Textureless Point Tracking

Group Project 2023 Fall

Xianyu Mo
Sustech
Shenzhen, China
12112712@mail.sustech.edu.cn

Jianzheng Huang
Sustech
Shenzhen, China
12110224@mail.sustech.edu.cn

Kangrui Chen
Sustech
Shenzhen, China
12110524@mail.sustech.edu.cn

January 16, 2024

Abstract

Textureless point tracking is one of the prior things on the agenda of augmented reality field. In this project, we will combine state-of-art techniques including optical flow, CNN, Transformer, segmentation, anchor point, and sliding window to design a video textureless point tracking algorithm. A baseline, CoTracker, is chosen as a baseline and we have been working on improve its structure and finding preferable datasets. We will latter using variant datasets to train, tune our model and test it on different benchmarks. Hoping to triumph over long-term textureless point tracking challenge.

1 Introduction

With the rapid rise of the augmented reality (AR) field, the interaction between the real world and the virtual world has become increasingly fascinating. The core of this interaction lies in seamlessly embedding three-dimensional objects into dynamic visual scenes to achieve visual augmentation. In the project, we focus on a preliminary challenge, textureless point tracking, for accurate placement of virtual objects into visual reality.

Previous work has facilitate the development of point tracking. While many efforts focus on dynamic objects, tracking points on textureless areas has been limited explored. There are still some algorithms seem perform well in general cases. One the the state-of-art one is CoTracker[1], which is chosen as a baseline of this project. However, by evaluating it via benchmarks and manual trial cases, we still found out some drawbacks remain to be resolved.

1. To achieve the best performance, CoTracker needs to generate a grid of supporting points for correlation over the whole screen, which put on a huge load for computation as the transformer complexity is quadratic in the number of tracked points.

2. Also, the generated supporting points are not guaranteed to be useful. They will all contribute in correlation features. In some cases, the grids are distributed on blurry or confusing pixels become burdensome for tracking targets.

3. Inspired by PIPs[31], CoTracker uses fixed windows to pursue long-term tracking while limiting the size of tensors. However, it also inherits the obstacles that include drifting and losing points covered beyond window size. Sliding window smooths the results partially. Long-time occluded points will still be lost or mistracked.

So in this project, we aiming to combine mutiple methods to improve the architecture of CoTracker and train the outcome using harder datasets with textureless-oriented tuning. To be more specific, we introduce two methods to improve CoTracker's performance. First, we apply segmentation techniques to keep the model from confusing foreground and background objects. This may potentially improve performance when there exists background objects that share similar texture and figures to our tracking points. Second, we apply more flexible point selection policies. Instead of simply choosing grids and neighbor points as "global" and "local" environment, we choose the most confidence matching points as anchors. Points in textureless areas are ensured for translational and rotational invariance by computing the Euclidean distance with respect to the anchor points selected by local feature matching algorithms.

2 Related Work

2.1 Feature Matching

Feature matching plays a crucial role in video-related point tracking tasks. Many previous work prepare their models with local feature matching techniques before tracking, and their strategies on feature matching methods fall into two main categories: detector-based and detector-free.

Detector-based approaches involve feature detection, description, and matching, using handcrafted features like SIFT [16] and ORB[20]. Efforts to improve the feature matching stage include D2Net [7], R2D2 [19], and SuperGlue[21] . However, detector-based methods face limitations in challenging scenarios.

In contrast, detector-free methods directly match dense features between pixels without relying on a local feature detector. While classical methods exist[10, 17], few surpass detector-based methods. Learning-based techniques, particularly cost-volume-based[9, 29] and Transformer-based methods[30], have transformed the field. While cost-volume-based methods show promise, the limited perceptual field of CNNs is a notable drawback. Transformer-based methods have recently addressed this issue and lead the benchmark.

Some previous work introduced geometry prior into their modules. SEM[3] select high-confidence anchor points, and encode the relative position between target points and the anchor points in a scale and rotation invariant manner. Yu et al. introduced the Adaptive Spot-guided Transformer[2], featuring a spot-guided aggregation module and an adaptive scaling module, disregarding irrelevant regions and utilizing depth information to adjust grids size dynamically for refinement.

2.2 Group Tracking

In real-time scenarios, the collective tracking of points holds the potential to leverage the inherent physical structures of objects and establish more comprehensive connections between successive

frames. OmniMotion[2] aspires to generate globally consistent full-length motion trajectories for all points within an entire video, even in the presence of occlusions. CoTracker[1] concurrently tracks groups of points, introducing inter-object features by introducing neighboring feature points or global grid points. Alternatively, more sophisticated grouping strategies can be applied to enhance accuracy.

2.3 Optical Flow

Optical flow, a foundational problem in computer vision, involves estimating dense instantaneous motion. Initially addressed through solving simple differential equations related to color constancy [11, 18, 4, 5], contemporary approaches leverage deep learning. Dosovitskiy et al. introduced FlowNet[8], the pioneer end-to-end convolutional network for optical flow, later refined in FlowNet2 [13]. RAFT[23], preserving high-resolution flow, inspired subsequent works [15, 26, 27] aiming to enhance model efficiency or flow accuracy.

Transformers [42] also entered the optical flow domain [12, 22, 25]. Flowformer [12], drawing inspiration from RAFT, introduced a transformer-based approach tokenizing the 4D cost volume. GMFlow [25] replaced the update network with softmax self-attention for refinement. Perceiver IO [14] proposed a unified transformer for various tasks, including optical flow. In contrast to traditional optical flow, our focus is long-term tracking, a challenge for optical flow estimators limited to approximate temporal integration.

2.4 Long-Term Tracking

To achieve pixel-level long-term tracking, several notable approaches has been proposed. This objective is traditionally tackled with manual designed methods like divide-and-conquer[32]. While leaning-based methods are much more popular recently. MFT[6] learns to select the most reliable sequences of optical flows to perform long-range tracking. PIPs++[28] and our baseline CoTracker[1] use frame sequences as fix-sized temporal windows and perform refinement inside, leveraging the context.

3 Methods

3.1 CoTracker

We formulize the task as follows:A video $V = (I_t)_{t=1}^{T}$ is a sequence of T RGB frames $I_t \in^{3\times H\times W}$ .A track $P_t^i = (x_t^i, y_t^i) \in^2, t = t^i, ..., T, i = 1, ..., N$, for each of the N points, where $t^i \in \{1, ..., T\}$ denotes the time when the track starts. A tracker is an algorithm that takes as input the video V and starting locations and times $(P_{t^i}^i, t^i)_{i=1}^{N}, v_{t^i}^i = 1$ and then output an estimate tracks $\hat{P}_t^i = (\hat{x}_t^i, \hat{y}_t^i), \forall t \geq t^i$ and an estimate visibility flag $\hat{v}_t^i$.

3.2 Feature extraction

FPN is a neural network structure commonly used for tasks like object detection and semantic segmentation. It employs multiple layers of CNNs to extract features, connected via ResNet. This network establishes a feature pyramid to obtain multi-scale feature representations, aiding precise localization and recognition of features across different scales in object detection and segmentation

tasks. When passing down features from higher layers, it supplements lower-level semantics. RAFT, LOFTR, and Cotracker all utilize this network as a means to encode images. Our model will also use this approach to extract features for each frame in the video, denoted as $\phi(I_t)$. Simultaneously, when initializing the tracked features Q_t^i, sampling will be performed on $\phi(I_t)$ followed by updates through a neural network. Here, $t \in \{1, 2, \ldots, T\}$, $i \in \{1, 2, \ldots N\}$, T represents the video length, and N is the number of tracked points. In other words: d-dimensional appearance features extracted by a CNN: $\phi(I_t) \in \mathrm{R}^{d \times \frac{H}{k} \times \frac{W}{k}}$ and k is a downsampling factor. Scaled versions: $\phi(I_t; s) \in \mathrm{R}^{d \times \frac{H}{sk} \times \frac{W}{sk}}$ with strides $s = 1, ..., S$ are obtained by average pooling in $s \times s$ neighbors. $(k = 8, S = 4)$ The appearance of the tracks is captured by feature vectors $Q_t^i \in \mathrm{R}^d$.

3.3 Transformer formulation

Transformer network: $\Psi : G \mapsto O$, where G_t^i are tracks encoded as a grid of input tokens, and O_t^i are a corresponding grid of output tokens representing updated tracks.

3.3.1 Correlation features

In order to achieve point localization in the SEM [1] (Structure from Motion) model, the feature operator first extracts characteristic points and matches them through epipolar geometry constraints, selecting the top K points with the highest confidence. Subsequently, points in textureless areas are ensured for translational and rotational invariance by computing the Euclidean distance with respect to the K points, along with ΔX and ΔY. In Cotracker, to ensure that each tracking point focuses on surrounding features, sampling $\phi(I_i)[P_i/ks + \delta]$ near the tracking point is performed, calculating the dot product between this sampled area and the tracking feature, which is called correlation features.
Correlation features: $C_t^i \in \mathrm{R}^S$ are obtained by comparing the track features Q and the image features $\phi(I_t; s)$ around the current track locations $\hat{P}_t^i$. Specifically, the vector C_t^i is obtained by stacking the inner products:

$$\left\langle Q_t^i, \phi(I_t; s)[\hat{P}_t^i/ks + \delta] \right\rangle s = 1, ..., S,\ \delta \in^2,\ ||\delta|| \leq \Delta,\ \Delta \in$$

where the offsets δ define a neighborhood of point $\hat{P}_t^i$. $(S = \Delta = 4)$The image features $\phi(I^t; s)$ are sampled at non-integer locations by using bilinear interpolation and zero padding.

3.3.2 Tokens

The input token $G(\hat{P}, \hat{v}, Q)$ code for position, visibility, appearance, and correlation of the tracks. This information is represented by stacking corresponding features:

$$G_t^i = (\hat{P}, \mathrm{logit}(\hat{v}), Q_t^i, C_t^i, \eta(\hat{P}_t^i - \hat{P}_1^i))$$

where the last component is derived from the estimated position: it is the sinusoidal positional encoding η of the track location with respect to the initial location at time $t = 1$.

The output token: $O(\hat{P}', Q')$, are the updated locations and appearance features

3.3.3 Iterated transformer applications

We apply the transformer M times in order to progressively improve the track estimates. Let $m = 0, 1, ..., M$ index the estimate, with $m = 0$ denoting initialization. Then:

$$O(\hat{P}^{(m+1)}, Q^{(m+1)}) = \Psi(G(\hat{P}^{(m)}, \hat{v}^{(0)}, Q^{(m)}))$$

The visibility mask $\hat{v}$ is updated at the end of the M transformations as $\hat{v}^{(M)} = \sigma(WQ^{(M)})$, where σ is the sigmoid activation function and W is a learned matrix of weights.

Initialization:

$$\hat{P}_t^{i,(0)} \leftarrow P_{t^i}^i, \ \hat{v}_t^{i,(0)} \leftarrow 1, \ \left[Q_t^{i,(0)}\right]_s \leftarrow \phi(I_{t^i}; s)[P_{t^i}^i/ks]$$

3.4 Point Selection

A key insight in our approach is the simultaneous tracking of multiple points, which enhances the model's ability to reason about video motion and track correlations. However, caution is warranted when evaluating the method on benchmark datasets, where human-annotated points often reside in salient locations on moving objects.

3.4.1 Anchor point

For the "anchor selection" strategy, We apply coarse matching on feature maps of neighborhood frames to get the anchor matching candidates set Candidate$(P) = P \cup$ TopK(M_c), where P is the set of all track points,M_c is the confidence matrix that comes from a coarse-level local feature matching algorithm and TopK stands for selecting the top-K points with high matching confidence. By including these anchor points during tracking, we can maintain the point-wise relative positions information at a high confidence level and thus potentially avoid misleading signals from occlusions and other interruptions that may occur in the video.

3.4.2 Compare

And we use them to compare with other simpler strategies,including the "global" strategy, choosing additional points systematically on a regular grid across the entire image, and the "local" strategy, involving selecting points in proximity to the target point to utilize a regular grid around the target to focus on its neighborhood. Notably, point selection is solely employed during inference.

3.5 Windowed Inference

Taking advantage of the transformer, we split the video in $J = \frac{2T'}{T1}$ windows of length T, with an overlap of $T/2$ frames. In each window, we apply the transformer M times, the output of the first window is used as the input to the second window and so on. Then we have a $M \times J$ grid of quantities $(\hat{P}^{(m,j)}, \hat{v}^{(m,j)}, Q^{(m,j)})$.

Specifically, the first $T/2$ components of $\hat{P}^{(0,j+1)}$ are copies of the last $T/2$ components of $\hat{P}^{(M,j)}$; the last $T/2$ components of $\hat{P}^{(0,j+1)}$ are instead copies of the last time $t = T/2 - 1$ from $\hat{P}^{(M,j)}$. The same rule is used for $\hat{v}$.

Originally, $Q^{(0,j)}$ is always initialized with initial track features Q. To tackle losing track of long-time occluded points, we introduce a trick brought by PIPs++[28] and adapt is into sliding

window technique, which is to maintain the recent-appearance features of tracks.

$$Q^{(0,j+1)} \leftarrow Q^{(0,j)} + \hat{v}^{(M,j)} \times \left(Q'^{(M,j)} - Q^{(0,j)}\right)$$

where Q' indicates each track's track feature of its most convincing frame to be visible.

By applying the following formula after each M iterated transformer applications, we update track features Q if and only if the points are credibly visible. The emphasis of confidence is to avoid edge-effect.

4 Experiment

4.1 Evaluation

We consider several datasets for training and evaluation. We evaluate CoTracker on the TAP-Vid benchmark datasets TAP-Vid-DAVIS and TAP-Vid-Kinetics. In these benchmarks, points are queried on objects at random frames and the goal is to predict positions and occlusion labels of queried points.

To analyze the experimental results of CoTracker on the above-mentioned dataset, we conducted experiments again and assessed the performance of CoTracker through manual evaluation. The model demonstrates excellent tracking capabilities in some scenarios. However, when there is a brief appearance of fast-moving objects occluding foreground objects in the video frames, some tracking points shift from the original object to the momentarily appearing fast object. Additionally, although CoTracker exhibits remarkable accuracy when tracking objects that reappear after a short disappearance, the precision of tracking tends to decrease with slightly irregular object movements. We suspect this is due to the model analyzing and learning the motion patterns of objects to make predictions. You can refer to Appendix A for more detail.

4.2 PointOdyssey

Additionally, We tried a state-of-art dataset, PointOdyssey[28], to fine-tune our model. PointOdyssey is a large-scale synthetic dataset, and data generation framework, for the training and evaluation of long-term fine-grained tracking algorithms, with annotations including trajectories, visibility, and masks.

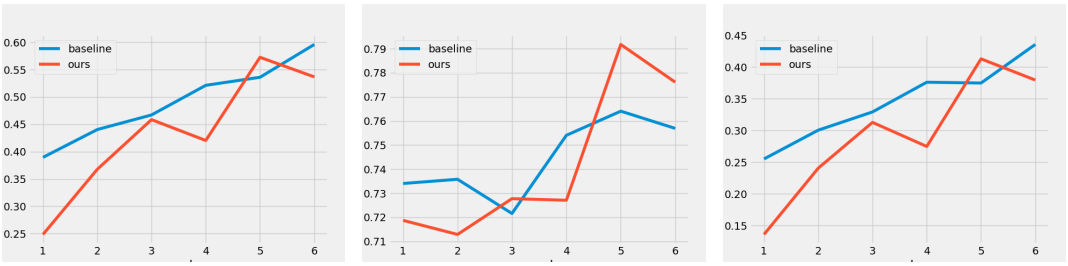
Figure 1: Extraction

We further extract a textureless subset using the mask information provided to tune our model. As textureless tracks always lie in static surfaces, we keep some relatively active points to prevent model from overfitting the camera movement.

We attempted to utilize information such as masks in PointOdyssey; however, during the actual training process, we opted to use the Kubric dataset. This decision was made because we observed that the model's training performance was unsatisfactory and encountered data gap when fine-tuning on it. In contrast, training on the Kubric dataset yielded acceptable results.

4.3 Training

The original model was trained on 32 Nvidia GPUs with 50000 steps, equivalent to 1,600,000 segments with 768 trajectories each. Due to the limitation of both graphic memory and FLOP, we only tried to train the baseline and ours on 64,680 segments with 384 trahectories to distinguish the difference on early stages.



(a) delta (b) OA (c) Jaccard

Figure 2: evaluation metrics

Three metrics, delta, OA, and Jaccard are used to evaluate the models, they represent position accuracy, visible classification accuracy, and true positive fraction respectively. Though delta and Jaccard seem similar at 5 or 6 epochs, the OA of our model exceeded the baseline significantly, indicating that our model gain a considerable improvement on perceiving the visibility of the tracks. However, the final differences are yet to be seen since the training was far less than the released version of CoTracker.

4.4 Optical Flow Guidance

For a model like ours that focuses on textureless tracking, it updates the position of query points gradually through iterative transformers. Consequently, the model predicts the next moment's position of query points. However, as the model tracks points based on their mutual correlations, this potentially leads to the influence of other points on a query point. We believe that providing some prior knowledge to these points, such as an estimation of where they will roughly be in the next moment (in other words, imparting optical flow information), could be beneficial. We can integrate optical flow information with other data and use it as input to the transformer. Additionally, due to the input of optical flow, the transformer may not need to iterate many times, thus improving efficiency. As for the optical flow model, currently, raft is a good option. Raft[23] is a lightweight model with high efficiency. In the future, we can integrate raft into our model.

4.5 Additional Mask Information

For existing models, points on different objects still interact with each other. For example, points at the edge of a computer screen can affect points outside the screen. The vibration at the edge of the computer can cause points in the surrounding area to vibrate as well. To mitigate the interaction between query points on different objects, we can provide the model with the mask of the current frame. This involves using a segmentation model to obtain masks for different objects and then concatenating the mask information with the original information. This combined data is used as input for the Transformer, allowing the model to learn about the interactions between different objects. This approach may lead to more precise transformations of query point positions, reducing the impact of points on different objects. Currently, one of the better models for this purpose is FastSAM[33], which is also a lightweight model with high segmentation efficiency.

Figure 3: Experiment

References

[1] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. arXiv preprint arXiv:2307.07635. 2023.

[2] Jiahuan Yu, Jiahao Chang, Jianfeng He, Tianzhu Zhang, and Feng Wu. Adaptive spot-guided transformer for consistent local feature matching. arXiv preprint arXiv:2303.16624, 2023.

[3] Chang, Jiahao, Jiahuan Yu and Tianzhu Zhang. Structured Epipolar Matcher for Local Feature Matching, CVPRW, 2023.

[4] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In Proc. ICCV, 1993.

[5] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. IJCV, 61, 2005.

[6] Michal Neoral, Jonáš Šerých, and Ji ˇrí Matas. Mft: Long-term tracking of every pixel, 2023.

[7] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[8] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In Proc. ICCV, 2015.

[9] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual resolution correspondence networks. Advances in Neural Information Processing Systems, 33, 2020.

[10] Berthold KP Horn and Brian G Schunck. Determining optical flow. Artificial intelligence, 17(1-3):185–203, 1981.

[11] Berthold KP Horn and Brian G Schunck. Determining optical flow. Artificial intelligence, 17(1-3), 1981.

[12] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In Proc. ECCV, 2022.

[13] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In Proc. CVPR, 2017.

[14] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs outputs. In Proc. ICLR, 2022.

[15] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning optical flow from a few matches. In Proc. CVPR, 2021.

[16] David G Lowe. Distinctive image features from scale invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004

[17] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision, volume 81. Vancouver, 1981.

[18] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proc. IJCAI, volume 2, 1981.

[19] Jerome Revaud, Philippe Weinzaepfel, Cesar Roberto de ´ Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In Advances in Neural Information Processing Systems, 2019.

[20] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564– 2571. Ieee, 2011.

[21] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4938–4947, 2020.

[22] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. arXiv, 2023.

[23] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In Proc. ECCV, 2020.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proc. NeurIPS, 2017.

[25] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In Proc. CVPR, 2022.

[26] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. High-resolution optical flow from 1d attention and correlation. In Proc. CVPR, 2021.

[27] Feihu Zhang, Oliver J Woodford, Victor Adrian Prisacariu, and Philip HS Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In Proc. CVPR, 2021.

[28] Y. Zheng, A. W. Harley, B. Shen, G. Wetzstein, and L. J. Guibas, Pointodyssey: A large-scale synthetic dataset for long-term point tracking, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp.19 855–19 865.

[29] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelovic, Akihiko ´ Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In Advances in Neural Information Processing Systems, pages 1658–1669, 2018.

[30] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8922–8931, 2021.

[31] A. W. Harley, Z. Fang, and K. Fragkiadaki, Particle video revisited: Tracking through occlusions using point trajectories, 2022.

[32] Michael Rubinstein, Ce Liu, and William Freeman. Towards longer long-range motion trajectories. In Proceedings of British Machine Vision Conference, 2012.

[33] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv preprint arXiv:2306.12156*, 2023.

A Experiment

Table 2 shows the experiments outcomes. We compare our strategies with previous work including CoTracker and our model perform better than most of the previous models. Although we still don't beat CoTracker in occlusion accuracy, it is possibly because the global grid points CoTracker use are much more than ours. Our method is less time-consuming because less query points are needed during tracking. It is also worth noticed that the authors of CoTracker also find this strategy useful and they added supportive anchor points in their later experiments, as shown in figure 4.

	Attention		Points		DAVIS First		
	time	joint	target	support	AJ ↑	δ_{avg} ↑	OA ↑
(a)	✓	✗	single	—	58.3	71.5	86.4
(b)	✓	✓	single	—	41.1	62.9	75.1
(c)	✓	✓	single	glob. 9×9	56.8	71.2	85.8
(d)	✓	✓	single	SIFT 8×8	57.4	72.1	85.8
(e)	✓	✓	single	loc. 10×10	60.4	75.2	87.5
(f)	✓	✓	single	glob. 5×5 +loc. 8×8	**62.2**	**75.7**	**89.3**
(g)	✓	✓	all	—	57.6	73.2	86.1
(h)	✓	✓	all	glob. 5×5	60.5	**75.8**	87.6
(i)	✓	✓	all	SIFT 8×8	**60.7**	75.7	**88.1**

Figure 4: additional anchor points

Table 1: Evaluation on TAP-Vid.

Method	AJ	δ^{i}_{avg}	OA
TAP-Net	33.0	48.6	78.8
PIPs	42.2	64.8	77.7
MFT	47.3	66.8	77.8
OmniMotion	-	-	-
TAPIR	56.2	70.0	86.5
CoTracker	**60.6**	**75.4**	**89.3**
Ours(8×5)	57.4	75.2	84.3
Ours(8×3)	56.6	74.3	83.9

Table 2: The table shows our idea and the comparison with previous work. Although we still don't beat CoTracker in occlusion accuracy, it is possibly because the global grid points CoTracker use are much more than ours. It is also worth noticed that the researchers of CoTracker also find this strategy useful and they added supportive anchor points in their later experiments.