Non-Causal Tracking by Deblatting

Denys Rozumnyi[1 [0000−0001−9874−1349]], Jan Kotera[2 [0000−0001−5528−5531]], Filip Šroubek[2 [0000−0001−6835−4911]], and Jiří Matas[1 [0000−0003−0863−4844]]

[1] Centre for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic
[2] Institute of Information Theory and Automation, Czech Academy of Sciences, Prague, Czech Republic

Abstract. Tracking by Deblatting[1] stands for solving an inverse problem of deblurring and image matting for tracking motion-blurred objects. We propose non-causal Tracking by Deblatting which estimates continuous, complete and accurate object trajectories. Energy minimization by dynamic programming is used to detect abrupt changes of motion, called bounces. High-order polynomials are fitted to segments, which are parts of the trajectory separated by bounces. The output is a continuous trajectory function which assigns location for every real-valued time stamp from zero to the number of frames. Additionally, we show that from the trajectory function precise physical calculations are possible, such as radius, gravity or sub-frame object velocity. Velocity estimation is compared to the high-speed camera measurements and radars. Results show high performance of the proposed method in terms of Trajectory-IoU, recall and velocity estimation.

1 Introduction

The field of visual object tracking has received huge attention in recent years [20,6,7]. The developed techniques cover many problems and various methods were proposed, such as single object tracking [9,1,19,17], long-term tracking [10], methods with re-detection and learning [3,13,12,18], or multi-view [8] and multi-camera [14] methods.

Detection and tracking of fast moving objects is an underexplored area of tracking. In a paper focusing on tracking objects that move very fast with respect to the camera, Rozumnyi et al. [15] presented the first algorithm that tracks such objects, i.e. objects that satisfy the Fast Moving Object (FMO) assumption – the object travels a distance larger than its size during exposure time. The method [15] operates under restrictive conditions – the motion-blurred object should be visible in the difference image and trajectories in each frame should be approximately linear.

Recently, a method called Tracking by Deblatting[1] (TbD) has been introduced by Kotera et al. [5] to alleviate some of these restrictions. TbD performs significantly better than [15] and for a larger range of scenarios. The method solves two inverse problems of deblurring and image matting, and estimates object trajectories as piece-wise parabolic curves in each frame individually.

In its core, TbD assumes causal processing of video frames, i.e. the trajectory reported in the current frame is estimated using only information from previous frames.

[1] Deblatting = *debl*urring and m*atting*

Tracking by Deblatting [5] Non-causal trajectory estimation Speed estimation

Fig. 1. Trajectory reconstruction using the proposed non-causal Tracking by Deblatting (middle) compared to the causal TbD [5] (left). Color denotes the trajectory accuracy, from red (complete failure) to green (high accuracy). Ground truth trajectory (yellow) from high-speed camera is shown under the estimated trajectory. Speed estimation is shown on the right. Ground truth speeds (olive) are noisy due to discretization and TbD speed estimation (lightgray) is inaccurate, which is fixed by the proposed TbD-NC (purple).

Applications of detection and tracking of fast moving objects do not usually require online and causal processing. Moreover, non-causal trajectory estimation brings many advantages, such as complete and accurate trajectories, which were among TbD limitations, e.g. failures at contact with a player or missing detection.

We study non-causal Tracking by Deblatting and show that global analysis of FMOs leads to accurate estimation of FMO properties, such as nearly uninterrupted trajectory, velocity and shape. The paper makes the following contributions:

- We introduce global non-causal method, referred here as TbD-NC, for estimating *continuous* object trajectories by optimizing a global criterion on the whole sequence. Segments without bounces are found by an algorithm based on dynamic programming, followed by fitting of polynomials using a least squares linear program. Recovered trajectories give object location in every real-valued time stamp.
- Compared to the causal tracker, TbD-NC reduces by a factor of 10 the number of frames where the trajectory estimation completely fails.
- We show that TbD-NC increases the precision of the recovered trajectory to a level that allows good estimates of object velocity and size. Fig. 1 shows an example.

2 Related Work

Tracking methods that consider motion blur have been proposed in [21,16,11], yet there is an important distinction between models therein and the problem considered here. Unlike in case of object motion, the blur is assumed to be caused by camera motion, which results in blur affecting the whole image and in the absence of alpha blending of the tracked object with the background.

To our knowledge, there are only a few published methods that tackle the problem of detection and tracking of motion-blurred objects. The first publication was the work by Rozumnyi et al. [15]. The method assumes linear motion and trajectories are calculated by morphological thinning of the difference image between the given frame and the estimated background. In this paper, the first dataset with FMOs was introduced, however

it contains only ground truth masks without trajectories and it cannot be used to evaluate trajectory accuracy. Deblurring of FMOs also appeared in the paper by Kotera et al. [4], focusing only on deblurring without taking into account tracking or detection.

TbD [5] is the only method that uses motion blur and deblurring to improve tracking results and performs parametric fit to estimate intra-frame trajectories. The TbD dataset presented therein is another dataset with FMOs which contains ground truth trajectories and can be used for evaluating trajectory accuracy. A brief overview of TbD follows. The acquisition model with fast moving objects proposed in [15,5] is defined as

$$I = H * F + (1 - H * M)B, \tag{1}$$

where I: $D \rightarrow \mathbb{R}^3$ is the current image frame defined in image domain $D \subset \mathbb{R}^2$, which is modelled by two terms. The first term is the motion-blurred object model F along the trajectory given by the blur kernel H: $D \rightarrow \mathbb{R}$. The second term represents the influence of the background B and it depends on the indicator function M of object model F. The blur is then modelled by convolution and the background is estimated as a median of previous 3 to 5 frames. The camera is assumed to be static. We consider color images in this work and the median operator as well as convolutions are performed on each color channel separately. TbD introduces a prior on the blur kernel H and it is represented in each frame t by a continuous trajectory function $\mathcal{C}_t$: $[0, 1] \rightarrow \mathbb{R}^2$. The TbD outputs are individual trajectories $\mathcal{C}_t$ and blur kernels H_t in every frame. The outputs serve as inputs to the proposed TbD-NC method.

3 Non-Causal Tracking by Deblatting

TbD-NC is based on post-processing of individual trajectories from TbD. The final output of TbD-NC consists of a single trajectory $\mathcal{C}_f(t)$: $[0, N] \subset \mathbb{R} \rightarrow \mathbb{R}^2$, where N is a number of frames in the given sequence. The function $\mathcal{C}_f(t)$ outputs precise object location for any real number between zero and N. Each frame has unit duration and the object in each frame is visible only for duration of exposure fraction $\epsilon \leq 1$. Function $\mathcal{C}_f(t)$ is continuous and piecewise polynomial

$$\mathcal{C}_f(t) = \sum_{k=0}^{d_s} \bar{c}_{s,k} t^k \quad t \in [t_{s-1}, t_s], s = 1..S, \tag{2}$$

with S polynomials, where polynomial with index s has degree d_s and it is represented by its coefficient matrix $\bar{c}_s \in \mathbb{R}^{2,d_s}$. Columns of the matrix, denoted as $\bar{c}_{s,k} \in \mathbb{R}^2$, correspond to coefficients of two polynomials for x and y axis. The degree depends on the size of time-frame in which the polynomial is fitted to. Variables t_s form a splitting of the whole interval between 0 and N, i.e. that $0 = t_0 < t_1 < ... < t_{S-1} < t_S = N$.

Polynomials of degree 2 (parabolic functions) can model only free falling objects under the gravitational force. In many cases forces, such as air resistance or wind, also influence the object. They are difficult to model mathematically by additional terms. Furthermore, we would like to keep the function linear with respect to the weights. Taylor expansion will lead to a polynomial of higher degree, which means that these

forces can be approximated by adding degrees to the fitted polynomials. We validated experimentally that 3rd and 4th degrees are essential to explain object motion in standard scenarios. Degrees 5 and 6 provide just a small improvement, whereas degrees higher than 6 tend to overfit. Circular motion can also be approximated by (2).

A rough overview of the structure of the proposed method follows. The whole approach to estimate the piecewise polynomial function (2) is based on three main steps. In the first step, the sequence is decomposed into non-intersecting parts. Each part is converted into a discrete trajectory by minimizing using dynamic programming an energy function which combines information from partial trajectories estimated by the causal TbD, curvature penalizer to force smooth trajectories and constraints on start and end points. In the second step, the discrete trajectory is further decomposed into segments by detecting bounces. Then, segments define frames which are used for fitting each polynomial. In the third step, polynomials of orders up to six are fitted into segments without bounces, which define the final trajectory function $\mathcal{C}_f(t)$.

Splitting into segments. When tracking fast moving objects in long-term scenarios, objects commonly move back and forth, especially in rallies. During their motion, FMOs abruptly change direction due to contact with players or when they bounce off static rigid bodies. We start with splitting the sequence into differentiable parts, i.e. detecting *bounces* – abrupt changes of object motion due to contact with other stationary or moving objects. Parts of the sequence between bounces are called *segments*. Segments do not contain abrupt changes of motion and can be approximated by polynomial functions. Theoretically, causal TbD could detect bounces by fitting piecewise linear functions in one frame, but usually blur kernels are noisy and detecting bounces in just one frame is unstable. This inherent TbD instability can be fixed by non-causal processing.

To find segments and bounces, we split the sequence into *non-intersecting parts* where the object does not intersect its own trajectory, i.e. either horizontal or vertical component of motion direction has the same polarity. Between non-intersecting parts we always report bounces. Energy minimization by dynamic programming is used to convert blur kernels H_t from all frames in the given non-intersecting part into a single discrete trajectory. The proposed dynamic programming approach finds the global minimum of the following energy function

$$E(P) = -\sum_{x=x_b}^{x_e}\sum_{t=t_{s-1}}^{t_s} H_t(x, P_x) + \kappa_1 \sum_{x=x_b+2}^{x_e} \Big|(P_x - P_{x-1}) - (P_{x-1} - P_{x-2})\Big| \\ +\kappa_2(\mathcal{C}^x_{t_{s-1}}(0) - x_b) + \kappa_3(x_e - \mathcal{C}^x_{t_s}(1))\,, \tag{3}$$

where variable P is a discrete version of trajectory $\mathcal{C}$ and it is a mapping which assigns y coordinate to each corresponding x coordinate. P is restricted to the image domain. The first term is a data term of estimated blur kernels in all frames with the negative sign in front of the sum which accumulates more values from blur kernels while our energy function is being minimized. The second term penalizes direction changes and it is defined as the difference between directions of two following points and it is an approximation of the second order derivative of P. This term makes trajectories smoother and κ_1 serves as a smoothing parameter. The last two terms enforce that the starting point

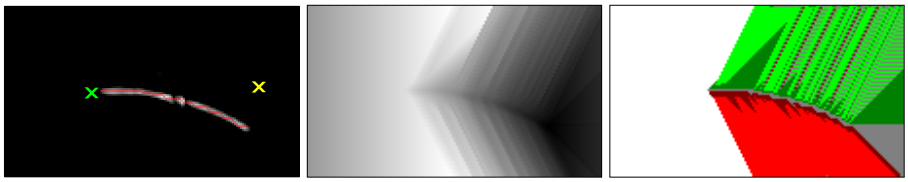

Fig. 2. Example of dynamic programming. Estimated discrete trajectory P is marked in red, starting point $\mathcal{C}_{t_{s-1}}(0)$ by green cross, and ending point $\mathcal{C}_{t_s}(1)$ by yellow cross. These points were deliberately moved further away to show robustness of the approach. Left image: accumulated blur kernels from two consecutive frames $H_{t_{s-1}}$ and H_{t_s} in joint coordinate system. Middle image: value of the energy function at each pixel from black (lowest) to white (highest). Right image: pixels where moving down by 1 is optimal are marked in dark green, down by 2 in bright green, up by 1 in dark red, up by 2 in bright red and moving straight in grey. Pixels, where reporting a starting point x_b is optimal, are white. The minimal value of the energy function is at the most right red pixel x_e in the left image. The whole trajectory is then estimated from right to left by backtracking until the next minimizing pixel is reported as a starting point (white space).

and the ending point are not far from the ones in the non-intersecting part. $\mathcal{C}^x_{t_{s-1}}(0)$ and $\mathcal{C}^x_{t_s}(1)$ denote x coordinate of the starting point at frame t_{s-1} and the ending point at frame t_s of causal TbD output. Note that in the last two terms there is no absolute value function and the sign is different, because they try to make trajectories shorter and they compete with the first term which prefers longer trajectories, e.g. either making trajectory longer is worth it in terms of values in blur kernels. Without the first term, the optimal trajectory would be of zero length, i.e. just a point. Discrete trajectory P is defined from x_b until x_e and these two variables are also being estimated. The ending point $\mathcal{C}_{t_s}(1)$ is assumed to be on the right side from the starting point $\mathcal{C}_{t_{s-1}}(0)$, and the image is flipped otherwise. All κ_i parameters were set to 0.1.

The energy E (3) is minimized by a dynamic programming (DP) approach. Accumulated blur kernels H_t are sorted column-wise (H_t) or row-wise (H_t transpose) to account for camera rotation or objects travelling from top to bottom. For both options we find the global minimum of E and the one with lower energy is chosen. Let us illustrate the approach for the column-wise sorting. The row-wise case is analogous. DP starts with the second column and processes columns from left to right. We compute energy E for each pixel by comparing six options and choosing the one with the lowest E: either adding to the trajectory one pixel out of five nearest pixels in the previous column with y coordinate difference between $+2$ and -2, or choosing the current pixel as the starting point. Both the minimum energy (Fig. 2 middle) and the decision option (Fig. 2 right) in every pixel is stored. When all columns are checked, the minimum in (Fig. 2 middle) is selected as the end point and the trajectory is estimated by backtracking following decisions in (Fig. 2 right). Backtracking finishes when a pixel is reached with the starting-point decision (white in Fig. 2 right).

When each non-intersecting part is converted into 1D signal, it becomes easier to find bounces. We are looking for points with abrupt changes of direction. When w pixels to the left and w pixels to the right of the given point have a change of direction higher than some threshold, then this point is considered a bounce. In case of circular motion

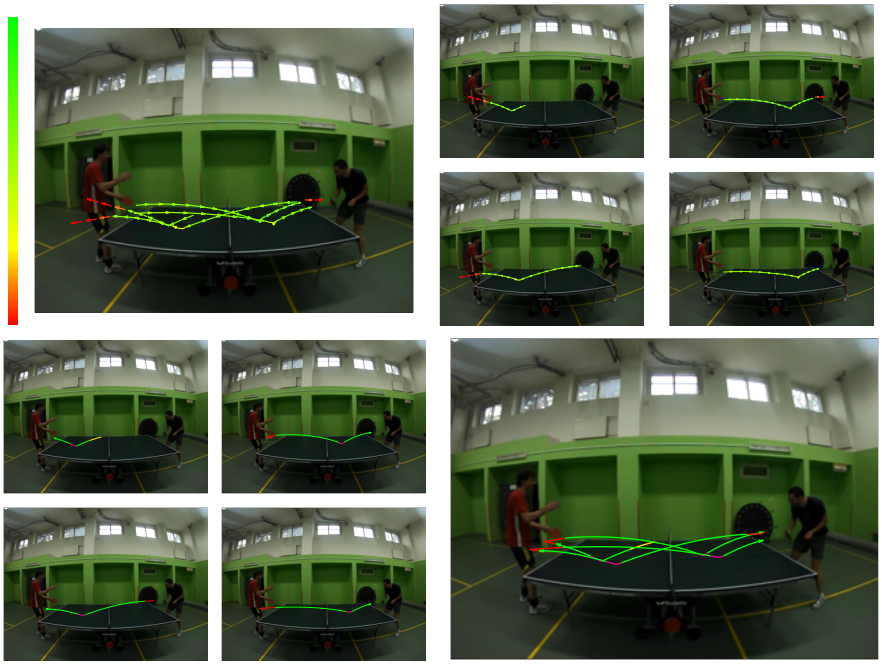

Fig. 3. TbD-NC processing steps. From left to right, top to bottom: causal TbD [5] output, splitting into segments, fitting polynomials to segments, final TbD-NC output. Top row: trajectories for all frames overlaid on the first frame, Trajectory-IoU accuracy measure color coded from red (failure) to green (success) by scale (top left corner). Bottom row: bounces between segments (magenta, red), fitted polynomials (green), extrapolation to the first and second frame (yellow). Arrows indicate motion direction. Best viewed when zoomed in a reader.

with no hard bounces, the approach finds a most suitable point to split the circle. After this step, the sequence is split into segments which are separated by bounces.

Fitting polynomials. The output discrete trajectory P has a two-fold purpose. First, it is used to estimate bounces and define segments, and second to estimate which frames belong to the segment and should be considered for fitting polynomials. To this end, we assign starting and ending points of each frame, i.e. $\mathcal{C}_t(0)$ and $\mathcal{C}_t(1)$, to the closest segment. For fitting we use only frames that completely belong to the segment, i.e. $\mathcal{C}_t(0)$ and $\mathcal{C}_t(1)$ are closer to this segment than to any other. The degree of a polynomial is a function of the number of frames ($N_s = t_s - t_{s-1} + 1$) belonging to the segment

$$d_s = \min(6, \lceil N_s/3 \rceil). \tag{4}$$

The polynomial coefficients are found by solving a linear least-squares problem

$$\begin{aligned} &\min_{\bar{c}_s} \textstyle\sum_{t=t_{s-1}}^{t_s} \|\mathcal{C}_f(t) - \mathcal{C}_t(0)\|^2 + \|\mathcal{C}_f(t+\epsilon) - \mathcal{C}_t(1)\|^2 \\ &\text{s. t.} \quad \mathcal{C}_f(t_{s-1}) = \mathcal{C}_{t_{s-1}}(0) \quad \text{and} \quad \mathcal{C}_f(t_s + \epsilon) = \mathcal{C}_{t_s}(1), \end{aligned} \tag{5}$$

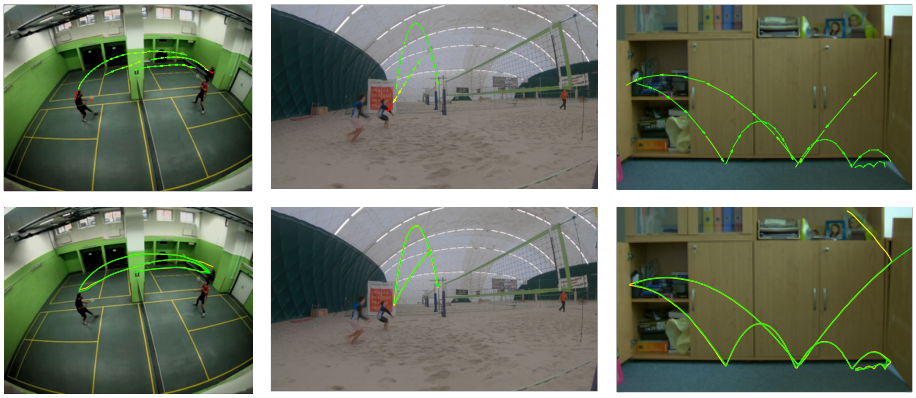

Fig. 4. Trajectory recovery for selected sequences from the TbD dataset. Top row: trajectories estimated by the causal TbD [5] overlaid on the first frame. TIoU (7) with ground truth trajectories from a high-speed camera is color coded by scale in Fig. 3. Bottom row: trajectory estimates by the proposed TbD-NC which outputs continuous trajectory for the whole sequence. The yellow curves underneath denote ground truth. Arrows indicate the direction of motion.

where s denotes the segment index. Equality constraints force continuity of the curve throughout the whole sequence, i.e. we get curves of differentiability class C^0. The least-squares objective enforces similarity to the trajectories estimated during the causal TbD pipeline. The final trajectory $\mathcal{C}_f$ is defined over the whole sequence and the last visible point in the frame t which is $\mathcal{C}_t(1)$ corresponds to $\mathcal{C}_f(t+\epsilon)$ in the sequence time-frame, where the exposure fraction ϵ is assumed to be constant in the sequence. The exposure fraction is estimated as an average ratio of the length of trajectories $\mathcal{C}_t$ in each frame and the distance between adjacent starting points

$$\epsilon = \frac{1}{N-1} \sum_{t=1}^{N-1} \frac{\|\mathcal{C}_t(1) - \mathcal{C}_t(0)\|}{\|\mathcal{C}_{t+1}(0) - \mathcal{C}_t(0)\|}. \tag{6}$$

Frames which are only partially in segments contain bounces. We replace them with a piecewise linear polynomial which connects the last point from the previous segment, bounce point found by dynamic programming and the first point from the following segment. Frames between non-intersecting parts are also interpolated by piecewise linear polynomial which connects the last point of the previous segment, point of intersection of these two segments and the first point of the following segment. Frames which are before the first detection or after the last non-empty $\mathcal{C}_t$ are extrapolated by the closest segment. Fig. 3 shows an example of splitting a sequence into segments which are used for fitting polynomials. More examples of full trajectory estimation are in Fig. 4.

4 Experiments

Experiments are done on the TbD dataset [5] with the ground truth trajectories from a high-speed camera. We use Trajectory Intersection over Union (TIoU) proposed by

Table 1. TIoU (7) and recall (Rcl) on the TbD dataset – comparison of TbD, FuCoLoT, FMO methods and the proposed TbD-NC. FuCoLoT is a standard, well-performing [7], near real-time tracker. For each sequence, the highest TIoU is highlighted in blue and recall in cyan.

Sequence	Frames	FuCoLoT [10]		FMO [15]		TbD [5]		TbD-NC	
		TIoU	Rcl	TIoU	Rcl	TIoU	Rcl	TIoU	Rcl
badminton_white	40	.286	0.39	.242	0.34	.694	0.97	.783	1.00
badminton_yellow	57	.123	0.22	.236	0.31	.677	0.91	.780	1.00
pingpong	58	.065	0.14	.064	0.12	.523	0.91	.643	1.00
tennis	38	.294	0.89	.596	0.78	.673	0.97	.750	1.00
volleyball	41	.496	0.79	.537	0.72	.795	0.97	.857	1.00
throw_floor	40	.275	0.63	.272	0.37	.810	1.00	.855	1.00
throw_soft	60	.463	0.95	.377	0.57	.652	0.97	.761	1.00
throw_tennis	45	.239	0.98	.507	0.65	.850	1.00	.878	1.00
roll_golf	16	.360	1.00	.187	0.71	.873	1.00	.894	1.00
fall_cube	20	.324	0.67	.408	0.78	.721	1.00	.757	1.00
hit_tennis	30	.330	0.93	.381	0.68	.667	0.93	.714	1.00
hit_tennis2	26	.226	0.79	.414	0.71	.616	0.83	.682	0.92
Average	39	.290	0.70	.352	0.56	.713	0.96	.779	0.99

Table 2. Comparison of TbD-NC with TbD [5]. TbD failure is defined as frames where TIoU (7) equals to zero. TbD-NC decreases the number of frames with failure by a factor of 10.

	TbD [TIoU]	TbD-NC [TIoU]	TbD [%]	TbD-NC [%]
TbD Fails	0.000	0.382	4.7	0.4
TbD TIoU>0	0.744	0.800	95.3	99.6

Kotera et al. [5] to measure the accuracy of estimated trajectories, which is defined as

$$\mathrm{TIoU}(\mathcal{C}, \mathcal{C}^*) = \int_t \mathrm{IoU}\left(M^*_{\mathcal{C}(t)},\, M^*_{\mathcal{C}^*(t)}\right) \mathrm{d}t, \tag{7}$$

where the estimated trajectory $\mathcal{C}$ is compared to the ground-truth trajectory $\mathcal{C}^*$. The ground truth object appearance mask M^* is used to measure IoU at different points x on the trajectory, denoted by M^*_x. Time t is discretized into 10 evenly spaced time-stamps to approximate integral.

Comparison to baselines on the TbD dataset is presented in Table 1. We use the recently introduced long-term tracker FuCoLoT [10] as a baseline standard tracker, the FMO method [15] as a baseline for a tracker specialized on fast moving objects and Tracking by Deblatting [5] (causal TbD with a template) as a well-performing method for establishing trajectories in each frame. The proposed TbD-NC outperforms all baselines in both recall and TIoU. Recall is 100% in all cases except one, where the first detection appeared only on the seventh frame and extrapolation to the first six frames was not successful. Table 2 shows that TbD-NC corrects complete failures of causal TbD when TIoU is zero, e.g. due to wrong predictions or other moving objects. TbD-NC also improves TIoU of successful detection by fixing small local errors, e.g. when the blur is misleading or fitting in one frame is not precise.

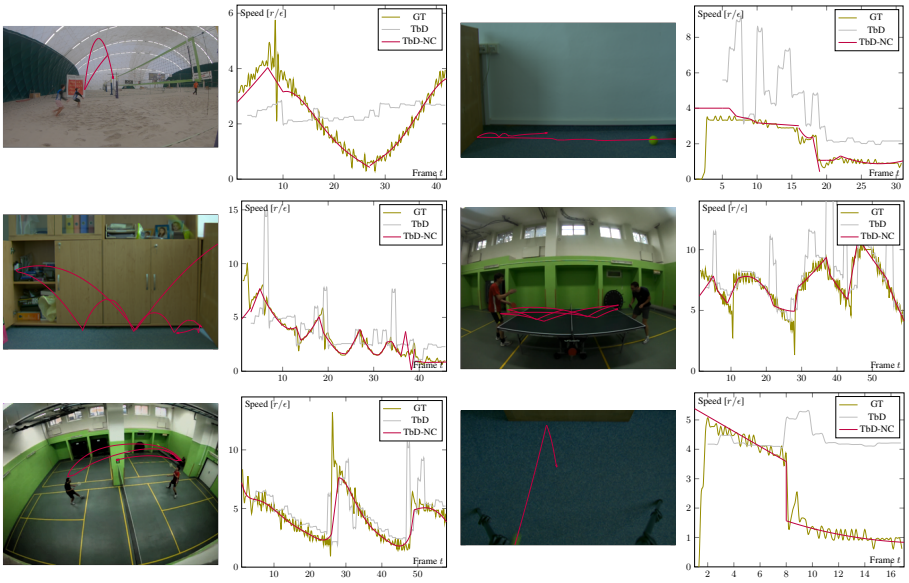

Fig. 5. Speed estimation using TbD-NC on selected sequences from the TbD dataset. Trajectories estimated by TbD-NC are overlaid on the first frame of each sequence. Graphs contain the speed estimation by TbD [5] (lightgray) and TbD-NC (purple) in radii per exposure compared to "ground truth" speeds (olive) calculated from high speed camera. The noise and oscillations in GT are caused by discretization. Mean differences to GT for all sequences are shown in Table 4.

Speed estimation. Tbd-NC provides the trajectory function $\mathcal{C}_f(t)$, which is defined for each real-valued time stamp t between 0 and the number of frames. Taking the norm of the derivative of $\mathcal{C}_f(t)$ gives a real-valued function of object velocity, measured in pixels per exposure. To normalize it with respect to the object, we divide it by the radius and report speed in radii per exposure. The results are visualized in Fig. 5 where sequences are shown together with their speed functions. The ground-truth speed was estimated from a high-speed camera footage having 8 times higher frame rate. The object center was detected in every frame and the GT speed was then calculated from the distance between the object centers in adjacent frames. Deliberately, we used no prior information (regularization) to smooth the GT speed and therefore it is noisy as can be seen in Fig. 5. We also report average absolute differences between GT and the estimated speed in Table 4. The error is mostly due to the noise in GT.

Speed estimation compared to radar guns. In sports, such as tennis, radar guns are commonly used to estimate the speed of serves. In this case, only the maximum speed is measured and the strongest signal usually happens immediately after the racquet hits the ball. Hrabalík [2] gathered the last 10 serves of the final match of 2010 ATP World Tour. The serves were found on YouTube from a spectator's viewpoint. Ground truth was available from another footage which showed the measured speeds from radar guns (example in Fig. 6). A real-time version of FMO detector in [2] achieved precise

Table 3. Speed estimation compared to the radar gun (GT). We used the last 10 serves of the final match of 2010 ATP World Tour. The lowest error for each serve is marked in blue.

Serve	Duration [frames]	GT [mph]	Hrabalík [2]		TbD-NC	
			Speed [mph]	Error [%]	Speed [mph]	Error [%]
1	23	108	105.6	2.2	108.0	0.0
2	32	101	103.8	2.8	101.6	0.6
3	62	104	106.5	2.4	110.4	6.1
4	75	113	101.7	10.0	115.8	2.5
5	82	104	91.9	11.6	106.9	2.8
6	30	127	127.4	0.3	126.3	0.6
7	34	112	116.1	3.7	107.5	4.0
8	78	125	123.2	1.4	130.3	4.2
9	67	99	88.3	10.8	89.7	9.4
10	90	108	110.2	2.0	106.2	1.6
Mean	57	110.1	107.5	4.7	110.3	3.2

Spectator's view | Front view (speed in top left) | Cropped

Fig. 6. Radar gun measurements. Speed was automatically estimated by TbD-NC method from the video on the left. Ground truth acquisition from YouTube video is shown in the middle and the right images. Table 3 compares estimates to the ground truth.

estimates of the speeds with the average error of 4.7 %, where the error is computed as an absolute difference to the ground truth velocity divided by the ground truth velocity.

Unfortunately, the ATP footage from spectator's viewpoint is of a very poor quality and the tennis ball is visible only as several pixels. Deblurring does not perform well when a video has low resolution or the object of interest is poorly visible. To test only the performance of full trajectory estimation (TbD-NC), we manually simulated FMO detector by annotating only start and end points of the ball trajectory in several frames after the hit for every serve. Then the time-stamp t_{hit} is found, such that the final trajectory $\mathcal{C}_f(t_{hit})$ at this point is the closest to the hit point. Then $\|\mathcal{C}'_f(t_{hit})\|$ is the speed measured by TbD-NC. The pixel-to-miles transformation was computed by measuring the court size in the video (1519 pixels) and dividing it by the tennis standards (78 feet). The camera frame rate was set to the standard 29.97 fps. Additionally, due to severe camera motion, the video was stabilized by computing an affine transformation between consecutive frames using feature matching as in [15]. Table 3 compares the speed estimated by TBD-NC and FMO methods to the ground truth from the radar. The proposed TbD-NC method is more precise than the FMO method and in several cases the speed is estimated with GT error close to zero.

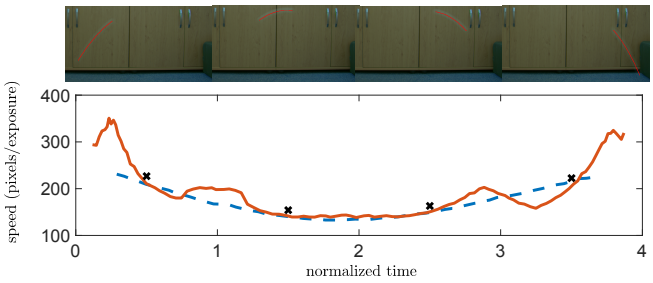

Fig. 7. Estimating the object velocity from blur kernels. In four consecutive frames (top row), object trajectories were estimated with TbD. The bottom plot shows the velocity calculated from the blur kernels (solid red) and the ground-truth (dashed blue line) obtained by a high-speed camera. Black crosses show the average velocity per frame calculated from the trajectory length.

Speed from the blur kernel. Apart from estimating speed by taking the norm of the derivative of $\mathcal{C}_f(t)$, we can also directly estimate speed from the blur kernel H. The values in the blur kernel are directly proportional to time the object spent in that location. For example, if half of the exposure time the object was moving with a constant velocity and than it stopped and stayed still, the blur kernel will have constant intensity values terminated with a bright spot that will be equal to the sum of intensities of all other pixels. Estimating speed from blur intensity values is however not very reliable due to noise in H. Fig. 7 illustrates a case where this approach works. All pixels in the blur kernel H which lay on the trajectory $\mathcal{C}$ are used for calculating the object velocity.

Shape and gravity estimation. In many situations, gravity is the only force that has non-negligible influence. Then, fitting polynomials of second order is sufficient. If parameters of the polynomial are estimated correctly, and the real gravity is given, then transforming pixels to meters in the region of motion is feasible. Gravity is represented by a parameter a, which has units $[\text{px}(\frac{1}{f}s)^{-2}]$, where the frame rate is denoted by f. If we assume the gravity of Earth $g \approx 9.8[ms^{-2}]$, f is known and a is estimated by curve fitting, the formula to convert pixels to meters becomes $p = g/(2af^2)$, where p are meters in one pixel on the object in motion. The radius estimation by this approach is shown in Table 4. Only half of the TbD dataset is used, i.e. sequences where the object was undergoing only motion given by the gravity (throw, fall, ping pong, volleyball). In other cases such as roll and hit, the gravity has almost no influence and this approach cannot be used. The badminton sequences have large air resistance and the tennis sequence was recorded outside during strong wind. When gravity was indeed the only strong force, the estimation has average error 4.1 %. The variation of gravity on Earth is mostly neglectable, but knowing exact location where videos have been recorded might even improve results. Alternatively, when the real object size is known, we can estimate gravity, e.g. when throwing objects on another planet and trying to guess which planet it is. In this case, the formula can be rewritten to estimate g. Results are also shown in Table 4 and the average error is 5.3 % when compared to the gravity on Earth. This shows robustness of the approach in both estimating radius and gravity.

Table 4. Estimation of radius, speed and gravity by TbD-NC on the TbD dataset [5]. The speed estimation is compared to GT from a high-speed camera. Radius is calculated when assuming Earth gravity, or vice versa. Standard object sizes are taken as GT for radius.

Sequence	Speed	Radius			Gravity	
	Mean Diff. $[r/\epsilon]$	GT [cm]	Est. [cm]	Err. [%]	Est. $[ms^{-2}]$	Err. [%]
badminton_white	0.57	-	-	-	-	-
badminton_yellow	0.65	-	-	-	-	-
pingpong	0.66	2.00	1.99	0.3	9.53	2.8
tennis	0.56	-	-	-	-	-
volleyball	0.45	10.65	10.47	1.7	10.50	7.2
throw_floor	0.61	3.60	3.47	3.7	10.21	4.2
throw_soft	0.42	3.60	3.72	3.3	9.52	2.9
throw_tennis	1.31	3.43	3.69	7.6	9.19	6.2
roll_golf	2.54	-	-	-	-	-
fall_cube	2.24	2.86	2.63	8.0	10.66	8.8
hit_tennis	0.43	-	-	-	-	-
hit_tennis2	1.28	-	-	-	-	-
Average	0.98	-	-	4.1	9.93	5.3

Temporal super-resolution. Among other applications of TbD-NC are fast moving object removal and temporal super-resolution. The task of temporal super-resolution stands for creating a high-speed camera footage out of a standard video and consists of three steps. First, a video free of fast moving objects is produced which is called fast moving object removal. For all FMOs which are found in every frame, we replace them with the estimated background. Second, intermediate frames between adjacent frames are calculated as their linear interpolation. Objects which are not FMOs will look natural after linear interpolation. Then, trajectory $\mathcal{C}_f(t)$ is split into the required number of pieces, optionally with shortening to account for the desired exposure fraction. Third, the object model (F, M) is estimated and used to synthesize the formation model with FMOs (1). Examples of these applications are provided in the supplementary material.

5 Conclusion

We proposed a non-causal Tracking by Deblatting (TbD-NC) which estimates accurate and complete trajectories of fast moving objects in videos. TbD-NC is based on globally minimizing an optimality condition which is done by dynamic programming. High-order polynomials are then fitted to trajectory segments without bounces. The method performs well on the recently proposed TbD dataset and complete failures appear 10 times less often. From the estimated trajectories, we are able to calculate precise object properties such as velocity or shape. The speed estimation is compared to the data obtained from a high-speed camera and radar guns. Novel applications such as fast moving objects removal and temporal super-resolution are shown.

Acknowledgements. This work was supported by the Czech Science Foundation grant GA18-05360S and the Czech Technical University student grant SGS17/185/OHK3/3T/13.

References

1. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: Proceedings of the British Machine Vision Conference. BMVA Press (2014). https://doi.org/10.5244/C.28.65
2. Hrabalík, A.: Implementing and applying fast moving object detection on mobile devices, master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering (2017)
3. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1409–1422 (July 2012). https://doi.org/10.1109/TPAMI.2011.239
4. Kotera, J., Šroubek, F.: Motion estimation and deblurring of fast moving objects. In: 2018 25th IEEE International Conference on Image Processing (ICIP). pp. 2860–2864 (Oct 2018). https://doi.org/10.1109/ICIP.2018.8451661
5. Kotera, J., Rozumnyi, D., Šroubek, F., Matas, J.: Intra-frame Object Tracking by Deblatting. arXiv e-prints arXiv:1905.03633 (May 2019)
6. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Čehovin, L., Vojíř, T., Häger, G., Lukežič, A., Fernández, G., et al.: The visual object tracking VOT2016 challenge results. In: Hua, G., Jégou, H. (eds.) Computer Vision – ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II. pp. 777–823. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3
7. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Zajc, L.Č., Vojíř, T., Bhat, G., Lukežič, A., Eldesokey, A., et al.: The sixth visual object tracking VOT2018 challenge results. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018 Workshops. pp. 3–53. Springer International Publishing, Cham (2019)
8. Kroeger, T., Dragon, R., Van Gool, L.: Multi-view tracking of multiple targets with dynamic cameras. In: Jiang, X., Hornegger, J., Koch, R. (eds.) Pattern Recognition. pp. 653–665. Springer International Publishing, Cham (2014)
9. Lukežič, A., Vojíř, T., Zajc, L.C., Matas, J., Kristan, M.: Discriminative correlation filter with channel and spatial reliability. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4847–4856 (July 2017). https://doi.org/10.1109/CVPR.2017.515
10. Lukežič, A., Zajc, L.Č., Vojíř, T., Matas, J., Kristan, M.: FuCoLoT – a fully-correlational long-term tracker. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision – ACCV 2018. pp. 595–611. Springer International Publishing, Cham (2019)
11. Ma, B., Huang, L., Shen, J., Shao, L., Yang, M., Porikli, F.: Visual tracking under motion blur. IEEE Transactions on Image Processing **25**(12), 5867–5876 (Dec 2016). https://doi.org/10.1109/TIP.2016.2615812
12. Moudgil, A., Gandhi, V.: Long-term visual object tracking benchmark. arXiv preprint arXiv:1712.01358 (2017)
13. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 445–461. Springer International Publishing, Cham (2016)
14. Ristani, E., Tomasi, C.: Features for multi-target multi-camera tracking and re-identification. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6036–6046 (June 2018). https://doi.org/10.1109/CVPR.2018.00632
15. Rozumnyi, D., Kotera, J., Šroubek, F., Novotný, L., Matas, J.: The world of fast moving objects. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4838–4846 (July 2017). https://doi.org/10.1109/CVPR.2017.514
16. Seibold, C., Hilsmann, A., Eisert, P.: Model-based motion blur estimation for the improvement of motion tracking. Computer Vision and Image Understanding **160**, 45 –

56 (2017). https://doi.org/10.1016/j.cviu.2017.03.005, http://www.sciencedirect.com/science/article/pii/S1077314217300590
17. Tang, M., Yu, B., Zhang, F., Wang, J.: High-speed tracking with multi-kernel correlation filters. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4874–4883 (June 2018). https://doi.org/10.1109/CVPR.2018.00512
18. Tao, R., Gavves, E., Smeulders, A.W.: Tracking for half an hour. arXiv preprint arXiv:1711.10217 (2017)
19. Vojíř, T., Noskova, J., Matas, J.: Robust Scale-Adaptive Mean-Shift for Tracking, pp. 652–663. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38886-6
20. Wu, Y., Lim, J., Yang, M.: Online object tracking: A benchmark. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2411–2418 (June 2013). https://doi.org/10.1109/CVPR.2013.312
21. Wu, Y., Ling, H., Yu, J., Li, F., Mei, X., Cheng, E.: Blurred target tracking by blur-driven tracker. In: 2011 International Conference on Computer Vision. pp. 1100–1107 (Nov 2011). https://doi.org/10.1109/ICCV.2011.6126357