the evaluation.

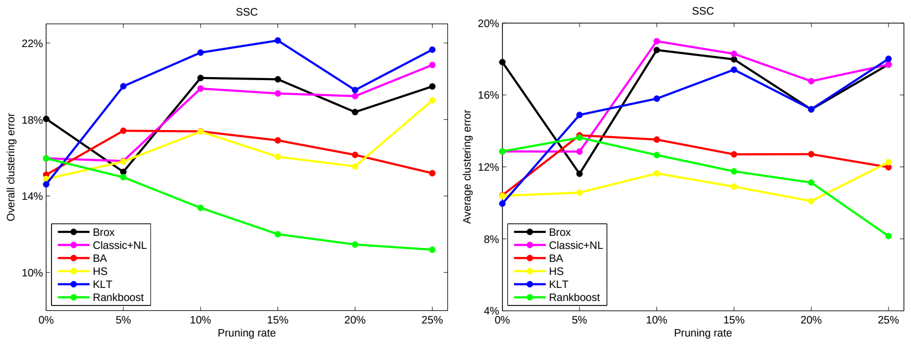

Fig. 6: The overall clustering error (left) and average clustering error (right) from 12 annotated sequences for different pruning rates.

In this paper we use the state-of-the-art motion segmentation algorithm sparse subspace clustering (SSC) [15]. As explained before, there may be some moving objects in the first frame that are not present in all frames. In order to evaluate the segmentation error accurately, we only consider the 'major' motions in the video, so the number of motions for the motion segmentation is set to the value given by the ground truth in the Hopkins 155 dataset. We omit other motions when calculating the segmentation error.

The segmentation error is averaged over the 12 video sequences that have the dense ground truth motion segmentation of the first frame. Because of using fewer image sequences for this evaluation than for the RMSE measure (12 instead of 47), this measure is expected to be less accurate.

Furthermore, the motion segmentation imposes limitations on the maximum pruning rate that can be used. Since the rank of the measurement matrix from one rigid motion is at most four, many segmentation methods require that the number of trajectories in one motion is at least four. Because of this requirement, the maximum pruning rate that can be handled by the motion segmentation is 25%. The RMSE$_\tau$ error described above does not suffer from this limitation, allowing a pruning rate of 80% or more.

In figure 6 are shown the overall clustering error and the average clustering error for different pruning rates, averaged over the 12 sequences with dense GT motion segmentation. These measures were explained in Section 4.2.

One could find that the RankBoost algorithm enables to reduce the overall clustering error effectively, constantly decreasing as the pruning rate is increased. Moreover, the RankBoost algorithm also constantly reduces the average clustering error when the pruning rate is at least 10%. These segmentation experiments also show that the RankBoost algorithm improves the quality of the feature trajectories.

5 Conclusions

In this paper, we presented a RMSE error measure based on factorization of the affine camera model. By comparing the percentage of trajectories whose RMSE error is above a threshold for a number of tracking algorithms, we find the relative order of the algorithms is consistent with that in the Middlebury dataset. Based on this RMSE trajectory error measure and RankBoost, we introduce an algorithm for ranking the quality of feature point trajectories. One feature of the algorithm is that it does not require any priori information to rank trajectories, and it can be used to rank the trajectories obtained with any feature tracker. The comparative study has demonstrated that the proposed algorithm obtains smaller RMSE errors than other selected feature tracking algorithms after pruning. An indirect measure based on motion segmentation was also employed to evaluate the performance of different trackers. The segmentation evaluation shows again that the RankBoost algorithm can effectively improve the quality of the obtained feature point trajectories.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A database and evaluation methodology for optical flow. IJCV **92** (2011) 1–31
2. Tomasi, C., Kanade, T.: Detection and tracking of point features. Technical report, Tech. Rept. CMU-CS-91132, Carnegie Mellon University (1991)
3. Needham, C., Boyle, R.: Performance evaluation metrics and statistics for positional tracker evaluation. In: Computer Vision Systems. Volume 2626. (2003) 278–289
4. Yin, F., Makris, D., Velastin, S., Orwell, J.: Quantitative evaluation of different aspects of motion trackers under various challenges. British Machine Vision Association (5) (2010) 1–11
5. Brox, T., Malik, J.: Object segmentation by long term analysis of point trajectories. European Conference on Computer Vision (2010) 282–295
6. Tron, R., Vidal, R.: A benchmark for the comparison of 3-D motion segmentation algorithms. In: CVPR. (2007)
7. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision **9** (1992) 137–154
8. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. ECCV (2004) 25–36
9. Sun, D., Roth, S., Black, M.: Secrets of optical flow estimation and their principles. In: CVPR. (2010) 2432–2439
10. Black, M., Anandan, P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. CVIU **63** (1996) 75–104
11. Horn, B., Schunck, B.: Determining optical flow. Artificial intelligence **17** (1981) 185–203
12. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. The Journal of Machine Learning Research **4** (2003) 933–969
13. Shi, J., Tomasi, C.: Good features to track. In: CVPR. (1994) 593–600
14. Lauer, F., Schnörr, C.: Spectral clustering of linear subspaces for motion segmentation. In: ICCV. (2009)
15. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: CVPR. (2009)

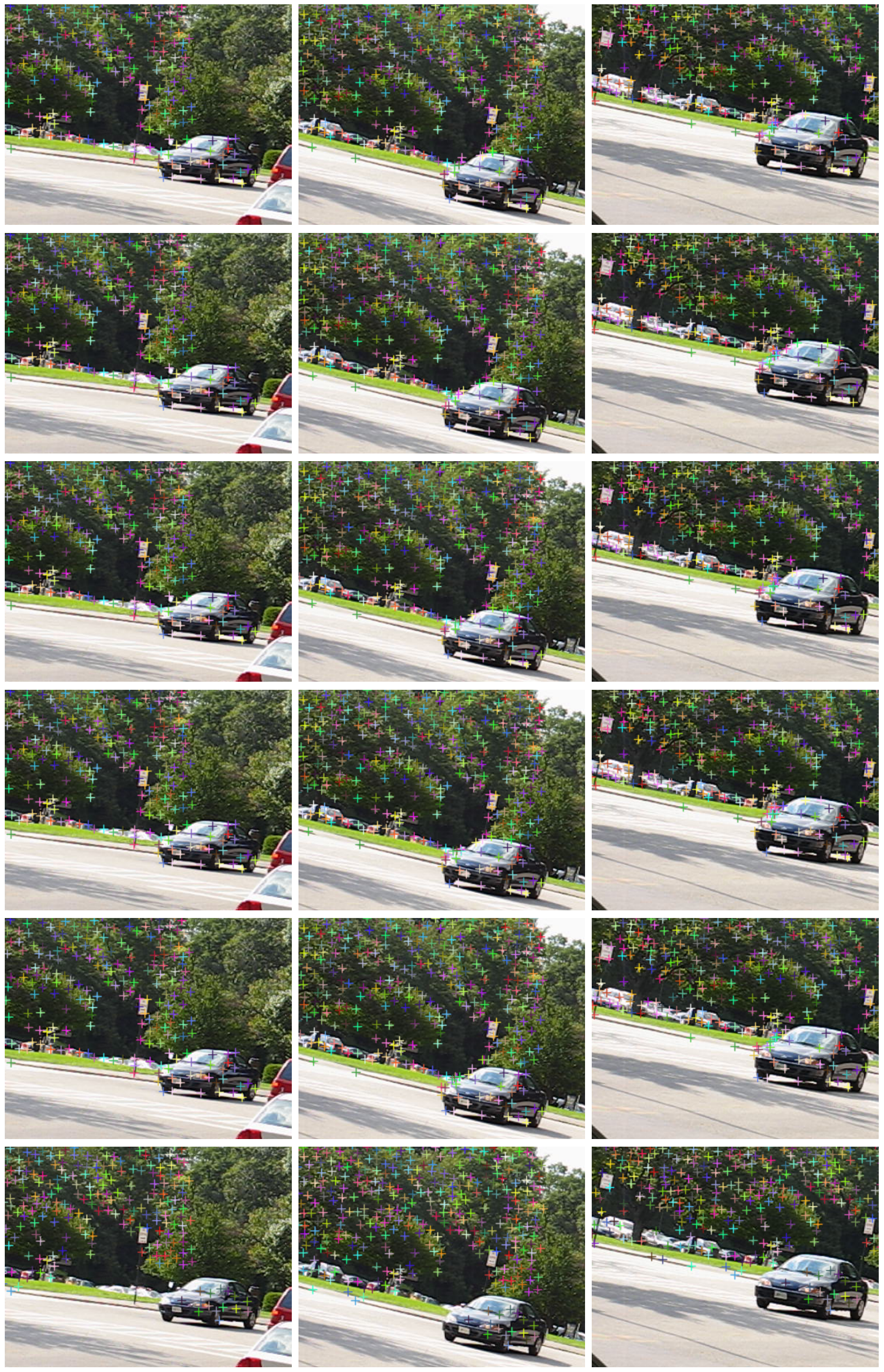
Fig. 7: Trajectories from different tracking algorithms on the `cars7` sequence from the Hopkins 155 dataset. Row 1 to row 6: Brox, Classic+NL, BA, HS, KLT, Rankboost. The number of trajectories is kept the same and the truncation rate is 20%. From left to right: frame 1, 13, 25. The images were cropped for clarity.