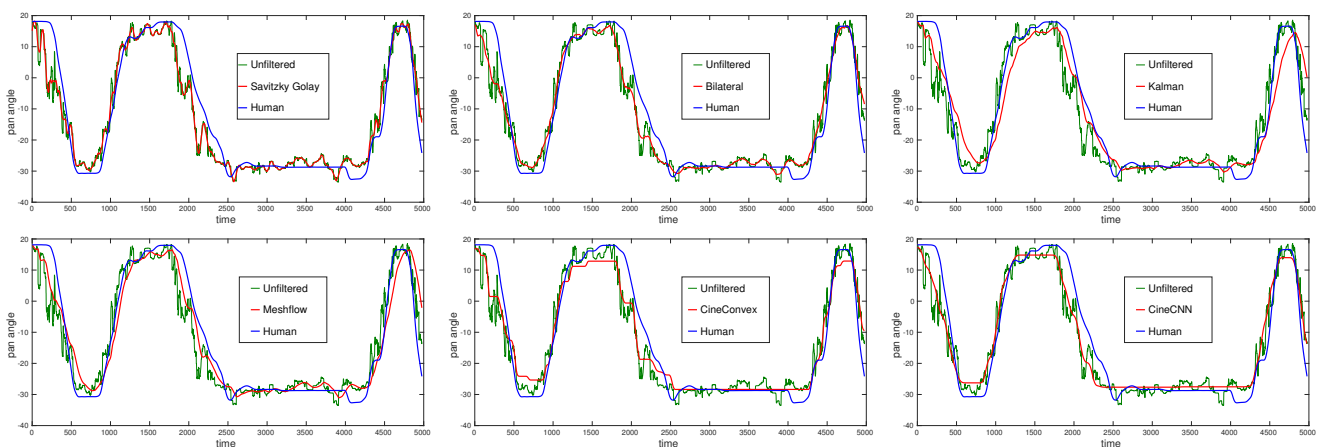


# CineFilter: Unsupervised Filtering for Real Time Autonomous Camera Systems

Sudheer Achary<sup>†</sup>, K L Bhanu Moorthy<sup>†</sup>, Ashar Javed<sup>‡</sup>, Nikita Shravan<sup>†</sup>,  
Vineet Gandhi<sup>†</sup>, Anoop Namboodiri<sup>†</sup>

<sup>†</sup> CVIT, KCIS, IIIT Hyderabad, <sup>‡</sup> Carnegie Mellon University



**Figure 1:** The pan angle of a human cameraman compared to those generated by filtering per frame pan angle predictions using different algorithms over a basketball video sequence. The plots show the filtered outputs corresponding to several baseline algorithms and our proposed approach. The proposed CineFilters mimic professional cameraman like behaviour and results in piece-wise static, linear and parabolic segments. Other filters fail to give perfect static segments, have sharp corners and sudden direction changes. Although they smooth the trajectories well, they are not appropriate to be used for applications in autonomous camera systems.

## Abstract

Autonomous camera systems are often subjected to an optimization operation to smoothen and stabilize the rough trajectory estimates. Most common filtering techniques do reduce the irregularities in data; however, they fail to mimic the behavior of a human cameraman. Global filtering methods modeling human camera operators have been successful; however, they are limited to offline settings. In this paper, we propose two online filtering methods called Cinefilters, which produce smooth camera trajectories that are motivated by cinematographic principles. The first filter (CineConvex) uses a sliding window-based convex optimization formulation, and the second (CineCNN) is a CNN based encoder-decoder model. We evaluate the proposed filters in two different settings, namely a basketball dataset and a stage performance dataset. Our models outperform previous methods and baselines on quantitative metrics. The CineConvex and CineCNN filters operate at about 250fps and 1000fps, respectively, with a minor latency (half a second), making them apt for a variety of real-time applications.

## 1. Introduction

Autonomous camera systems that track a person, object, or action of interest often employ a filtering operation on top of raw per frame estimations (e.g., a virtual camera following an instructor by cropping a window from a static camera every frame). Trajectory optimization is also useful in video stabilization methods, where

the video is re-rendered as it would have followed the stabilized trajectory. The main goal of trajectory optimization models is to transform the original trajectory to one that mimics the behavior of a skilled cameraman, as close as possible. The cinematographic literature [TB09] suggests that a good pan/tilt shot should be composed of three components: a static period of the camera at the beginning,

a smooth camera movement, which “leads” to the movement of the subject and a static period of the camera at the end. The prior art has shown [GKE11, GRG14, TWLT18] that such a behavior can be modeled as an optimization problem leading to trajectories with piece-wise constant, linear, and parabolic segments.

However, most of these methods [GKE11, GRG14, TWLT18] are offline and require the entire trajectory during the stabilization process. Such offline optimization methods cannot be adopted for real-time applications such as live broadcasts (e.g., sports, lectures, live performances). Chen [CLC\*16] made an attempt to tackle this problem in real-time by learning a regression model using a specific ground truth signal generated by a human operator for a basketball game dataset. The learned model, however, is inherently tied to the behavior specific to the particular basketball setting and its ground truth. Such a supervised approach is not applicable to arbitrary trajectory optimization.

Real-time prediction of smooth camera trajectories is an ill-posed problem since, at any given instance, the predicted trajectory can be smoothed in multiple ways depending on the future input sequences. Thus, using direct optimization on trajectories like previous offline works in a piece-wise manner (sliding window) for the online prediction can make the system very brittle. In this paper, we study the effect of allowing little peek into the future and applying causal constraints on pre-stabilized trajectories. We show that the proposed sliding window formulation of *CineConvex* filter can closely mimic an offline global optimization with a latency of just half a second. We further investigate parametric models trained with data that can learn the multimodal distribution of a smooth camera trajectory in an online setting. We hypothesize that predictive models can reduce the dependence on the latency, especially in high noise scenarios. Furthermore, the optimization can be done at much faster rates.

We propose *CineCNN*, a novel CNN based filtering approach that can learn from patterns existing across multiple samples in the dataset. A key issue when training such models is the availability of large labeled datasets. However, our training is unsupervised and uses a set of generic objective functions on top of total variation denoising. Thus, we do not require any ground truth labels for training, and *CineCNN* is independent of the idiosyncrasies of human labels from specific settings. Moreover, our objective functions also encourage cinematographically motivated behaviors like the prediction of true static segments without any residual motions and avoidance of sudden jerks, which is something missing from the past works. A motivating illustration is shown in Figure 1, where we compare our approach with filtering mechanisms prevalent in previous works. In summary, we make the following contributions:

1. We propose *CineConvex* formulation to adapt offline trajectory optimization [GKE11, GRG14, TWLT18] into an online one. Filtering is posed as a convex optimization in sliding window fashion with minor latency and constraints on the optimized trajectory from the previous window.
2. We propose *CineCNN* for learning a camera trajectory smoothing function. A total variation denoised input is sent to the CNN to convert into trajectories which mimic cameraman behaviour. The predictive model offers various advantages, especially in terms of computation load and robustness to noise.
3. Our models can predict smooth trajectories in real-time (1000 FPS for the *CineCNN* and 250 FPS for *CineConvex* on an Intel Xeon CPU). *CineCNN* has a low model complexity (400KB in size), which enables easy deployment in live settings.
4. Both our approaches work in an unsupervised way and do not need any expensive ground truth labels as compared to previous approaches [CLC\*16]. This makes our model-training independent of the setting in which the label was obtained, thus making it easier to extend to new settings.
5. Our quantitative evaluations show that our approach can mimic professional cameraman behavior and outperforms the baselines and the prior art.

## 2. Related Work

Research in autonomous camera systems dates back more than two decades. One of the earliest systems was proposed by Pinhanez and Bobick [Pin95], which aimed at automated camera framing in cooking shows. Their system was based on two types of cameras, a spotting camera that watched the entire area of interest and a robotic tracking camera that followed the verbal instructions from a director to frame the desired targets automatically. The Autoauditorium [Bia04] system extended this idea for lecture videos (a single presenter in front of a screen). The robotic camera in the Autoauditorium system crudely followed the presenter, whose position was estimated each frame using background subtraction on the spotting camera.

A series of works then followed, and we review the computational models for camera movement, proposed in these approaches. Yokoi *et al.* [YF05] present a method for automated editing of lecture videos. Their work replaces the robotic camera by a virtual camera, i.e., a cropping window moving inside a high-resolution video. They use temporal frame differencing to detect the Region of Interest (RoI) around the presenter and use bilateral filtering to remove the jittery motion introduced by per-frame RoI estimations. A similar digital tracking approach was employed in Microsoft’s icam2 system [ZRCH08]. The contemporary work by [SFKM05] also tracks cropped RoI from a panoramic video; however, it employs a Kalman filter for removing the jitter. Such filtering approaches successfully reduce jitter; however, they are not cinematically inspired, and they lead to unmotivated camera movements and fail to keep the camera static. Heuristics have been applied to tackle some of these challenges. For instance, [SFKM05] keeps RoI unchanged if the Kalman filter predictions of new positions are within a specified distance of the registered position, and the new estimated velocity is below a threshold. However, such heuristics are not applicable in generalized scenarios.

The virtual camerawork has been investigated for the autonomous broadcast of sports. Unlike the classroom environment, where framing a single person is relatively simple, the sports videos have multiple players and fast-moving objects. Diago *et al.* [DO04] propose an offline system that uses the audience face direction to build an automatic pan control system (pan angle of the broadcasting camera). Ariki *et al.* [AKK06] proposed a heuristic-based editing strategy. However, these approaches focus on the per-frame pan angle estimation and skip details on the camera motion models or smoothing algorithms. Chen *et al.* [CDV10] uses Gaus-

sian Markov Random Fields (MRF) to obtain smooth virtual camera movements. They induce smoothness by inducing inter-frame smoothness priors. Recent work [RKGS18] has shown that while inter-frame smoothness priors do induce smoothness, it fails to give aesthetically pleasing camera behavior. They augment it with an additional post-processing optimization to render professional-looking camera trajectories.

The virtual videography [HWG07] system was one of the earliest works to model camera movement inspired by filmmaking literature [C\*95, TB09]. They define that a good tracking shot should consist only of smooth motions in a single direction that accelerate and decelerate gradually to avoid jarring the viewer while maintaining the correct apparent motion of the subject. They define a customized parametric function for the motion model and solve for parameters for each moving shot individually. Liu *et al.* [LG06] demonstrate the applicability of such a motion model in the application of automatic Pan and Scan. Jain *et al.* [JSSH15] build upon their work and model the camera trajectories as parametric piece-wise spline curves. However, such parametric motion models have limited applicability due to assumptions on the content (a single pan in each shot). Grundmann *et al.* [GKE11] show that ‘professional cameraman like’ trajectories consist of piece-wise static, linear, and parabolic segments. They show the applicability of such a motion model for video stabilization. Their approach seems motivated by cinematic ideas, generalizes well, and is deployed on large scale systems like Youtube. A similar formulation has been employed in applications of virtual cinematography in panoramic [GRG14] and 360 videos [TWLT18]. However, the optimization posed in these works [GKE11, GRG14, TWLT18] is offline (and non-causal) and in our work, we extend it to online and causal settings.

Carr *et al.* [CMM13] proposed a hybrid camera to aesthetic video generation in the context of basketball games. They combined robotic cameras to coarsely track the game and augmented it with virtual camera simulations to get smoother camera movements. They show that the loss of image resolution can be minimized by using a hybrid system. They investigate a causal moving average filtering and a non-causal  $l_1$  trend filtering [KKBG09] to filter the crude trajectories obtained by following the centroid of player detections. They extend their work by learning per frame pan angle predictors based on the player positions in the game and further smooth it using Savitzky Golay filter [SG64]. In [CLC\*16], they merge the camera position prediction and smoothing into a unified framework. These works [KKBG09, CLC\*16] rely on a supervised signal generated by a synchronized human camera operator, which is difficult to obtain and also makes them domain-specific. In contrast, the proposed filter in our work is unsupervised.

Our work is also related to the work of Liu *et al.* [LTY\*16], which proposed a framework for online video stabilization of casually captured videos. Their method performs video stabilization by optimizing vertex level motion trajectories. It runs with a single frame latency and runs optimization locally at every frame using a few previous frames. However, such exact optimization only applies to minimal cost functions like mean squared error + L2 norm smoothness (as in [LTY\*16]). Such minimal functions fail to give the desired cinematic behavior. Moreover, running the optimization

on every frame is computationally expensive when a closed-form solution does not exist. We tackle this problem by controlling strides in a sliding window optimization. The proposed *CineConvex* filter: (a) uses a cinematically motivated cost function, and (b) provides more structure due to historical constraints and peek into the future frames. Moreover, beyond a standalone optimization for each window, the proposed *CineCNN* filter can learn priors/structures from previous data as well as temporal dependencies within a sequence.

Our work, limits to the problem of one-dimensional trajectory filtering/stabilization. The work is agnostic to the application and is directly applicable in variety of frameworks [YF05, SFKM05, CDV10, HWG07, CMM13, CLC\*16, LTY\*16, GRG14, ZRCH08]. We compare our approach against trajectory filtering approaches employed in these frameworks.

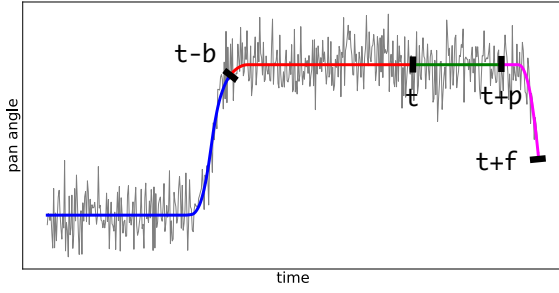
### 3. The CineFilter Model

In this section, we describe our models and their unsupervised optimization procedure. Before going into the problem formulation, we list the desiderata that guide our modeling choices.

1. The smoothing filter should be online and run in real-time.
2. It should not require labeled supervisory data (e.g. ground truth trajectories from human experts).
3. Unmotivated camera movements should be avoided. For instance, when the subject in the frame is stationary, the camera behavior should be static to ensure a pleasant viewing experience.
4. Trajectory smoothing should be robust to outliers in the case of sudden changes in camera trends (should avoid abrupt jerks).
5. It should avoid the accumulation of drift, which is a common problem observed in real-time systems.

With the above noted, we first define our problem as that of predicting a smooth camera trajectory given a stream of incoming noisy trajectory positions. Let  $X_t = \{x_0, x_1, \dots, x_t\}$  be the noisy input sequence that has arrived until frame  $t$  and  $Y_{t-1} = \{y_0, y_1, \dots, y_{t-1}\}$  be the smoothed output sequence predicted until the previous frame  $t-1$ . We wish to learn a nonlinear causal filter specified by a parametric model  $y_t = f(X_t, Y_{t-1})$ , which predicts the smoothed output for the current frame  $t$ . At any timestep, predicting the next smooth frame is ideally a function of both the past and future trajectory directions, which is feasible to model using offline approaches. In the online scenario that we wish to operate in, learning the above mapping is an ill-posed problem as multiple solutions can exist depending on where the future trajectory goes. Therefore it makes sense to accommodate a small number of future frames into the model to constrain the direction of the trajectory.

In this work, we propose two different methods to solve the above problem. The first is an online filter that performs convex optimization on a combination of objectives. Since the final objective is convex, we can run a per-sample solver to obtain a global minimum at each timestep. However, these methods have a dependency on the solver at deployment time and are computationally intensive if run at each frame in an online fashion. These methods also do not learn a data-based model of trajectory behavior; hence, they might not be fit for cases with high variance in data statistics. The second model we propose is a learning-based solution combining total



**Figure 2:** The sliding window configuration for our CineFilter models. At timestep  $t$ , the model stores  $b$  timesteps of the past buffer, has access to  $f$  timesteps of the future and shifts with a stride of  $p$  after each prediction.

variation denoising with a 1D convolution neural network (CNN). The CNN gives advantage over convex optimization approach as it can learn trajectory patterns from data. Moreover, at inference, a forward pass through this model is faster and computationally cheaper than a convex solver.

The ideal camera trajectory should be composed of three types of segments, namely static segments, constant velocity segments, and segments with constant acceleration, all transitioning in a smooth manner. As opposed to previous works that use ground truth data from human operators or create large datasets for deep learning based methods, we build our model through an unsupervised multi-objective loss function that enforces such behavior without the need to collect labeled data.

### 3.1. CineConvex filter

We first describe the convex optimization-based solution for on-line smoothing. The model works in a sliding window manner. The sliding window configuration is shown in Figure 2. It consists of three fragments. The first fragment is called the present window of size  $p$  from timestep  $t$  to  $t + p$ . This fragment gets updated in the final predictions after optimizing the current sliding window. It is also the step-size by which we shift the window after each optimization. The second fragment is called the buffer window, which spans timestep  $t - b$  to  $t$ . It is the historical trajectory information we use in the optimization. The third fragment is called the future window which holds timestep  $t$  to  $t + f$  with  $t + f > t + p > t$  and includes the present window. It provides future context for the optimization. We optimize the trajectories from timestep  $t - b$  to  $t + f$  and shift all windows by  $p$  after each optimization.

The optimization procedure for each sliding window includes: (a) a term to enforce the predicted trajectory to be close to the original trajectory in some distance metric and (b) L1-norm of the first-order, second-order and third-order derivatives over the optimized trajectory to induce piece-wise static, linear and parabolic behavior. L1-norm has the property to avoid residual motions (e.g., when the path is meant to be static, it leads to truly static outputs) and avoids the superposition between the constant, linear, and parabolic segments. The final objective  $J^m$  with respect to timestep  $t$ , where  $m$  indexes the  $m^{\text{th}}$  optimization procedure, is given by:

$$J^m(t) = \lambda_0 D_0^m(t) + \lambda_1 D_1^m(t) + \lambda_2 D_2^m(t) + \lambda_3 D_3^m(t) \quad (1)$$

where,

$$D_0^m(t) = \sum_{i=t-b}^{t+f} (x^m(i) - y^m(i))^2 \quad (2)$$

$$D_1^m(t) = \sum_{i=t-b}^{t+f} |y^m(i+1) - y^m(i)| \quad (3)$$

$$D_2^m(t) = \sum_{i=t-b}^{t+f} |y^m(i+2) - 2y^m(i+1) + y^m(i)| \quad (4)$$

$$D_3^m(t) = \sum_{i=t-b}^{t+f} |y^m(i+3) - 3y^m(i+2) + 3y^m(i+1) - y^m(i)| \quad (5)$$

The  $\lambda$ s are hyperparameters found using cross-validation and  $p, b$  and  $f$  are values we fix heuristically.

Note that this CineConvex model has a latency of  $f$  timesteps, stores an extra  $b$  timesteps from the past, and is run after every  $p$  timesteps. Reducing  $f$  can reduce the latency, but will lead to less smooth results since multiple future trajectory directions are possible. Decreasing  $p$  will lead to frequent trajectory updates but will affect the speed of the filtering operation due to the larger number of optimizations required. Finally, increasing  $b$  will provide further historical context; however, that will increase the time required for each individual optimization. Hence the window-related parameters offer a way to balance the speed and accuracy trade-off.

Another problem encountered during this optimization is to maintain continuity over optimizations on consecutive sliding windows. To mitigate this, we place a hard equality constraint on the past trajectories being currently predicted ( $y^m(t-b)$  to  $y^m(t)$ ) and the past trajectories already predicted during the previous optimization ( $y^{m-1}(t-b)$  to  $y^{m-1}(t)$ ). Finally, the objective is changed to the following:

$$J^m(t) = \lambda_0 D_0^m(t) + \lambda_1 D_1^m(t) + \lambda_2 D_2^m(t) + \lambda_3 D_3^m(t) \quad (6)$$

$$\text{s.t., } y^m(k) = y^{m-1}(k) \quad \forall k \in \{t-b, \dots, t\}$$

We use the state of the art Gurobi solver [Sol19] for minimizing the objective function (the fastest solver in the MIPLIB 2017 Benchmark [mip18]).

### 3.2. CineCNN filter

We can pose the problem of estimating the current smooth trajectory position as a sequence modeling problem. However, popular sequence prediction models like HMMs, RNNs, and LSTMs operate sequentially and have a high memory footprint. Moreover, we found in our experiments that 1D encoder-decoder CNN based architecture better learns the local structure over the more complex recurrent counterparts and also provides improved performance. To promote perfectly static trajectories, wherever possible, we first filter the input signal using a 1D Total Variation (TV) denoising algorithm [Con13] and then pass it to the CNN. The TV output leads into staircase artefacts whenever the camera movement occurs. The



CNN smooths out the staircase artefacts and the results in trajectories that mimic professional camera behavior (having smooth transitions between perfectly static segments). We use an extremely fast direct (non iterative) TV denoising algorithm [Con13]. In the absence of smooth trajectory labels, we use a similar unsupervised loss function for training the CNN as the convex optimization, without the first-order term. The loss function penalizes the squared distance between the original trajectory and the predictions along with the second, and third-order derivative terms.

The final loss function for the model is as follows:

$$L = \lambda_0 D_0 + \lambda_2 D_2 + \lambda_3 D_3 \quad (7)$$

The first three terms  $D_1, D_2, D_3$  are the same as before, but from timestep 1 to  $n$ .

While training, the input sequence from all trajectories are divided into overlapping subsets of  $n = 512$  frames. The inference happens on a sliding window of 32 frames, which is possible since the network is fully convolutional. During inference, when only the first frame has arrived, we left-pad the input with repeated values of the first trajectory position and use this as the input. Since there is no optimization at test time, but only a filter-forward pass through the model, we can make a prediction at each time step i.e., with  $p = 1$  in figure 2. Also, note that there is no explicit constraint that enforces trajectory continuity across predictions like the one needed in the convex optimization formulation. The model has a structural inductive bias (1D convolution) that merges information from local trajectory positions, thus aiding continuity. The data itself, which is fed as a sequence, also provides an additional implicit constraint on the smoothness of the predictions.

### 3.3. Implementation details

For *CineCNN*, we use a 5-layered 1D encoder-decoder based CNN model that has skip connections similar to U-Net architecture with a kernel size of 3 and with 16, 32, 32, 16 and 1 filters respectively. Each CNN layer has a relu based activation on the outputs. The videos from TLP dataset [MG17] and Basketball dataset [CLC\*16] (described in Section 4.1) are used to train the model. We sub-sample from the full trajectory of the video at random frames and create a set of input trajectories each of a fixed length of 512 and use these as single instances of training data for the model.

We create 98k such instances through sub-sampling. The model is trained for 20 epochs on the basketball and the TLP datasets, each with a batch size of 16. The adam optimizer is used during training. We follow a learning rate schedule that decays the learning rate by a factor of 0.1 if the validation loss plateaus for more than 4 epochs. The weights associated with each loss term are obtained empirically and  $(\lambda_0, \lambda_1, \lambda_2, \lambda_3)$  for stage performance dataset are (1.0, 1000, 50, 2000) and for basketball dataset are (1.0, 2000, 100, 3000). The values for  $p, f, b$  are 8, 16, 64, respectively, for the *CineConvex* model. For the *CineCNN*, we set  $p = 1, f = 16$  and  $b = 16$ .

## 4. Experiments

We evaluate our approach for the automated broadcast of basketball matches [CLC\*16] and staged performances [GRG14]. We com-

pute the results of *CineConvex* and *CineCNN* with four algorithms commonly used for online filtering in previous works. We also perform ablation studies to demonstrate the effect of the parameters  $p, f, b$  on model performance. We now describe the datasets, baselines, and evaluation strategy in detail.

### 4.1. Datasets

**Basketball dataset:** We use the Basketball dataset proposed by Chen et al. [CLC\*16]. This dataset consists of a video recording of a high school basketball match taken from two different cameras. One wide-angle camera is installed near the ceiling and looks at the entire basketball area. The feed from the wide-angle camera is used to detect players and compute features summarizing the current state of the scene. The second broadcast camera is placed at the ground level and is manually operated by a human expert. The evaluation task is to predict the pan angle for a robotic camera, given the current state of the match observed by the wide camera. The pan angle of the human-operated camera is considered as the ground truth (calculated by computing its homography with respect to the wide-angle camera). The dataset consists of 50 segments of 40 seconds each (overall 32 minutes of in-play data), out of which 48 segments are used for training, and 2 are used for validation and testing. Similar to [CLC\*16], we train a Random Forest regressor to obtain per frame pan angle predictions. For learning the Random Forest regressor, the game features computed using the wide-angle camera are used as input and the human operator pan angle as ground truth labels. The per-frame predictions give an extremely noisy output, which are then subjected to a filtering operation. We perform comparisons on the output of CineFilter models and the baselines with the human operator trajectory as the ground truth.

**Stage Performance dataset:** We build a Stage Performance dataset that comprises of two wide-angle recordings of staged performances each of 12 and 10 minutes, respectively (a dance and a theatre performance). The videos are selected from the Track Long and Prosper (TLP) Dataset [MG17]. The original recordings were done using a static wide-angle camera covering the entire action in the scene. The noisy object trajectory sequences for these recordings are obtained using the MDNet tracker [NH16]. A complete 12 minute sequence is used for training, and the 10 minute sequence is used for testing. The filters are evaluated on the task of virtual camerawork, following an actor on the stage based on the output of the tracker. Since there is no ground truth available for these videos, we use the offline optimizer from [GRG14] as the ground truth trajectory.

### 4.2. Baselines

**Savitzky Golay:** SG filter performs data smoothing using least squares to fit a polynomial of a chosen degree within a window of consecutive data points around every point. It takes the central value in the window as the new smoothed data point and this step is performed at each point on the time series. SG filter is non-causal and in our case we choose a window size of 51, giving a latency of 25 frames. The degree of the polynomial is set to 3.

**Kalman Filter:** Kalman Filter is a recursive Bayesian estimation method that can be used for updating the current smooth

camera state at every time step using previous predictions and the current observation. We set the parameters of the filter similar to [CLC\*16].

**Bilateral Filter:** Bilateral filter is a non linear, adaptive, edge preserving and noise reducing smoothing filter. It uses a weighted-average approach where the weights at any given data point is computed using two Gaussians, one Gaussian on the difference of spatial distance and the other on the difference between the magnitudes of the given point with every other point in the surrounding window. It is also non-causal and we use a window size of 64, giving the latency of 32 frames.

**MeshFlow:** MeshFlow [LTY\*16] runs with a single frame latency and runs optimization locally at every frame using a few previous frames. The optimization functions consists of two different terms. The first term is the MSE over the predicted and original values of the 1D signal. The second term is a MSE over the first order differential of the predicted signal. Window size of 20 values is considered for the experiments.

### 4.3. Evaluation Metric

For quantitative experiments on the Basketball dataset, we use the data from the human operator as ground truth and compute two different metrics, one measuring the closeness to the original signal and other measuring the movement profile. The first metric (precision metric) is the mean squared error between the filtered trajectory predictions and the ground truth trajectories (pan angle corresponding to the expert human operator). Assuming that the human operator selects the pan angle that best showcases the activity happening in the game, this term penalizes trajectories that deviate from the angles at which important activities are happening. The second metric (smoothness metric) is the absolute difference in the slopes between the predicted and the ground truth trajectory. It measures the ability to mimic human cameraman-like behavior. It penalizes any movement of the predicted trajectory when the human operator is static and also penalizes the predicted trajectory if it moves in a different direction or at a different speed than the ground truth. We define the two terms as Precision Loss and Smoothness Loss:

$$precision = \sum_t (y(t) - \hat{y}(t))^2 \quad (8)$$

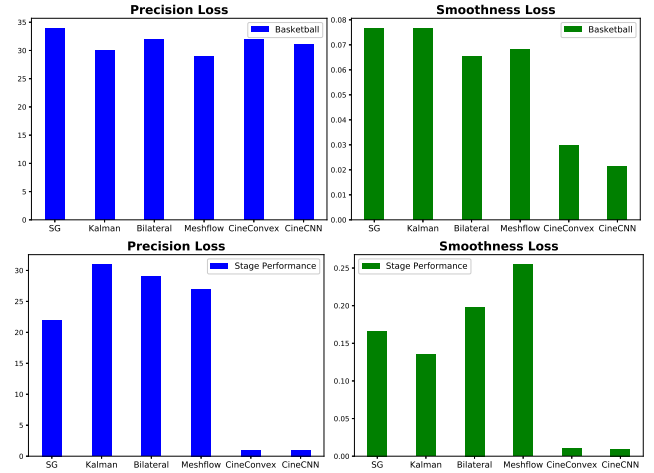
$$smoothness = \sum_t \left| \frac{dy(t)}{dt} - \frac{d\hat{y}(t)}{dt} \right| \quad (9)$$

$y(t)$  is the prediction,  $\hat{y}(t)$  is the ground truth from the human operator, lower the precision & smoothness loss, the better the predictions.

## 5. Results

### 5.1. Quantitative Evaluation

We compare *CineConvex* and *CineCNN* against the baselines on the precision loss (Equation 8) and smoothness loss (Equation 9) metrics. The results are summarized in Figure 3.



**Figure 3:** Precision & Smoothness loss for our approach and the baselines on the Basketball dataset (top row) and the Stage Performance dataset (bottom row).

For the Basketball dataset, which has the noisier motion of the two datasets, we see that the proposed models are competent on the precision metric to other approaches; however, they bring significant improvements over the smoothness metric. *CineCNN* gives more than three times improvement over the other baselines. We also observe that *CineCNN* gives better performance in high noise situations over the *CineConvex* filter.

For the Stage Performance dataset, which has relatively noise-free trajectories, *CineConvex* and *CineCNN* give similar performances. The proposed methods notably outperform the baselines over both precision and smoothness metric. The use of a ground truth generated using similar loss terms (offline) might give an added advantage to the proposed models. However, offline optimization [GKE11, GRG14] is shown to extremely effective and considered to closely mimic human cameraman behaviour.

## 6. Conclusion

In this paper, we propose two unsupervised methods for the on-line filtering of noisy trajectories. *CineConvex* formulation poses filtering as a convex optimization over individual sliding windows and is solved using iterative convex solvers. *CineCNN* formulation models filtering as a prediction using a convolutional neural network. The unsupervised nature of the proposed methods, high operational frame rates, and low memory footprint make them a generic choice across a large variety of applications. We contrast the performance of the proposed methods in comparison to commonly used online filters, across two diverse applications of basketball games (fast-paced, multiple players) and theatre plays (following individual actors on a stage). Thorough quantitative experiments show that the proposed methods give superior performance over the existing filtering techniques and provide a preferable alternative for trajectory filtering in online automated camera systems.

## References

- [AKK06] ARIKI Y., KUBOTA S., KUMANO M.: Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Eighth IEEE International Symposium on Multimedia (ISM'06)* (2006), IEEE, pp. 851–860. 2
- [Bia04] BIANCHI M.: Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (2004), ACM, pp. 117–123. 2
- [C\*95] CANTINE J., ET AL.: *Shot by shot: A practical guide to filmmaking*. ERIC, 1995. 3
- [CDV10] CHEN F., DE VLEESCHOUWER C.: Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision and Image Understanding* 114, 6 (2010), 667–680. 2, 3
- [CLC\*16] CHEN J., LE H. M., CARR P., YUE Y., LITTLE J. J.: Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 4688–4696. 2, 3, 5, 6
- [CMM13] CARR P., MISTRY M., MATTHEWS I.: Hybrid robotic/virtual pan-tilt-zoom cameras for autonomous event recording. In *Proceedings of the 21st ACM international conference on Multimedia* (2013), ACM, pp. 193–202. 3
- [Con13] CONDAT L.: A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters* 20, 11 (2013), 1054–1057. 4, 5
- [DO04] DAIGO S., OZAWA S.: Automatic pan control system for broadcasting ball games based on audience's face direction. In *Proceedings of the 12th annual ACM international conference on Multimedia* (2004), ACM, pp. 444–447. 2
- [GKE11] GRUNDMANN M., KWATRA V., ESSA I.: Auto-directed video stabilization with robust 11 optimal camera paths. In *CVPR 2011* (2011), IEEE, pp. 225–232. 2, 3, 6
- [GRG14] GANDHI V., RONFARD R., GLEICHER M.: Multi-clip video editing from a single viewpoint. In *Proceedings of the 11th European Conference on Visual Media Production* (2014), ACM, p. 9. 2, 3, 5, 6
- [HWG07] HECK R., WALLICK M., GLEICHER M.: Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3, 1 (2007), 4. 3
- [JSSH15] JAIN E., SHEIKH Y., SHAMIR A., HODGINS J.: Gaze-driven video re-editing. *ACM Transactions on Graphics (TOG)* 34, 2 (2015), 21. 3
- [KKBG09] KIM S.-J., KOH K., BOYD S., GORINEVSKY D.:  $\ell_1$  trend filtering. *SIAM review* 51, 2 (2009), 339–360. 3
- [LG06] LIU F., GLEICHER M.: Video retargeting: automating pan and scan. In *Proceedings of the 14th ACM international conference on Multimedia* (2006), ACM, pp. 241–250. 3
- [LTY\*16] LIU S., TAN P., YUAN L., SUN J., ZENG B.: Meshflow: Minimum latency online video stabilization. In *European Conference on Computer Vision* (2016), Springer, pp. 800–815. 3, 6
- [MG17] MOUDGIL A., GANDHI V.: Long-term visual object tracking benchmark. *arXiv preprint arXiv:1712.01358* (2017). 5
- [mip18] MIPLIB 2017, 2018. <http://miplib.zib.de>. 4
- [NH16] NAM H., HAN B.: Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 4293–4302. 5
- [Pin95] PINHANEZ C. S.: *Intelligent studios: Using computer vision to control TV cameras*. 1995. 2
- [RKGS18] RACHAVARAPU K. K., KUMAR M., GANDHI V., SUBRAMANIAN R.: Watch to edit: Video retargeting using gaze. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 205–215. 3
- [SFKM05] SUN X., FOOTE J., KIMBER D., MANJUNATH B.: Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Transactions on Multimedia* 7, 5 (2005), 981–990. 2, 3
- [SG64] SAVITZKY A., GOLAY M. J.: Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry* 36, 8 (1964), 1627–1639. 3
- [Sol19] SOLVER G.: Gurobi optimization, 2019. 4
- [TB09] THOMPSON R., BOWEN C.: *Grammar of the Shot*. Taylor & Francis, 2009. 1, 3
- [TWLT18] TANG C., WANG O., LIU F., TAN P.: Joint stabilization and direction of 360\deg videos. *ACM Transactions on Graphics (TOG)* (2018). 2, 3
- [YF05] YOKOI T., FUJIYOSHI H.: Virtual camerawork for generating lecture video from high resolution images. In *2005 IEEE International Conference on Multimedia and Expo* (2005), IEEE, pp. 4–pp. 2, 3
- [ZRCH08] ZHANG C., RUI Y., CRAWFORD J., HE L.-W.: An automated end-to-end lecture capture and broadcasting system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 4, 1 (2008), 6. 2, 3