International Journal of Computer Vision (2021) 129:2583–2604
https://doi.org/10.1007/s11263-021-01480-w

Tracking by Deblatting

Denys Rozumnyi[1,2] · Jan Kotera[3] · Filip Šroubek[3] · Jiří Matas[1]

Received: 29 December 2019 / Accepted: 12 May 2021 / Published online: 22 June 2021

Abstract
Objects moving at high speed along complex trajectories often appear in videos, especially videos of sports. Such objects travel a considerable distance during exposure time of a single frame, and therefore, their position in the frame is not well defined. They appear as semi-transparent streaks due to the motion blur and cannot be reliably tracked by general trackers. We propose a novel approach called Tracking by Deblatting based on the observation that motion blur is directly related to the intra-frame trajectory of an object. Blur is estimated by solving two intertwined inverse problems, blind deblurring and image matting, which we call deblatting. By postprocessing, non-causal Tracking by Deblatting estimates continuous, complete, and accurate object trajectories for the whole sequence. Tracked objects are precisely localized with higher temporal resolution than by conventional trackers. Energy minimization by dynamic programming is used to detect abrupt changes of motion, called bounces. High-order polynomials are then fitted to smooth trajectory segments between bounces. The output is a continuous trajectory function that assigns location for every real-valued time stamp from zero to the number of frames. The proposed algorithm was evaluated on a newly created dataset of videos from a high-speed camera using a novel Trajectory-IoU metric that generalizes the traditional Intersection over Union and measures the accuracy of the intra-frame trajectory. The proposed method outperforms the baselines both in recall and trajectory accuracy. Additionally, we show that from the trajectory function precise physical calculations are possible, such as radius, gravity, and 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.

Keywords Fast moving objects · Visual object tracking · Deblatting · Deblurring · Trajectory estimation · Energy minimization

1 Introduction

The field of visual object tracking has received considerable attention in recent years; see (Wu et al. 2013; Kristan et al. 2016, 2019). The developed techniques cover many problems. Various methods were proposed, such as single object tracking in (Lukežič et al. 2017; Danelljan et al. 2014; Vojíř et al. 2013; Tang et al. 2018) and multi-object tracking that employ the tracking-by-detection paradigm in (Hornakova et al. 2020; Braso and Leal-Taixe 2020). Other methods include long-term tracking as in (Lukežič et al. 2019), methods with re-detection and learning in (Kalal et al. 2012; Mueller et al. 2016; Moudgil and Gandhi 2017; Tao et al. 2017), multi-view methods in (Kroeger et al. 2014), and multi-camera in (Ristani and Tomasi 2018).

Communicated by Simone Frintrop.

This work was supported by the Czech Science Foundation grant GA18-05360S, the Czech Technical University student grant SGS17/185/OHK3/3T/13, and by the Praemium Academiae awarded by the Czech Academy of Sciences. D. Rozumnyi was also supported by Google Focused Research Award.

✉ Denys Rozumnyi
rozumden@cmp.felk.cvut.cz; denys.rozumnyi@inf.ethz.ch

Jan Kotera
kotera@utia.cas.cz

Filip Šroubek
sroubekf@utia.cas.cz

Jiří Matas
matas@cmp.felk.cvut.cz

[1] Visual Recognition Group, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

[2] Department of Computer Science, ETH Zurich, Zurich, Switzerland

[3] The Czech Academy of Sciences, Institute of Information Theory and Automation, Prague, Czech Republic

Tracking by Deblatting | Non-causal trajectory estimation | Speed estimation

Fig. 1 Trajectory reconstruction starts with causal Tracking by Deblatting (TbD, left), followed by non-causal Tracking by Deblatting (TbD-NC, middle). Color denotes trajectory accuracy, from red (complete failure) to green (high accuracy). The ground truth trajectory from high-speed camera is shown in yellow. Speed estimates are shown on the right. Ground truth speeds (olive) are noisy due to discretization and TbD speed estimation (lightgray) is inaccurate, which is fixed by TbD-NC (purple) (Color figure online)

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. 2017) 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. However, this method operates under restrictive conditions – the motion-blurred object should be visible in the difference image, and trajectories in each frame should be approximately linear. The first FMO dataset introduced by (Rozumnyi et al. 2017) 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 and Šroubek, 2018), focusing only on deblurring without taking into account tracking or detection.

General trackers, both long and short term, provide information about the object location in a frame in the form of a single rectangle, e.g. in the VOT challenge by (Kristan et al. 2019). The true, continuous trajectory of the object center is thus sampled with the frequency equal to the video frame rate. For slow moving objects, such sampling is adequate. For fast moving objects, especially if their trajectory is not linear (due to bounces, gravity, friction), a single location estimated per frame cannot represent the true trajectory well, even if the fast moving object is inside the reported bounding box. Moreover, general trackers typically fail even in achieving that as was shown in (Rozumnyi et al. 2017).

Tracking methods that consider motion blur have been proposed in (Wu et al. 2011; Seibold et al. 2017; Ma et al. 2016), yet there is an important distinction between models therein and the problem considered here. Unlike in the case of object motion, blur is assumed to be caused by camera motion that creates blur affecting the whole image without alpha blending of the tracked object with the background.

We propose a novel methodology for tracking fast-moving blurred objects. The approach untangles the image formation by solving two inverse problems: *motion deblurring* and *image matting*. We therefore call the method *Tracking by Deblatting*, TbD in short. The deblatting procedure simultaneously recovers the trajectory of the object, its shape, and appearance. This is formulated as an optimization problem, which is then solved using the Alternating Direction Method of Multipliers (ADMM); see (Boyd et al. 2011). We introduce a strong prior on the blur kernel and force it to lie on a 1D curve that represents the object trajectory within a frame. Unlike a standard general tracker, TbD does not need a template of the object since the representation of the shape and appearance of the object is recovered on the fly. Experiments show that the estimated trajectory is often highly accurate; see Fig. 1.

TbD is formulated as causal processing of video frames, i.e. the trajectory reported in the current frame is estimated using only information from previous frames. Applications of detection and tracking of fast moving objects do not usually require online and causal processing. We therefore also study non-causal Tracking by Deblatting that estimates continuous trajectory for the whole sequence by fitting piecewise polynomial curves. Non-causal trajectory estimation is more robust and brings advantages, such as complete and accurate trajectories, which are among TbD limitations, e.g. failures at contact with a player or missing detection. We show that the non-causal analysis of FMOs leads to accurate estimation of FMO properties, such as nearly uninterrupted trajectory, velocity, and shape, which can be further used in applications of temporal super-resolution, object removal, and gravity estimation.

The paper makes the following **contributions**:

- We propose Tracking by Deblatting (TbD) to estimate intra-frame object trajectories that solves an inverse problem of deblurring and image matting.
- We introduce a global non-causal method, called 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 linear least squares. Recovered trajectories give object location as a function of continuous time.
- 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.
- We derive an effective solution of the proposed constraint optimization problem by the alternating direction method of multipliers (ADMM, Sect. 2.1).

This work is an extension of our earlier conference publications (Rozumnyi et al. 2019) and (Kotera et al. 2019). Some parts have been reported in Master Thesis by (Rozumnyi 2019). In addition to the earlier versions, we improve the loss function in the dynamic programming part and introduce an extended TbD dataset that contains slower motions. To handle such slower motions, we additionally improve polynomial curves fitting. We also include experimental results in case of all-speed tracking. We further study the influence of rotation and polynomial degree on the performance of the proposed method.

The paper is organized as follows. In Sect. 2, the core idea of TbD is introduced including the concept of causal long-term tracking; details of the deblatting optimization problem are deferred to appendix. Sect. 3 introduces the non-causal extension of TbD and presents trajectory estimation for the whole video sequence. Used parameters and algorithm settings are explained in Sect. 4. Experiments are divided into three sections: Sect. 5 provides quantitative evaluation, Sect. 6 demonstrates the ability of TbD to track objects of varying speed, and Sect. 8 illustrates applications of object speed and radius estimation, gravity estimation, and temporal super-resolution. Running time is reported in Sect. 7. Limitations are discussed in Sect. 9, and the paper is concluded in Sect. 10. Efficient Python (for CPU) and PyTorch (for speed-up on GPU) implementations are open-sourced.[1]

2 Tracking by Deblatting

The proposed method formulates tracking as an inverse problem. Consider a single color video frame $I\colon D \to \mathbb{R}^3$ defined on a rectangular domain $D \subset \mathbb{R}^2$, which is either of size of the video frame or of a small region of interest. In the frame I, an object $F\colon D^o \to \mathbb{R}^3$ moves along a continuous

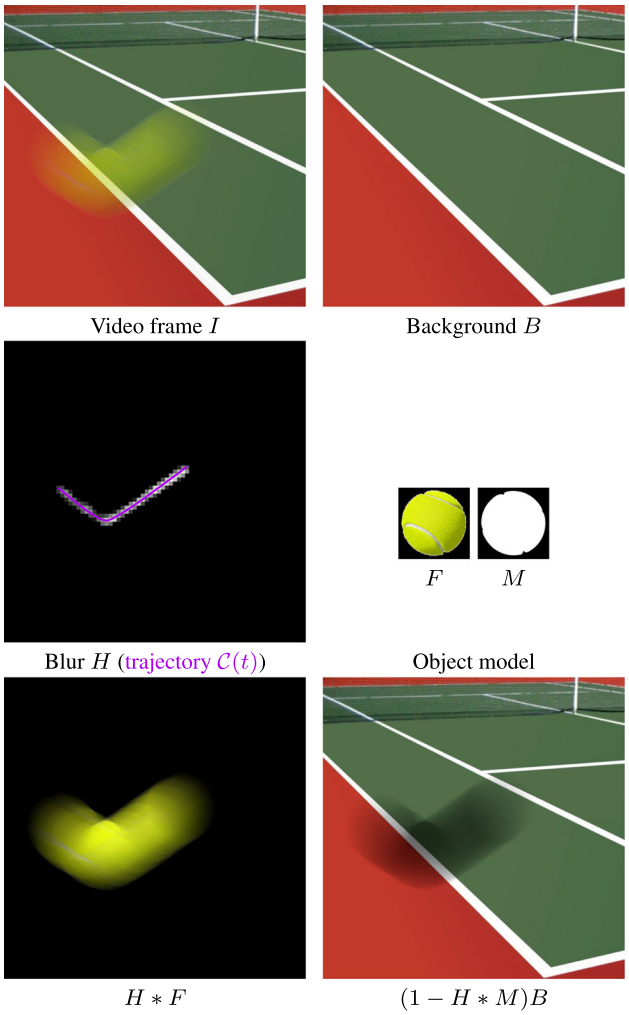


Fig. 2 The image formation model, Eq. (1). Top: known variables – video frame I with the blurred object (left) and background image B (right). Middle: estimated variables – motion blur H and the object model (sharp object and shape mask). Bottom: the first and second terms of Eq. (1). Note that blur H covers the same domain D as the input video frame and effectively encodes the position of the object in the image. Trajectory $\mathcal{C}(t)$ is a piece-wise polynomial curve fitted to the blur. Object appearance F and its shape mask M are defined on domain $D^o \ll D$

trajectory $\mathcal{C}(t)\colon [0, 1] \to D$ in front of background $B\colon D \to \mathbb{R}^3$. The size of the object domain D^o is assumed to be much smaller than the size of D. The frame formation model then becomes

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

where $*$ denotes convolution, $H\colon D \to \mathbb{R}$ is a blur kernel, and $M\colon D^o \to \mathbb{R}$ is the binary mask of the object shape (i.e. the indicator function of F). We refer to the pair (F, M) as the object model. The mutual relation between the blur kernel H and the trajectory $\mathcal{C}$ is defined as follows: the blur is the image of the trajectory rendered into the domain D,

[1] https://github.com/rozumden/deblatting_python.

Table 1 The most important variables

Var.	Domain	Description
N	$\in \mathbb{N}$	Number of frames
D	$\subset \mathbb{R}^2$	Image domain
D^o	$\subset D \subset \mathbb{R}^2$	Object domain
I	$D \to \mathbb{R}^3$	Input image
B	$D \to \mathbb{R}^3$	Background
F	$D^o \to \mathbb{R}^3$	Object appearance
M	$D^o \to \mathbb{R}$	Object mask
H	$D \to \mathbb{R}$	Blur kernel
$\mathcal{C}(t)$	$[0, 1] \to D$	Parametric trajectory
$\mathcal{C}_f(t)$	$[0, N] \to D$	Full trajectory (Sect. 3)
P	$\mathbb{N} \to \mathbb{N}$	Discrete trajectory (Sect. 3)
ϵ	$\in [0, 1] \subset \mathbb{R}$	Exposure fraction (Sect. 3)

i.e. $H(x) = \int_0^1 \delta(x - \mathcal{C}(t))\mathrm{d}t$ for $x \in D$, where $\delta(x - \mathcal{C}(t))$ is the delta function at position $\mathcal{C}(t)$, and the trajectory is a piece-wise polynomial curve fitted to the blur. The first term of the formation model is the tracked object blurred by its own motion, and the second term is the background partially occluded by the object with blending coefficients given by $H*M$. A pictorial explanation of the formation model (1) is in Fig. 2. Inference in this formation model consists of solving simultaneously two inverse problems: blind deblurring and image matting. The solution is the estimated blur kernel H and the object model (F, M). The most important variables used in the manuscript are summarized in Table 1.

Motion blur in (1) is modeled by convolution, which implies an assumption that the object model remains constant during the frame exposure time. Scenarios that satisfy the assumption precisely are, e.g., an object of arbitrary shape undergoing only translational motion or a spherical object of uniform color undergoing arbitrary motion under spatially-uniform illumination. The object motion must be in a plane parallel to the camera image plane to guarantee constant size of the object. In addition, the model assumes that the background in the close vicinity of the object location ($H*M > 0$) is also constant during the frame exposure time. For the purpose of tracking and trajectory estimation, we claim that the formation model (1) is sufficient as long as the assumptions hold at least approximately, which is experimentally validated on the presented dataset.

The proposed TbD method is iterative and processes a new frame I_{i+1} in a causal manner using only knowledge acquired from earlier frames $I_1, \ldots, I_i$; see Fig. 3 (shaded area) for an overview. Inputs are the current estimates of the object model F_i and M_i, background B_i, and a region of interest D_i in I_{i+1}, which is the neighborhood of the predicted object location. Three main steps are performed in TbD:

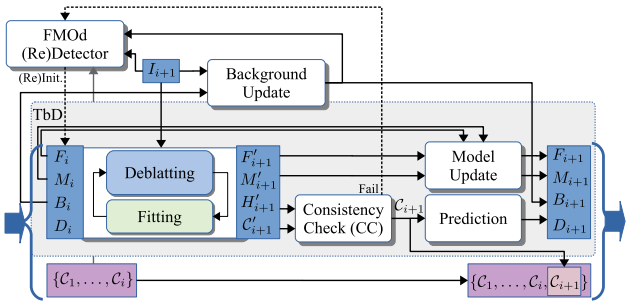


Fig. 3 Long-term Tracking by Deblatting (Sect. 2). The FMO detector (FMOd – top left box) is activated during initialization or if the consistency check fails

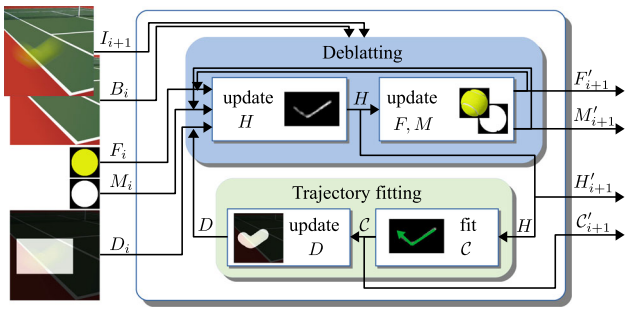


Fig. 4 *Deblatting*, i.e. *debl*urring and m*atting* – Sect. 2.1, with trajectory fitting – Sect. 2.2

1. *Deblatting*: Iteratively solve blind deblurring and matting in the image region D_i using model (1) and estimate F'_{i+1}, M'_{i+1}, and H'_{i+1}; see Sect. 2.1.
2. *Trajectory fitting*: Estimate physically plausible motion trajectory (parametric curve) $\mathcal{C}'_{i+1}$ corresponding to H'_{i+1} and optionally adjust D_i according to $\mathcal{C}'_{i+1}$; see Sect. 2.2.
3. *Consistency check & model update*: Verify that the error of the mapping $H \to \mathcal{C}$ is below threshold τ, predict the new region of interest D_{i+1} for the next frame, and update the object model to F_{i+1} and M_{i+1}.

A more detailed illustration of Steps 1 and 2 is in Fig. 4. Step 1 stops after reaching either a given relative tolerance or a maximum number of iterations. Steps 1 and 2 are repeated only if the newly fitted trajectory $\mathcal{C}$ touches the boundary of the image domain D – in this case the new domain is the d-neighborhood of trajectory $\mathcal{C}$ where d is the object diameter. This approach helps to eliminate the detrimental influence of other moving objects on the blur kernel estimation.

Consistency check The consistency check (CC) represents the newly fitted curve $\mathcal{C}'_{i+1}$ as a blur kernel and measures an error between this blur kernel and H'_{i+1} estimated in the deblatting step. The CC passes if the error is below the threshold τ. Then, the estimated trajectory is extrapolated to the next frame, and D_{i+1} becomes the new d-neighborhood of the extrapolation.

To update the object model we use exponential forgetting

$$F_{i+1} = \gamma F_i + (1-\gamma) F'_{i+1} \tag{2}$$

and similarly for M_{i+1}.

To enable long-term tracking, the FMO detector (FMOd) from (Rozumnyi et al. 2017) determines the new input if CC fails. First, FMOd tries to detect the object in an gradually enlarged D. If it succeeds, the main TbD pipeline is reinitialized with D set as a neighborhood of the FMOd detection. If FMOd fails, TbD returns the extrapolation of trajectory $\mathcal{C}_i$ as the best guess of $\mathcal{C}_{i+1}$ and tracking is restarted anew on the next frame. The background B_i is estimated as a temporal median of frames B_{i-1}, $B_{i-2}, \ldots$, optionally including video stabilization by homography fitting if necessary. The first detection is also performed automatically by FMOd. We consider color images in this work. The median operator as well as convolutions are performed on each color channel separately. The object appearance model is learned "on the fly" starting trivially with uniform $F_0 \equiv 1$, $M_0 \equiv 1$, equivalent to a white square. Alternatively, the user provides a template of the tracked object, e.g. a rectangular region from one of the frames where the object is still.

More details of deblatting and trajectory fitting are provided in the next two subsections.

2.1 Deblatting

The core step of TbD is the extraction of motion information H from the input frame, which we formulate as a blind deblurring and matting problem. Inputs are the frame I, domain D, background B, and previously estimated (or initially selected by the user) object appearance $\hat{F}$. The inverse problem corresponding to (1) is formulated as

$$\min_{F,M,H} \frac{1}{2} \|H * F + (1 - H * M)B - I\|_2^2 + \frac{\lambda}{2} \|F - M\hat{F}\|_2^2 + \alpha_F \|\nabla F\|_1 + \alpha_H \|H\|_1 \tag{3}$$

s.t. $0 \leq F \leq M \leq 1$ and $H \geq 0$ in D, $H \equiv 0$ elsewhere. The first term in (3) is fidelity to the acquisition model (1). The second λ-weighted term is a form of "template-matching", an agreement with the prescribed appearance. The template $\hat{F}$ is multiplied by the shape mask M because if $\hat{F}$ is initially supplied by user as a rectangular region from a video frame, it contains both the object and the surrounding background. The template is used to establish the scale of the object (denoted by D^o) and the appearance model (F, M). When processing the i-th frame, we set $\hat{F} = F_{i-1}$ as the updated appearance estimate (2) from the previous frame. The first L^1 term is the total variation that promotes smoothness of the recovered object appearance. The second L^1 regularization penalizes non-sparse blurs.

If M is a binary mask, as initially defined, then the condition $F \leq M$ states that F cannot be nonzero where M is zero – pixels outside the object must be zero. Formally, it means that the support of F is contained in the support of M. However, we relax the binary restriction and allow M to attain fractional values in the range [0, 1]. Such relaxation is beneficial for computational reasons and accounts for mixed pixels on object borders or for artifacts such as shadows. In the relaxed setting we consider the appearance model as an RGBA image where RGB channels are stored in F, and the alpha channel A is stored in M. The constraint corresponding to this relaxation is then $F \leq M$, assuming the intensity range of F alone limited to [0, 1]. The inequality constraint $H \geq 0$ prohibits unphysical negative values of H. The blur must also vanish outside its feasibility domain D.

Alternating minimization We solve (3) by minimizing in a coordinate-descend manner with respect to H and (F, M) separately. The whole deblatting procedure then consists of the following steps:

1. Initialize $M := M_{i-1}$ (if available from previous detection) or $M \equiv 1$; initialize $\hat{F} := F_{i-1}$, $F := M\hat{F}$.
2. Fix (F, M) and update H by solving (15).
3. Check convergence, exit if satisfied.
4. Fix H and update (F, M) by solving (21), go to 2.

All the optimization details are provided in Appendix 1. The minimization w.r.t. H is stated in (15), and w.r.t. (F, M) is stated in (21).

Examples of the deblatting alone are in Figs. 5 and 6. Fig. 5 contains from left to right: the cropped input frame, the corresponding frame from the high-speed camera, estimated blur kernel H, and estimated object model (F,M). In the top row, we see that the shape of the badminton shuttlecock, though not circular, is estimated correctly. In the bottom row, we see that if the non-uniform object undergoes only small rotation during motion, the appearance estimation can also be good. In this case, the shape estimation is difficult due to the mostly homogeneous background similar to the object. Fig. 6 illustrates an interesting example of deblatting behavior in the case of a shadow. The input frame with an object casting a shadow is in the top left corner, and the corresponding part from the high-speed camera is below. If we set the size of F too small, the model cannot cope with the shadow, and the estimated blur contains artifacts in the locations of the shadow as is visible in the top row. If instead we make the support of F sufficiently large, the estimated mask compensates for the shadow, and the estimated blur is clean as shown in the bottom row.

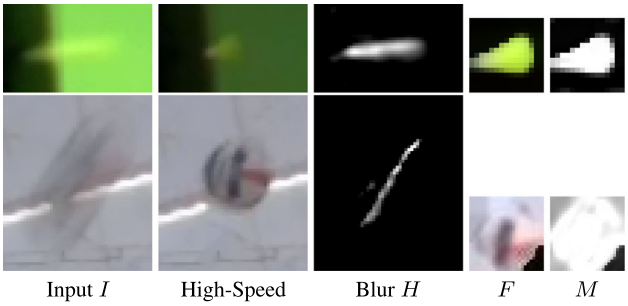


Fig. 5 Deblatting examples (top row - shuttlecock, bottom row - volleyball). From left to right: the input image I, the corresponding highspeed camera frame; estimated blur H, estimated appearance F, and shape M

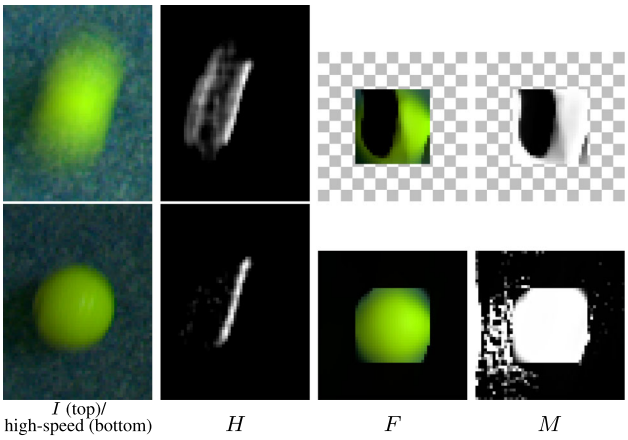


Fig. 6 Shadow and blur estimation: single example showing different shadow effects. Top (undersized domain): the domain of F is set too small and the shadow causes artifacts in H. Bottom (oversized domain): the domain of F is larger, M can compensate for the shadow and the blur H is estimated correctly

2.2 Trajectory Fitting

Fitting the blur kernel H, which is a gray-scale image, with a trajectory $\mathcal{C}(t)$: $[0, 1] \rightarrow \mathbb{R}^2$ serves three purposes. First, we use the error of the fit in the Consistency Check to determine if H is the motion blur induced by the tracked object and thus whether to proceed with tracking, or to declare the deblatting step a failure and to reinitialize it with different parameters. Second, the trajectory as an analytic curve can be used for motion prediction whereas H cannot. Third, $\mathcal{C}$ defines the intra-frame motion, which is the desired output of the proposed method.

The fitting is analogous to vectorization of raster images. It is formulated as the maximum a posteriori estimation of the parametric trajectory $\mathcal{C}$, given blur kernel H, with the physical plausibility of the trajectory used as a prior. Let $\mathcal{C}$ be a curve defined by a set of parameters θ (e.g. polynomial coefficients) and $H_{\mathcal{C}}$ be a raster image of the corresponding trajectory $\mathcal{C}$ – calculated by rendering the curve into the discrete image. We say that the curve $\mathcal{C}$ is the *trajectory fit* of H if θ minimizes

$$\min_{\theta} \|H_{\mathcal{C}} - H\| \quad \text{s.t. } \mathcal{C} \in \Psi, \tag{4}$$

where Ψ is the set of admissible curves.

We assume that in each frame, the tracked object is in free flight except for a possible bounce or impulse from other objects and the environment. We thus define Ψ as a set of continuous piece-wise quadratic curves – quadratic to account for acceleration due to gravity and piece-wise to account for abrupt change of motion during bounces. The curve $\mathcal{C} \in \Psi$, $\mathcal{C} : [0, 1] \rightarrow \mathbb{R}^2$ is defined as

$$\mathcal{C}(t) = \begin{cases} \sum_{k=0}^{2} c_{1,k} t^k & 0 \leq t < \tilde{t}, \\ \sum_{k=0}^{2} c_{2,k} t^k & \tilde{t} \leq t \leq 1, \end{cases} \tag{5}$$

s.t. $\sum_k^2 c_{1,k} \tilde{t}^k = \sum_k^2 c_{2,k} \tilde{t}^k$. Parametrization of the non-smooth point (bounce) is denoted by $\tilde{t}$. Since the variable t represents merely the curve parametrization and does not correspond to any physical quantity, such as curve length or exposure time, we can fix $\tilde{t}$ to any suitable value (e.g. $1/2$), and the corresponding polynomial coefficients are then calculated accordingly. When the fitting is done, we reparameterize coefficients c such that the length proportions w.r.t. t are correct. Single linear or quadratic curves are considered as special case for which it formally holds: $\tilde{t} = 1$ and $c_{2,k} \equiv 0$. Problem (4) is non-convex, and thus a good initial guess is important for gradient-descent optimization to perform well. To this end, we employed a four-step procedure:

1. Identify the most salient linear and quadratic segments in H by RANSAC.
2. Connect segments to form a curve $\mathcal{C}$ of the kind (5).
3. Refine $\mathcal{C}$ to be a locally optimal fit of H in terms of point-wise distance.
4. Calculate the loss (4) and choose the best candidate.

See Fig. 7 for illustrations of the above steps.

Step 1 – Identify Let us view the blur H as a set of pixels with coordinates x_j and intensities $w_j > 0$. Sequential RANSAC finds line segments as follows: sample two points, find inliers of the corresponding line, find the most salient consecutive run of points on this line, and in each round remove the winner from the sampling pool. The saliency is defined as the sum of pixel intensities in the inlier set. The estimated blur H sometimes contains gaps, deviating from the expected contiguous line. We therefore relax the term "consecutive" and allow discontinuities of maximum 2 pixels between points on the line. The search stops when there are no more points to sample from, or when the saliency of any new potential segment falls below one percent of the total intensity of all

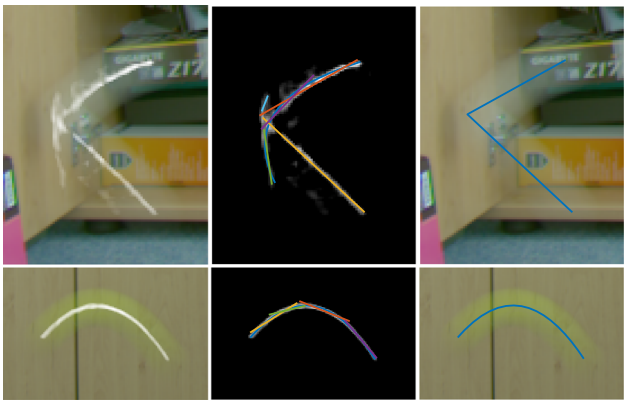


Fig. 7 Trajectory fitting. *Left* input image with estimated blur superimposed in white, *middle* linear and parabolic segments found by **RANSAC**, *right* final fitted trajectory

points. This stopping criterion helps to avoid unnecessary processing, which would anyway improve the line segment only negligibly. We denote the set of collected linear segments as $\mathcal{M}_1$. Parabolic arcs are found similarly. We sample four points, find two corresponding parabolas, and project the remaining points on the parabolas to determine the distance, the inlier set, and the arc-length parametrization of inliers (required for correct ordering and mutual distance calculation of inliers). Then, we again find the most salient consecutive run. We denote the set of collected parabolic segments as $\mathcal{M}_2$.

Step 2 – Connect The solution will be close to a curve formed from one or two segments (linear or parabolic) found so far. Let $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{M}_1$ be two linear segments. If the intersection P of the corresponding lines is close to the segments (w.r.t. some threshold), the curve connecting $\mathcal{C}_1 \rightarrow P \rightarrow \mathcal{C}_2$ is a candidate for the piece-wise linear trajectory fit. This way we construct a set $\mathcal{M}_3$ of all candidate and similarly $\mathcal{M}_4$ with candidates of parabolic pairs.

Step 3 – Refine Curves in $\mathcal{M} = \bigcup \mathcal{M}_i$ are approximate candidates for the final trajectory, yet we first refine them to be locally optimal robust fits to H. Let the blur kernel H be interpreted as a set of pixels at coordinates x_j with nonzero intensities w_j. We say that a curve $\mathcal{C}$ defined by a set of parameters θ is locally optimal fit to $\{x_j\}$ if θ is the minimizer of the problem

$$\min_{\theta} \sum_{x_j \in K} w_i \operatorname{dist}(x_j, \mathcal{C}) + \lambda_d \int_0^1 \operatorname{dist}(\mathcal{C}(t), \{x_j\}) \mathrm{d}t \qquad (6)$$

where $K = \{x_j | \operatorname{dist}(x_j, \mathcal{C}) < \rho\}$, $\operatorname{dist}(x, \mathcal{C})$ is the distance of the point x to the curve $\mathcal{C}$ and $\operatorname{dist}(\mathcal{C}(t), \{x_j\})$ is the distance of the curve point $C(t)$ to the set $\{x_j\}$. In the first term, K is a set of inliers defined by the distance threshold ρ, and $\mathcal{C}$ is the distance-optimized fit to inliers. The second term restricts curve length. Ideally, the estimated blur kernel H is a curve 1px thick. Therefore, the inlier threshold ρ should be close to one. We set $\rho = \sqrt{2}$, which is the maximum distance of neighbors in the standard 8-connected neighborhood.

The gradient of (6) is intractable since the distance of a point x to a non-convex set (in our case the curve $\mathcal{C}$) is intractable. We therefore resort to a procedure similar to the Iterative Closest Point (ICP) algorithm. We refine every curve in $\mathcal{M}$ by solving (6) with the ICP-like algorithm. In each iteration, we fix the currently closest curve counterpart $y_j = \mathcal{C}(t_j)$ for each point x_j by solving the equation $t_j = \operatorname{argmin}_t \operatorname{dist}(x_j, \mathcal{C}(t))$, and in (6) we approximate $\operatorname{dist}(x_j, \mathcal{C}) \approx \|x_j - y_j\|$. We proceed analogically for $\operatorname{dist}(\mathcal{C}(t), \{x\})$. Then, Eq. (6) becomes a tractable function of θ. We find the solution using the Iteratively Re-weighted Least-Squares algorithm and proceed with the next iteration of ICP. The algorithms converges in a few iterations, and the optimization is fast.

Step 4 – Finalize For each refined curve $\mathcal{C} \in \mathcal{M}$, we construct $H_{\mathcal{C}}$, measure the error $\|H_{\mathcal{C}} - H\|$, and choose the best candidate as the trajectory fit $\mathcal{C}_i(t) : [0, 1] \rightarrow \mathbb{R}^2$ of the current frame I_i. The TbD Consistency Check is performed after every deblatting loop (Fig. 3) by evaluating the criterion of the best trajectory fit $\mathcal{C}_i$

$$\|H_{\mathcal{C}_i} - H_i\| / \|H_i\| < \tau. \qquad (7)$$

The goal of TbD is to produce a precise intra-frame motion trajectory, and not only a single position per frame in the form of a bounding box. Fig. 7 shows examples of trajectory estimation. The left column is the input image with the estimated blur kernel superimposed in white, and the right column shows the estimated motion trajectory. The efficacy of trajectory fitting is a crucial part of the framework. The estimated blur can contain various artefacts (e.g. in the top example due to the ball shadow), and the trajectory fit still recovers the actual motion.

The TbD outputs are individual trajectories $\mathcal{C}_i$'s and blur kernels H_i's in every frame. The outputs serve as inputs to the proposed non-causal TbD 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 is assumed to have unit duration, and the object in each frame is visible only

for duration of exposure fraction $\epsilon \leq 1$. The sequence is divided into S segments defined by timestamps t_s's such that $0 = t_0 < t_1 < \ldots < t_s < \ldots < t_{S-1} < t_S = N$. Splitting into segments is discussed in Sect. 3.1. Similarly to polynomial fitting in TbD (Sect. 2.2), $\mathcal{C}_f(t)$ is represented as a piece-wise polynomial function

$$\mathcal{C}_f(t) = \begin{cases} \sum_{k=0}^{d_1} \bar{c}_{1,k} t^k & 0 \leq t < t_1, \\ \vdots & \vdots \\ \sum_{k=0}^{d_S} \bar{c}_{S,k} t^k & t_{S-1} \leq t \leq N, \end{cases} \tag{8}$$

In each segment s, we fit x and y polynomials of degree d_s with coefficients $\bar{c}_s := \{\bar{c}_{s,k} | k = 0, \ldots, d_s\}$, where $\bar{c}_{s,k} \in \mathbb{R}^2$ are coefficients of the k-th degree. Unlike in TbD trajectory fitting (5), where we assume at most two quadratic polynomials ($S = 2, d_s = 2$), here the number of polynomials is equal to the number of segments S, which is typically more than 2, and the degree d_s in each segment can differ. The degree depends on the number of frames in the segment, i.e. $t_s - t_{s-1}$, as explained in Sect. 3.2. We also enforce the final trajectory be continuous, and the segment endpoints be consistent within the whole trajectory.

Polynomials of degree two model only free falling objects under the gravitational force and were sufficient for fitting short curves in TbD. However, when fitting curves spanning longer time intervals, forces such as air friction and wind start to be noticeable. These forces can be approximated by Taylor expansion, which is equivalent to adding higher degrees to the fitted polynomials. We validated experimentally, as shown Fig. 9, that the 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. Notice that circular motion can also be approximated by (8).

A rough overview of the structure of the proposed method follows. The whole approach to estimate the piece-wise polynomial function (8) is based on three main steps. In the first step, the sequence is decomposed into non-intersecting parts. Using dynamic programming, each part is converted into a discrete trajectory by minimizing an energy function. The energy function combines information from partial trajectories estimated by the causal TbD, the curvature penalizer to force smooth trajectories, and the trajectory length penalizer. In the second step, the discrete trajectory is further decomposed into segments by detecting bounces. Then, segments define frames that are used for fitting each polynomial. In the third step, we fit polynomials of degree up to six that define the final trajectory function $\mathcal{C}_f(t)$. Each step is thoroughly explained in the following subsections.

3.1 Splitting into Segments

When tracking fast moving objects in long-term scenarios, objects commonly move back and forth, especially in rallies. During their motion, fast moving objects 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 piece-wise 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. We convert blur kernels H_t from all frames in the given non-intersecting part into a single discrete trajectory by dynamic programming. The proposed dynamic programming approach finds the global minimum of the following energy function

$$\begin{aligned} E(P) = &- \sum_{x=x_b}^{x_e} \sum_{t=t_{s-1}}^{t_s} H_t(x, P_x) l_t \\ &+ \kappa_1 \sum_{x=x_b+2}^{x_e} \left| (P_x - P_{x-1}) - (P_{x-1} - P_{x-2}) \right|^2 \\ &+ \kappa_2 (x_e - x_b) , \end{aligned} \tag{9}$$

where variable P is a discrete version of trajectory $\mathcal{C}$, and it is a mapping that 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 that accumulates more values from blur kernels while our energy function is being minimized. Each blur kernel is multiplied by the trajectory length l_t estimated from TbD in order to normalize each blur kernel and force each pixel on the trajectory to have value approximately 1. The second term penalizes direction changes and is defined as the difference between directions of two following points – an approximation of the second order derivative of P. The value is squared, so that several consecutive small changes are more preferable than one large change in direction. This term makes trajectories smoother, and κ_1 serves as a smoothing parameter. Parameter κ_1 is set to 0.5, assuming that values of pixels at trajectory are near 1. The last term enforces shorter trajectories by penalizing each

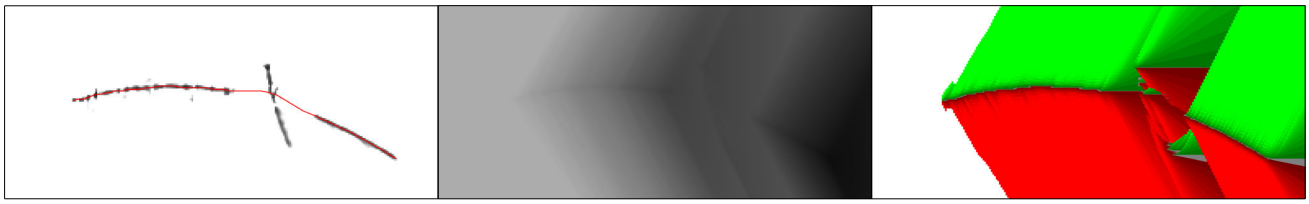


Fig. 8 Example of dynamic programming. Left image: accumulated blur kernels (reverted for visualization) from four consecutive frames between $H_{t_{s-1}}$ and H_{t_s} in the joint coordinate system, with the estimated discrete trajectory P marked in red. Middle image: value of the energy function at each pixel from black (lowest) to white (highest). Right image: pixels where optimal move is downwards are marked in green (brighter means steeper), upwards in red (brighter means steeper), and moving straight in gray. 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) (Color figure online)

additional pixel. Parameter κ_2 is set to 0.1, which ensures that values of pixels along the trajectory are on average more that κ_2 and forbids prolonging the trajectory to get pixels with the value less than κ_2. The algorithm is not sensitive to values of κ_1 and κ_2, and any value in the range between 0.05 and 0.7 achieves similar results. Discrete trajectory P is defined from x_b until x_e, and these two variables are also being estimated. In short, dynamic programming estimates the trajectories that correspond to causal trajectories as much as possible, while being smooth (controlled by κ_1) and short (controlled by κ_2).

Energy minimization Energy function E (9) is minimized by a dynamic programming (DP) approach. To account for camera rotation or objects travelling from top to bottom, we consider independently two cases: the accumulated blur kernels H_t and rotated H_t by 90 degrees. For both options, we find the global minimum of E and the one with lower energy is chosen. We validated experimentally that the pixel with the lowest energy has an average distance of 2.8 pixels to the ground truth ending point. Considering both the original and the rotated version is important in order to improve rotation invariance of the proposed method, as experimentally validated in Fig. 12. Let us illustrate the approach for the original non-rotated case; see Fig. 8. The rotated 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 all options and choosing the one with the lowest E: either adding to the trajectory one sub-pixel out of nearest ones in the previous column with y coordinate difference between $+2$ and -2, or choosing the current pixel as the starting point. Threshold ± 2 indicates that the non-causal trajectories cannot have angles more than 60 degrees in one step. Larger threshold (i.e. angle) can help to find better trajectories, but then the complexity of the dynamic programming will increase and also the trajectory will be less smooth. The pixels are discretized by a step size of 0.2, which means that 21 possible sub-pixels are checked. The values in blur kernels are linearly interpolated. Both the minimum energy (Fig. 8 middle) and the decision option (Fig. 8 right) in every pixel is stored. When all columns are checked, a pixel with the minimum energy (Fig. 8 middle) is selected as the end point and the trajectory is estimated by backtracking following decision options (Fig. 8 right). Backtracking finishes when a pixel is reached with the starting-point decision (white in Fig. 8 right).

Bounces When each non-intersecting part is converted into 1D signal, it becomes easier to find bounces, i.e. points with abrupt changes of direction. The given point is considered a bounce when both points on the left and on the right with the distance w to the given point have a change of direction greater than 3 pixels with the same sign. Threshold w controls sensitivity of the bounce. In the FMO setting, smaller trajectories imply low speed and more bounces. Thus, we set the sensitivity automatically for each point based on the trajectory length in the closest frame, i.e. $w = l_t/4$. In the case of circular motion with no bounces, the approach finds the most suitable point to split the circle. After this step, the sequence is split into segments that are separated by bounces.

3.2 Fitting Polynomials

The output discrete trajectory P is used to estimate bounces and define segments. It also determines which frames belong to the segment and should be considered for fitting polynomials. To this end, we assign starting ($\mathcal{C}_t(0)$) and ending ($\mathcal{C}_t(1)$) points of each frame to the closest segment. For fitting, we use only frames that completely belong to the segment, i.e. the whole trajectory in the frame is closer to this segment than to any other segment. 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

$$d_s = \min(6, N_s/3). \tag{10}$$

We restrict polynomials to degree up to 6, as higher degrees tend to overfit (Fig. 9). With this setting, we observed no oscillations that are typical for overfitting, but they were visible for degrees higher than 8. Our interpretation is that

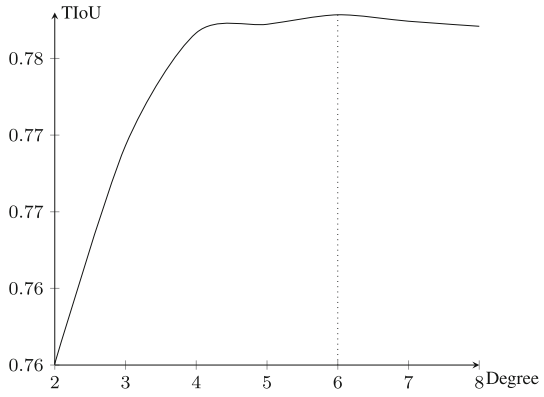


Fig. 9 The influence of maximal polynomial degree. The dotted line shows the location of the best setting: polynomial of degree 6. Vertical axis: Trajectory-IoU (14) on the TbD dataset

the trajectories provide sufficiently strong constraints. The degree is adaptive – if the trajectory is short, the degree is decreased according to Eq. (10). The polynomials are further constrained by the continuity conditions between frames.

The polynomial coefficients are found by solving:

$$\min_{\bar{c}_s} \sum_{t=t_{s-1}}^{t_s} \int_0^1 \|\mathcal{C}_f(t+\tau\epsilon) - \mathcal{C}_t(\tau)\|^2 \mathrm{d}\tau, \tag{11}$$

s.t. $\mathcal{C}_f(t_{s-1}) = \mathcal{C}_{t_{s-1}}(0)$ and $\mathcal{C}_f(t_s+\epsilon) = \mathcal{C}_{t_s}(1)$. After the integral approximation as the sum of two end-points, the minimization problem becomes

$$\min_{\bar{c}_s} \sum_{t=t_{s-1}}^{t_s} \|\mathcal{C}_f(t) - \mathcal{C}_t(0)\|^2 + \|\mathcal{C}_f(t+\epsilon) - \mathcal{C}_t(1)\|^2, \tag{12}$$

s.t. $\mathcal{C}_f(t_{s-1}) = \mathcal{C}_{t_{s-1}}(0)$ and $\mathcal{C}_f(t_s+\epsilon) = \mathcal{C}_{t_s}(1)$, where s denotes the segment index. The minimization w.r.t. polynomial coefficients $\bar{c}_s = \{\bar{c}_{s,k} | k = 0, \ldots, d_s\}$ is a linear least-squares problem for each segment independently. 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 least-square cost function is a common choice that is computation convenient. The final trajectory $\mathcal{C}_f$ is defined over the whole sequence. The last visible point in the frame t, i.e. $\mathcal{C}_t(1)$, corresponds to $\mathcal{C}_f(t+\epsilon)$ in the sequence time-frame. The exposure fraction ϵ is assumed to be constant in the sequence. It is estimated as an average ratio between trajectory lengths l_t and the expected length of full-exposure trajectory:

$$\epsilon = \frac{1}{N-1} \sum_{t=1}^{N-1} \frac{l_t}{l_t + \|\mathcal{C}_{t+1}(0) - \mathcal{C}_t(1)\|}. \tag{13}$$

Frames that belong only partially to segments contain bounces. We replace them with a piece-wise linear polynomial that connects the last point from the previous segment, bounce point found by DP, and the first point from the following segment. Frames between non-intersecting parts are also interpolated by piece-wise linear polynomial that connects the last point of the previous segment, point of intersection of these two segments, and the first point of the following segment. Frames that are before the first detection or after the last non-empty $\mathcal{C}_t$ are extrapolated by the closest segment. Fig. 10 shows an example of splitting a sequence into segments that are used for fitting polynomials. More examples of full trajectory estimation are in Fig. 11.

4 Choice of Parameters

All parameters of the proposed method can be split into fixed and adaptive. Most parameters are fixed to a certain value that has been logically chosen based on the problem characteristics. The correct choice of parameters is validated by running an additional experiment. Fig. 16 shows examples of several randomly found YouTube videos with fast moving objects. Correctly detected objects and estimated trajectories indicate that the chosen set of parameters can generalize well to other unseen videos.

Fixed parameters We use the following L^1 weight on H in deblatting (3): $\alpha_H = 0.2$. The TV weight on F in Eq. (3) is set to $\alpha_F = 0.001$. For deblurring, we set relative tolerance to 0.01 and maximum number of iterations to 15. The background is estimated as a median of last 5 frames. Template-matching term λ in Eq. (3), (21) is fixed to 0.1, as it provides the best results (Fig. 13). The threshold for Consistency Check τ in Eq. (7) is set to 0.15. The value of other fixed parameters is explained directly in the main text when the parameter is defined.

Adaptive parameters The scale of the object, denoted by domain D^o, is found by the FMO detector from (Rozumnyi et al. 2017) as a sphere with radius equal to the maximal value of the distance transform of the detected stroke. If the template is given as in TbD-T1, domain D^o is given as well as part of the template. Parameter w of the sensitivity of the bounce is set adaptively depending on the trajectory length in one frame. Degree d of the fitted polynomial depends on the number of frames that belong to the segment. The exposure fraction ϵ is also set adaptively based on the average ratio between consecutive trajectory lengths.

5 Experiments

We show the results of Tracking by Deblatting and compare it with other trackers on the task of long-term tracking of motion-blurred objects in real-life video sequences. As a baseline, we chose the FMO detector (FMOd, (Rozumnyi et al. 2017)), specifically proposed for detection and tracking of fast moving objects, and the Discriminative Correlation Filter with Channel and Spatial Reliability (CSR-DCF, (Lukežič et al. 2017)), which performs well on standard benchmarks such as VOT (Kristan et al. 2016). CSR-DCF was not designed to track objects undergoing large changes in velocity within a single sequence and would perform poorly in the comparison. We therefore augmented CSR-DCF by FMOd reinitialization every time it outputs the same bounding box in consecutive frames, which is considered a fail. We use FMOd for automatic initialization of both TbD and CSR-DCF to avoid manual input. We skip the first two frames of every sequence to establish background B and initialize CSR-DCF. the background B is estimated as a moving median of the past 3 - 5 frames. The rest of the sequence is processed causally.

The comparison with baseline methods was conducted on a new dataset consisting of 12 sequences with different objects in motion and settings: different kinds of sports, objects in flight or rolled on the ground, indoor/outdoor. The sequences contain abrupt changes of motion, such as bounces and interactions with players, and a wide range of speeds. Videos were recorded with a high-speed camera at 240 fps with exposure time 1/240s (exposure fraction $\epsilon \rightarrow 1$). The sequences for evaluation with 30 fps were generated by averaging 8 consecutive frames. The dataset was annotated with trajectories obtained from the original high-speed camera footage. We compare the method performance in predicting the motion trajectory in each frame. We therefore generalize Intersection over Union (IoU), the standard measure of position accuracy, to trajectories and define a new measure *Trajectory-IoU* (TIoU):

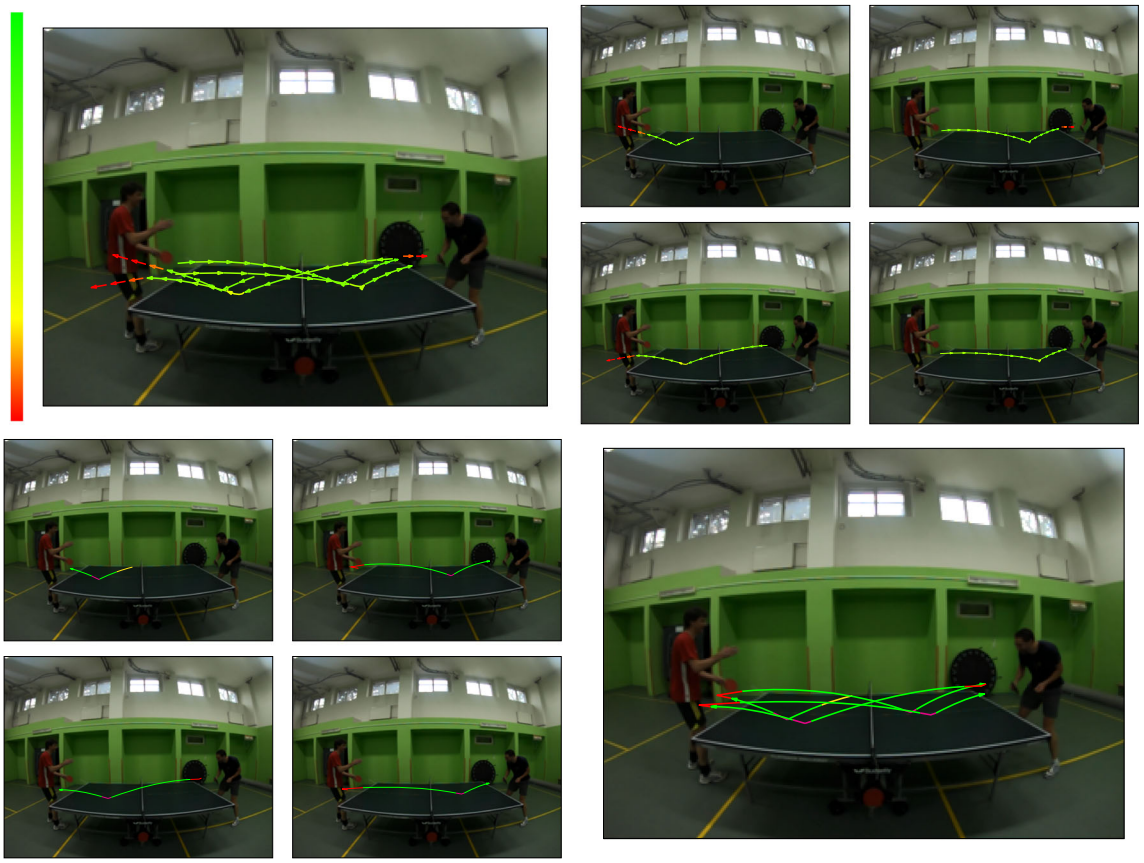


Fig. 10 TbD-NC processing steps (Sect. 3). From left to right, top to bottom: causal TbD output, splitting into segments, fitting polynomials to segments, final TbD-NC output. Top row: trajectories for all frames overlaid on the first frame, Trajectory-IoU accuracy measure color coded from red (failure) 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 (Color figure online)

Fig. 11 Trajectory recovery for sequences selected from the TbD dataset. Top row: trajectories estimated by the causal TbD overlaid on the first frame. TIoU (14) with ground truth trajectories from a high-speed camera is color coded by scale in Fig. 10. Bottom row: trajectory estimates by the proposed TbD-NC that outputs continuous trajectory for the whole sequence. The yellow curves underneath denote the ground truth. Arrows indicate the direction of the motion

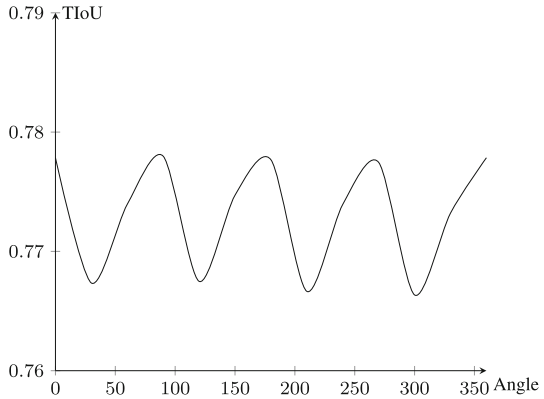


Fig. 12 The influence of rotation on TbD-NC. All inputs to the method are rotated by a certain degree (0-360) and compared to the ground truth rotated by the same angle. The method is invariant to rotations by 90, 180, and 270 degrees. Performance scores repeat with the period of 90 degrees. The lowest performance is achieved when the rotation is 45 degrees due to interpolation errors. Vertical axis: Trajectory-IoU (14) on the TbD dataset

$$\mathrm{TIoU}(\mathcal{C}, \mathcal{C}^*; M^*) = \int_t \mathrm{IoU}\left(M^*_{\mathcal{C}(t)},\ M^*_{\mathcal{C}^*(t)}\right) \mathrm{d}t, \tag{14}$$

where $\mathcal{C}$ is the predicted trajectory, $\mathcal{C}^*$ is the ground-truth trajectory, M^* is a disk mask with true object diameter obtained from the ground truth, and M_x denotes M placed at location x. TIoU can be regarded as the standard IoU averaged over each position on the estimated trajectory. In practice, we discretize the exposure time into evenly spaced timestamps and calculate IoU between the ground-truth and the prediction. Since the CSR-DCF tracker only outputs positions, it estimates only linear trajectories from positions in neighboring frames.

The results of the comparison are presented in Table 2. We evaluated three flavors of TbD that differ in the presence of the initial user-supplied template $\hat{F}$ and the learning rate γ of the object model in (2). The presented flavors are:

- TbD-T0,0: Object template is not available, model update is instantaneous (memory-less), $\gamma = 0$.
- TbD-T0,0.5: Object template is not available, model is updated with the learning rate $\gamma = 0.5$.
- TbD-T1,1: Object template is available, model remains constant and equal to the template, $\gamma = 1$.
- TbD-NC: non-causal TbD-T1,1 with full trajectory estimation (Sect. 3).

Empirical justification of chosen learning rates is presented in Fig. 13. We evaluated all learning rates from 0 to 1 with the step size 0.05 for each method, i.e. TbD-T1 and TbD-T0. For each step size, the average TIoU was computed over a subset of the TbD dataset, and the best performing setting was chosen. When the template is not available, updating model smoothly with the rate between 0.4 and 0.6 generally outperforms other settings irrespective of the chosen template-matching weight λ. We have therefore selected $\gamma = 0.5$, which is slightly better than the instantaneous update ($\gamma = 0$) and no update at all ($\gamma = 1$ keeps the first estimate as the template). When the template is available,

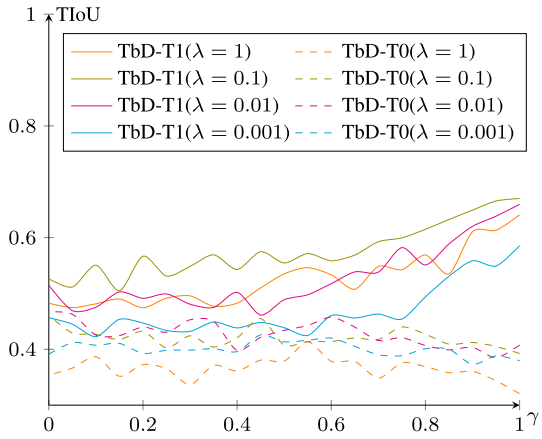


Fig. 13 Exponential forgetting factor estimation for TbD-T0 and TbD-T1 methods. The graph compares performance in terms of Trajectory-IoU over a subset of the TbD dataset with varying exponential forgetting factors for updating the object model. TbD-T0 has no object template, and the best performance is achieved for $\gamma = 0.5$. TbD-T1 was provided with the object template, and the best performing setting was for $\gamma = 1$

it is preferable to keep the template rather than updating it, however this conclusion depends on the template quality. Even when no update is done ($\gamma = 1$), it is still necessary to minimize the loss (3) with respect to F. Template $\hat{F}$ usually contains only object-specific details. Image noise and other phenomena such as shadows or illumination changes should not be part of the template, but minimization with respect to F models these variables and helps to correctly estimate H.

The TbD outperforms baseline methods on average by a wide margin, both in the traditional recall measure (a detection is called true positive if it has non-zero TIoU) as well as in trajectory accuracy TIoU. FMOd is less accurate and more prone to false positives as it lacks any prediction step and by design ignores slow objects. CSR-DCF, despite reinitializations by FMOd, fails to detect fast moving objects accurately. Among TbD flavors, it is no surprise that availability of the object template is beneficial and outperforms other versions. However, even if the template is not available, TbD can learn the object model, and updating the appearance model gradually during tracking is preferable to instantaneous updates.

To evaluate the performance of the core part of TbD that consists of deblatting and trajectory fitting alone, we provide results of a special version of the proposed method called "TbD with oracle", TbD-O. This behaves like regular TbD but with a perfect trajectory prediction step. We use the ground-truth trajectory to supply the region D to the deblatting step exactly as if it were predicted by the prediction step, effectively bypassing the long-term tracking logic of TbD. The rest is identical to TbD-T1,1. TbD with oracle tests the performance and potential of the deblatting and trajectory estimation alone, because failures do not cause long-term damage – success in one frame is independent of success in the previous frame.

Non-causal Tracking by Deblatting (TbD-NC) is based on the TbD-T1,1 version. TbD-NC provides a performance boost both in recall and TIoU. Recall is 100% in all cases except for one, where the first detection appeared only on the

Table 2 Trajectory Intersection over Union (TIoU) and Recall (Rcl) on the TbD dataset – comparison of CSR-DCF tracker and the FMO method in Rozumnyi et al. (2017) to TbD and TbD-NC. CSR-DCF proposed by Lukežič et al. (2017) is a standard, well-performning as shown in Kristan et al. (2019), near-real time tracker. TbD tracker settings: TbD without template and with exponential forgetting factors (2) $\gamma = 0$ (TbD-T0, 0) and $\gamma = 0.5$ (TbD-T0, 0.5), TbD with template and $\gamma = 1$ (TbD-T1, 1), non-causal version of the previous TbD setting (TbD-NC), TbD with oracle (TbD-O). The highest TIoU for each sequence is highlighted in bolditalics and the highest recall in italics. TbD-O shows the highest attainable TIoU for TbD as a reference point when predictions are precise

Sequence name	Frames	CSR-DCF		FMO method		TbD-T0, 0		TbD-T0, 0.5		TbD-T1, 1		TbD-NC		TbD-O
		TIoU	Rcl	TIoU	Rcl	TIoU	Rcl	TIoU	Rcl	TIoU	Rcl	TIoU	Rcl	TIoU
Badminton_white	40	.286	0.39	.242	0.34	.659	0.92	.657	0.92	.694	0.97	***.783***	*1.00*	.792
Badminton_yellow	57	.123	0.22	.236	0.31	.615	0.89	.626	0.89	.677	0.91	***.780***	*1.00*	.788
Pingpong	58	.064	0.12	.064	0.12	.581	0.89	.590	0.89	.523	0.91	***.643***	*1.00*	.697
Tennis	38	.278	0.64	.596	0.78	.596	0.92	.554	0.89	.673	0.97	***.750***	*1.00*	.827
Volleyball	41	.533	0.82	.537	0.72	.552	0.87	.591	0.90	.795	0.97	***.857***	*1.00*	.836
Throw_floor	40	.287	0.71	.272	0.37	.760	*1.00*	.776	*1.00*	.810	*1.00*	***.864***	*1.00*	.864
Throw_soft	60	.470	0.97	.377	0.57	.584	0.90	.564	0.90	.652	0.97	***.765***	*1.00*	.707
Throw_tennis	45	.444	0.95	.507	0.65	.693	*1.00*	.777	*1.00*	.850	*1.00*	***.880***	*1.00*	.872
Roll_golf	16	.331	*1.00*	.187	0.71	.414	*1.00*	.346	*1.00*	.873	*1.00*	***.894***	*1.00*	.898
Fall_cube	20	.324	0.67	.408	0.78	.597	*1.00*	.590	*1.00*	.721	*1.00*	***.757***	*1.00*	.744
Hit_tennis	30	.329	0.93	.381	0.68	.564	0.93	.570	0.93	.667	0.93	***.725***	*1.00*	.828
Hit_tennis2	26	.214	0.79	.414	0.71	.476	0.83	.496	0.83	.616	0.83	***.682***	*0.92*	.738
Average	39	.307	0.68	.352	0.56	.591	0.93	.595	0.93	.713	0.96	***.782***	*0.99*	.799

Table 3 Comparison of TbD-NC with TbD. TbD failure is defined as frames where TIoU (14) 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

seventh frame, and extrapolation to the first six frames was not successful. Table 3 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.

A visual demonstration of the tracking by the proposed method on the TbD dataset is shown in Fig. 11. Each image shows results of tracking in one sequence from the evaluation dataset superimposed on a single image from the sequence. Arrows depict trajectories detected in a particular frame, and the color encodes the corresponding TIoU from green=1 to red=0 (false positive). We can see that the causal TbD trajectories are estimated successfully with the exception of frames where the object is in direct contact with other moving objects, which throws off the local estimation of background. This instability is fixed by the non-causal TbD in the bottom row.

Table 4 shows aggregated results for the FMO dataset introduced by (Rozumnyi et al. 2017). This dataset does not contain ground-truth trajectory data. Therefore, we report traditional precision/recall measure that is derived from the detection and ground-truth bounding-box IoU. On this dataset, the proposed TbD and TbD-NC methods are slightly better in recall, owing to the fact that initial detection is done by FMOd – if FMOd fails, then TbD cannot start the tracking. However, the proposed methods are significantly better in terms of precision. TbD-T1 is not evaluated, as templates for FMO dataset are not available.

6 All-Speed Tracking

The inner part of the TbD method consists of deblatting and fitting that allow estimating robust intra-frame object locations. Speed of the object can be arbitrary, albeit performance is better for higher speed when the object is not perfectly round and homogeneous. We evaluated the performance of the TbD-NC method on the extended TbD dataset (eTbD) that contains the same sequences as the TbD dataset but with on average around twice more frames with objects slowing down and staying still. Originally, the eTbD dataset was created first. Then, the TbD dataset was made by cropping the eTbD dataset, such that all speeds are represented equally.

For normalization, we represent speed in radii per exposure that measures the number of radii the object travels in one exposure time, as shown in Fig. 15. Speeds less than one radii per exposure $[r/\epsilon]$, i.e. not FMOs, represent half of frames in the eTbD dataset, and the other half contains FMOs. Table 5 shows results on the eTbD dataset for the TbD-NC method and compares it to other baselines.

In Fig. 14, we report histograms of performance of all-speed tracking for every method, measured by the average

Table 5 Trajectory Intersection over Union (TIoU) and Recall (Rcl) on the eTbD dataset. Extended version of the TbD dataset is used to evaluate the performance of TbD-NC in long-term scenarios and on objects with different speeds, ranging from still objects to very fast moving objects. The number of frames is denoted by "#" sign. The proposed TbD-NC performs better than the baselines FuCoLoT in (Lukežič et al. 2019) and the FMO method in (Rozumnyi et al. 2017). TIoU and Recall are lower than on the TbD dataset (Table 2) due to more challenging tasks in the eTbD dataset

Sequence	#	FuCoLoT		FMO		TbD-NC	
		TIoU	Recall	TIoU	Recall	TIoU	Recall
badmin._w.	125	.232	0.40	.142	0.19	***.635***	*0.85*
badmin._y.	125	.155	0.33	.229	0.30	***.536***	*0.84*
pingpong	95	.062	0.10	.100	0.15	***.604***	*0.98*
tennis	118	.245	*0.84*	***.554***	0.74	.420	0.58
volleyball	72	.500	0.79	.430	0.56	***.814***	*0.97*
throw_floor	73	.147	0.34	.153	0.21	***.896***	*1.00*
throw_soft	150	.516	0.98	.303	0.51	***.790***	*1.00*
throw_ten.	71	.232	0.99	.347	0.46	***.867***	*1.00*
roll_golf	16	.360	*1.00*	.187	0.71	***.894***	*1.00*
fall_cube	28	.414	0.77	.341	0.65	***.759***	*1.00*
hit_tennis	57	.330	0.96	.225	0.42	***.772***	*1.00*
hit_tennis2	26	.226	0.79	.414	0.71	***.681***	*0.92*
Average	80	.285	0.69	.285	0.47	***.722***	*0.93*

The highest TIoU for each sequence is highlighted in bolditalics and the highest recall in italics.

Table 4 Precision and recall of the TbD tracker (setting: TbD without template and with exponential forgetting factor (2) $\gamma = 0.5$), TbD-NC on top of it, and the FMO method in (Rozumnyi et al. 2017). We report the average on the 16 sequences of the FMO dataset

	FMO method		TbD-T0, 0.5		TbD-NC	
	Precision	Recall	Precision	Recall	Precision	Recall
Average	59.2	35.5	81.6	41.1	83.4	45.1

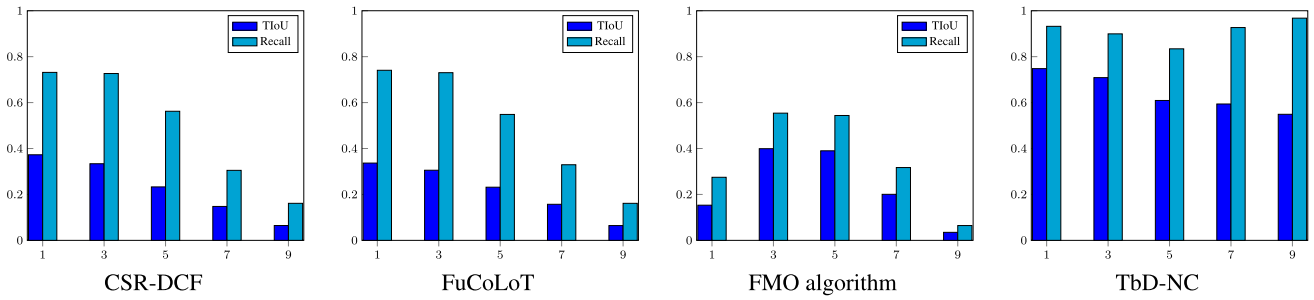


Fig. 14 All-speed tracking. Trajectory-IoU and recall on the extended TbD dataset (eTbD) for different algorithms (from left to right) – CSR-DCF in (Lukežič et al. 2017), FuCoLoT in (Lukežič et al. 2019), FMO algorithm in (Rozumnyi et al. 2017), and non-causal Tracking by Deblurring (TbD-NC). Horizontal axis denotes speed measured in radii per exposure. Vertical axis: the success rate measured by TIoU and the recall

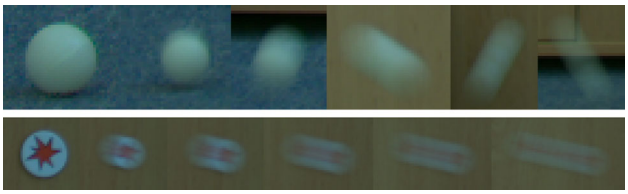


Fig. 15 Objects with varying speeds (0, 1, 3, 5, 7, 9) in radii per exposure, which removes dependence on camera settings and object size

TIoU in blue and by recall in cyan. Histogram bins represent different speeds ranging from 1 to 9 radii per exposure. We also compare TbD-NC to FuCoLoT tracker proposed by (Lukežič et al. 2019), which is a long-term extension of CSR-DCF tracker. General trackers such as CSR-DCF and FuCoLoT have similar performance that declines quickly for higher speeds. The FMO method proposed by (Rozumnyi et al. 2017) has peek performance for speeds between 3 and 5 radii per exposure. Lower or higher speeds decrease TIoU and recall drastically. The FMO method is based on difference images. Thus, very high speeds cause low contrast images, which makes the object almost invisible in the difference image. The FMO method was not designed to track not so fast moving objects. Its performance also drops for slow objects. The TbD method solves both problems and indeed connects the world of fast moving objects and the world of slow or still objects. For very high speeds, the TbD method does not suffer from low contrast images, because the image formation model is still valid. TbD-NC has a slightly decreasing TIoU for higher speeds, but its recall is close to one in all cases. Lower TIoU for higher speeds can be explained by the difficulty of deblatting and fitting when the object is severely blurred. When a severely blurred object has a color similar to the background, the part of the loss function that minimizes the L^1 norm of the blur kernel penalizes the blur too much. For sequences where this is the case, we lowered the weight α_H of the L^1 term that enforces sparsity of the blur kernel and reduces small non-zero values.

All-speed tracking posed another problem of estimating background when the object is close to still. The median of previous several frames is not sufficient. To this end, we increased the number of frames used for estimating the background to 20 previous frames, which is used when object speed is less than a threshold. For still objects with zero speed, the background is not updated.

7 Running Time

When the TbD tracker is initialized and frame-to-frame tracking operates smoothly, the average speed is close to 2 seconds per frame. When TbD fails, the necessary reinitialization takes 10-15 seconds. In total, the average speed on the dataset is 4 seconds per frame. TbD-NC is used only once for the sequence and takes on average 5 seconds per sequence. Runtimes are reported for Matlab implementations on a single CPU. More efficient CPU implementation in Python (approximately speed-up by 2) and GPU implementation in PyTorch (speed-up by 10) are open-sourced. In comparison, the FMO detector from (Rozumnyi et al. 2017) needs on average 1.5 seconds per frame. State-of-the-art trackers, such as CSR-DCF, are real-time and run at 30 frames per second.

8 Applications

Among the most interesting applications of the proposed method are temporal super-resolution and speed, shape, and gravity estimation, which are studied in the following subsections.

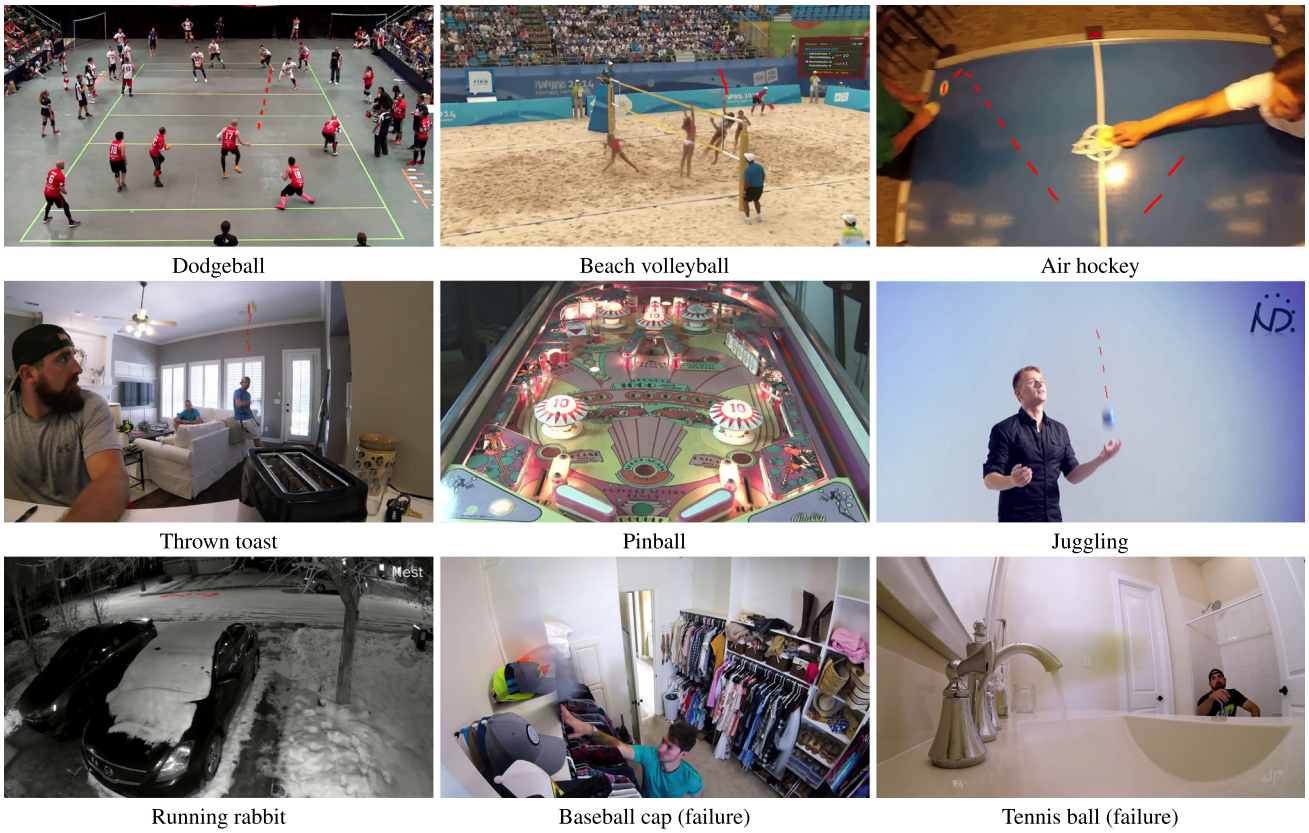


Fig. 16 Examples of sequences found on YouTube that contain fast moving objects. Estimated object trajectories by TbD from multiple frames are rendered in red into the last frame. The bottom middle image shows an example where the proposed method fails because of the object that is far from spherical and that is also undergoing significant rotation. The bottom right image shows a failure due to the object motion towards the camera. A list of videos with timestamps of FMO events is available at http://cmp.felk.cvut.cz/fmo

8.1 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 that 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 that are not FMOs 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 according to (1). Examples of these applications are provided in the supplementary material.

8.2 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 calculated velocity is a projection of real velocity to the image plane, e.g. perceived velocity. The results are visualized in Fig. 17, where sequences are shown together with their speed functions. The ground-truth speed was estimated from a high-speed camera footage with 8 times higher frame rate. The object center was detected in every frame. Then, the ground-truth speed was calculated from the distance between the object centers in adjacent frames. Deliberately, we used no prior information (regularization) to smooth the speed from high-speed camera. Therefore, it is noisy as can be seen in Fig. 17. We also report average absolute differences between the speed from the high-speed camera and the esti-

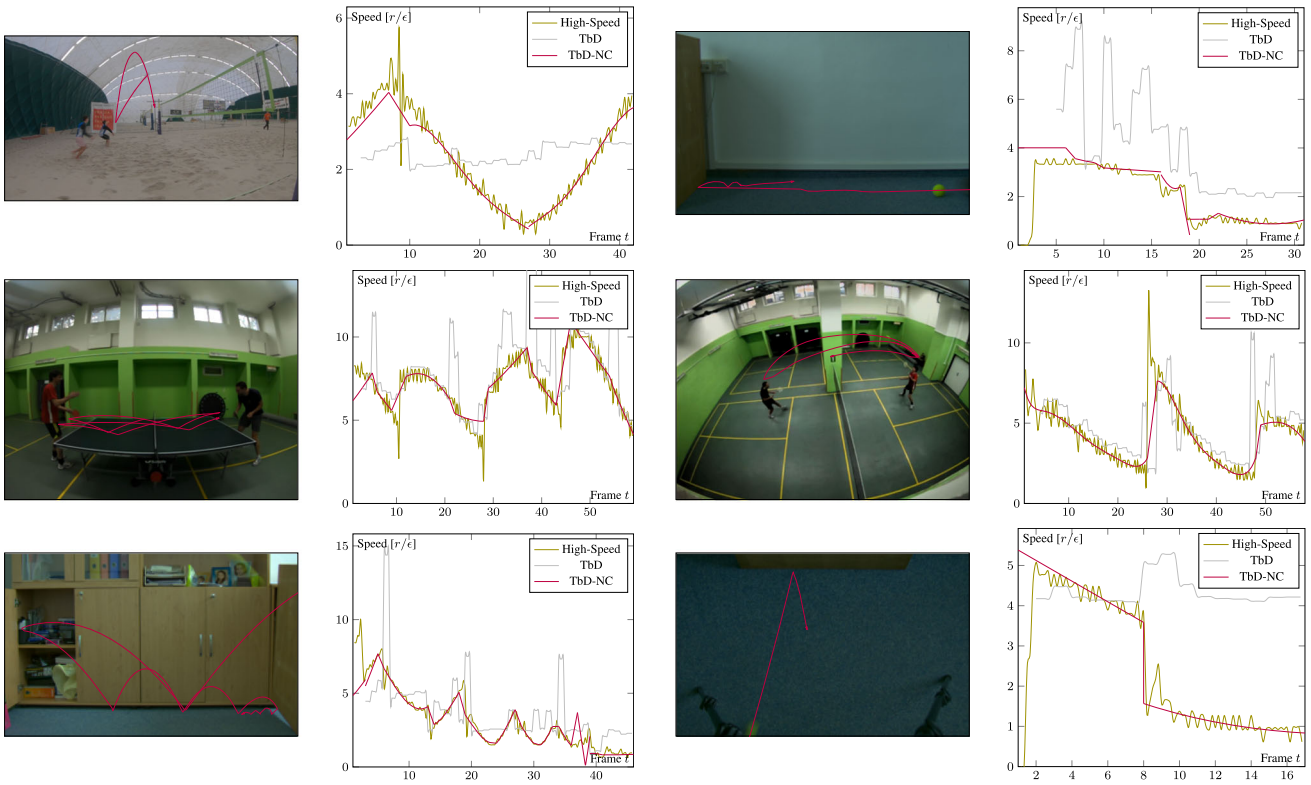


Fig. 17 Speed estimation using TbD-NC on selected sequences from the TbD dataset. Trajectories estimated by TbD-NC are overlaid on the first frame of each sequence. Graphs contain the speed estimation by TbD (lightgray) and TbD-NC (purple) in radii per exposure compared to speed calculated from the high-speed camera (olive). The noise and oscillations in high-speed camera estimates are caused by discretization. Errors for all sequences are shown in Table 7

Table 6 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 bolditalics

Serve	Duration	GT	(Hrabalík 2017)		TbD-NC	
	[frames]	[mph]	Speed [mph]	Error [%]	[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***

mated speed in Table 7. The error is mostly due to the noise in high-speed camera estimates.

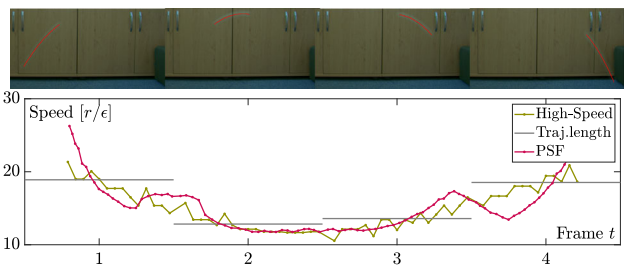


Fig. 18 Estimating object speed from blur kernels. In four consecutive frames (top row), object trajectories were estimated with TbD. The bottom plot shows the speed calculated from intensity values of blur kernels (solid red - averaged by a uniform filter of length 10) and from positions detected on a high-speed footage (olive - no averaging). Gray lines show the average velocity per frame calculated from the trajectory length (Color figure online)

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. The strongest signal usually happens immediately after the racquet hits the ball. (Hrabalík 2017) 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 that showed the measured speeds from radar guns (example in Fig. 19). A real-time version of FMO detector in (Hrabalík 2017) achieved accurate 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. The tennis ball is also 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 non-causal part of the proposed method (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 (Rozumnyi et al. 2017). Table 6 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. In several cases, the speed is estimated with GT error close to zero.

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, i.e. object speed is inversely proportional to the intensity value in the blur kernel. For example, if half of the exposure time the object was moving with a constant velocity, and then 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. Fig. 18 illustrates the speed estimation from the blur kernel and compares with rough estimation from the blur length. In practice, the speed estimation from the blur kernel is not very reliable due to the noise in H.

8.3 Shape and gravity estimation

In many situations, gravity is the only force that has non-negligible influence. In such cases, fitting second-degree polynomials is sufficient. If coefficients of the polynomial are estimated correctly, and the real gravity is given, then transforming pixels to meters in the region of motion is feasible. The coefficient a of the second-degree term corresponds to the gravity in units $[\text{px}(\frac{1}{f}s)^{-2}]$, where f is the frame rate. Assuming that the gravity of Earth is $g \approx 9.8[ms^{-2}]$. and that f is known, the formula to convert pixels to meters becomes $p = g/(2af^2)$, where p is in meters per pixel on the object in motion. The radius estimation by this approach is shown in Table 7. We use only sequences from TbD dataset with objects that were undergoing motion given only 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 drag (air resistance), and the tennis sequence was recorded outside during strong wind. When gravity was the only strong force, the estimation has an average error of 4.1 %. The variation of gravity on Earth is mostly negligible (< 0.2 %), 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 7, 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. If the fast moving object is a cloth in the wind, more physical quantities can be estimated as in (Runia et al. 2020).

9 Limitations

The introduced method has several limitations.

9.1 Stabilized Camera

Since the method requires an estimate of the background, the camera motion must be negligible within 3 to 5 frames needed for the estimate. However, we argue that in usual scenarios the background moves slowly in comparison to a fast moving object. Therefore, in the experiments, we stabilized all sequences by fitting a homography between consecutive frames. The method works best when the camera is nearly static.

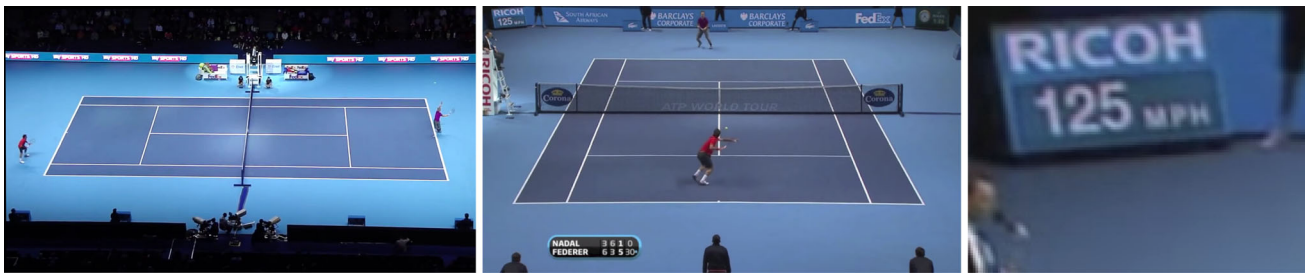
9.2 Object Appearance

We model the motion blur in Eq. (1) by convolution, which is obviously a simplification of the general case and is valid only if, from the camera vantage point, the object appearance F and its silhouette M remain constant during the exposure time. We thus make the following assumptions about the object and its motion:

– The object is spherical or can be approximated by a sphere. Fig. 16 (bottom middle image) illustrates a failure when this assumptions is grossly invalid. However,

Table 7 Estimation of radius, speed, and gravity by TbD-NC on the TbD dataset. 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



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

Fig. 19 Radar gun measurements. Speed was automatically estimated by the 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 6 compares estimates to the ground truth

if this assumption is only moderately inaccurate, e.g. in the case of a cube, a shuttlecock, or a toast (Fig. 16), the proposed method is still applicable.

– The object inner rotation and longitudinal motion do not produce significant changes to the perceived object appearance and size. This implies that the object must travel approximately parallel to the camera plane. An example where this assumptions is violated and the proposed method fails is shown in Fig. 16 (bottom right image).
– The background in the close vicinity of the object location in the frame is constant during exposure.

These assumptions serve to justify the simplifications made in (1) and establish the intended target scenarios of the proposed method. Generalization for the case of changing size, e.g. when the object is not moving in a plane orthogonal to optical axis is covered in a follow-up article by (Rozumnyi et al. 2020).

10 Conclusion

We proposed a novel approach – Tracking by Deblatting – intended for sequences in which the object of interest undergoes non-negligible motion within a single frame, which needs to be specified by intra-frame trajectory rather than a single position. The proposed methodology is based on an observation that motion blur is related to the motion trajectory of the object. Motion trajectories are estimated by a robust method combining blind deblurring, image matting, and shape estimation, followed by fitting a piecewise linear or quadratic curve that models physically plausible trajectories. As a result, we can precisely localize the object with higher temporal resolution than by conventional trackers.

As a second contribution we proposed non-causal Tracking by Deblatting (TbD-NC) that estimates accurate and complete trajectories of fast moving objects. TbD-NC is based on globally minimizing an optimality condition by dynamic programming. High-degree polynomials are then fitted to trajectory segments without bounces.

The proposed method was evaluated on a newly created TbD dataset of videos recorded with a high-speed camera using a novel Trajectory-IoU metric that generalizes the traditional Intersection over Union and measures the accuracy of the intra-frame trajectory. The proposed method outperforms baseline techniques both in recall and trajectory accuracy. The TbD-NC method performs well on the TbD dataset with complete failures appearing 10 times less often than the causal TbD. From the estimated trajectories, we are able to calculate precise object properties such as the object size and velocity. The speed estimation is compared to the data obtained from a high-speed camera and radar guns. More applications such as fast moving objects removal and temporal super-resolution are shown in the supplementary material.

Due to simplifications in blind deblurring to make optimization feasible, the method is currently limited to objects that do not significantly change their perceived shape and appearance within a single frame. The method works best for approximately round and uniform objects.

Funding Open Access funding provided by ETH Zurich.

A Deblatting

The objective of the deblatting operation is solving (3), which we do in a coordinate-descend manner. We fix F and M and solve the corresponding problem for H ("H-step"). Then, we fix H and solve for F and M simultaneously ("FM-step"). These two steps are repeated until convergence. However, both steps lead to nonlinear equations if solved directly. Therefore, it is convenient to split them further into simpler subproblems. Motivated by this, we use the ADMM (see e.g. (Boyd et al. 2011) and references therein). We introduce suitable variable substitutions and turn the original problem into a constrained problem. Then, the constrained problem is solved using the *augmented Langrangian method* – a weighted quadratic penalty corresponding to each substitution constraint is added to the standard Langrangian. By minimizing the Langrangian alternatingly w.r.t. the individual variables, the original problem effectively splits into multiple elementary subproblems that are easy to solve.

In the H-step we solve the problem

$$\min_H \frac{1}{2}\|H * F + (1 - H * M)B - I\|_2^2 + \alpha_H \|H\|_1, \quad (15)$$

s.t. $H \geq 0$. To separate the quadratic data term from the non-smooth L^1 regularizer and the positivity constraint, we introduce substitution $z := H$. According to the above outlined procedure, see also (Boyd et al. 2011, eqs. (3.5-7)), solving (15) then results in the following iteration steps performed in a loop:

$$z := \arg\min_z \left(\alpha_H \|z\|_1 + \frac{\rho}{2}\|H - z + u\|_2^2\right) \quad \text{s.t. } z \geq 0, \quad (16)$$

$$H := \arg\min_H \left(\frac{1}{2}\|H * F + (1 - H * M)B - I\|_2^2 + \frac{1}{2}\frac{\rho}{2}\|H - z + u\|_2^2\right), \quad (17)$$

$$u := u + H - z, \quad (18)$$

where u is a new variable corresponding to the Lagrange multiplier of the constraint $z = H$ (initialized to zero), and ρ is a positive weight of the quadratic augmentation of the Langrangian (a fixed user-defined parameter).

The update step (18) is trivial and dictated by the ADMM. The problem (16) presents a separate scalar problem for each pixel z_i. By direct differentiation w.r.t. z_i and inspection, it can be seen that the solution is a soft-thresholding of the minimizer of the quadratic term combined with projection onto $\mathbb{R}_0^+$,

$$z = \max\left(H + u - \frac{\alpha_H}{\rho},\, 0\right). \quad (19)$$

By differentiating w.r.t H, the problem (17) leads to a linear system for H

$$\left((\mathbf{F} - B\mathbf{M})^*(\mathbf{F} - B\mathbf{M}) + \rho\right) H = (\mathbf{F} - B\mathbf{M})^*(I - B) + \rho(z - u), \quad (20)$$

where $\mathbf{F}$ and $\mathbf{M}$ are operators performing convolution with F and M, respectively, and $(\cdot)^*$ denotes the corresponding adjoint operator. In practice, we use vector-matrix notation. All variables are vectors representing the vectorized images. Then, operators are matrices, and adjoints are their transpositions. Multiplication of two images is performed elementwise. We solve this linear system using the conjugate gradient method.

In the FM-step, we solve

$$\min_{F,M} \frac{1}{2}\|H * F + (1 - H * M)B - I\|_2^2$$

$$+\frac{\lambda}{2}\|F - M\hat{F}\|_2^2 + \alpha_F\|\nabla F\|_1, \tag{21}$$

s.t. $0 \leq F \leq M \leq 1$. We introduce two substitutions, $z_1 := \nabla F$ and $z_2 := \begin{bmatrix} F \\ M \end{bmatrix}$, to isolate minimization of the total variation term and the convex constraint, respectively. Then, the optimization of the resulting constrained problem consists of the following iteration steps performed in a loop:

$$z_1 := \arg\min_{z_1} \left(\alpha_F\|z_1\|_1 + \frac{\rho_1}{2}\|\nabla F - z_1 + u_1\|_2^2\right), \tag{22}$$

$$z_2 := \arg\min_{z_2} \left(\frac{\rho_2}{2}\left\|\begin{bmatrix} F \\ M \end{bmatrix} - z_2 + u_2\right\|_2^2\right) \quad \text{s.t. } z_2 \in C, \tag{23}$$

$$\begin{aligned}\begin{bmatrix} F \\ M \end{bmatrix} := \arg\min_{F,M} \Bigg(&\frac{1}{2}\|H * F + (1 - H * M)B - I\|_2^2 \left\|\begin{bmatrix} F \\ M \end{bmatrix}\right\|_2^2 \\ &+ \frac{\lambda}{2}\|F - M\hat{F}\|_2^2 + \frac{\rho_1}{2}\|\nabla F - z_1 + u_1\|_2^2 \\ &+ \frac{\rho_2}{2}\left\|\begin{bmatrix} F \\ M \end{bmatrix} - z_2 + u_2\right\|_2^2\Bigg),\end{aligned} \tag{24}$$

$$u_1 := u_1 + \nabla F - z_1, \tag{25}$$

$$u_2 := u_2 + \begin{bmatrix} F \\ M \end{bmatrix} - z_2, \tag{26}$$

where, similarly to the H-step, u_1 and u_2 are Lagrange multipliers of the constraints corresponding to the substitutions (both initialized by zero), and ρ_1 and ρ_2 are fixed user-defined weights of the quadratic augmentation terms of the Lagrangian. The set C constraints each "pixel" of the combined variable z_2 and is defined as $C = \{[f_1, f_2, f_3, m]^T \in \mathbb{R}^4; 0 \leq f_i \leq m \leq 1\}$. Note that $C \subset \mathbb{R}^4$, since each pixel in F has three (RGB) channels, and M is a single-channel mask. Therefore, the constraint $z_2 \in C$ must be interpreted pixelwise.

The update step (22) is analogous to (16) and leads to element-wise soft-thresholding of the image derivatives,

$$z_1 = \text{sign}(\nabla F + u_1)\max\left(|\nabla F + u_1| - \frac{\alpha_f}{\rho_1}, 0\right). \tag{27}$$

The problem (23) amounts to projection of the minimizer of the quadratic term to C,

$$z_2 = \text{proj}_C\left(\begin{bmatrix} F \\ M \end{bmatrix} + u_2\right). \tag{28}$$

This projection can be implemented directly but we employ a different yet equally fast approach. The set C is an intersection of 4 convex sets, one axis-aligned orthant and three half-spaces. Each of these sets is easy to project to. Therefore, we use the alternating projection method by (Boyle and Dykstra 1986) (best see (Tibshirani 2017)), which converges in just a few iterations. Lastly, by differentiating w.r.t. F and M, the update step (24) leads to a linear system for (F, M),

$$\begin{aligned}&\begin{bmatrix} \mathbf{H}^*\mathbf{H} + \rho_1\nabla^*\nabla + \lambda + \rho_2 & -\mathbf{H}^*B - \lambda\hat{F} \\ -\mathbf{H}^*B - \lambda\hat{F} & \mathbf{H}^*B^2\mathbf{H} + \lambda\hat{F}^2 + \rho_2 \end{bmatrix}\begin{bmatrix} F \\ M \end{bmatrix} \\ &= [\mathbf{H}, -B\mathbf{H}]^*(I - B) + \rho_1\nabla^*(z_1 - u_1) + \rho_2(z_2 - u_2),\end{aligned} \tag{29}$$

where $\mathbf{H}$ denotes operator performing convolution with H. As in the H-step, we solve this system using the conjugate gradient method. Values of the $\rho_{(\cdot)}$ parameters were chosen experimentally. In our experience, too large values slow down the convergence, while too low values cause oscillations.

References

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122. https://doi.org/10.1561/2200000016.

Boyle, J. P., & Dykstra, R. L. (1986). A method for finding projections onto the intersection of convex sets in hilbert spaces. *Advances in Order Restricted Statistical Inference* (pp. 28–47). New York, New York, NY: Springer.

Braso, G., & Leal-Taixe, L. (2020) Learning a neural solver for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M. (2014) Accurate scale estimation for robust visual tracking. In: Proceedings of the British Machine Vision Conference, BMVA Press, https://doi.org/10.5244/C.28.65

Hornakova, A., Henschel, R., Rosenhahn, B., Swoboda, P. (2020) Lifted disjoint paths with application in multiple object tracking. In: The 37th International Conference on Machine Learning (ICML)

Hrabalík, A. (2017). *Implementing and applying fast moving object detection on mobile devices, master's thesis*. Faculty of Electrical Engineering: Czech Technical University in Prague.

Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(7), 1409–1422. https://doi.org/10.1109/TPAMI.2011.239.

Kotera, J., Šroubek, F. (2018) Motion estimation and deblurring of fast moving objects. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp 2860–2864, https://doi.org/10.1109/ICIP.2018.8451661

Kotera, J., Rozumnyi, D., Šroubek, F., Matas, J. (2019) Intra-frame object tracking by deblatting. In: The IEEE International Conference on Computer Vision (ICCV) Workshops

Kristan M, Leonardis A, Matas J, Felsberg M, Pflugfelder R, Čehovin L, Vojíř T, Häger G, Lukežič A, Fernández G, et al. (2016) 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, Springer International Publishing, Cham, pp 777–823, https://doi.org/10.1007/978-3-319-48881-3

Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Zajc, L. Č, et al. (2019). The sixth visual object tracking VOT2018 challenge results. In L. Leal-Taixé & S. Roth (Eds.), *ECCV 2018 Workshops* (pp. 3–53). Cham: Springer International Publishing.

Kroeger, T., Dragon, R., & Van Gool, L. (2014). Multi-view tracking of multiple targets with dynamic cameras. In X. Jiang, J. Hornegger, & R. Koch (Eds.), *Pattern Recognition* (pp. 653–665). Cham: Springer.

Lukežič, A., Vojíř, T., Zajc, L. C., Matas, J., Kristan, M. (2017) Discriminative correlation filter with channel and spatial reliability. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4847–4856, https://doi.org/10.1109/CVPR.2017.515

Lukežič, A., Zajc, L. Č., Vojíř, T., Matas, J., & Kristan, M. (2019). FuCoLoT - a fully-correlational long-term tracker. In C. V. Jawahar, H. Li, G. Mori, & K. Schindler (Eds.), *Computer vision - ACCV 2018* (pp. 595–611). Cham: Springer.

Ma, B., Huang, L., Shen, J., Shao, L., Yang, M., & Porikli, F. (2016). Visual tracking under motion blur. *IEEE Transactions on Image Processing*, *25*(12), 5867–5876. https://doi.org/10.1109/TIP.2016.2615812.

Moudgil, A., Gandhi., V. (2017) Long-term visual object tracking benchmark. arXiv preprint arXiv:1712.01358

Mueller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for uav tracking. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision - ECCV 2016* (pp. 445–461). Cham: Springer.

Ristani, E., Tomasi., C. (2018) Features for multi-target multi-camera tracking and re-identification. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6036–6046, https://doi.org/10.1109/CVPR.2018.00632

Rozumnyi, D. (2019). *All-speed long-term tracker exploiting blur, master's thesis*. Faculty of Electrical Engineering: Czech Technical University in Prague.

Rozumnyi, D., Kotera, J, Šroubek, F., Novotný, L., Matas, J. (2017) The world of fast moving objects. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4838–4846, https://doi.org/10.1109/CVPR.2017.514

Rozumnyi, D., Kotera, J., Šroubek, F., & Matas, J. (2019). Non-causal tracking by deblatting. In G. A. Fink, S. Frintrop, & X. Jiang (Eds.), *Pattern Recognition* (pp. 122–135). Cham: Springer.

Rozumnyi, D., Kotera, J., Šroubek, F., Matas, J. (2020) Sub-frame appearance and 6d pose estimation of fast moving objects. In: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Runia, TFH, Gavrilyuk, K, Snoek, C. G. M., Smeulders, A. W. M. (2020) Cloth in the wind: A case study of physical measurement through simulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Seibold, C., Hilsmann, A., & Eisert, P. (2017). Model-based motion blur estimation for the improvement of motion tracking. *Computer Vision and Image Understanding*, *160*, 45–56. https://doi.org/10.1016/j.cviu.2017.03.005.

Tang, M., Yu, B., Zhang, F., Wang, J. (2018) High-speed tracking with multi-kernel correlation filters. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4874–4883, https://doi.org/10.1109/CVPR.2018.00512

Tao, R., Gavves, E., Smeulders, A. W. (2017) Tracking for half an hour. arXiv preprint arXiv:1711.10217

Tibshirani, R. J. (2017). Dykstra's algorithm, ADMM, and coordinate descent: Connections, insights, and extensions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 517–528). USA: Curran Associates.

Vojíř, T., Noskova, J., & Matas, J. (2013). *Robust scale-adaptive mean-shift for tracking*. Berlin: Springer. https://doi.org/10.1007/978-3-642-38886-6.

Wu, Y., Ling, H., Yu, J., Li, F., Mei, X., Cheng, E. (2011) Blurred target tracking by blur-driven tracker. In: 2011 International Conference on Computer Vision, pp 1100–1107, https://doi.org/10.1109/ICCV.2011.6126357

Wu, Y., Lim, J., Yang, M. (2013) Online object tracking: A benchmark. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, pp 2411–2418, https://doi.org/10.1109/CVPR.2013.312