



Факультет
компьютерных наук

Образовательная
программа ПМИ

Object Tracking: everything & everywhere

Подготовили студенты:

Пономарева Ольга
Теняев Александр
Алимханов Карим
Бабаев Минходж
Солодовников Михаил
Некрасов Артём
Хайруллин Инсаф



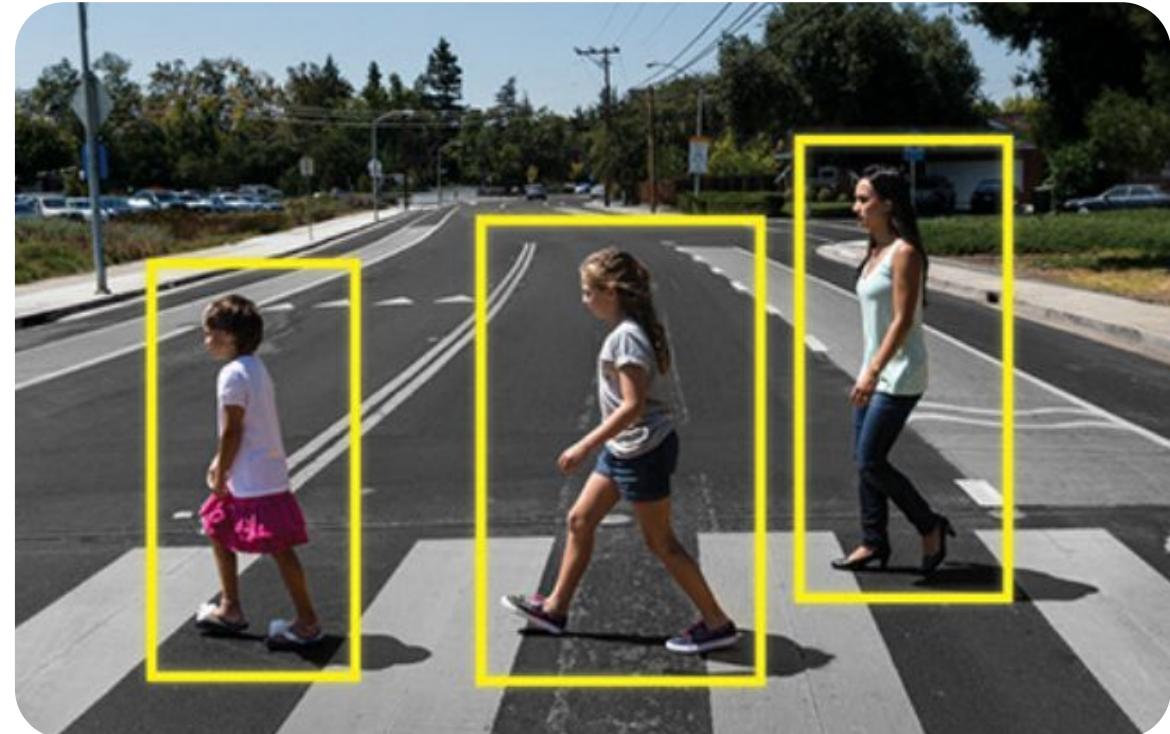
1. Object Tracking
2. DeepSort
3. Motion Estimation: Sparse (SLAM) vs Dense (RAFT)
4. Модель Motion tracking: CoTracker
5. Модель Motion tracking: OmniMotion

Object Detection

Определение объектов на картинке/кадре

Нейронная сеть определяет объект и записывают координаты рамок прямоугольников.

Алгоритм работает только с одним изображением





Модели и их принцип работы

4

Две ключевые модели: YOLO и Mask R-CNN.

Обе модели основаны на архитектуре Faster R-CNN

YOLO

Фокусируется на точности границ
рамок объектов

Однопроходная сеть

Mask R-CNN

Сосредоточен на разделении
объектов на разные группы и
экземпляры.

Создает маски для каждого объекта



Различия моделей

5

	YOLO	Mask R-CNN
Точность	Средняя/низкая	Высокая
Скорость	Высокая	Низкая
Комплексность	Легкая и в реализации, и в вычислениях	Вычислительнозатратная и имеет более сложную реализацию
Применение	Быстрый анализ большого количества изображений или видео	Высокая точность в определении границ объектов и разделении на группы.

Object Tracking

Отслеживание объектов в кадре

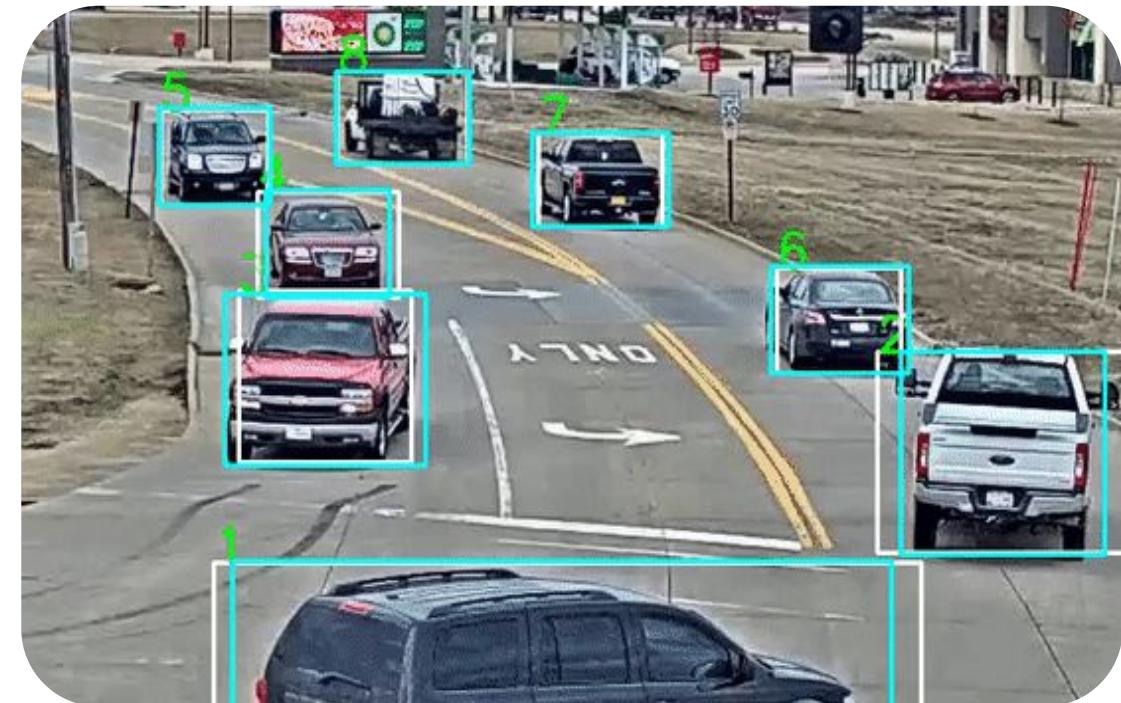
Задача:

- определить объекты в кадре
- связать информацию предыдущих кадров

Включает в себя:

- Object Detection
- Алгоритмы для идентификации объекта

с





DeepSort

7

DeepSort (Deep Learning SORT) - это алгоритм отслеживания объектов в видео. Он представляет собой улучшенную версию алгоритма SORT (Simple Online and Realtime Tracking) и использует глубокое обучение для повышения точности отслеживания.

Задачи:

1. Повышение точности отслеживания
 2. Ассоциация объектов между кадрами
 3. Поддержание уникальных идентификаторов объектов

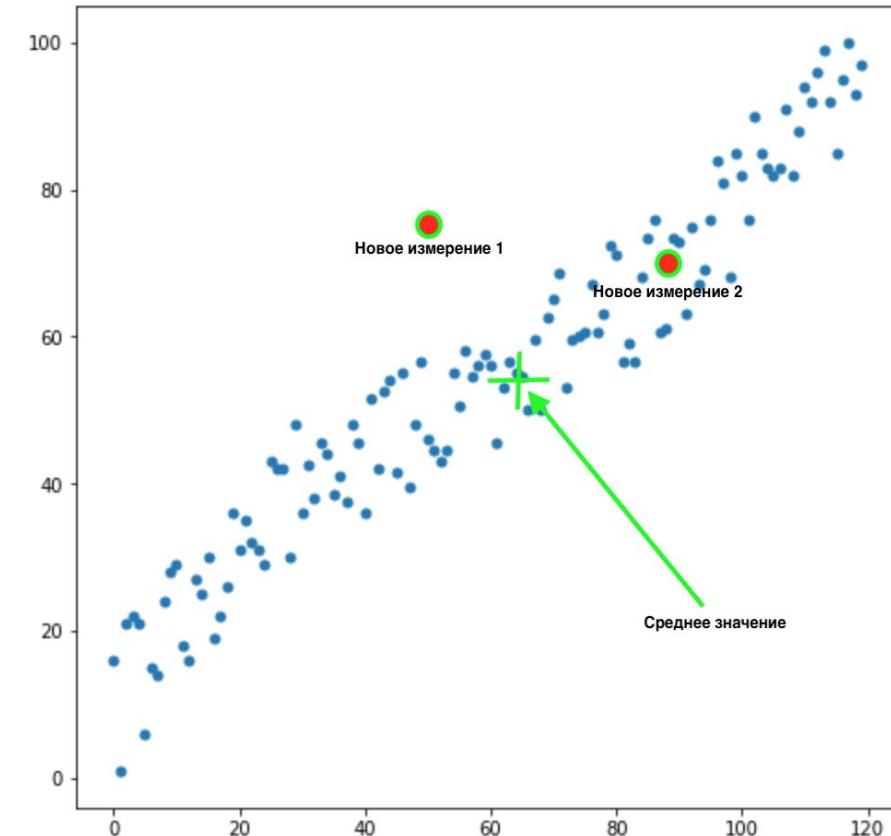


Расстояние Махалонобиса

$$d^{(1)}(i, j) = (\mathbf{d}_j - \mathbf{y}_i)^T \mathbf{S}_i^{-1} (\mathbf{d}_j - \mathbf{y}_i),$$

где \mathbf{d}_j - j-е обнаружение \mathbf{y}_i , \mathbf{S}_i - среднее значение и матрица ковариации i-го распределения треков.

$$\begin{matrix} & x_1 & x_2 & x_3 \\ x_1 & cov(x_1, x_1) & cov(x_1, x_2) & cov(x_1, x_3) \\ x_2 & cov(x_2, x_1) & cov(x_2, x_2) & cov(x_2, x_3) \\ x_3 & cov(x_3, x_1) & cov(x_3, x_2) & cov(x_3, x_3) \end{matrix}$$



Косинусное расстояние:

$$d^{(2)}(i, j) = \min\{1 - \mathbf{r}_j^T \mathbf{r}_k^{(i)} \mid \mathbf{r}_k^{(i)} \in \mathcal{R}_i\}.$$

где \mathbf{r}_j - это дескриптор внешнего вида для j-го обнаружения, $\mathbf{r}_k^{(i)}$ - это дескриптор внешнего вида для k-го элемента дорожки распределения i-й дорожки.

Объединяем:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda)d^{(2)}(i, j)$$



Фильтр Калмана

10

Фильтрация Калмана -это алгоритм, который обеспечивает оценки некоторых неизвестных переменных с учетом измерений, наблюдаемых во времени

Показатели – ($u, v, a, h, u', v', a', h'$), где u, v – это позиция предсказанного прямоугольника по X и Y, a – это соотношение сторон предсказанного прямоугольника (aspect ratio), h – высота прямоугольника, u', v', a', h' – производные по каждой величине.

Listing 1 Matching Cascade

Input: Track indices $\mathcal{T} = \{1, \dots, N\}$, Detection indices $\mathcal{D} = \{1, \dots, M\}$, Maximum age A_{\max}

- 1: Compute cost matrix $\mathbf{C} = [c_{i,j}]$ using Eq. 5
 - 2: Compute gate matrix $\mathbf{B} = [b_{i,j}]$ using Eq. 6
 - 3: Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
 - 4: Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
 - 5: **for** $n \in \{1, \dots, A_{\max}\}$ **do**
 - 6: Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
 - 7: $[x_{i,j}] \leftarrow \text{min_cost_matching}(\mathbf{C}, \mathcal{T}_n, \mathcal{U})$
 - 8: $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i, j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
 - 9: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
 - 10: **end for**
 - 11: **return** \mathcal{M}, \mathcal{U}
-

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

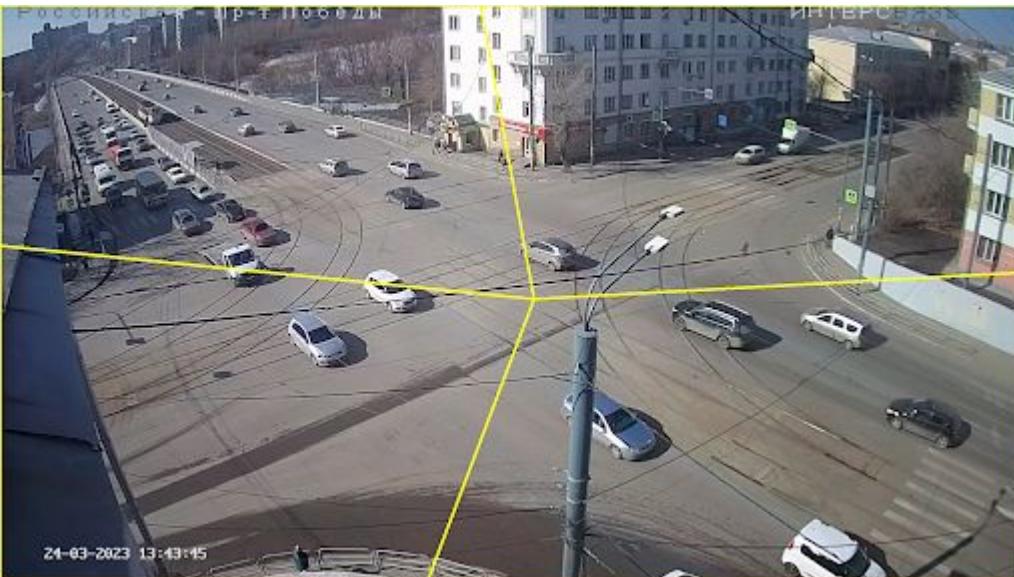
Table 1: Overview of the CNN architecture. The final batch and ℓ_2 normalization projects features onto the unit hypersphere.

Примеры

Расчет транспортного потока на основе YOLOv5 и DeepSORT на базе Deepstream

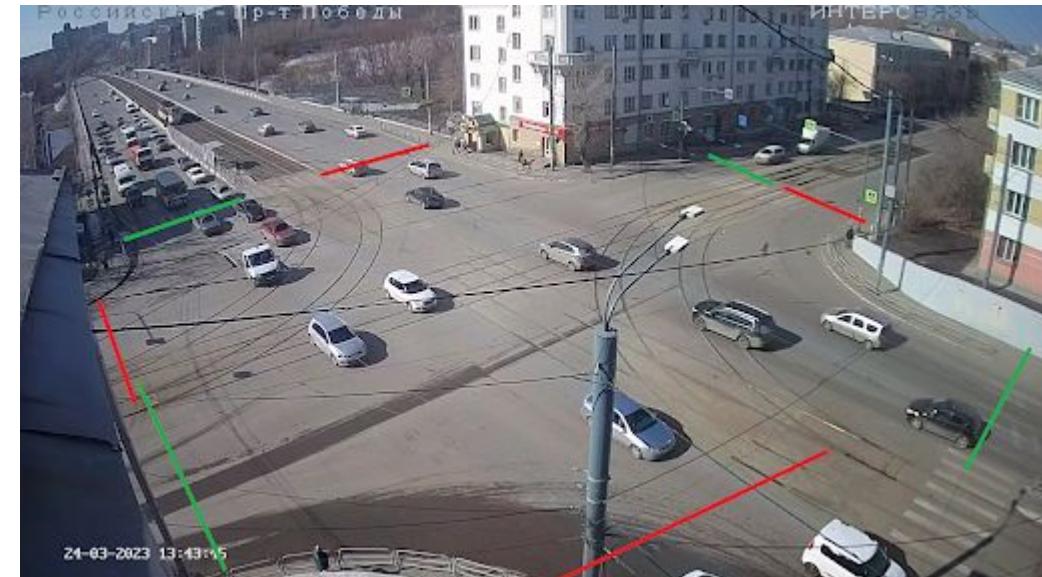
- 1) Принять видеопоток.
- 2) Пропустить его через детектор YOLOv5.
- 3) Полученные детекции обработать с помощью DeepSORT.
- 4) Проанализировать треки, полученные на основе работы DeepSORT

1 Вариант



Указание линий, при пересечении которых авто заносится в статистику.

2 Вариант



Разбивка кадра на зоны, при пересечении которых также обновляется статистика.

		MOTA ↑	MOTP ↑	MT ↑	ML ↓	ID ↓	FM ↓	FP ↓	FN ↓	Runtime ↑
KDNT [16]*	BATCH	68.2	79.4	41.0%	19.0%	933	1093	11479	45605	0.7 Hz
LMP-p [17]*	BATCH	71.0	80.2	46.9%	21.9%	434	587	7880	44564	0.5 Hz
MCMOT_HDM [18]	BATCH	62.4	78.3	31.5%	24.2%	1394	1318	9855	57257	35 Hz
NOMTwSDP16 [19]	BATCH	62.2	79.6	32.5%	31.1%	406	642	5119	63352	3 Hz
EAMTT [20]	ONLINE	52.5	78.8	19.0%	34.9%	910	1321	4407	81223	12 Hz
POI [16]*	ONLINE	66.1	79.5	34.0%	20.8%	805	3093	5061	55914	10 Hz
SORT [12]*	ONLINE	59.8	79.6	25.4%	22.7%	1423	1835	8698	63245	60 Hz
Deep SORT (Ours)*	ONLINE	61.4	79.1	32.8%	18.2%	781	2008	12852	56668	40 Hz

Table 2: Tracking results on the MOT16 [15] challenge. We compare to other published methods with non-standard detections. The full table of results can be found on the challenge website. Methods marked with * use detections provided by [16].

Motion Estimation: Sparse (SLAM) vs Dense (RAFT)

Оценка движения (motion estimation) - один из важнейших методов в компьютерном зрении и робототехнике, который включает в себя определение движения объектов по последовательности изображений или видео.

Методы оценки движения:

- Block Matching Algorithms (BMA)
- Optical Flow Methods
- Pel-recursive Algorithms
- Frequency Domain Methods



Постановка задачи:

1. Для блочного сопоставления:

- Пусть $B_i(t)$ обозначает блок i -го кадра в момент времени t .
- Необходимо найти вектор движения $v_i(t)$, который минимизирует функцию ошибки (например, среднеквадратическую) между блоком $B_i(t)$ и блоком $B_j(t+1)$ в следующем кадре:

$$v_i(t) = \arg \min_v \sum_{x,y \in B_i} (I(x, y, t) - I(x + v_x, y + v_y, t + 1))^2$$

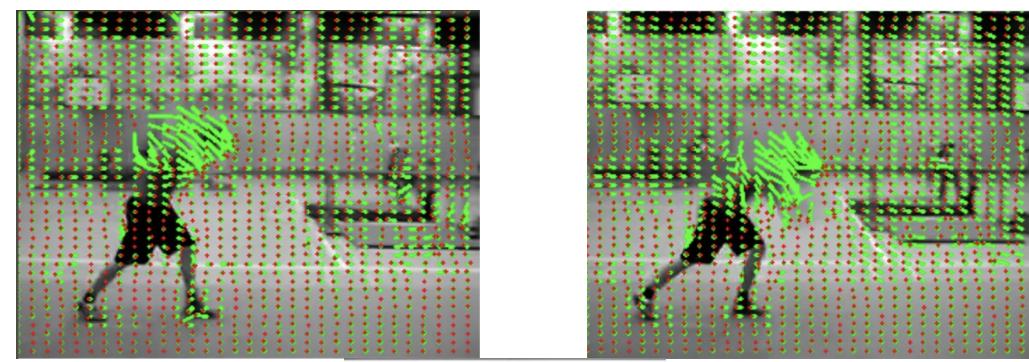
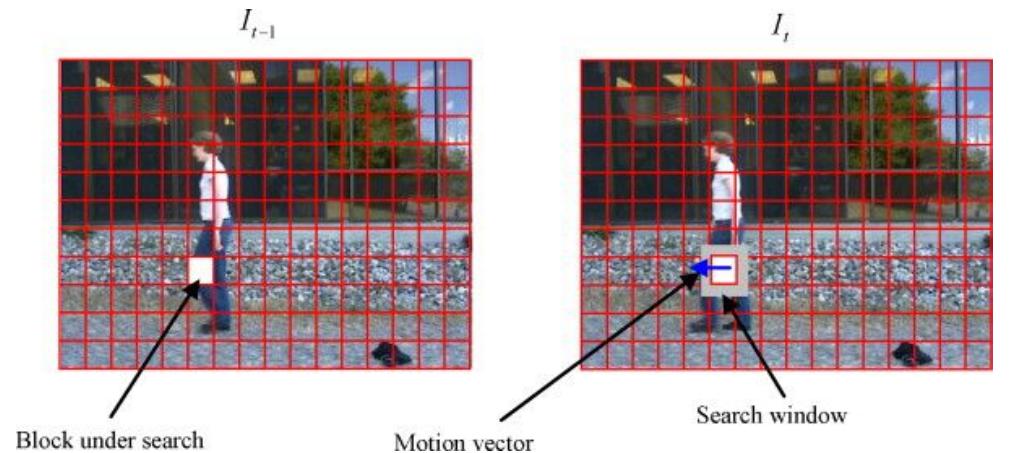
где $I(t)$ - последовательность видеокадров, $t = 0, 1, 2, \dots, T$ - временные метки кадров, (x, y) - координаты пикселей внутри блока, v_x и v_y - компоненты вектора движения $v_i(t)$.

2. Для оптического потока:

- Для каждого пикселя (x, y) определить вектор движения $(u(x, y, t), v(x, y, t))$, который удовлетворяет уравнению оптического потока:

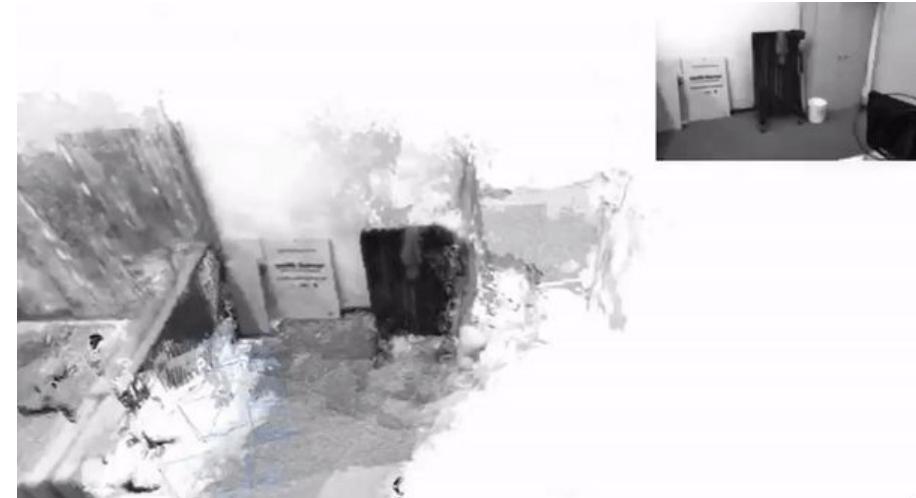
$$I_x(x, y, t)u(x, y, t) + I_y(x, y, t)v(x, y, t) + I_t(x, y, t) = 0$$

где I_x, I_y, I_t - частные производные интенсивности изображения по пространственным координатам и времени, а полный вектор оптического потока для пикселя (x, y) можно представить как $(u(x, y), v(x, y))$.

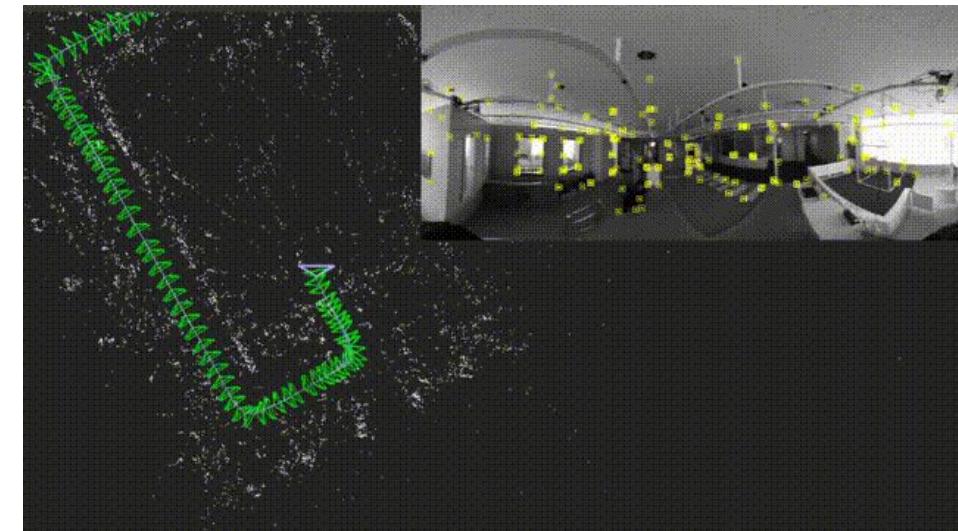
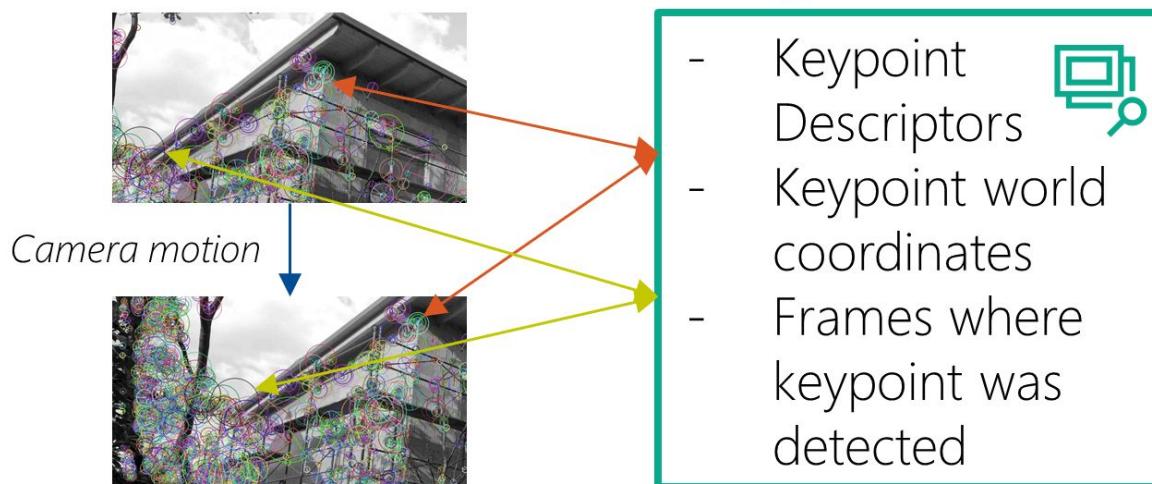


SLAM (*simultaneous localization and mapping* или одновременная локализация и построение карты) — метод, используемый в мобильных автономных средствах для построения карты в неизвестном пространстве или для обновления карты в заранее известном пространстве с одновременным контролем текущего местоположения и пройденного пути.

Некоторые реализации метода используются в беспилотных автомобилях, летательных аппаратах, автономных подводных аппаратах, планетоходах, и даже внутри человеческого тела.



Keypoint tracking — это процесс идентификации и отслеживания уникальных, устойчивых особенностей (ключевых точек) в последовательности изображений. Эти ключевые точки помогают устройству понять его перемещение относительно окружающей среды. Ключевые точки обычно представляют собой углы, края или текстурированные области, которые легко отличить друг от друга и найти на нескольких кадрах.





Сильные стороны:

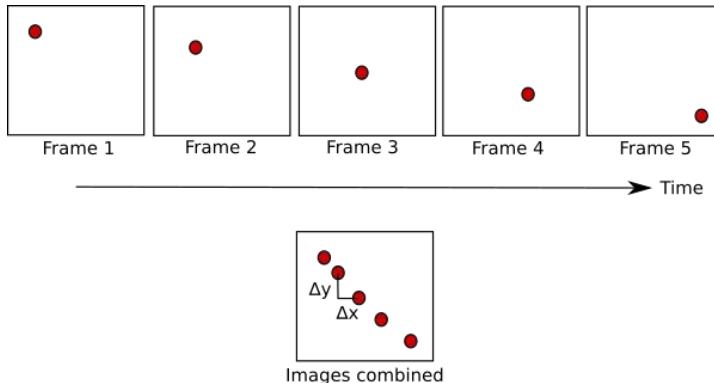
- SLAM может интегрировать данные от различных сенсоров, таких как лидары, камеры, IMU (инерциальные измерительные устройства) и ультразвуковые датчики.
- Многие современные алгоритмы SLAM разработаны с учетом возможности работы в реальном времени, что позволяет системам обновлять карту и локализацию без задержек, что критически важно, например, для автономных транспортных средств.
- SLAM используется в робототехнике, автономных транспортных средствах, дополненной реальности и многих других областях.

Недостатки:

- В условиях низкого освещения, плохого контраста или отсутствия текстур (например, на гладких поверхностях) алгоритмы детекции ключевых точек могут не срабатывать эффективно, что снижает точность.
- Маленькие ошибки в оценке движения могут накапливаться со временем, приводя к дрейфу и искажениям карты.
- В некоторых случаях ускорение работы алгоритма может потребовать компромиссов в точности локализации и картирования.

RAFT (Recurrent All-pairs Field Transforms или рекуррентные преобразования всех пар полей) - это современный метод для оценки оптического потока в видеоизображениях, представляет собой сочетание архитектур CNN и RNN.

Оптический поток – это векторное представление, которое количественно определяет перемещение элемента между двумя кадрами в видео.



Оптический поток
для красной точки

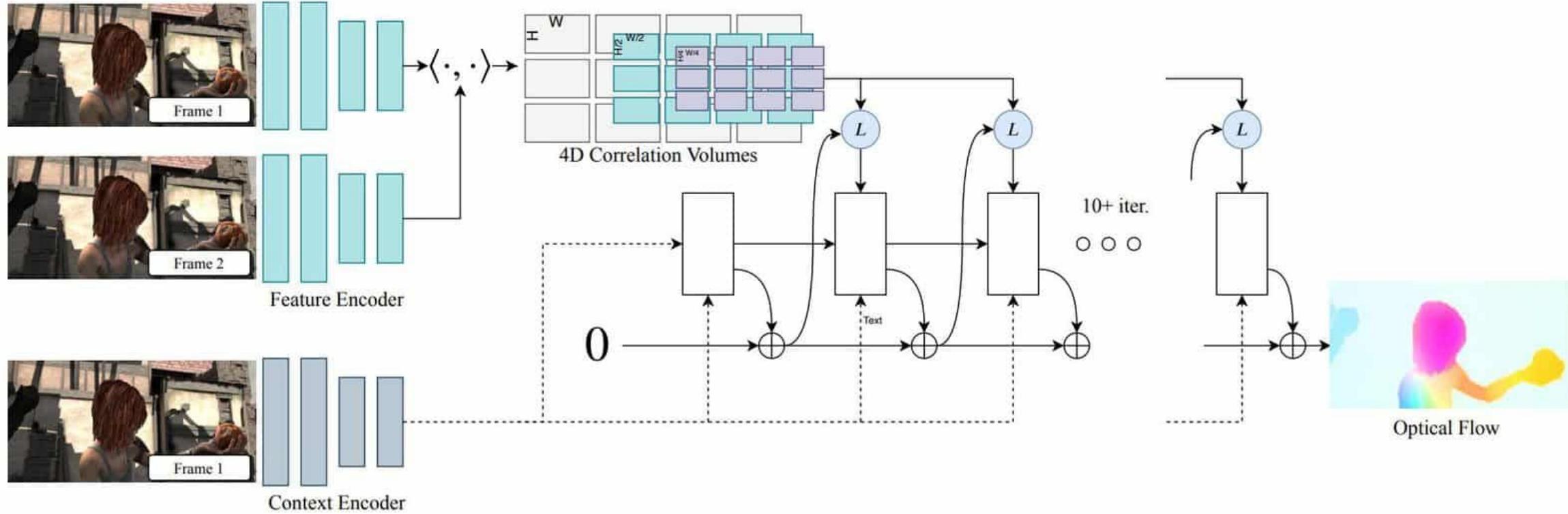


Оценка разреженного оптического потока



Оценка плотного оптического потока

RAFT. Архитектура



Сильные стороны:

- RAFT демонстрирует выдающуюся точность в оценке оптического потока на различных наборах данных. Модель способна улавливать мелкие детали движения благодаря плотному полю корреляции и рекуррентным обновлениям.
- Оценивает оптический поток для каждого пикселя изображения, создавая плотное поле движения, что позволяет получать подробные и гладкие карты движения.
- Использование рекуррентной нейронной сети для итеративного обновления поля оптического потока позволяет модели постепенно улучшать свои предсказания.

Недостатки:

- Полный подход на основе всех пар пикселей требует значительных вычислительных ресурсов и памяти.
- В сценах с окклюзиями некоторые пиксели могут исчезнуть из поля зрения, и, следовательно, не будет существовать соответствия между кадрами для таких пикселей.





Модель Motion tracking: OmniMotion

24



OmniMotion representation

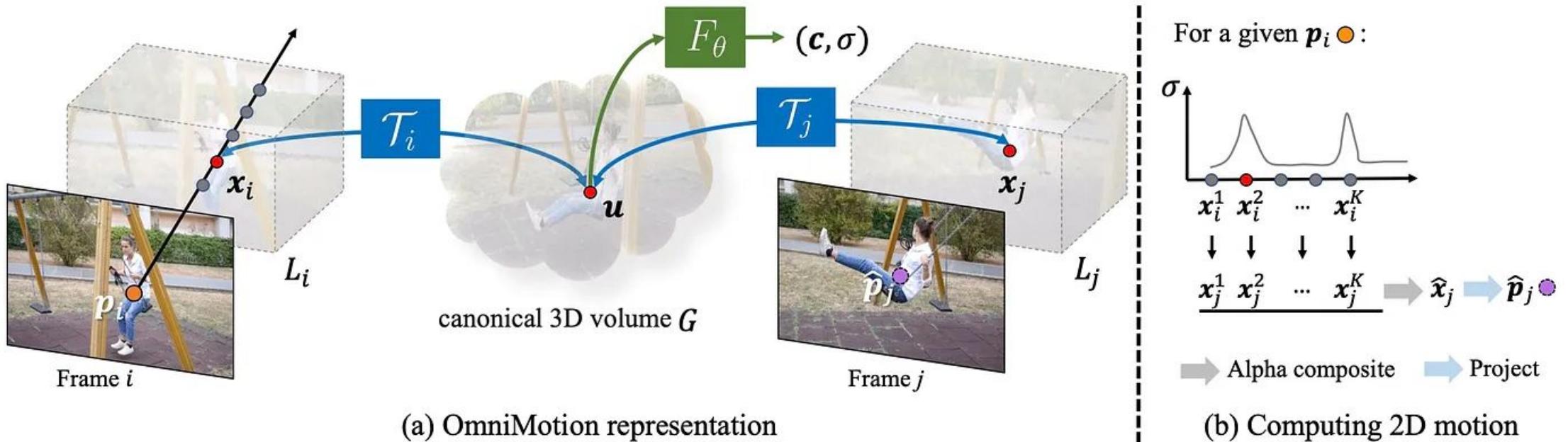


Figure 2: *Method overview.* (a) Our OmniMotion representation is comprised of a canonical 3D volume G and a set of bijections \mathcal{T}_i that map between each frame’s local volume L_i and the canonical volume G . Any local 3D location x_i in frame *i* can be mapped to its corresponding canonical location u through \mathcal{T}_i , and then mapped back to another frame *j* as x_j through the inverse mapping \mathcal{T}_j^{-1} . Each location u in G is associated with a color c and density σ , computed using a coordinate-based MLP F_θ . (b) To compute the corresponding 2D location for a given query point p_i mapped from frame *i* to *j*, we shoot a ray into L_i and sample a set of points $\{x_i^k\}_{k=1}^K$, which are then mapped first to the canonical space to obtain their densities, and then to frame *j* to compute their corresponding local 3D locations $\{x_j^k\}_{k=1}^K$. These points $\{x_j^k\}_{k=1}^K$ are then alpha-composited and projected to obtain the 2D corresponding location \hat{p}_j .

Loss functions

$$\mathcal{L} = \mathcal{L}_{\text{flo}} + \lambda_{\text{pho}} \mathcal{L}_{\text{pho}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}$$

$$\mathcal{L}_{\text{flo}} = \sum_{\mathbf{f}_{i \rightarrow j} \in \Omega_f} \|\hat{\mathbf{f}}_{i \rightarrow j} - \mathbf{f}_{i \rightarrow j}\|_1 \quad - \text{ Optical flow loss}$$

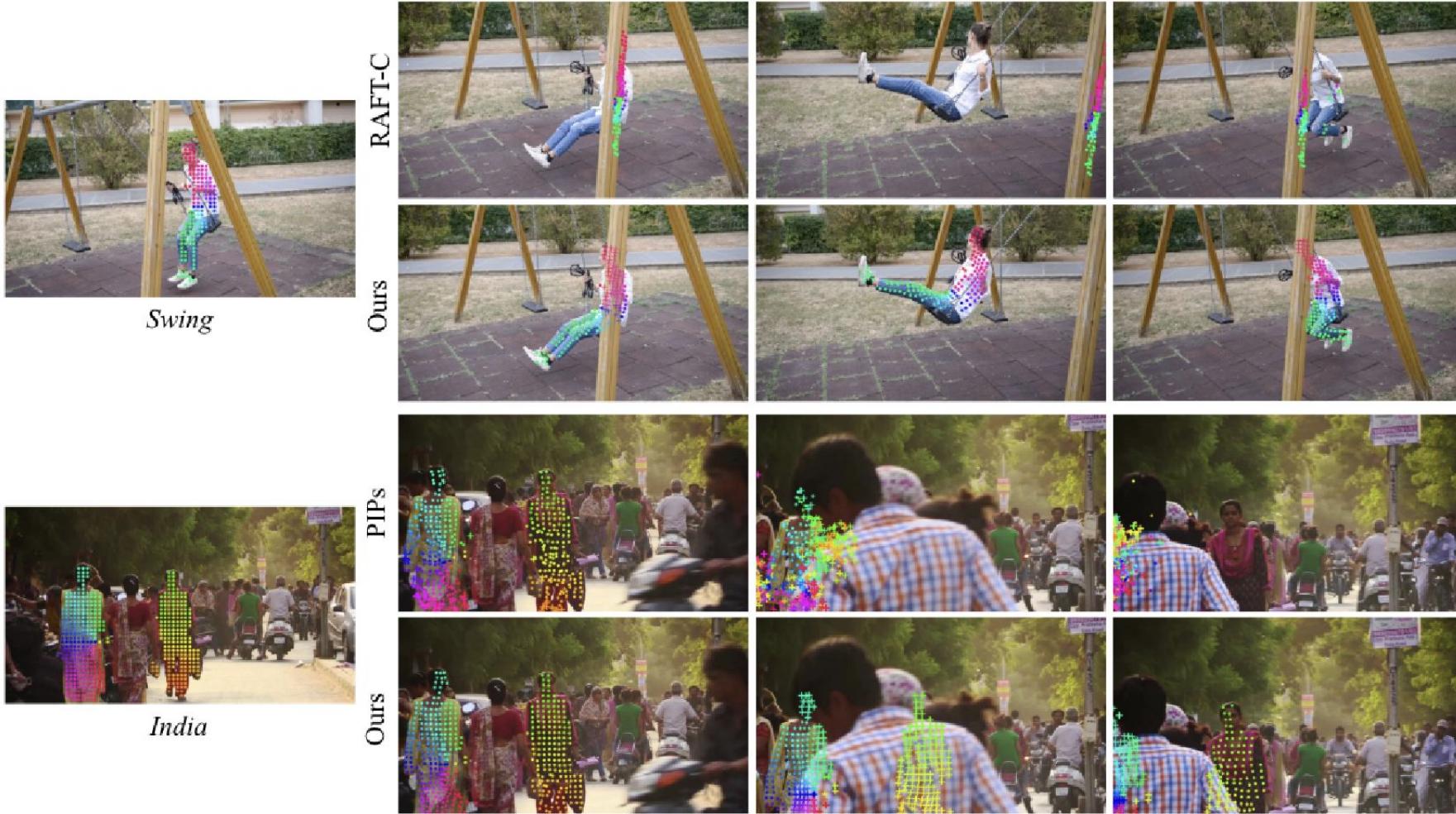
$$\mathcal{L}_{\text{pho}} = \sum_{(i, \mathbf{p}) \in \Omega_p} \|\hat{\mathbf{C}}_i(\mathbf{p}) - \mathbf{C}_i(\mathbf{p})\|_2^2 \quad - \text{ Photometric loss}$$

$$\mathcal{L}_{\text{reg}} = \sum_{(i, \mathbf{x}) \in \Omega_x} \|\mathbf{x}_{i+1} + \mathbf{x}_{i-1} - 2\mathbf{x}_i\|_1 \quad - \text{ Temporal Smoothness}$$

Method	Kinetics				DAVIS				RGB-Stacking			
	AJ \uparrow	$< \delta_{\text{avg}}^x \uparrow$	OA \uparrow	TC \downarrow	AJ \uparrow	$< \delta_{\text{avg}}^x \uparrow$	OA \uparrow	TC \downarrow	AJ \uparrow	$< \delta_{\text{avg}}^x \uparrow$	OA \uparrow	TC \downarrow
RAFT-C [66]	31.7	51.7	84.3	0.82	30.7	46.6	80.2	0.93	42.0	56.4	91.5	0.18
RAFT-D [66]	50.6	66.9	85.5	3.00	34.1	48.9	76.1	9.83	72.1	<u>85.1</u>	92.1	1.04
TAP-Net [15]	48.5	61.7	86.6	6.65	38.4	53.4	81.4	10.82	61.3	73.7	91.5	1.52
PIPs [23]	39.1	55.3	82.9	1.30	39.9	56.0	81.3	1.78	37.3	50.6	89.7	0.84
Flow-Walk-C [5]	40.9	55.5	84.5	<u>0.77</u>	35.2	51.4	80.6	0.90	41.3	55.7	<u>92.2</u>	<u>0.13</u>
Flow-Walk-D [5]	46.9	65.9	81.8	3.04	24.4	40.9	76.5	10.41	66.3	82.7	91.2	0.47
Deformable-Sprites [81]	25.6	39.5	71.4	1.70	20.6	32.9	69.7	2.07	45.0	58.3	84.0	0.99
Ours (TAP-Net)	<u>53.8</u>	<u>68.3</u>	<u>88.8</u>	<u>0.77</u>	<u>50.9</u>	<u>66.7</u>	<u>85.7</u>	<u>0.86</u>	<u>73.4</u>	84.1	<u>92.2</u>	0.11
Ours (RAFT)	55.1	69.6	89.6	0.76	51.7	67.5	<u>85.3</u>	0.74	77.5	87.0	93.5	<u>0.13</u>

Table 1: *Quantitative comparison of our method and baselines on the TAP-Vid benchmark [15]*. We refer to our method as *Ours*, and present two variants, *Ours (TAP-Net)* and *Ours (RAFT)*, which are optimized using input pairwise correspondences from TAP-Net [15] and RAFT [66], respectively. Both *Ours* and *Deformable Sprites* [81] estimate global motion via test-time optimization on each individual video, while all other methods estimate motion locally in a feed-forward manner. Our method notably improves the quality of the input correspondences, achieving the best position accuracy, occlusion accuracy, and temporal coherence among all methods tested.

Comparisons





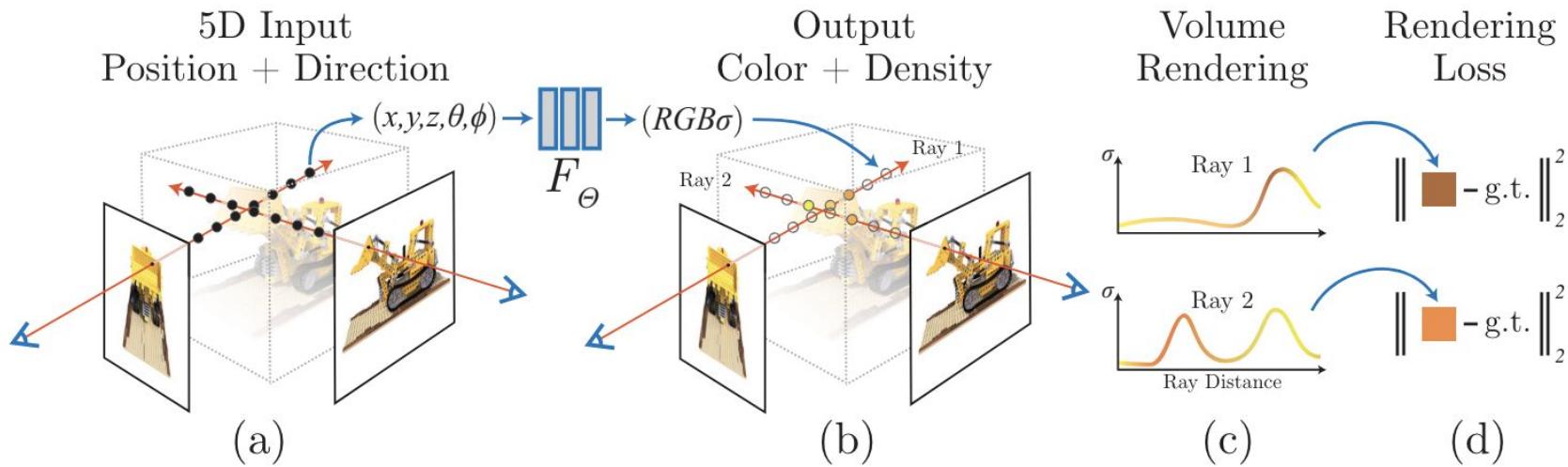


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

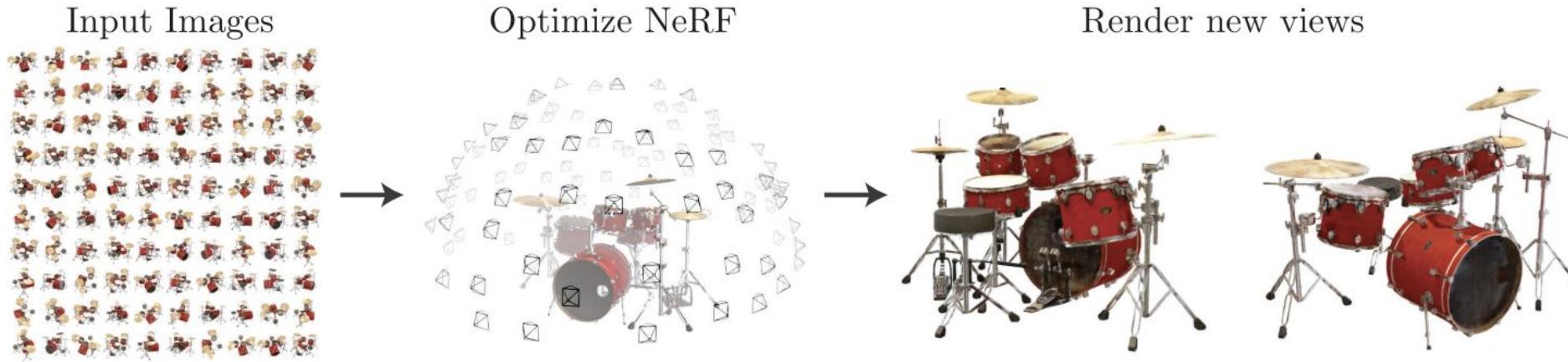


Fig. 1: We present a method that optimizes a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. We use techniques from volume rendering to accumulate samples of this scene representation along rays to render the scene from any viewpoint. Here, we visualize the set of 100 input views of the synthetic *Drums* scene randomly captured on a surrounding hemisphere, and we show two novel views rendered from our optimized NeRF representation.

Функция кодирования

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p))$$



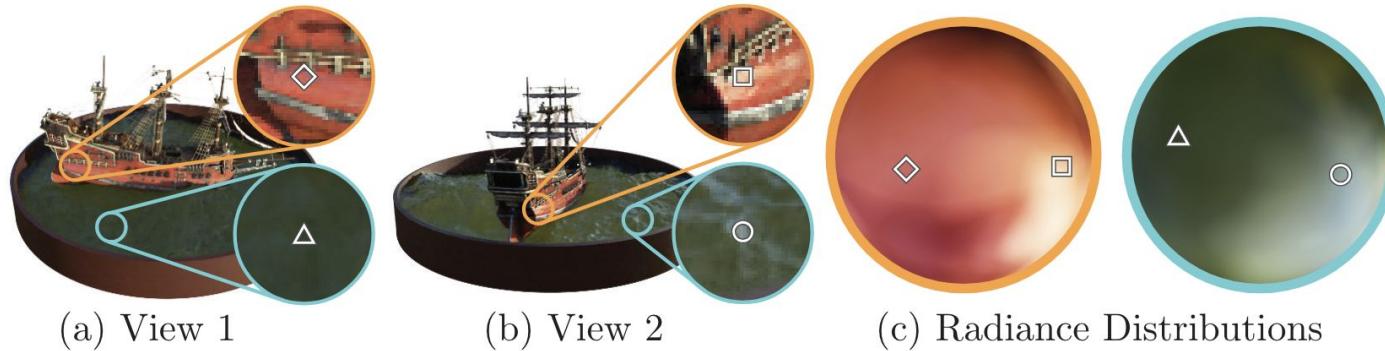


Fig. 3: A visualization of view-dependent emitted radiance. Our neural radiance field representation outputs RGB color as a 5D function of both spatial position \mathbf{x} and viewing direction \mathbf{d} . Here, we visualize example directional color distributions for two spatial locations in our neural representation of the *Ship* scene. In (a) and (b), we show the appearance of two fixed 3D points from two different camera positions: one on the side of the ship (orange insets) and one on the surface of the water (blue insets). Our method predicts the changing specular appearance of these two 3D points, and in (c) we show how this behavior generalizes continuously across the whole hemisphere of viewing directions.

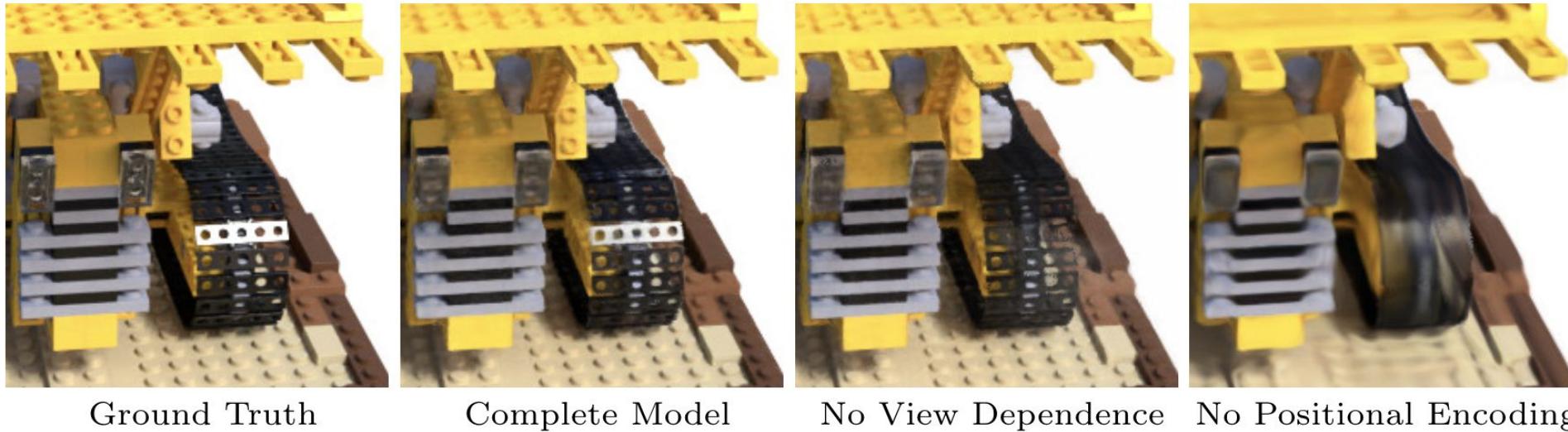


Fig. 4: Here we visualize how our full model benefits from representing view-dependent emitted radiance and from passing our input coordinates through a high-frequency positional encoding. Removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the positional encoding drastically decreases the model's ability to represent high frequency geometry and texture, resulting in an oversmoothed appearance.



NeRF: вывод

35

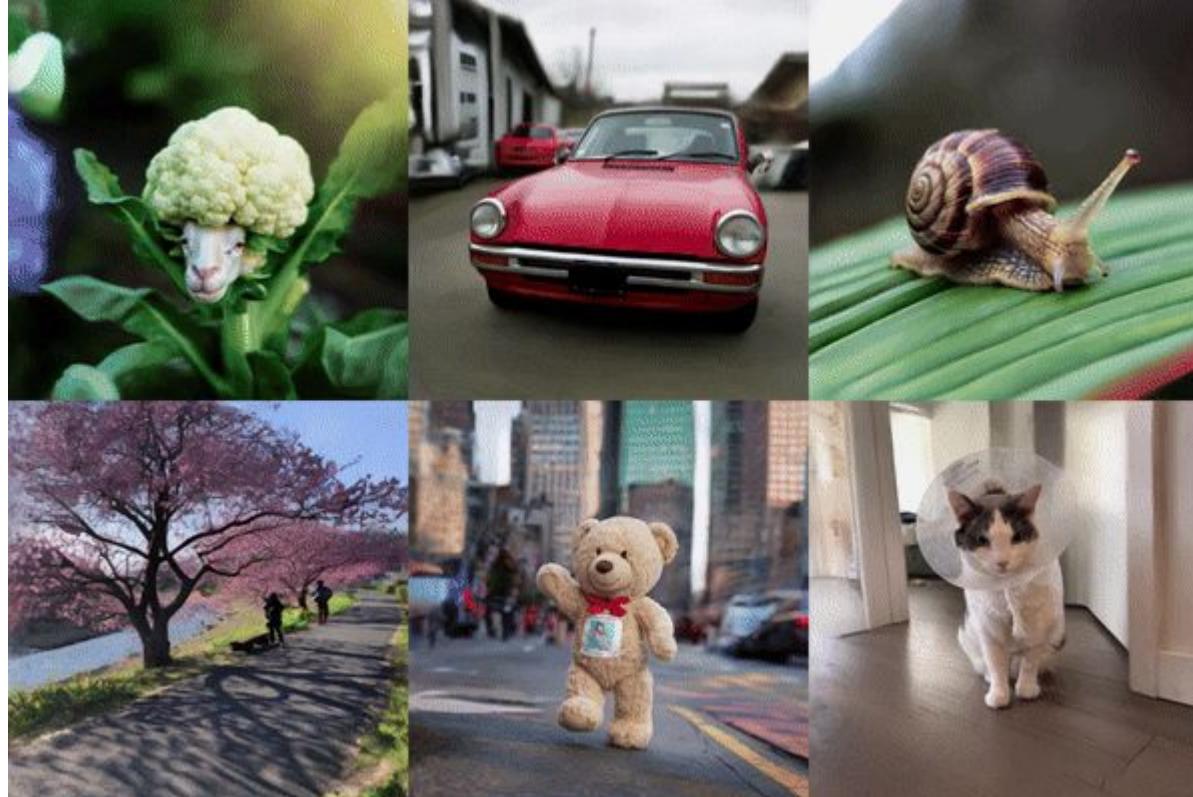
- Непрерывное Представление Сцены
- 5D Векторные Функции
- Позиционное Кодирование
- Зависимость от Направления Взгляда



Breaking News

CAT3D:Create Anything in 3D with Multi-View Diffusion Models

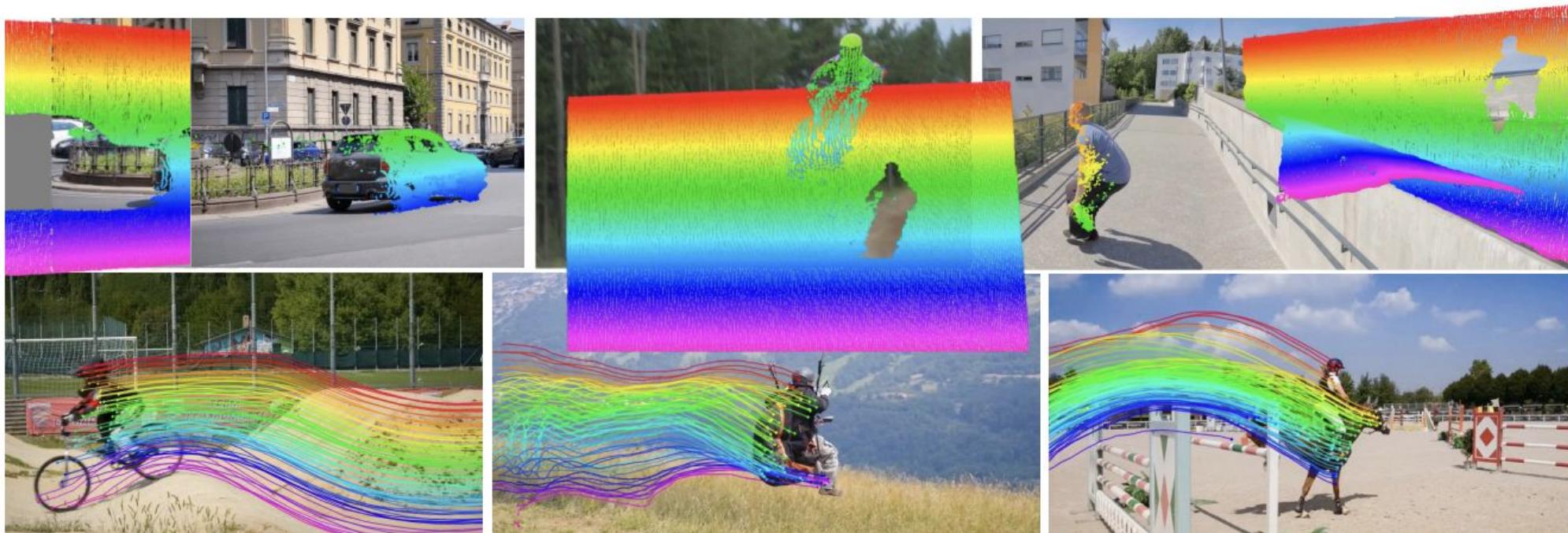
36



<http://cat3d.github.io/>

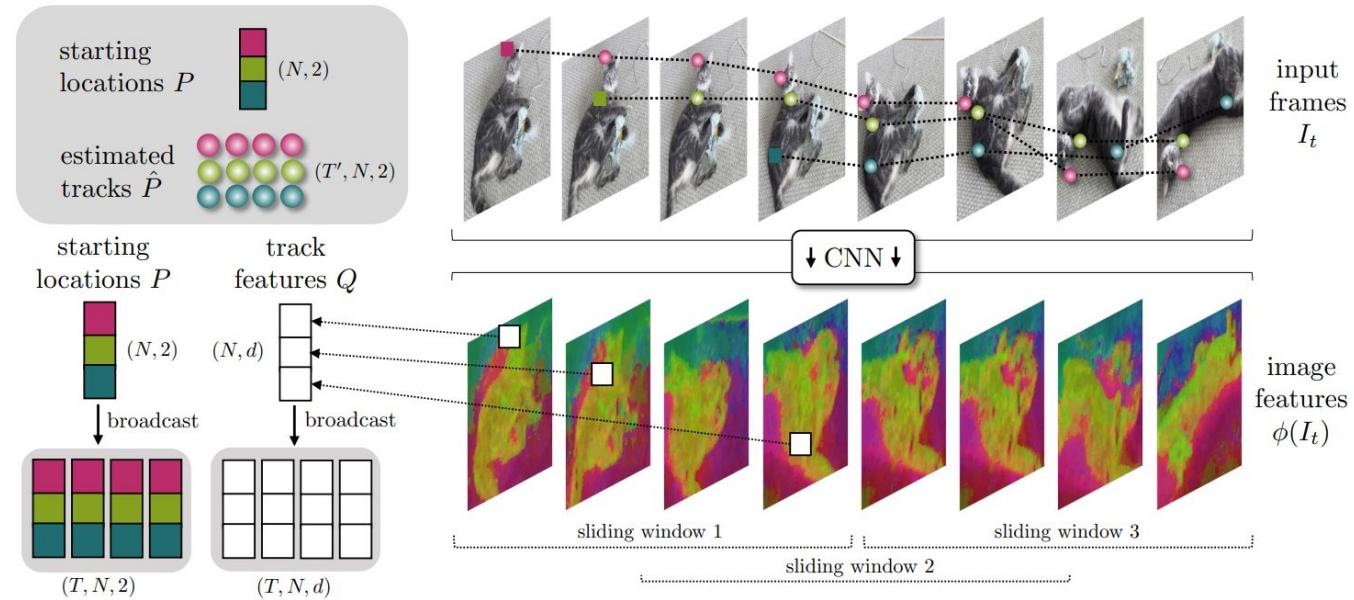
Модель Motion tracking: CoTracker

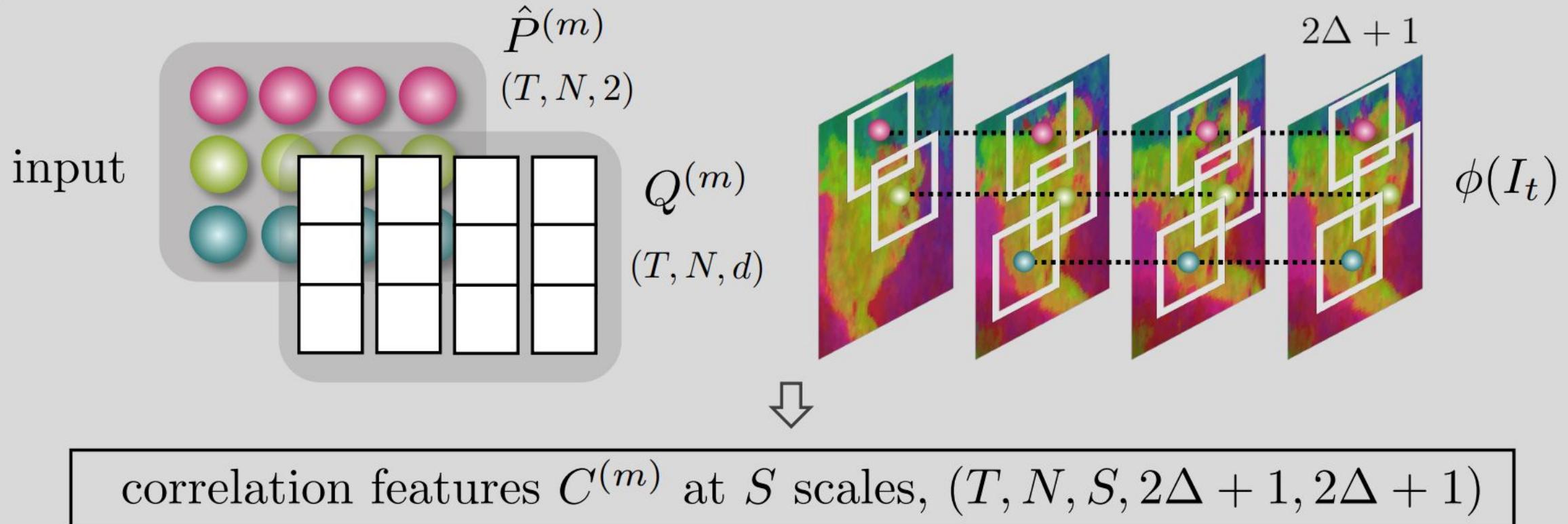
CoTracker — это новая модель на основе трансформеров для отслеживания плотных точек в кадре по всей последовательности видео. В отличие от большинства современных подходов, которые отслеживают точки независимо, CoTracker отслеживает точки совместно, что приводит к значительному улучшению точности и устойчивости отслеживания.



Основные понятия:

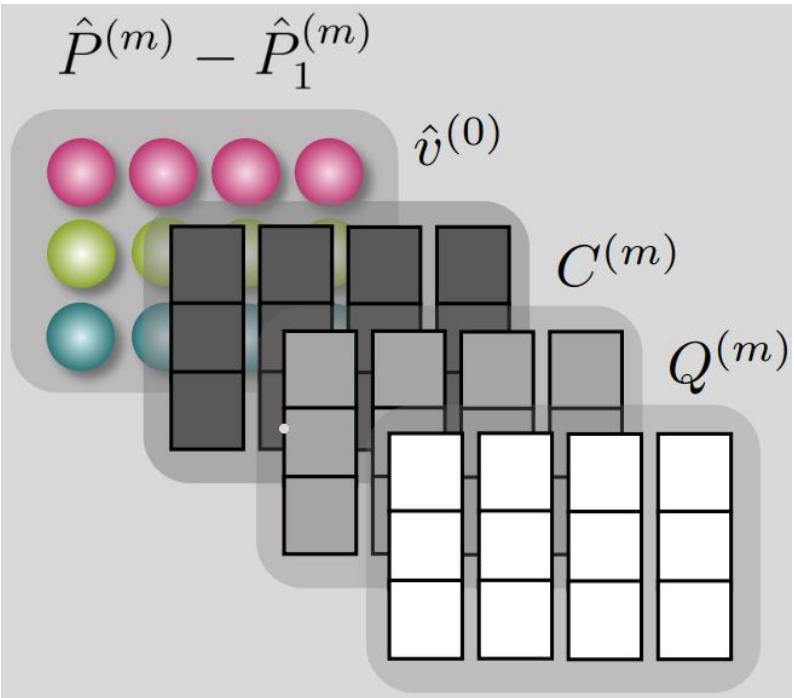
- Image Features
- Track Features
- Correlation Features
- Tokens
- Iterated transformer applications



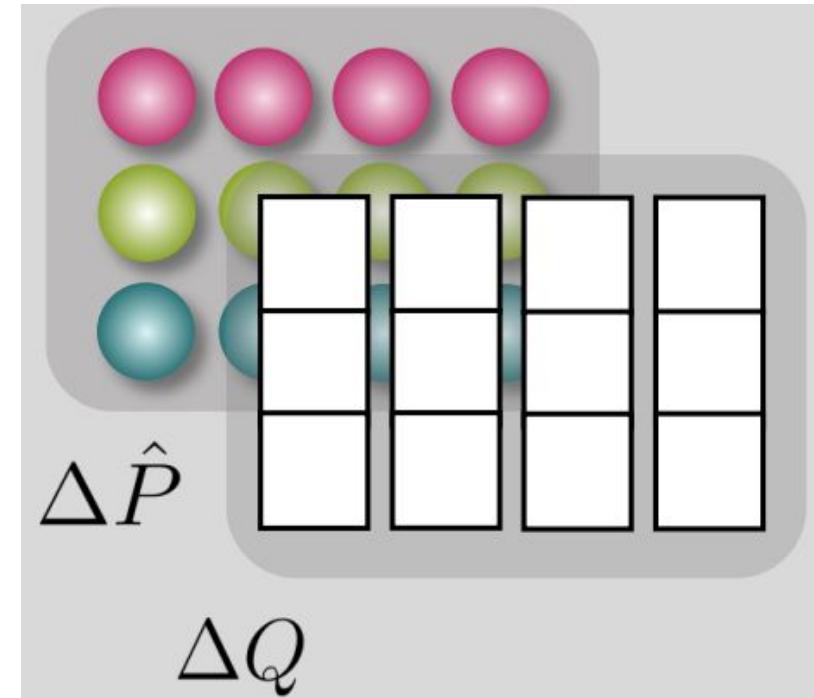


$$[C_t^i]_{s\delta} = \langle Q_t^i, \phi_s(I_t)[\hat{P}_t^i/ks + \delta] \rangle$$

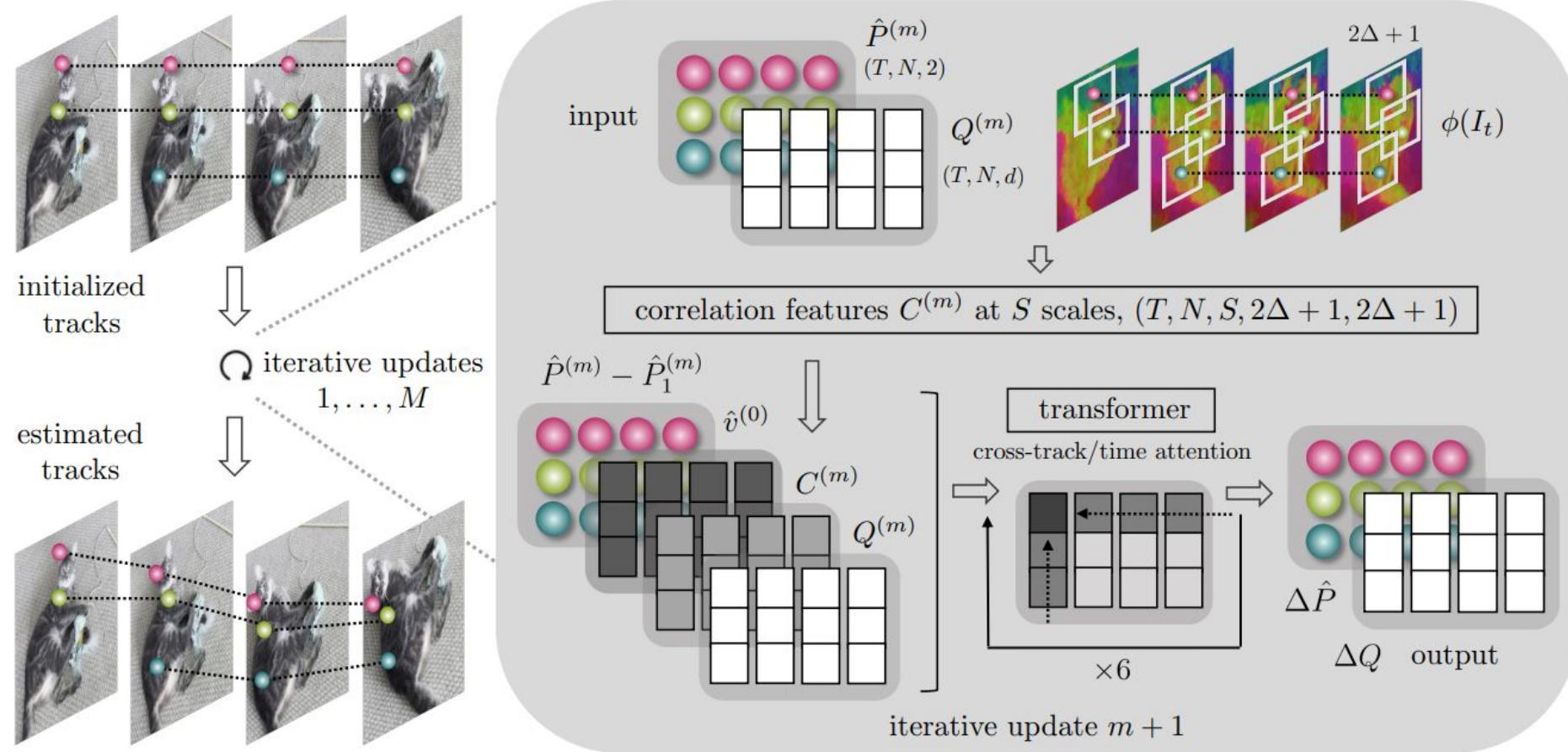
Input token



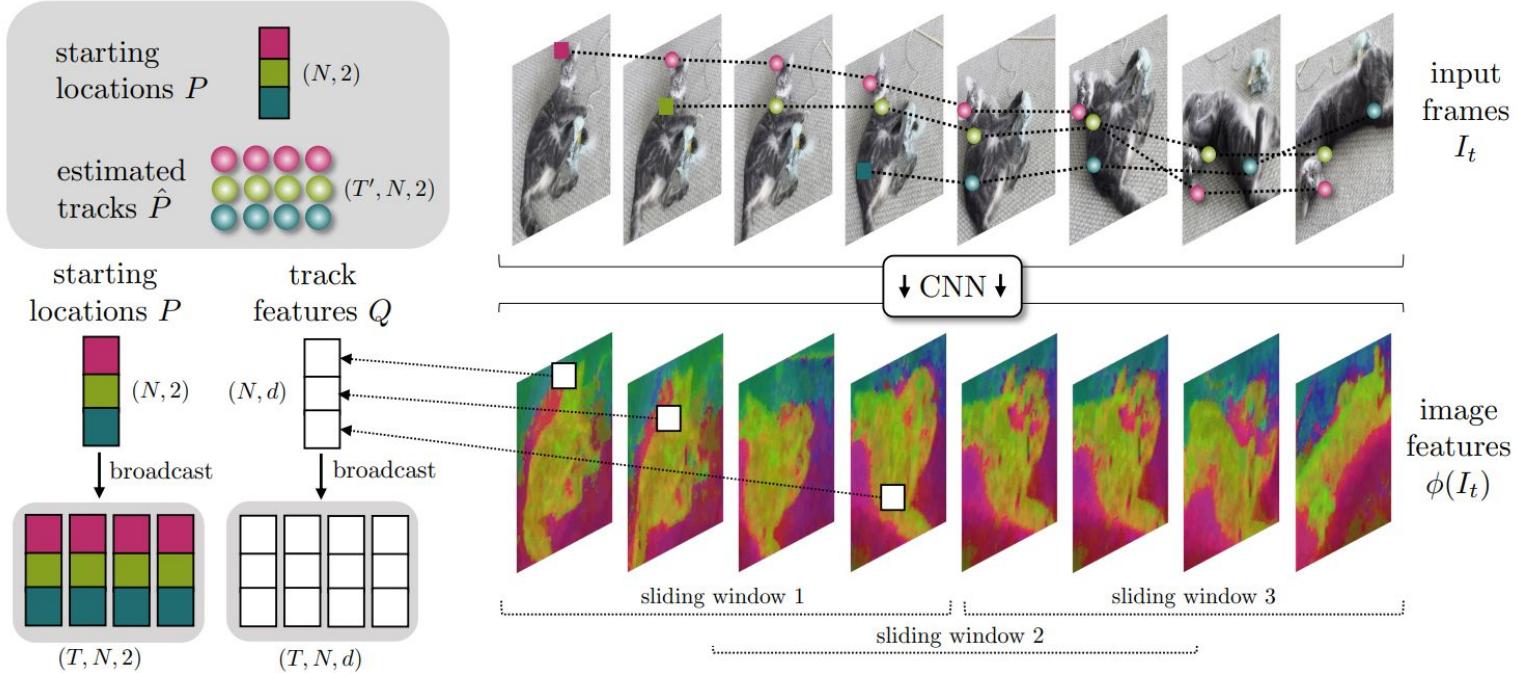
Output token



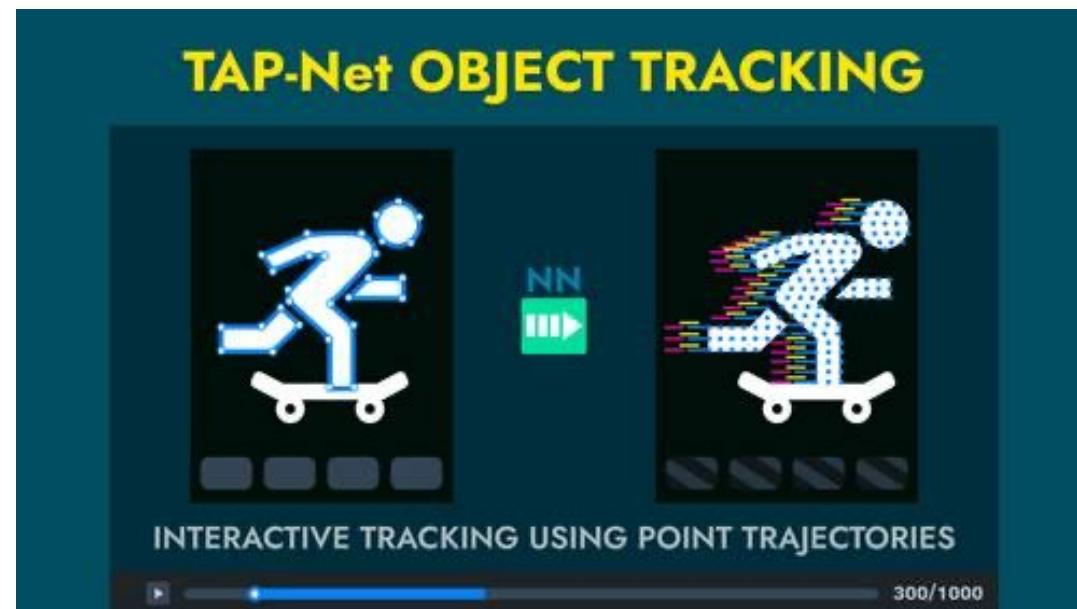
$$G_t^i = (\hat{P}_t^i - \hat{P}_1^i, \hat{v}_t^i, Q_t^i, C_t^i, \eta(\hat{P}_t^i - \hat{P}_1^i)) + \eta'(\hat{P}_1^i) + \eta'(t)$$



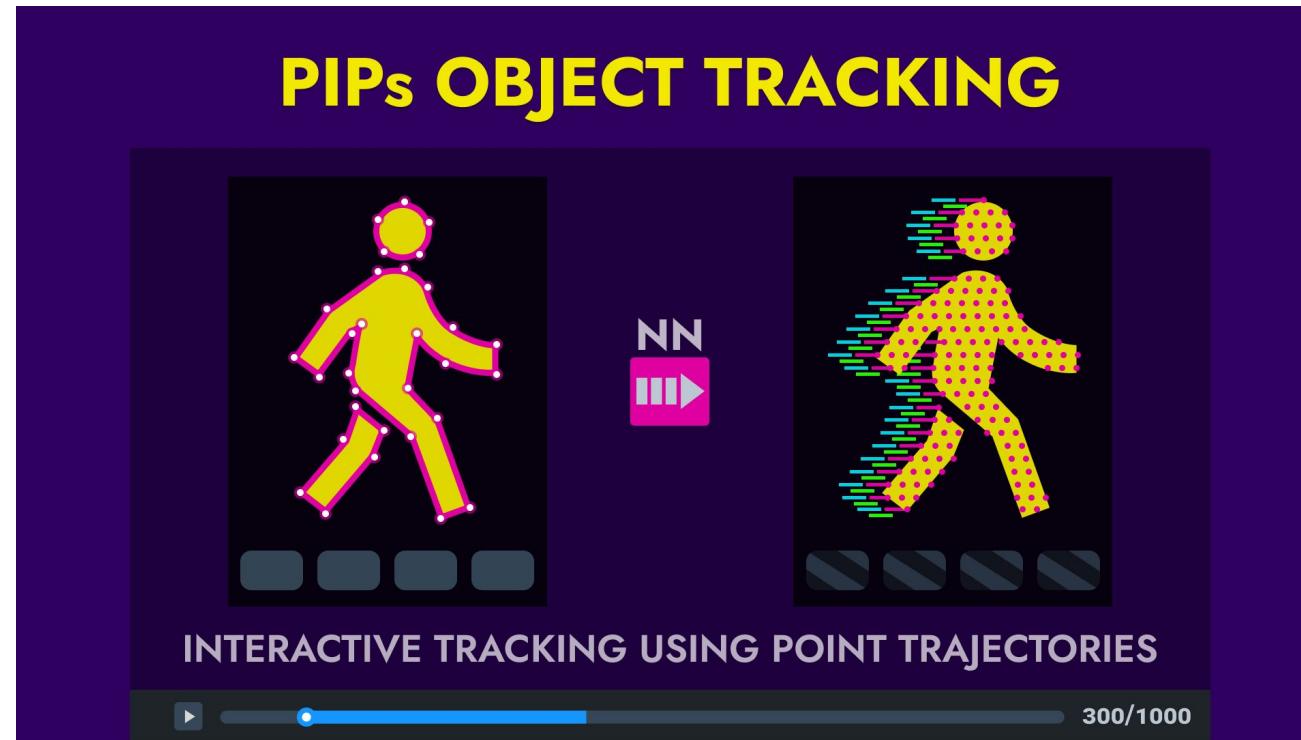
Sliding windows



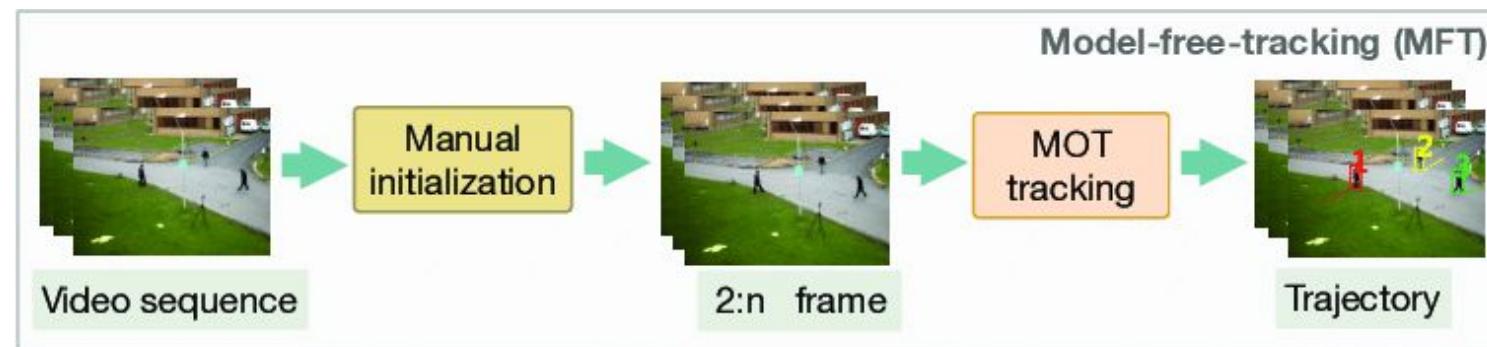
TAP-Net (Topology-Aware Pose Network) — это трекер, разработанный для оценки позы в видео, который сочетает в себе топологически-информированные сети и методики обработки временной информации. Основная цель TAP-Net заключается в улучшении точности и стабильности оценки позы человека в последовательности кадров видео.



PIPs (Plug-and-Play Iterative Pose Tracking) — это передовой трекер для отслеживания позы человека в видео. Он отличается своей архитектурой и подходом к задаче отслеживания, что позволяет ему эффективно обрабатывать длинные видеопоследовательности и обеспечивать высокую точность и стабильность.



MFT (Multi-Frame Tracking) — это система отслеживания позы человека в видео, разработанная для эффективной работы с последовательностями кадров, чтобы улучшить точность и стабильность предсказаний. MFT использует информацию из нескольких кадров одновременно для контекстуального анализа и более точного определения позы.



Полезно ли совместное отслеживание?

В Табл. мы демонстрируем важность совместного отслеживания точек, что является основной мотивацией CoTracker. Это достигается за счет удаления внимания между треками из трекера, таким образом полностью игнорируя корреляцию между треками.

	Attention		Points		DAVIS First		
	time	joint	target	support	AJ \uparrow	$\delta_{\text{avg}} \uparrow$	OA \uparrow
(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

Table 3. **Importance of joint tracking.** We compare using time and cross-track attention, tracking single or multiple target points, and using additional support points.

Как CoTracker сравнивается с предыдущими работами?

в Табл. сравниваем CoTracker с передовыми трекерами на трех наборах данных для бенчмарков, улучшая все метрики с значительным отрывом во всех случаях.

Method	Training data	DAVIS First			DAVIS Strided			Dynamic Replica		
		AJ \uparrow	$\delta_{\text{avg}}^{\text{vis}} \uparrow$	OA \uparrow	AJ \uparrow	$\delta_{\text{avg}}^{\text{vis}} \uparrow$	OA \uparrow	$\delta_{\text{avg}} \uparrow$	$\delta_{\text{avg}}^{\text{vis}} \uparrow$	$\delta_{\text{avg}}^{\text{occ}} \uparrow$
TAP-Net [10]	TAP-Vid-Kubric	33.0	48.6	78.8	38.4	53.1	82.3	45.5	53.3	20.0
OmniMotion [46]	—	—	—	—	51.7	67.5	85.3	—	—	—
PIPs [15]	FlyingThings++	42.2	64.8	77.7	52.4	70.0	83.6	41.0	47.1	21.0
MFT [31]	Kubric + others	47.3	66.8	77.8	56.1	70.8	86.9	—	—	—
PIPs++ [52]	PointOdyssey	—	69.1	—	—	73.7	—	55.5	64.0	28.5
TAPIR [11]	TAP-Vid-Kubric	56.2	70.0	86.5	61.3	73.6	88.8	56.8	66.1	27.2
CoTracker (Ours)	TAP-Vid-Kubric	62.2	75.7	89.3	65.9	79.4	89.9	61.6	68.9	37.6

Помогают ли виртуальные треки CoTracker масштабироваться?

В Таб. оцениваются преимущества использования виртуальных треков. Для фиксированного объема памяти (80 ГБ) использование виртуальных треков позволяет нам отслеживать в 7.4 раза больше точек, чем без них; фактически, мы можем отслеживать всю сетку 263×263 , что почти равно плотному отслеживанию для входного разрешения видео. Количество виртуальных треков не влияет на масштабируемость, но влияет на производительность

Unrolled Training	DAVIS First		
	AJ \uparrow	$\delta_{\text{avg}} \uparrow$	OA \uparrow
✗	44.6	60.5	75.3
✓	62.2	75.7	89.3



CoTracker Вывод

49

Хотя CoTracker лучше других трекеров, он все же делает ошибки в отслеживании, которых легко мог бы избежать человек. Еще одно ограничение заключается в том, что он все еще обрабатывает относительно короткие окна точек за раз и поэтому может потерпеть неудачу для точек, которые остаются скрытыми на протяжении нескольких скользящих окон. По той же причине он может не отслеживать точки в течение тысяч кадров. Хотя совместное отслеживание и развернутая тренировка смягчают это, офлайн-приложения трекера выиграли бы от учета всех кадров видео сразу.