Методы сопровождения объектов

Леонид Бейненсон
Летняя школа 2016, Itseez/Intel/ННГУ
Июль 2016

План:

- Общее описание методов сопровождения объектов
- Оптический поток и алгоритм Лукаса-Канаде
- Алгоритм сопровождения объектов “Median flow”

Сопровождение объектов

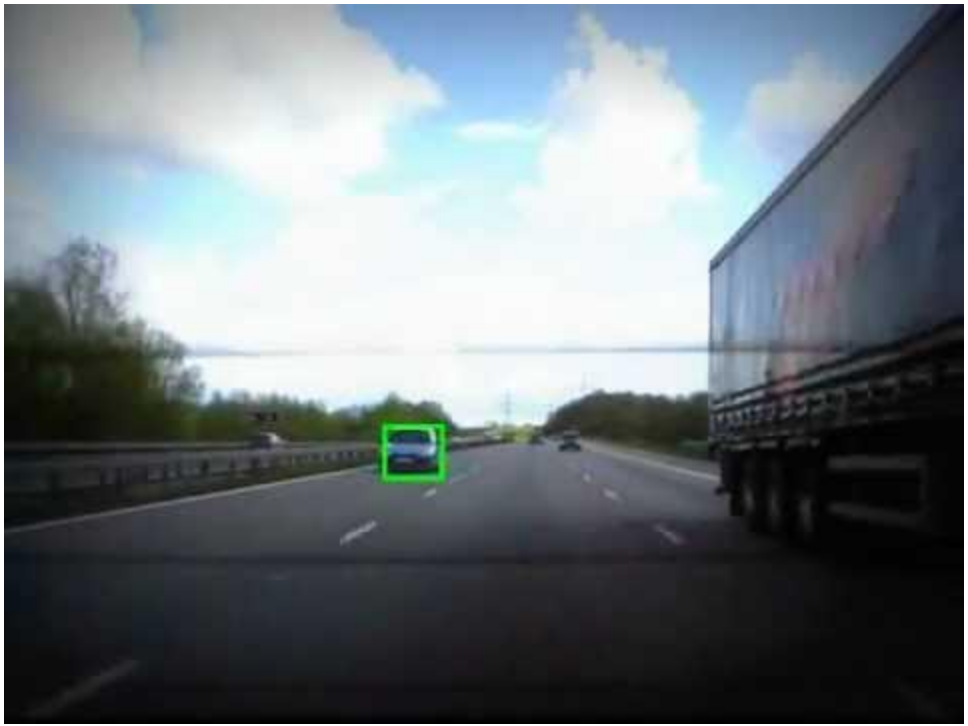
Что такое “трекинг”

Трекер -- алгоритм сопровождения объекта:

- Трекеру передается фрейм из видеопоследовательности (камера/видео)
- “Некто” (детектор или человек) выделяет объект(ы) на фрейме и передает его трекеру
- Трекеру подаются следующие фреймы из видеопоследовательности
- Трекер должен сказать где объект(ы) на этих фреймах

Что такое “трекинг”

Что такое “трекинг”



0:44 - 1:00

Популярные сценарии для трекинга

- Видеонаблюдение (камеры безопасности)
  на улицах или в помещениях
- Автомобильный
  объекты -- другие машины и пешеходы (оценка траектории)
- Видеокамера или камера на телефоне
  автофокусировка

Почему трекинг -- сложная задача

- Объекты не жесткие, изменяют форму
  (могут махать руками и ногами)
- Объекты с разных точек зрения могут выглядеть по разному
  (могут крутиться)
- Объект может зайти за преграду, или вообще уйти из кадра
  (а потом вернуться)
- Изменение размеров (особенно при слежении за автомобилями)

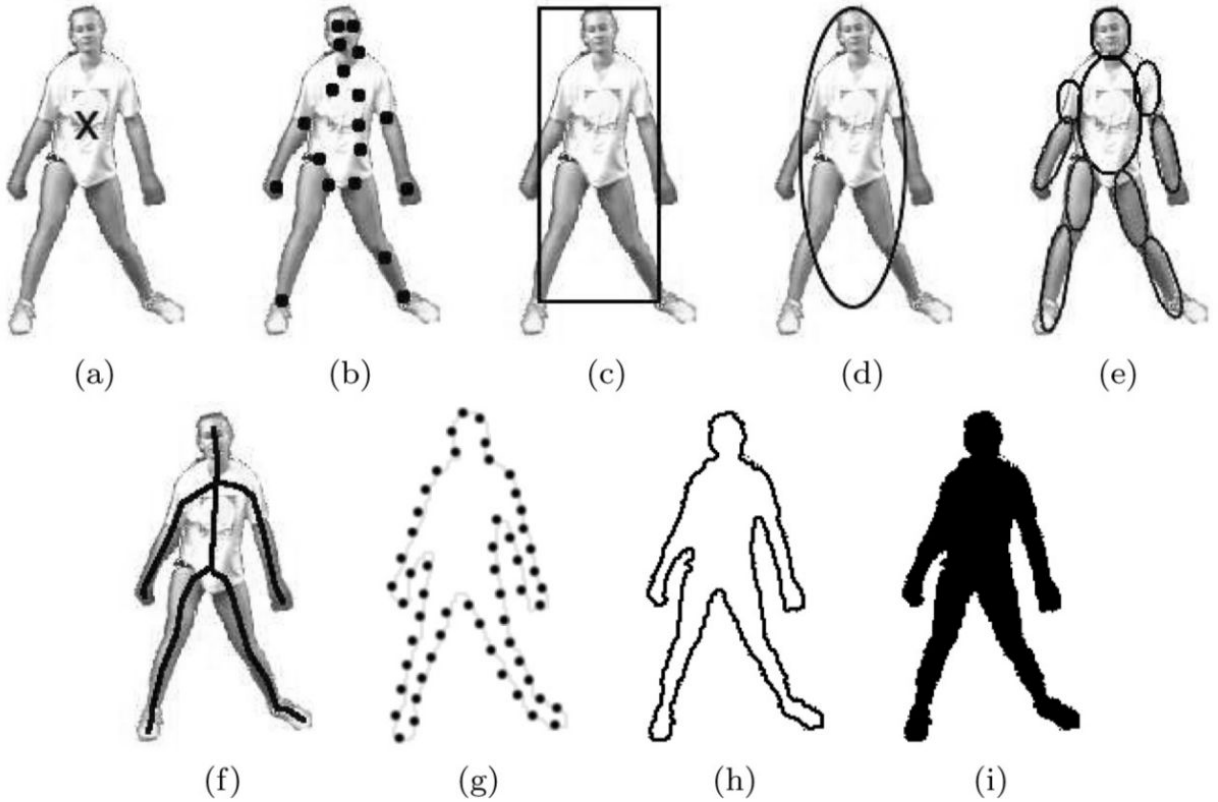
Популярные сценарии для трекинга

- Видеонаблюдение (камеры безопасности)
  на улицах или в помещениях
- Автомобильный
  объекты -- другие машины и пешеходы (оценка траектории)
- Видеокамера или камера на телефоне
  автофокусировка

Идеальный алгоритм, который бы подходил для всех этих случаев?

НЕ СУЩЕСТВУЕТ

Как представлять объект



Какие характерные черты объекта выбирать

- Распределение цветов (гистограммы)
- Характерные кусочки
- Ребра
- Другие характерные черты -- как в детекторах

Плюс можно следить за частями объекта (отдельно за руками, ногами, головой), а потом объединять их в целое.

Виды трекеров

Идеологически трекеры подразделяются на:

- Генеративные
- Дискриминативные
- Комбинированные

Следует иметь в виду, что разделение это не формальное, и некоторые трекеры трудно отнести к той или иной группе

Прямой подход (генеративные трекеры)

1. Задетектировать характерные черты объекта,
2. На следующем фрейме - искать части изображения с теми же характерными чертами
   (неподалеку от предыдущего положения объекта)
3. Найти те части, которые наиболее похожи
   (но недалеко от предыдущего положения)
4. Определить новое положение объекта

Простой пример генеративного трекера

- Модель объекта -- прямоугольник вокруг него
- Характерная черта объекта -- значение всех пикселов в прямоугольнике
- Сравнение кандидата с объектом -- SAD (Sum of Absolute Differences)
  - Выравниваем два прямоугольника по центру и по размеру
  - Вычитаем один из другого
  - Складываем модули значений по всем пикселям разности
  - Результат (сумма) -- мера “несхожести” кандидатов

- Чтобы найти следующее положение объекта -- перебираем в окрестности объекта все прямоугольники с размером +/- 10% от начального
  Тот, который лучше всего подходит -- следующий.

Сравнение характерных черт объекта и фона (дискриминативные трекеры)

1. Задетектировать характерные черты объекта
2. Задетектировать характерные черты фона
3. На следующем фрейме - искать части изображения так, чтобы они были как можно более похожи на объект и, одновременно, непохожи на фон (неподалеку от предыдущего положения объекта)
4. Найти из них те части, которые лучше всего подходят по критерию “объект, а не фон”
5. Определить новое положение объекта

Простой пример дискриминативного трекера

- Модель объекта -- прямоугольник вокруг него
- На первом кадре обучаем детектор (например, SVM+HOG)
  - В качестве положительного примера -- сам объект
  - В качестве отрицательных примеров -- прямоугольники рядом с объектом (могут пересекать объект, но не сильно)
- На каждом следующем фрейме запускаем получившийся детектор в окрестности предыдущего положения объекта
- Новое положение -- то, в котором получили наибольший отклик детектора

Простой пример дискриминативного трекера

позитив

негативы

Простой пример дискриминативного трекера

- Модель объекта -- прямоугольник вокруг него
- На первом кадре обучаем детектор (например, SVM+HOG)
  - В качестве положительного примера -- сам объект
  - В качестве отрицательных примеров -- прямоугольники рядом с объектом (могут пересекать объект, но не сильно)
- На каждом следующем фрейме запускаем получившийся детектор в окрестности предыдущего положения объекта
- Новое положение -- то, в котором получили наибольший отклик детектора

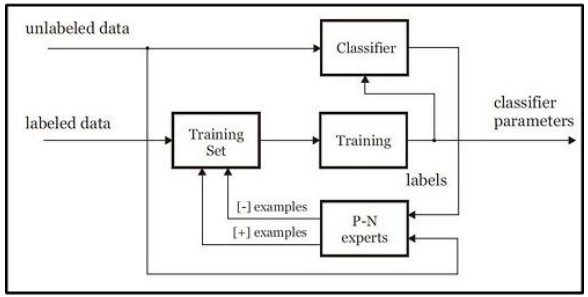
Будет ли это работать?

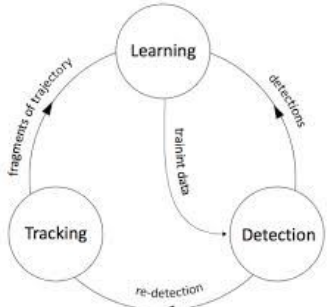
Иногда будет!
Трекер KCF (Kernelized Correlation Filter) основан на этой идее, плюс на использовании преобразования Фурье, чтобы подсчитывать отклик за одну свертку

Комбинированный трекер

- Содержит в себе несколько трекеров (например, один генеративный и один дискриминативный)
- Когда получает новый фрейм, запускает эти трекеры для отслеживаемого объекта
- “Комбинирует” результаты от трекеров с помощью хитрых эвристик

Пример -- TLD трекер
(Tracking-Learning-Detection, 2011)



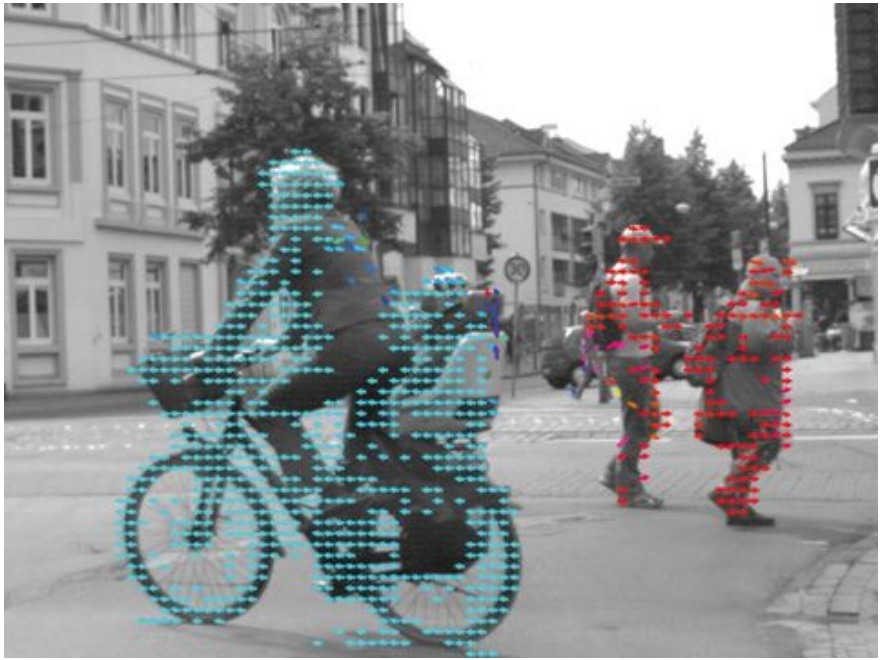


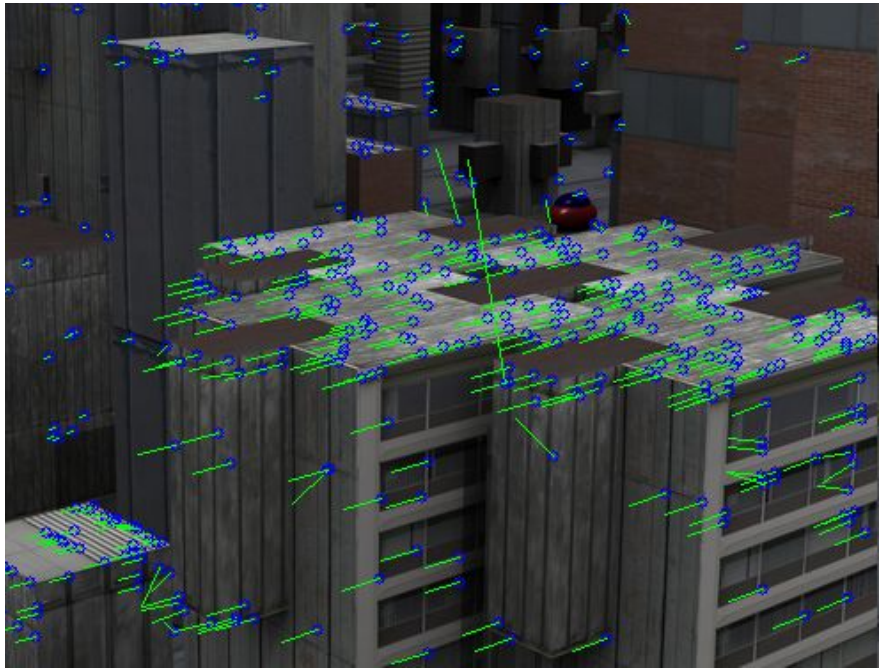
Оптический поток

Оптический поток

Оптический поток (плотный)

Оптический поток (разреженный)

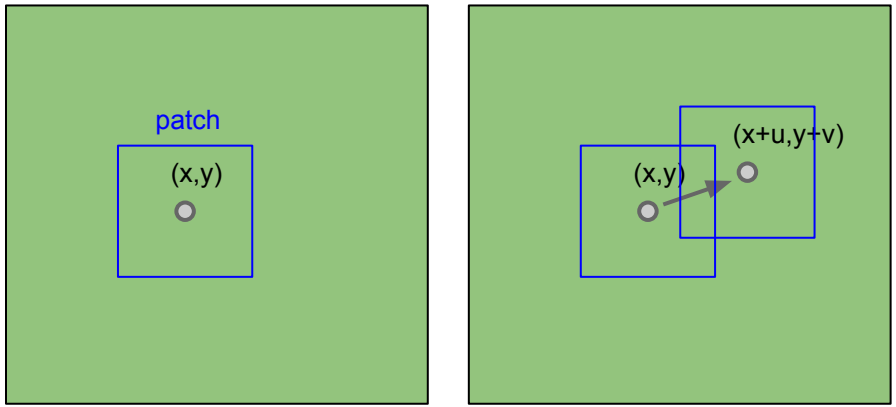




Разреженный оптический поток с точки зрения трекинга

С точки зрения трекинга разреженный оптический поток -- это задача слежения за объектом, состоящим из точки + небольшая окрестность вокруг нее (например, квадрат 5x5).

Модель объекта -- центральная точка, размер неизменен, нужно предсказание движения -- вектора (*u, v*) -- на 1 фрейм вперед

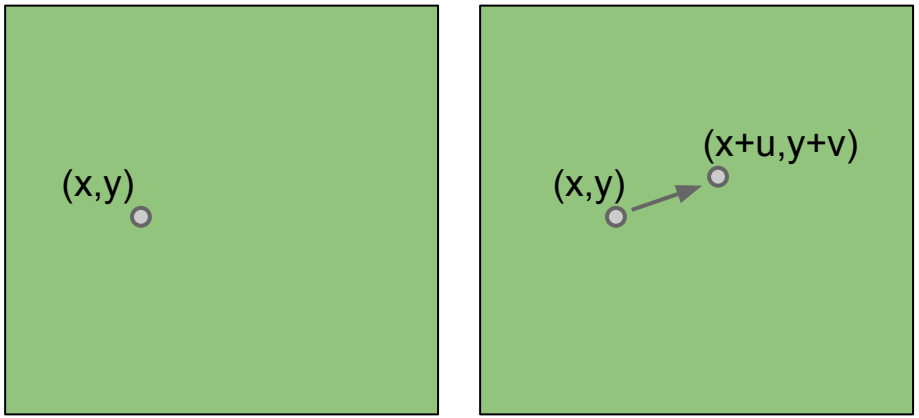


$(u,v) = ?$

Уравнение оптического потока

Пусть $I(x, y, t)$ -- функция, задающая значение пиксела (x, y) на фрейме t.

И пусть у нас некоторая точка имеет свою траекторию $(x(t), y(t))$



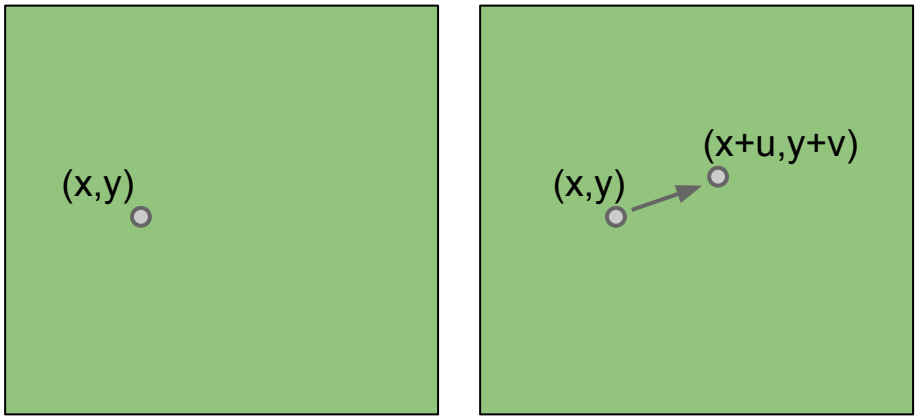
Предположим, что яркость точки не меняется:

$$\frac{dI(x(t), y(t), t)}{dt} = 0$$

Уравнение оптического потока

Пусть $I(x, y, t)$ -- функция, задающая значение пиксела (x, y) на фрейме t.

И пусть у нас некоторая точка имеет свою траекторию $(x(t), y(t))$



Предположим, что яркость точки не меняется:

$$\frac{dI(x(t), y(t), t)}{dt} = 0$$

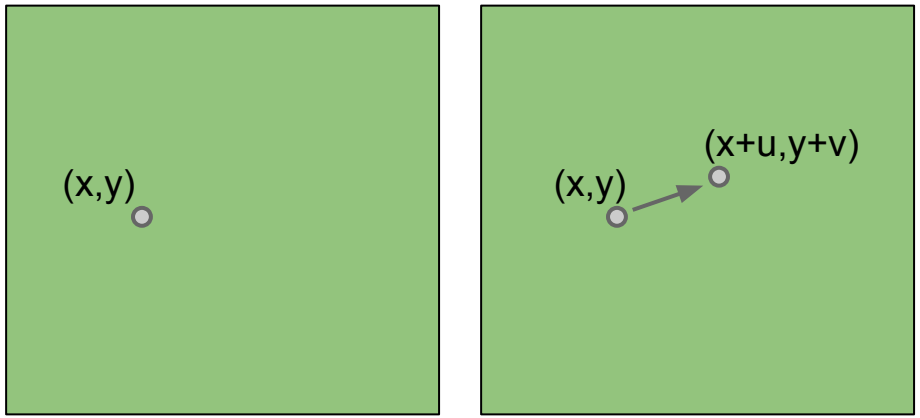
Уравнение оптического потока:

$$I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t = 0$$

Уравнение оптического потока

Пусть $I(x, y, t)$ -- функция, задающая значение пиксела (x, y) на фрейме t.

И пусть у нас некоторая точка имеет свою траекторию $(x(t), y(t))$



I(x,y,t)

I(x,y,t+dt)

Предположим, что яркость точки не меняется:

$$\frac{dI(x(t), y(t), t)}{dt} = 0$$

Уравнение оптического потока:

$$I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t = 0$$

или u=dx, v=dy, dt=1 (t -- номер кадра):

$$I_x u + I_y v = -I_t$$

Уравнение оптического потока

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

$$I(x + u, y + v, t + 1) \approx I + I_x u + I_y v + I_t$$

-- разложим в ряд Тейлора, производные выше первой отбросим

$$I_x u + I_y v = -I_t$$

-- уравнение оптического потока для точки (x,y)

Уравнение оптического потока

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

$$I(x + u, y + v, t + 1) \approx I + I_x u + I_y v + I_t$$

-- разложим в ряд Тейлора, производные выше первой отбросим

$$I_x u + I_y v = -I_t$$

-- уравнение оптического потока для точки (x,y)

Интересный факт:

Следующий фрейм используется только при вычислении I_t

При программировании аппроксимируем производные у функции $I(x, y, t)$ численно

То есть, I_x и I_y аппроксимируем по трем точкам, а I_t -- как разность

$I(x, y, t+1) - I(x, y, t)$

Метод Лукаса-Канаде

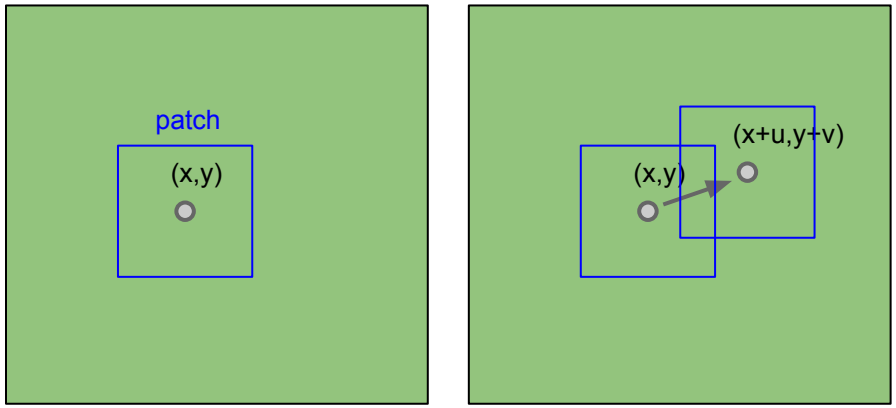
$I_x u + I_y v = -I_t$ -- уравнение оптического потока для точки (x,y)

Две неизвестных на точку, одно уравнение. Что делать?

Метод Лукаса-Канаде

$$I_x u + I_y v = -I_t$$ -- уравнение оптического потока для точки (x,y)

Две неизвестных на точку, одно уравнение. Что делать? Рассмотреть движение точки с окрестностью.



$$\begin{pmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x_1, y_1) \\ \dots \\ -I_t(x_n, y_n) \end{pmatrix}$$

-- уравнение оптического потока для окрестности точки (x,y)

Метод Лукаса-Канаде

$$\begin{pmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x_1, y_1) \\ \dots \\ -I_t(x_n, y_n) \end{pmatrix}$$

-- уравнение оптического потока для окрестности точки (x,y)

$$A \begin{pmatrix} u \\ v \end{pmatrix} = b$$

два столбца
n строк

один столбец
две строки

один столбец
n строк

Информация из следующего фрейма используется только в правой части!
-- при вычислении столбца b

Метод Лукаса-Канаде

$$\begin{pmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x_1, y_1) \\ \dots \\ -I_t(x_n, y_n) \end{pmatrix}$$

-- уравнение оптического потока для окрестности точки (x,y)

$$A \begin{pmatrix} u \\ v \end{pmatrix} = b$$

В случае окрестности 5x5 имеем 25 уравнений и 2 неизвестные -- система переопределена

$$\left\| A \begin{pmatrix} u \\ v \end{pmatrix} - b \right\| \longrightarrow min$$

-- воспользуемся методом наименьших квадратов

Метод Лукаса-Канаде

$$\begin{pmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x_1, y_1) \\ \dots \\ -I_t(x_n, y_n) \end{pmatrix}$$

-- уравнение оптического потока для окрестности точки (x,y)

$$A \begin{pmatrix} u \\ v \end{pmatrix} = b$$

В случае окрестности 5x5 имеем 25 уравнений и 2 неизвестные -- система переопределена

$$\left\| A \begin{pmatrix} u \\ v \end{pmatrix} - b \right\| \longrightarrow min$$

-- воспользуемся методом наименьших квадратов

$$A^T A \begin{pmatrix} u \\ v \end{pmatrix} = A^T b$$

-- система 2x2, дает решение МНК

Метод Лукаса-Канаде

$$\begin{pmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x_1, y_1) \\ \dots \\ -I_t(x_n, y_n) \end{pmatrix}$$

-- уравнение оптического потока для окрестности точки (x,y)

$$A \begin{pmatrix} u \\ v \end{pmatrix} = b$$

В случае окрестности 5x5 имеем 25 уравнений и 2 неизвестные -- система переопределена

$$\left\| A \begin{pmatrix} u \\ v \end{pmatrix} - b \right\| \longrightarrow min$$

-- воспользуемся методом наименьших квадратов

$$A^T A \begin{pmatrix} u \\ v \end{pmatrix} = A^T b$$

-- система 2x2, дает решение МНК

$$\begin{pmatrix} u \\ v \end{pmatrix} = \left(A^T A \right)^{-1} A^T b$$

Метод Лукаса-Канаде

- Метод ЛК применяется итеративно:
  - Вычислили поток
  - “Сдвинули” изображение
    …
  - Вычислили поток
  - “Сдвинули” изображение

  И так пока на сошлись или число итераций не превысит порог

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$

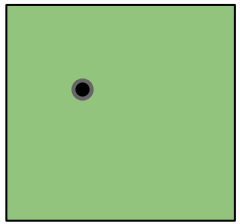
Метод Лукаса-Канаде

- Метод ЛК применяется итеративно:
  - Вычислили поток
  - “Сдвинули” изображение
    …
  - Вычислили поток
  - “Сдвинули” изображение

  И так пока на сошлись или число итераций не превысит порог

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$

итерация 1



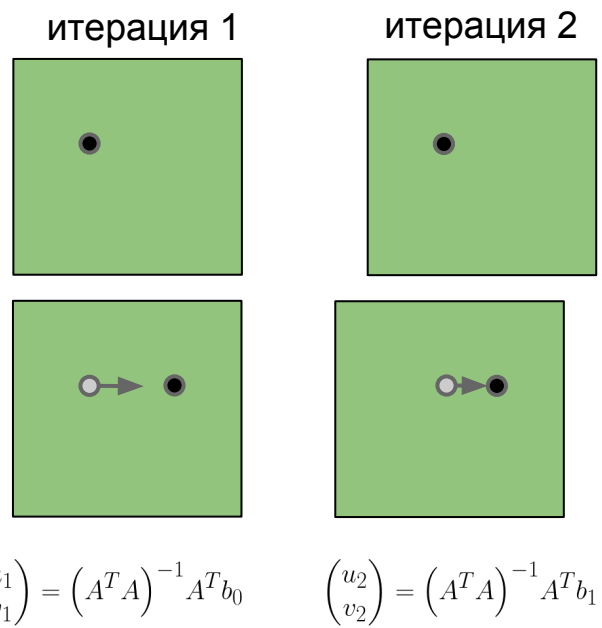
$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_0$$

Метод Лукаса-Канаде

- Метод ЛК применяется итеративно:
  - Вычислили поток
  - “Сдвинули” изображение
    …
  - Вычислили поток
  - “Сдвинули” изображение

  И так пока на сошлись или число итераций не превысит порог

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$

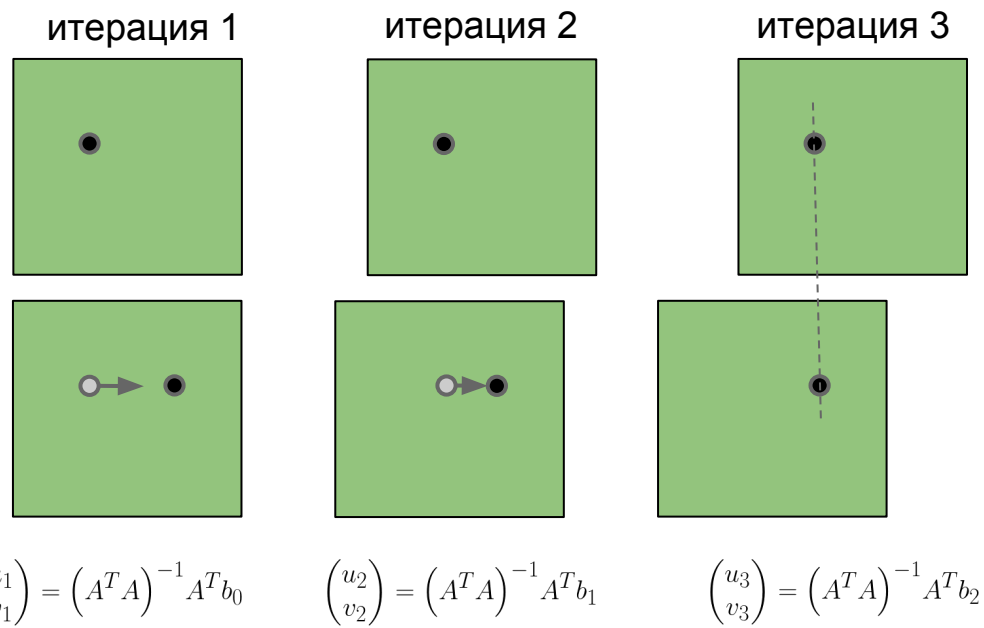


Метод Лукаса-Канаде

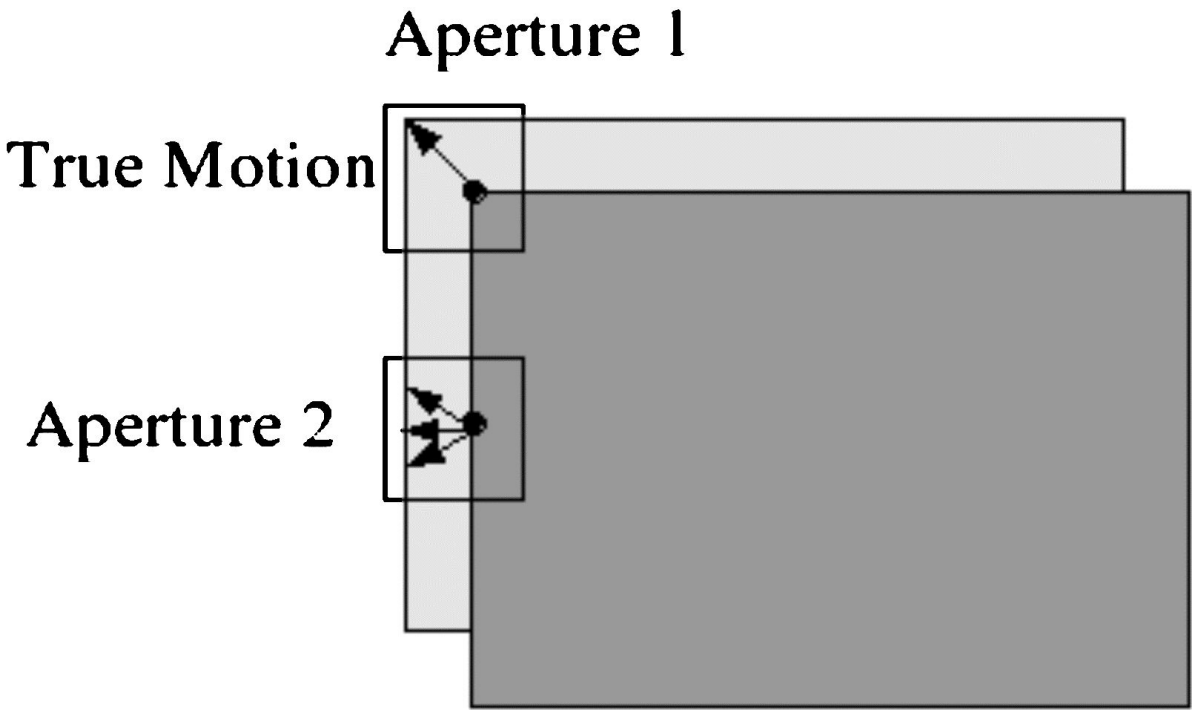
- Метод ЛК применяется итеративно:
  - Вычислили поток
  - “Сдвинули” изображение
    …
  - Вычислили поток
  - “Сдвинули” изображение

  И так пока на сошлись или число итераций не превысит порог

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$



Aperture problem



Не всегда по локальной окрестности можно понять куда сдвинулась точка

Aperture problem: метод ЛК

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$ -- формула, которая вычисляется на каждой итерации

$$M = A^T A = \sum_{x,y} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$ -- структурный тензор
(сумма производится по всем точкам окрестности 5x5)

Aperture problem выражается в том,
что матрица M оказывается вырожденной
или плохообусловленной:

$$cond(M) = \lambda_{2(max)} / \lambda_{1(min)}$$

Aperture problem: метод ЛК

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$ -- формула, которая вычисляется на каждой итерации

$$M = A^T A = \sum_{x,y} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$ -- структурный тензор
(сумма производится по всем точкам окрестности 5x5)

Как бороться с проблемой?

Aperture problem выражается в том,
что матрица M оказывается вырожденной
или плохообусловленной:

$$cond(M) = \lambda_{2(max)} / \lambda_{1(min)}$$

Aperture problem: метод ЛК

$$\begin{pmatrix} u_{i+1} \\ v_{i+1} \end{pmatrix} = \left(A^T A\right)^{-1} A^T b_i$$ -- формула, которая вычисляется на каждой итерации

$$M = A^T A = \sum_{x,y} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$ -- структурный тензор
(сумма производится по всем точкам окрестности 5x5)

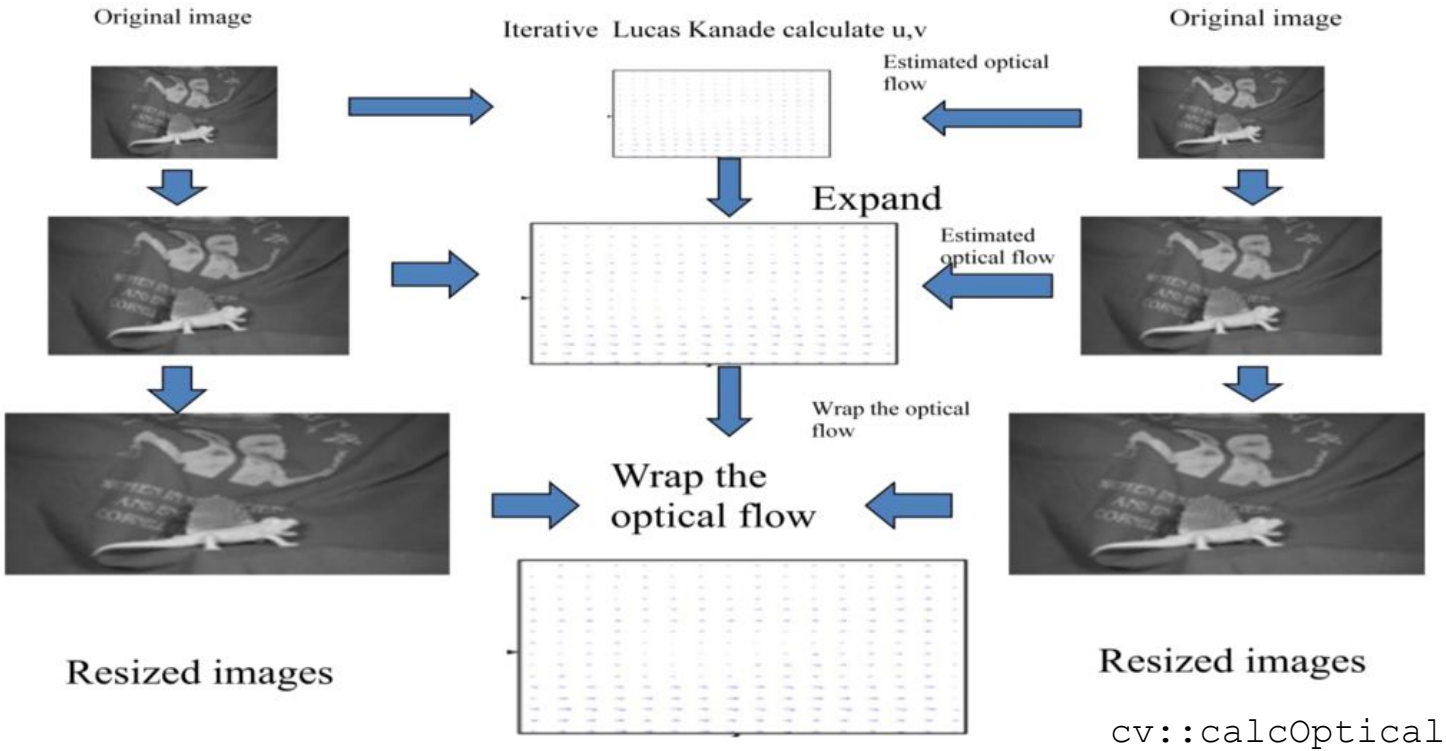
Aperture problem выражается в том,
что матрица M оказывается вырожденной
или плохообусловленной:

$$cond(M) = \lambda_{2(max)} / \lambda_{1(min)}$$

Как бороться с проблемой?

- Функция оптического потока должна возвращать “статус” каждой точки
- Считать только в хороших точках (угловых)
- Постобработка
(т.е. попытаться сделать приложение устойчивым к встречающимся иногда ошибкам оценки потока)

Пирамидальный метод ЛК



`cv::calcOpticalFlowPyrLK`

Трекер “Median flow”

Общее описание трекера “Median flow”

- Представление объекта -- прямоугольник
- Характерные черты -- “кусочки” объекта у которых ищутся их новые положения
- Основной метод для нахождения следующего положения каждого “кусочка” - оптический поток

Общее описание трекера “Median flow”

Шаг 1:

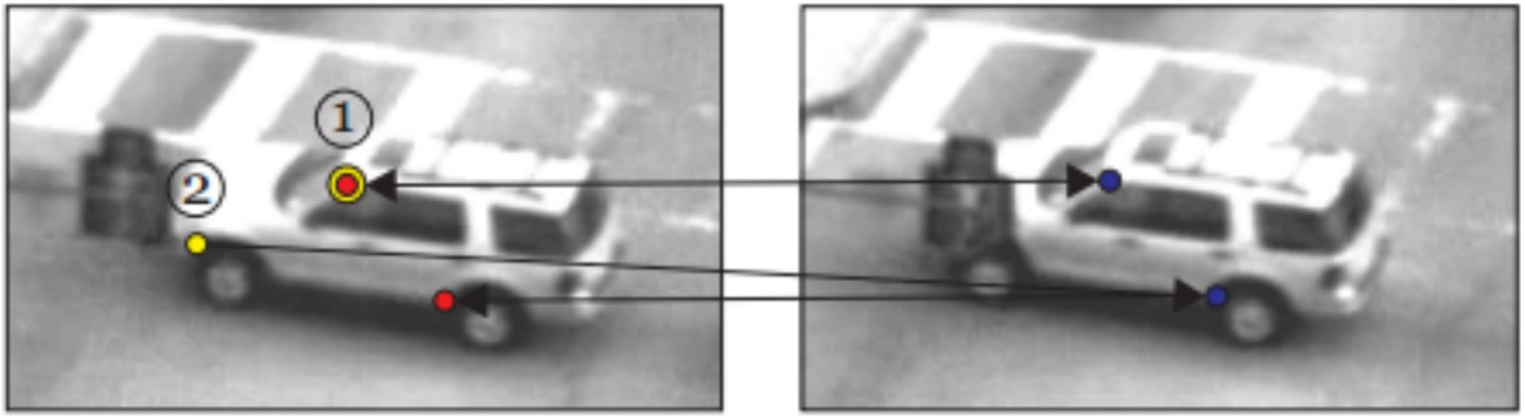
- Выбрать точки на объекте
  - Самое простое - квадратная сетка с шагом в 5 пикселей
  - Можно запустить один из детекторов “углов”

    например, `cv::goodFeaturesToTrack`

- Вычислить оптический поток для всех этих точек (например, `cv::calcOpticalFlowPyrLK`)
- Проверить “статус” для каждой точки
  - для точек, у которых возникли проблемы с плохо обусловленной матрицей, статус будет ‘0’, такие точки надо выбросить

Общее описание трекера “Median flow”

Стандартная проблема -- похожие элементы у объекта



Если у объекта есть несколько похожих частей, то оптический поток может ошибиться и выбрать “не ту” часть
-- такие соответствия нельзя использовать для оценки перемещения объекта

Общее описание трекера “Median flow”

Стандартная проблема -- похожие элементы у объекта

Для решения этой проблемы - **Шаг 2**:

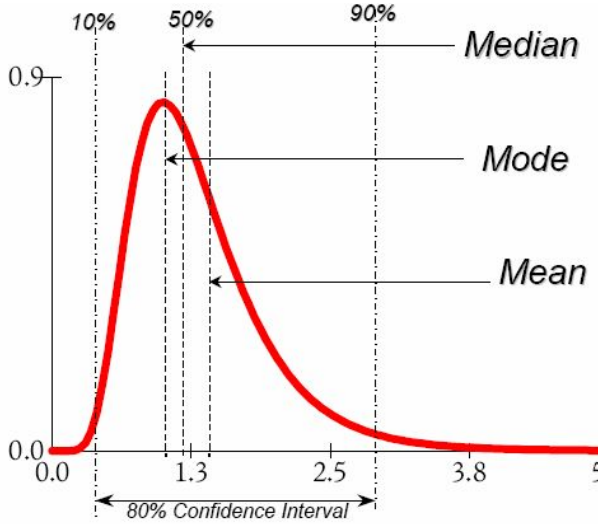
- Для всех найденных новых положений точек посчитать оптический поток обратно -- от второго фрейма к первому,
  используя найденные на Шаге 1 новые положения точек как исходные
- Выбросить все точки, у которых на этом оптическом потоке будет плохой статус
- Проверить, после обратного оптического потока получилась ли прежняя точка (плюс-минус 3 пиксела по осям x и y)
  Если точка ушла далеко -- тоже выбросить.

Общее описание трекера “Median flow”

После этого - оценка нового положения объекта.
Для этого потребуется понятие медианы.

Медиана для набора чисел
-- значение, которое будет ровно посередине,
если этот массив чисел отсортировать.

(Если количество чисел четное, то после
сортировки будет два серединных числа
- можно брать их среднее арифметическое.)

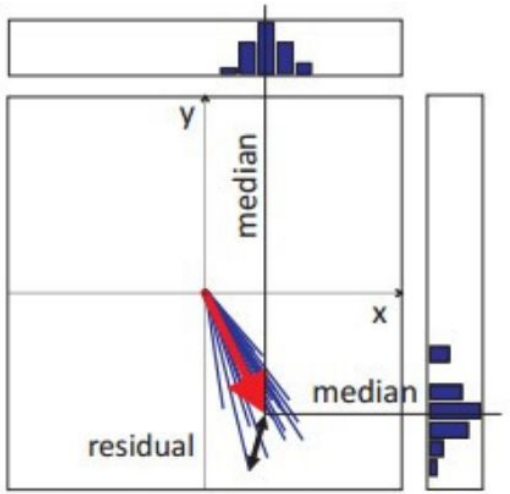


Общее описание трекера “Median flow”

После этого - оценка нового положения объекта

Шаг 3:

- Вычислить для всех оставшихся точек вектор смещения:
  `(новое_положение) - (старое_положение)`
- Вычислить медиану по оси x для всех этих векторов смещения
- Вычислить медиану по оси y для этих векторов
- Получившийся вектор (x, y) -- оценка того, как сместился центр объекта

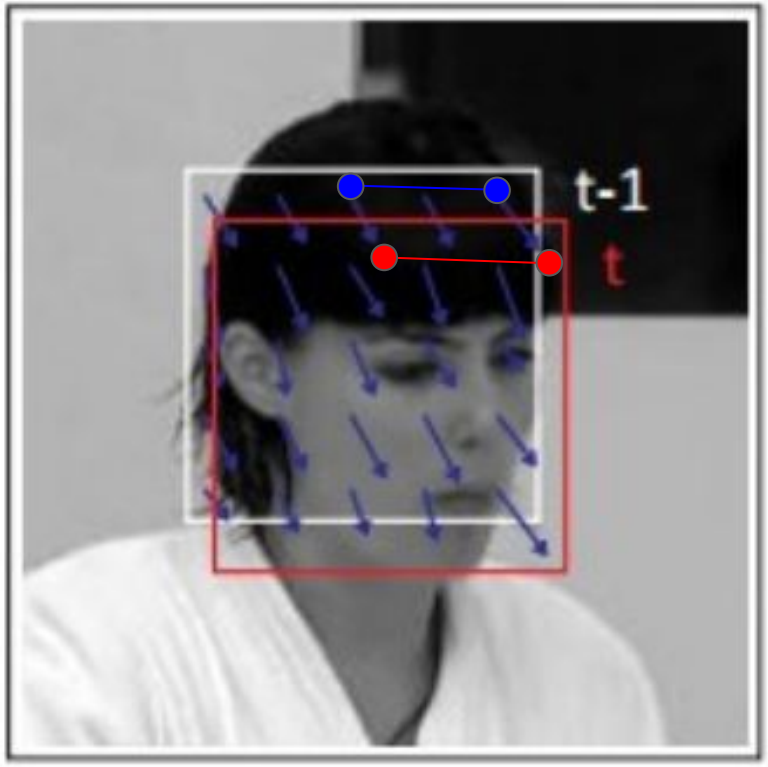


Общее описание трекера “Median flow”

После этого - оценка нового размера объекта

Шаг 4:

- Перебрать все пары из оставшихся точек
  Для каждой пары
  - Вычислить расстояние `D_old` между точками на п
  - Вычислить расстояние `D_new` между точками на в (после оптического потока)
  - Вычислить отношение `D_new / D_old`
  - Положить вычисленное отношение в массив `coe`
- Вычислить медиану всех чисел в массиве `co`
- Получившееся число -- оценка коэффициента размер объекта



Вопросы?

Дополнительные материалы

Как сравнивать трекеры

Система оценки качества для трекеров:

- Набор видеопоследовательностей
- Файлы разметки (“истинные данные”)
  - для каждого фрейма отмечены объекты с идентификаторами
- Файлы инициализации
  - для кажого фрейма отмечены объекты, которые “появились” на этом фрейме
- Скрипт, который получает файл разметки и файл с результатами трекера и вычисляет некоторые *критерии качества*
- (опционально) Скрипт “запускатель”, который берет много трекеров, запускает их все на всех видеопоследовательностях (подавая файлы инициализации), собирает результаты и вычисляет критерии качества

Как сравнивать трекеры

Критерий качества:

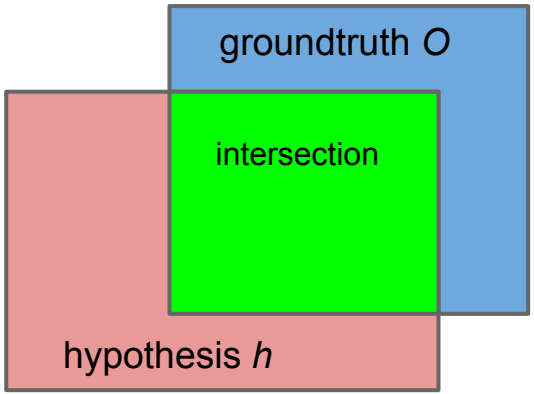
- Должен показывать, насколько трекер устойчив
  -- то есть,
  - как часто он теряет объект
  - как часто он “путает” два объекта

- Должен показывать, насколько трекер точно следит за объектом
  -- то есть, как далеко его представление о положение объекта отклоняется от истины

Как сравнивать трекеры

Классический простой критерий качества:

Выбираем отслеживаемый объект и для каждого фрейма считаем “соответствие” результатов трекера истинному положению:



$$corresp(o, h) = \frac{area(\quad)}{area(\quad)}$$

После этого считаем для этого объекта в каком проценте фреймов мы имеем это соответствие $corresp(o,h) > 0.5$

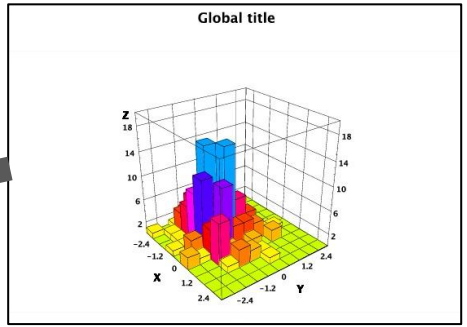
Пример генеративного трекера Mean-Shift by Comaniciu et al., 2000

- Объект представляется как эллипсоид (т.е. все пиксели внутри него)
- В качестве характерной черты объекта -- гистограмма RGB цветов 16x16x16
- Чтобы вычислить схожесть двух гистограмм используется коэффициент Бхаттачарьи

$$BC(p,q) := \sum_{u=1}^{N} \sqrt{p_u \cdot q_u}$$

- Следующее положение объекта вычисляется за несколько итераций алгоритмом Mean-Shift (похож на градиентный спуск по оптимизации *BC*)

Пример генеративного трекера Mean-Shift by Comaniciu et al., 2000



Пример дискриминативного трекера ASMS (Adaptive Scale Mean Shift)

В Mean Shift трекере оптимизируется коэффициент *BC*(*кандидат, модель*)

Если максимизировать не сам коэффициент *BC(кандидат, модель)*, а отношение

BC(*кандидат, модель*) / *BC*(*кандидат, фон вокруг модели*)

то получим уже дискриминативный трекер.

На этой идее основан ASMS
(лучший из real-time трекеров по версии VOT 2015)

Пример работы