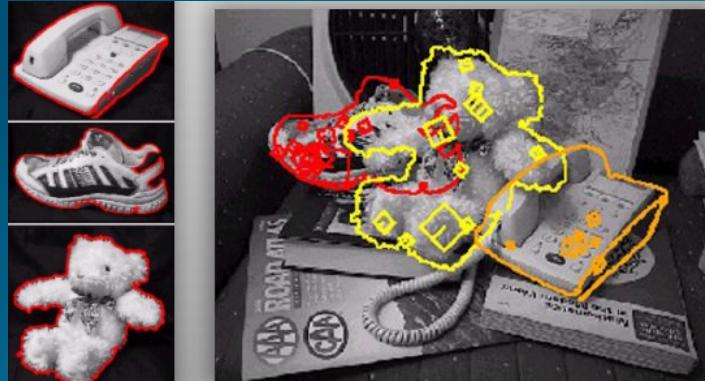


# ТЕХНИЧЕСКОЕ ЗРЕНИЕ: ОБНАРУЖЕНИЕ И РАСПОЗНАВАНИЕ ОБЪЕКТОВ



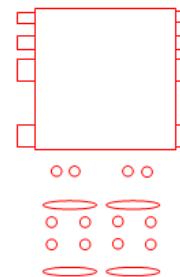
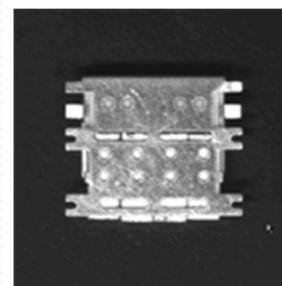
Бовырин Александр Владимирович

# Постановка задачи

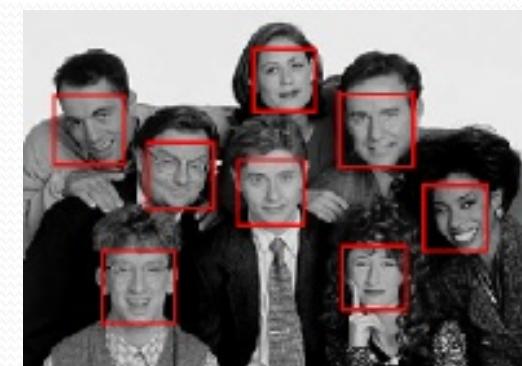
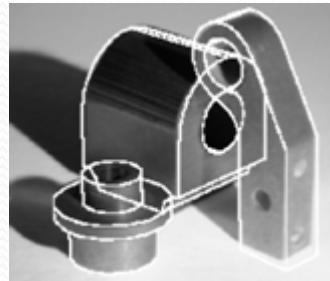
- Задача состоит в обнаружении на изображении заданного объекта(ов).



- Применение:  
поиск лиц,  
поиск пешеходов,  
поиск вещей,  
сортировка на конвейере,  
поиск в цифровых базах данных  
(запрос по шаблону), ...



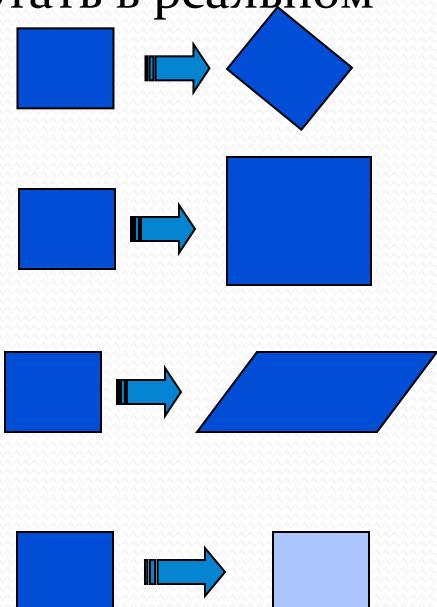
- Вариации задачи:
  - мало примеров для обучения
  - много примеров для обучения



# Постановка задачи

Требования к системе обнаружения

- Ошибки первого рода (ложное детектирование, *false positive*) должны быть редки, ошибки второго рода (пропуск объекта, *false negative*) допускаются ещё реже.
- Ограничено время поиска. Система должна работать в реальном времени (25FPS на VGA)
- Алгоритм поиска может быть инвариантным к
  - Смещению
  - Масштабированию
  - Повороту
  - Аффинным преобразованиям
  - Перспективным преобразованиям
  - Частичному перекрытию объекта
  - Изменению цветовых характеристик
  - Неравномерному освещению
  - Значительному уровню шума
  - 3D повороту объекта



# Методы обнаружения

Шаг 1.

Генерация гипотез  
(вопрос: “где?”)

Перебор всех возможных  
гипотез  
(метод скользящего окна)

Шаг 2.

Проверка гипотезы  
(вопрос: “что?”)

Оценка позы всего объекта по  
позе характерных частей  
объекта

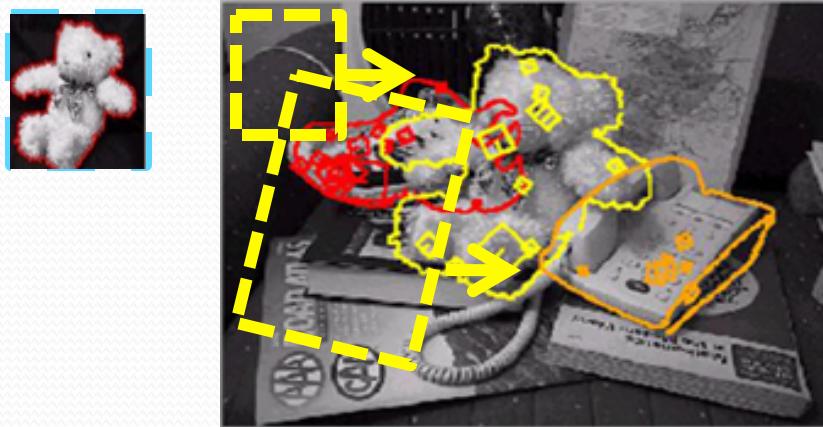
Стохастический поиск

Проверка корреляции  
изображения шаблона и  
гипотезы

Проверка согласованности  
частных гипотез  
(поиск клики, обобщённое  
преобразование Хафа)

Машинное обучение  
(adaboost, Nnets, SVM,...)

# Метод скользящего окна. Поиск прямым сравнением.



Методом скользящего окна проверяются все возможные гипотезы позиции, масштаба и поворота объекта.

Недостатки: низкая скорость, дискретность шага, трудность выбора порога для срабатывания детектора в зависимости от условий освещённости и контрастности объектов.

# Эмпирические методы поиска.

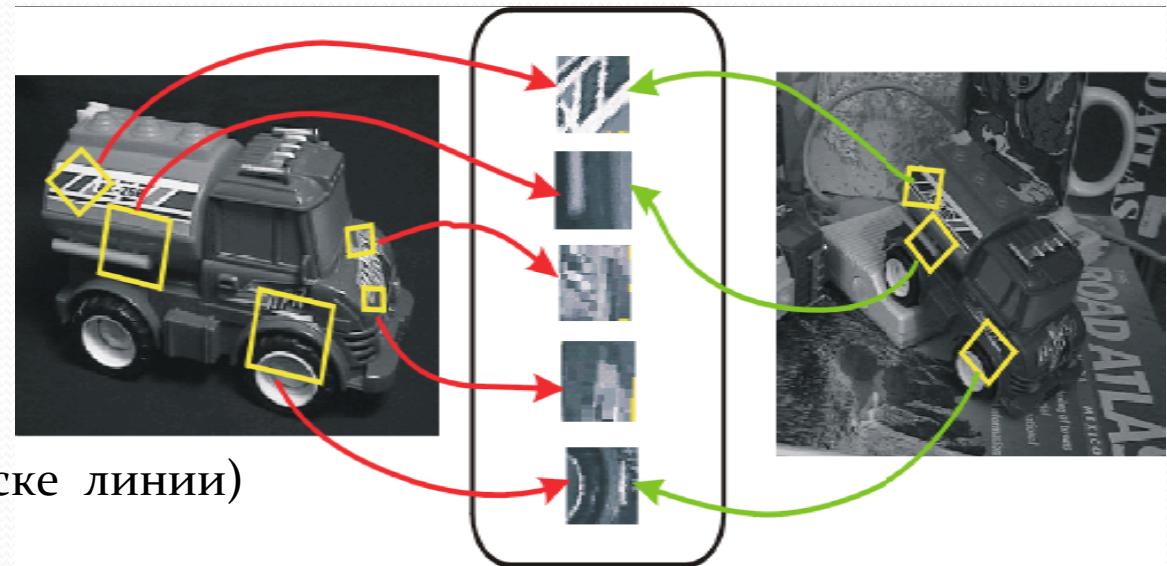
- Обнаружение номеров автомобилей



- Возможные эмпирические правила:
  - 1) находится вытянутый светлый сегмент характерного размера.
  - 2) проверяется наличие внутренних контуров (букв), проверяется их форма.

# Методы поиска снизу-вверх. ("Сборочный" метод.)

- Методы поиска объектов из этой группы осуществляют сборку объекта по частям.  
(т.е. сначала происходит поиск частей объекта, затем проверяется их взаимное расположение – оно должно удовлетворять допустимым преобразованиям шаблонного объекта (напр. заданы допуски на возможные аффинные преобразования))
- Каждая гипотеза о местоположении части объекта генерирует гипотезу о положении всего объекта.  
(аналогично hough transform при поиске линии)



# Робастные методы декомпозиции объекта(сцены) на существенные части

- Для того чтобы искать объект “частями” необходимо разработать детектор частей объекта, который будет находить те же самые характерные особенности объекта при его съёмке с разных мест, в разное время и разными камерами!  
(геометрические искажения, изменения освещённости, отсутствие видимости некоторых частей...).

# Общая схема вычисления описателей(дескрипторов)

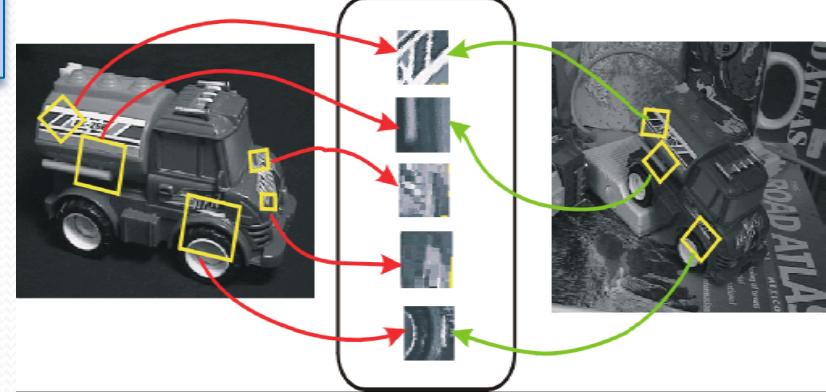
Локализация особенности  
(X, Y, Scale)



Вычисление ориентации особенности  
(alpha)



Вычисление вектора-признаков/  
дескрипторов (features[])



# Структура дескриптора

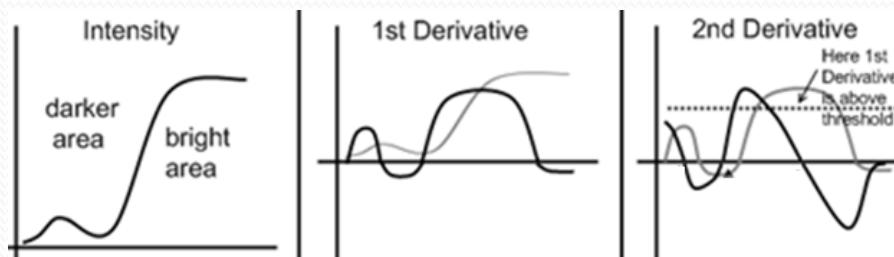
```
typedef struct KeyPoint
{
    • int x,y;
    • float scale; // scale
    • float angle;// orientation (in radian [-PI, PI])
    • float features[]; // feature vector. (128 мерный для SIFT
        описателя)
}
```

# Шаг1. Локализация особенности. Метод LoG (Laplacian of Gaussian)

- $f(x,y)$  –полутоновое изображение
- Для масштаба  $t$  определим “scale-space” представление

$$L(x,y,t) = g(x,y,t) * f(x,y). \quad g(x,y,t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/(2t)}$$

- Посчитаем лапласиан  $\nabla^2 L = L_{xx} + L_{yy}$
- Этот лапласиан имеет большие положительные значения в тёмных областях размером порядка  $\sqrt{t}$  и отрицательные в светлых.
- Значения лапласиана зависят от взаимосвязи размера тёмной или светлой характерной части объекта и масштаба  $t$





Scale-space representation  $L(x,y,\delta)$  at scale  $t = 0$ ,  
corresponding to the original image  $f$ .



Scale-space representation  $L(x,y,\delta)$  at scale  $t = 1$ .



Scale-space representation  $L(x,y,\delta)$  at scale  $t = 4$ .



Scale-space representation  $L(x,y,\delta)$  at scale  $t = 16$ .

# LoG (продолжение)

- Вместо лапласиана будем использовать *scale-normalized Laplacian operator*

$$\nabla_{norm}^2 L(x, y; t) = t(L_{xx} + L_{yy})$$

- Устойчивые к изменению масштаба тёмные и светлые части объекта соответствуют локальным максимумам и минимумам этого оператора (т.е. для каждой точки этого объёма надо проверить 26 соседей).

$$(\hat{x}, \hat{y}; \hat{t}) = \text{argmaxminlocal}_{(x, y; t)} (\nabla_{norm}^2 L(x, y; t)).$$

- Можно проверить, что выполняется свойство инвариантности: если дескриптор найден в точке  $(x_0, y_0; t_0)$ , то после изменения масштаба на  $s$  тот же самый дескриптор будет найден в точке  $(sx_0, sy_0; s^2 t_0)$

(Lindeberg 1998).

# SIFT – Scale Invariant Features Transform

“Distinctive image features from scale-invariant key points David G. Lowe, International Journal of Computer Vision, 2004”



**Sony Aibo  
(Evolution Robotics)**

[SIFT usage:](#)

- Recognize charging station
- Communicate with visual cards



# Локализация особенности методом DoG (Difference of Gaussians )

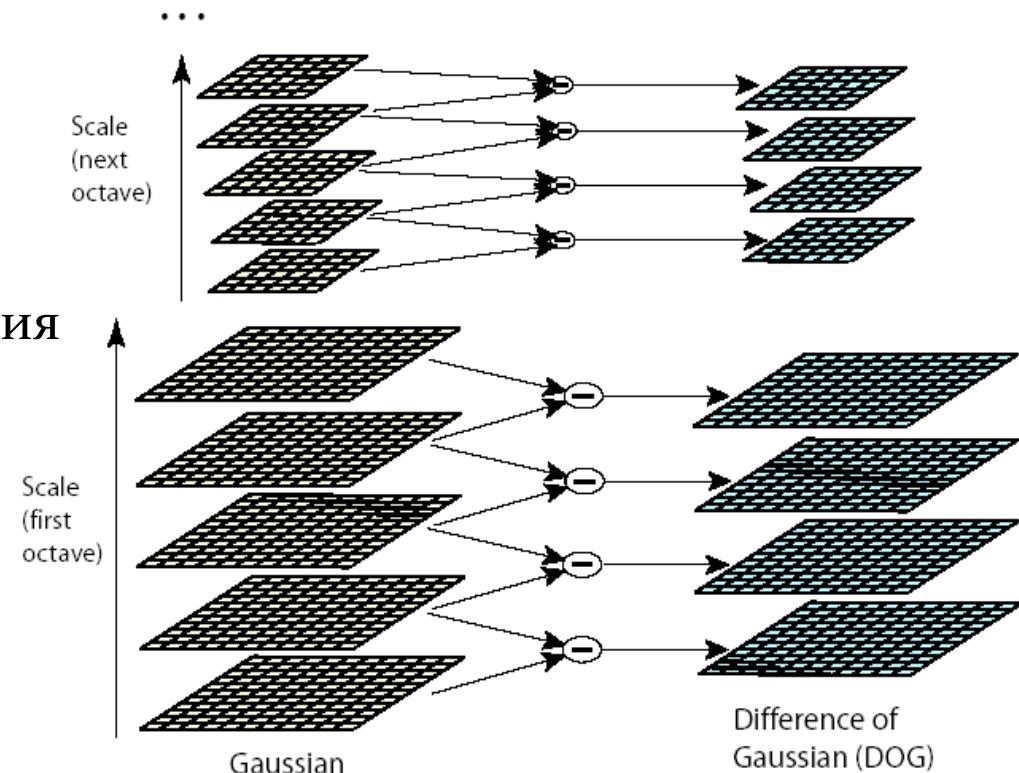
- Уравнение диффузии

$$\partial_t L = \frac{1}{2} \nabla^2 L$$

- Лапласиан пропорционален разнице гауссиан изображения

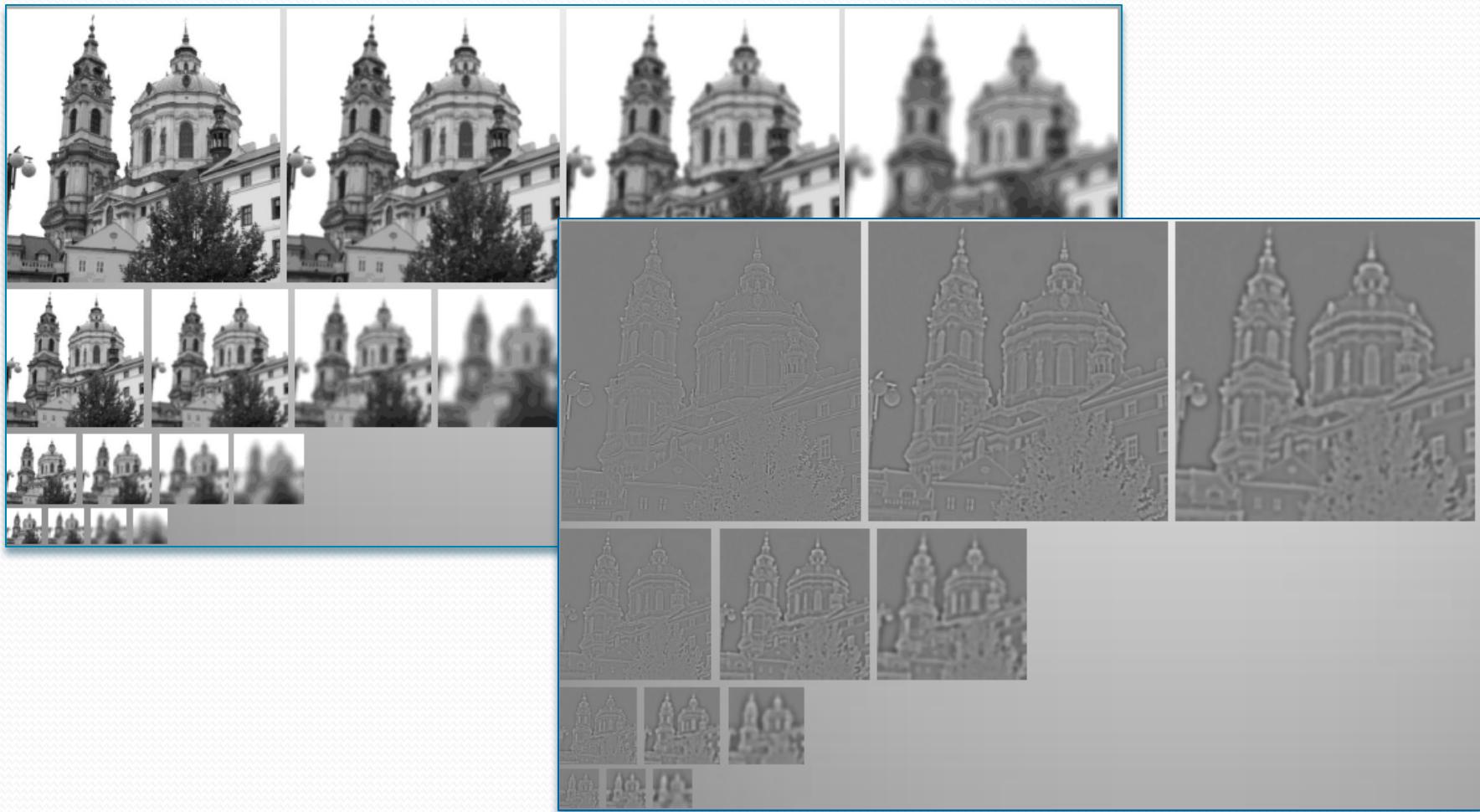
$$\frac{1}{2\Delta t} (L(x, y; t + \Delta t) - L(x, y; t - \Delta t))$$

- Для уменьшения вычислений scale-space разбивают на октавы



"Distinctive image features from scale-invariant key points David G. Lowe, accepted for publication in the International Journal of Computer Vision, 2004"

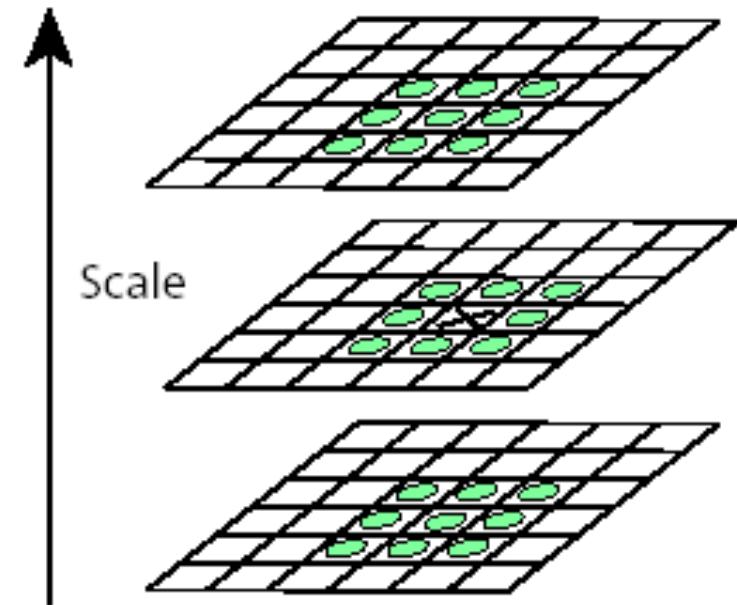
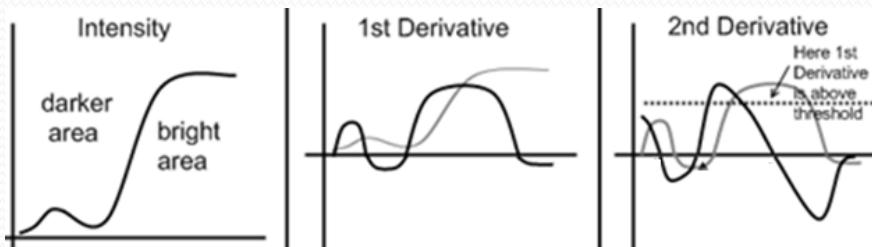
# Пример scale-space представления



# Локализация особенности

- Локализация характерных частей объекта заключается в поиске локальных экстремумов в массиве DoG изображений.

(поскольку лапласиан имеет большие положительные значения в тёмных областях размером порядка  $\sqrt{t}$  и отрицательные в светлых эти экстремумы будут соответствовать центрам характерных “неоднородностей” объекта.)



# Фильтрация неустойчивых дескрипторов

- Среди всех локальных экстремумов выбираются имеющие выраженный пик (большая разница с соседями).
- Также игнорируются экстремумы на рёбрах



# Основные шаги

Нахождение локальных экстремумов в  
Scale-space



Фильтрация шумовых экстремумов  
(non-on-edge, threshold on peak value)



Вычисление ориентации дескриптора



Вычисление вектора-признаков

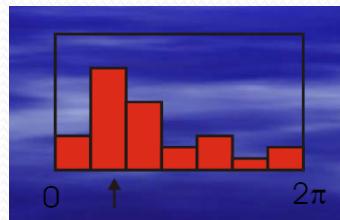
# Оценка ориентации

- Рассматривается окрестность особенности с радиусом пропорциональным масштабу особенности и строится гистограмма orientation градиентов

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$

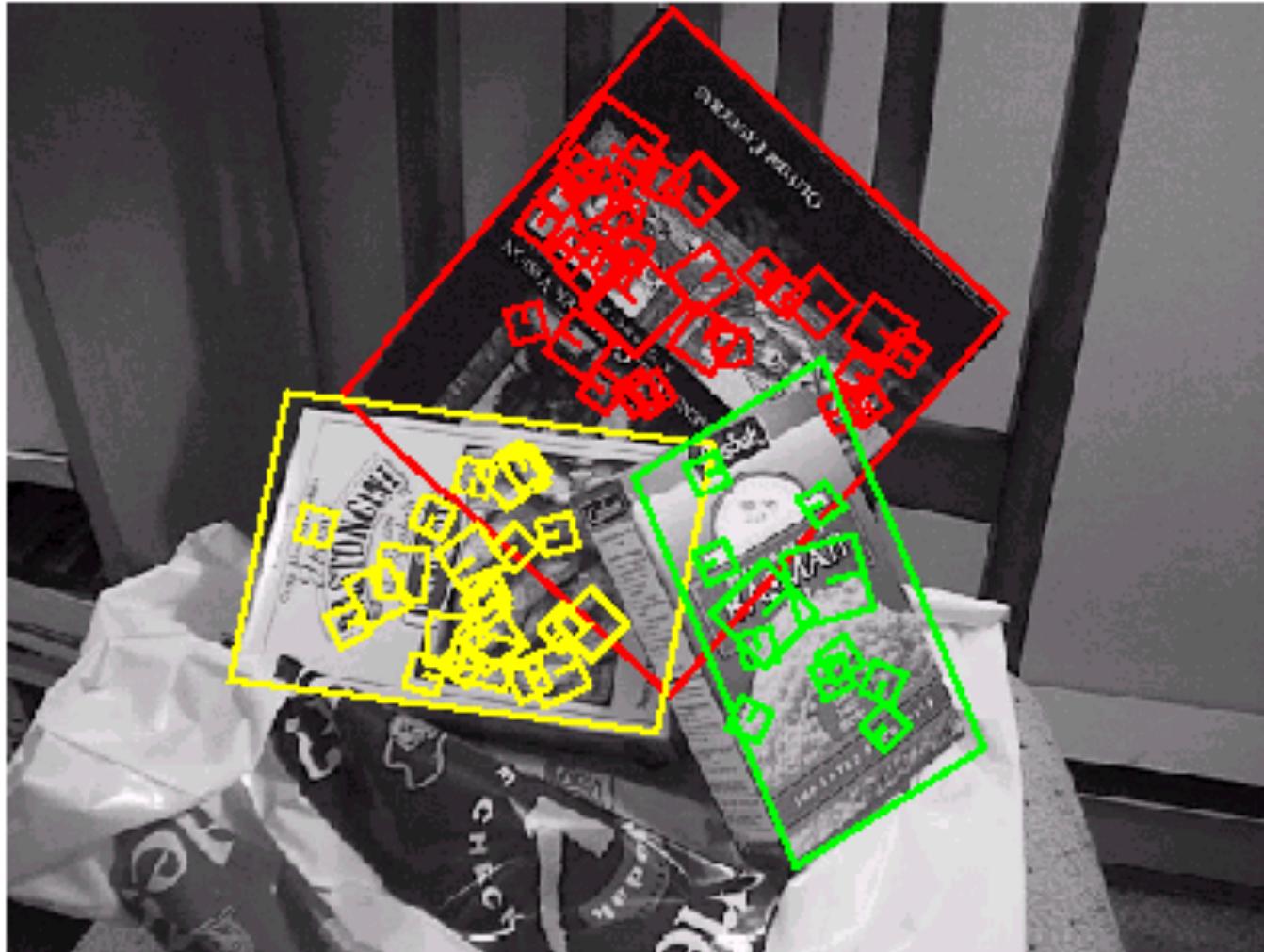
$$\theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1)) / (L(x + 1, y) - L(x - 1, y)))$$

- Максимальный бин в этой гистограмме соответствует самому репрезентативному направлению изменения градиента в этой окрестности. Угол соответствующий этому бину выбирается как оценка ориентации особенности.



Примечание. В оригинальной статье [Lowe] каждый градиент добавляется в гистограмму пропорционально его магнитуде и взвешивается с сигмой в 1.5 раза больше, чем масштаб дескриптора (т.е. центральные точки более важны).

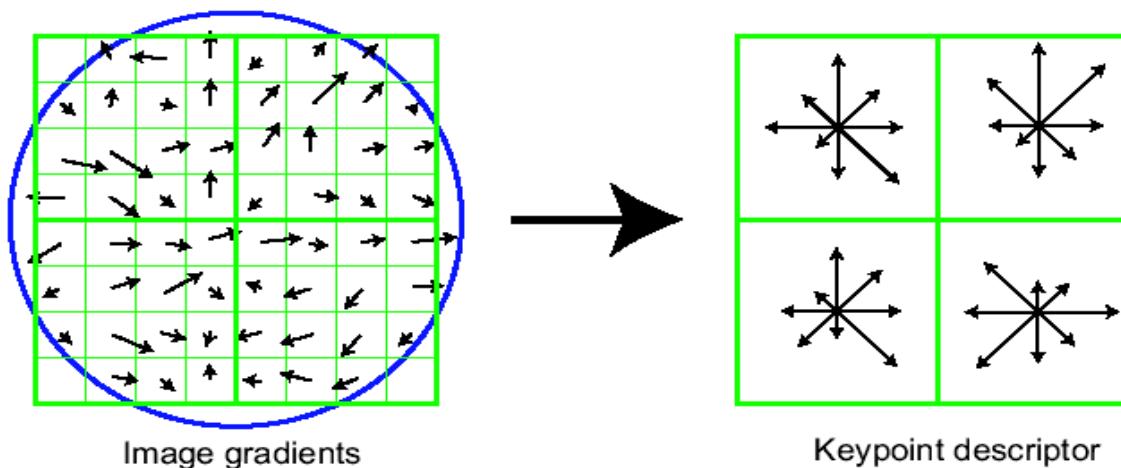
# Пример



# Вектор свойств особенности

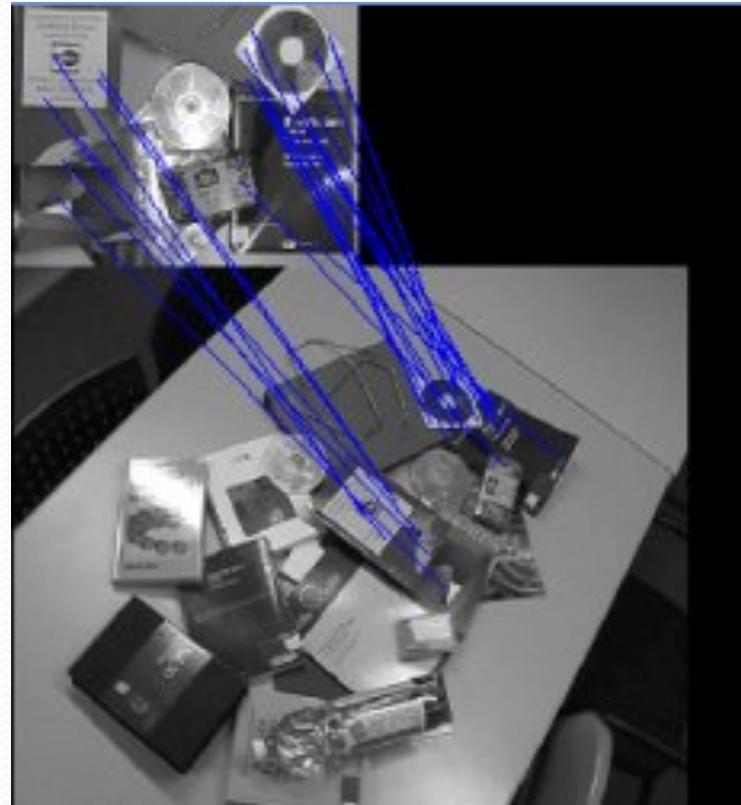
Вычисление вектора свойств особенности состоит из следующих шагов:

- 1) Область изображения, соответствующая особенности, поворачиваются так, что основное направление градиентов области направлено вдоль оси ОХ.
- Область разбивается на четыре квадранта. В каждом строится гистограмма ориентаций градиента, состоящая из 8 бинов.
- Гистограмма нормализуется так , что сумма бинов равнялась 1.



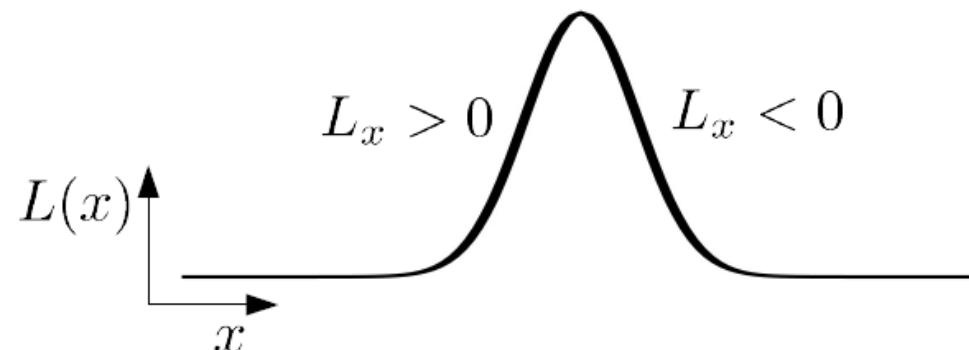
# Альтернатива SIFT – SURF: Speeded Up Robust Features

- <http://www.vision.ee.ethz.ch/~surf/index.html>



# SURF. Image Hessian Matrix

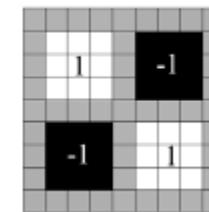
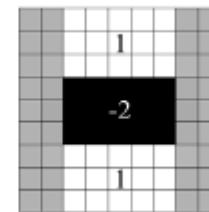
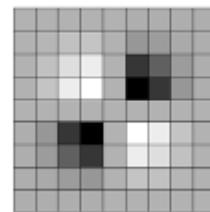
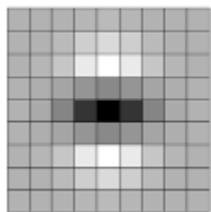
- Как и раньше обозначим  $L(\tilde{x}; t) \stackrel{\text{def}}{=} \mathbf{L}(\tilde{x}) \otimes \mathbf{G}(\tilde{x}, t)$
- Определим матрицу  $\mathbf{H} \stackrel{\text{def}}{=} \nabla \nabla^T \mathbf{L} = \begin{pmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{pmatrix}$
- Большие значения вторых производных соответствуют экстремумам.
- Большие значения определителя  $\mathbf{H}$  соответствуют экстремуму в обоих направлениях  $(x, y)$ .



$$\det(\mathbf{H}) = L_{xx}L_{yy} - L_{xy}^2$$

# SURF. Использование интегральных изображений.

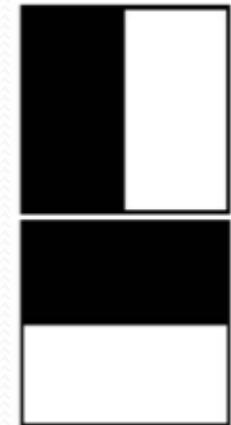
- $L_{xx}(x, \sigma)$  можно представить свёрткой второй производной гауссиана  $\frac{\partial^2}{\partial x^2}g(\sigma)$  с исходным изображением
- Аппроксимируя вторую производную гауссиана и используя интегральное изображение можно быстро посчитать эл-ты матрицы  $H$



- Позиция и масштаб дескриптора соответствует локальному максимуму  $\det(H_{approx})$
- Применение интегральных изображений позволяет избавится от вычислительно дорогих многократных сглаживаний изображений как в методе SIFT.

# SURF: ориентация особенности

- В окрестности точки, зависящей от масштаба особенности вычисляются Haar – вейвлеты. Окрестность равна  $6s$ , где  $s$  – масштаб особенности
- Масштаб вейвлетов  $4s$ . Вейвлеты вычисляются во всех точках окрестности с некоторым шагом.
- Сектором в  $\frac{\pi}{3}$  градусов сканируются все возможные ориентации и выбирается та, где сумма вейвлетов в секторе максимальна



# SURF: вектор свойств

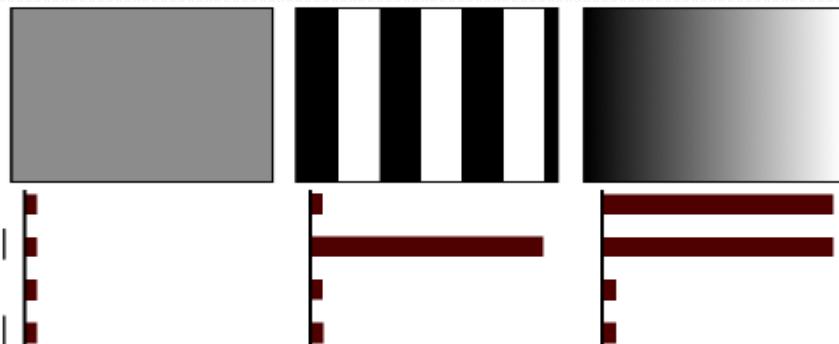
- Рассматривается окно с центром в особенности размером  $2os$  (где  $s$  масштаб особенности). Пиксели окна поворачиваются для выравнивания ориентации.
- Оно разбивается на  $4 \times 4$  области
- В каждой области с шагом 5 пикселей считаются Haar вейвлеты  $d_x$   $d_y$
- Считываются суммы в каждой области

$$\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|).$$

- Всего  $4 \times 4 \times 4$  компоненты.
- Для уменьшения влияния контраста вектор нормируется так что сумма компонент = 1

- Пример вектора свойств для различных изображений.

$$\begin{matrix} \sum dx \\ \sum |dx| \\ \sum dy \\ \sum |dy| \end{matrix}$$



# Поиск объектов с помощью обобщённого преобразования Хафа

- Для каждого дескриптора шаблонного объекта находится одно или несколько из соответствий на изображении (используется  $L_1$  или  $L_2$  расстояние между векторами свойств дескриптора).
- Каждое соответствие генерирует гипотезу в четырёхмерном пространстве гипотез ( $dx, dy, alpha, scale$ ).
- Гипотезы увеличивают соответствующие бины в четырёхмерной гистограмме (примечание. на практике используется структура хранения sparse matrix).
- Положению объекта соответствует гипотеза с максимальным значение бина в четырёхмерном пространстве Хафа (самая популярная гипотеза).

# Поиск объектов

Уточнение позиции объекта

- Аффинное преобразование шаблонного дескриптора в дескриптор на изображении можно записать как.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Если известно хотя бы три соответствия, то

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

- Или

$$Ax = b \quad x = [A^T A]^{-1} A^T b$$

# Поиск объектов

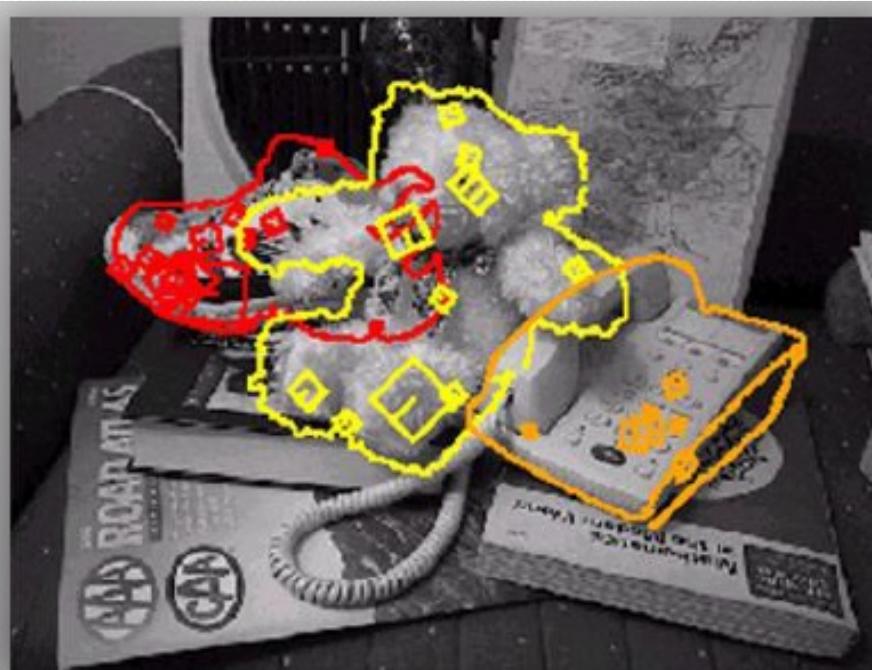
Уточнение позиции объекта

- Отбрасываются все гипотезы из максимального бина, которые не согласованы с полученной оценкой (отличаются больше, чем на половину размера бина).
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$
- После фильтрации параметры преобразования пересчитываются на оставшихся точках.
- Итерации повторяются пока есть, что отбросить или пока число точек больше 3.

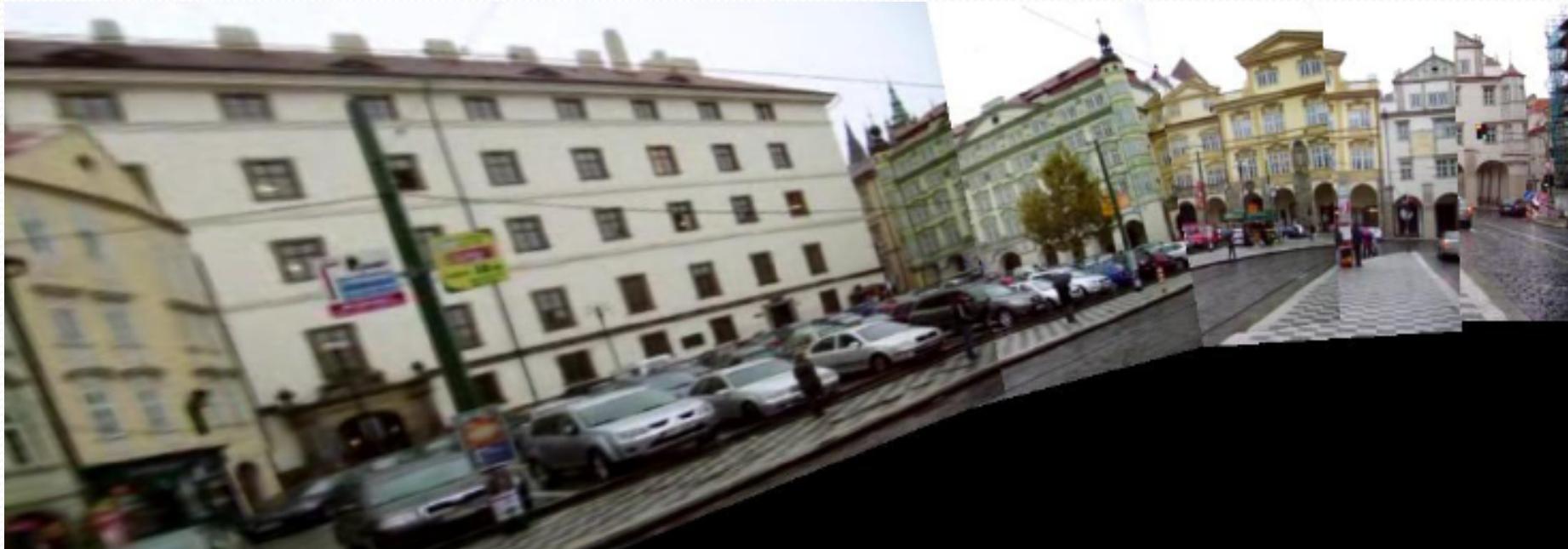
# Пример детекции



# Пример



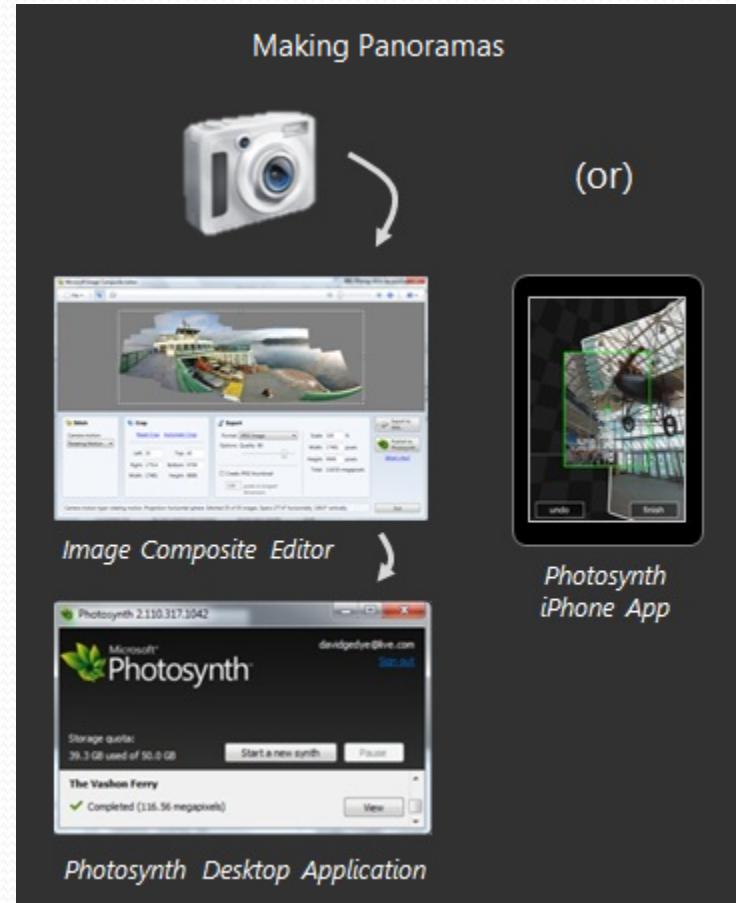
# Применение SIFT. Панорама.



- Панорамные фотографии.
- GIS: ортофотоплан,стыковка планшетов.
- Есть бесплатные продукты (Autopano , GPL license)

# Применение SIFT. <http://photosynth.net/>

- Поиск точек
- Сопоставление точек, построение “трэков”
- Восстановление 3D,
- Построение панорамы



# Применение SIFT

Sony Aibo  
(Evolution  
Robotics)

SIFT usage:

- Recognize charging station

- Communicate with visual cards



AIBO ERS-7 оснащен цветной камерой видеонаблюдения, установленной у него в пасти.

Покажите AIBO обложку компакт-диска, и он воспроизведет этот диск на встроенным громкоговорителе!

Покажите роботу AIBO его любимые игрушки: AIBOne (розовая косточка) и розовый мячик, и он повеселит вас невероятными трюками.

# MSER: Maximally stable extremal regions

## Локализация дескрипторов

**Image**  $I$  is a mapping  $I : \mathcal{D} \subset \mathbb{Z}^2 \rightarrow \mathcal{S}$ . Extremal regions are well defined on images if:

1.  $\mathcal{S}$  is totally ordered, i.e. reflexive, antisymmetric and transitive binary relation  $\leq$  exists. In this paper only  $S = \{0, 1, \dots, 255\}$  is considered, but extremal regions can be defined on e.g. real-valued images ( $S = R$ ).
2. An adjacency (neighbourhood) relation  $A \subset \mathcal{D} \times \mathcal{D}$  is defined. In this paper 4-neighbourhoods are used, i.e.  $p, q \in \mathcal{D}$  are adjacent ( $pAq$ ) iff  $\sum_{i=1}^d |p_i - q_i| \leq 1$ .

**Region**  $Q$  is a contiguous subset of  $\mathcal{D}$ , i.e. for each  $p, q \in Q$  there is a sequence  $p, a_1, a_2, \dots, a_n, q$  and  $pAa_1, a_iAa_{i+1}, a_nAq$ .

**(Outer) Region Boundary**  $\partial Q = \{q \in \mathcal{D} \setminus Q : \exists p \in Q : qAp\}$ , i.e. the boundary  $\partial Q$  of  $Q$  is the set of pixels being adjacent to at least one pixel of  $Q$  but not belonging to  $Q$ .

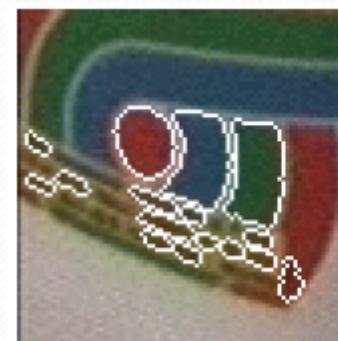
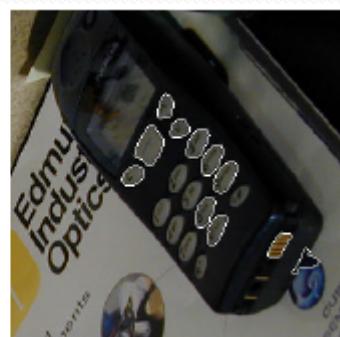
**Extremal Region**  $Q \subset D$  is a region such that for all  $p \in Q, q \in \partial Q : I(p) > I(q)$  (maximum intensity region) or  $I(p) < I(q)$  (minimum intensity region).

**Maximally Stable Extremal Region (MSER).** Let  $Q_1, \dots, Q_{i-1}, Q_i, \dots$  be a sequence of nested extremal regions, i.e.  $Q_i \subset Q_{i+1}$ . Extremal region  $Q_{i^*}$  is maximally stable iff  $q(i) = |Q_{i+\Delta} \setminus Q_{i-\Delta}| / |Q_i|$  has a local minimum at  $i^*$  ( $|\cdot|$  denotes cardinality).  $\Delta \in S$  is a parameter of the method.

# MSER(продолжение)

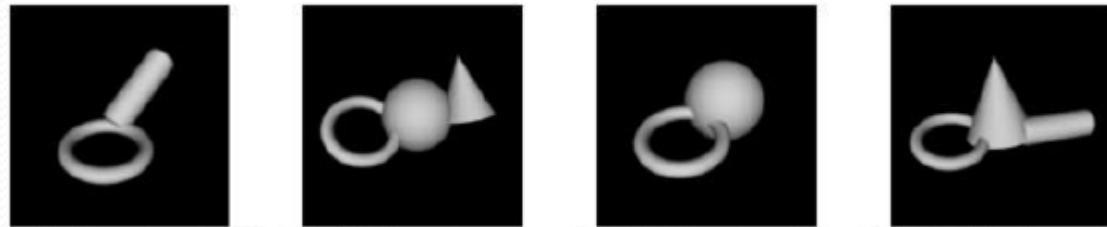
- Сложность локализации  $O(n \log \log n)$
- В качестве вектора признаков используются цветовые моменты.

[8] F. Fleuret, T. Moons, and L.J. van Gool. Recognizing color patterns irrespective of viewpoint and illumination. In *CVPR99*, pages I:368–373, 1999.

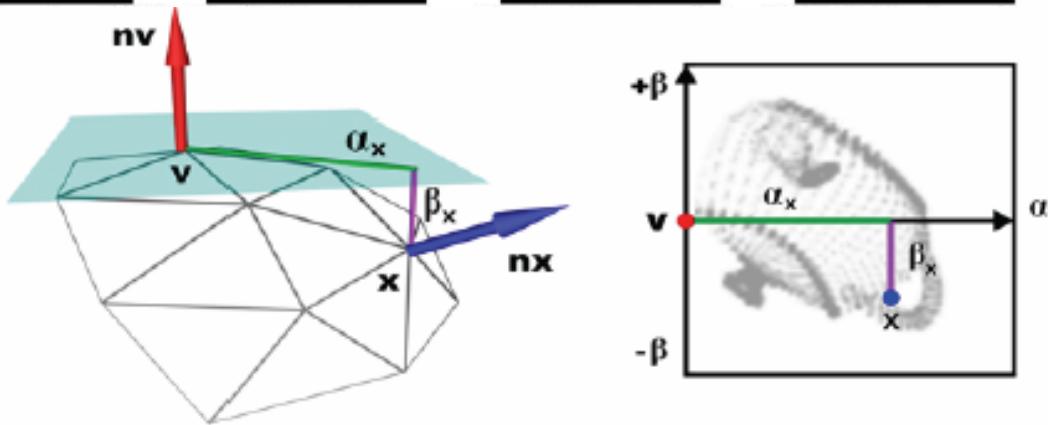


# Описание 3D объектов.

## Spin images.



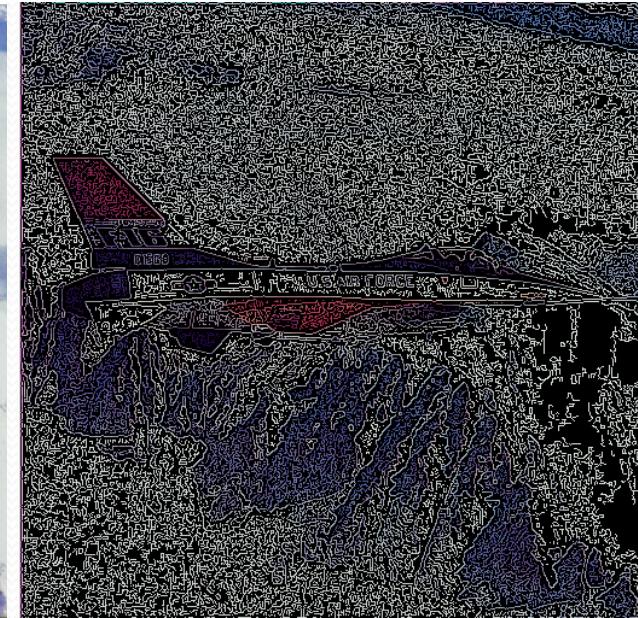
- Для каждой вершины находится нормаль.
- Рассматривается некоторая окрестность.
- Для каждой точки X окрестности вычисляются  $\alpha$  – расстояние до нормали,  $\beta$  – раст. до норм. плоскости
- Формируется  $\langle\alpha, \beta\rangle$  гистограмма = spin image.
- Таким образом каждая вершина формирует уникальный описатель некоторой окрестности 3D сетки.
- В ходе сравнения 3D объектов сравниваются эти уникальные описатели.



Применяется также для сопоставления 3D изображений с разных ракурсов.

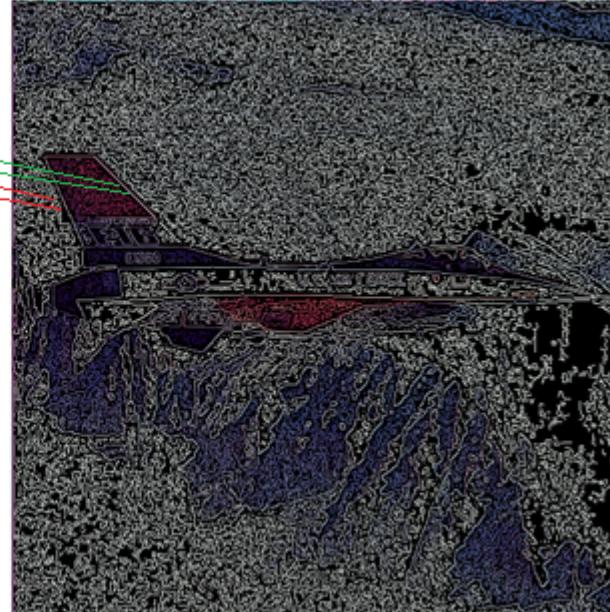
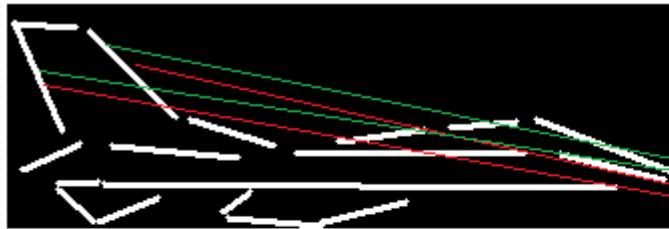
# Поиск объекта с помощью нахождения максимальной клики.

- Пусть шаблон искомого объекта задан рёберной моделью, которая представлена векторами рёбер  
 $T = \{X_{1i}, Y_{1i}, X_{2i}, Y_{2i}\}, i = 1..N$
- Нужно найти похожую конфигурацию рёбер  $I = \{X_{1i}, Y_{1i}, X_{2i}, Y_{2i}\}, i = 1..N_2$  на другом изображении с учётом того, что некоторые рёбра могут отсутствовать. Поиск должен быть инвариантным к заданным аффинным трансформациям объекта на изображении.



# Поиск объекта по краям с помощью нахождения максимальной клики.

- Каждому ребру  $E_1$  из шаблона можно сопоставить по длине, по наклону и по другим признакам несколько рёбер из набора рёбер изображения.

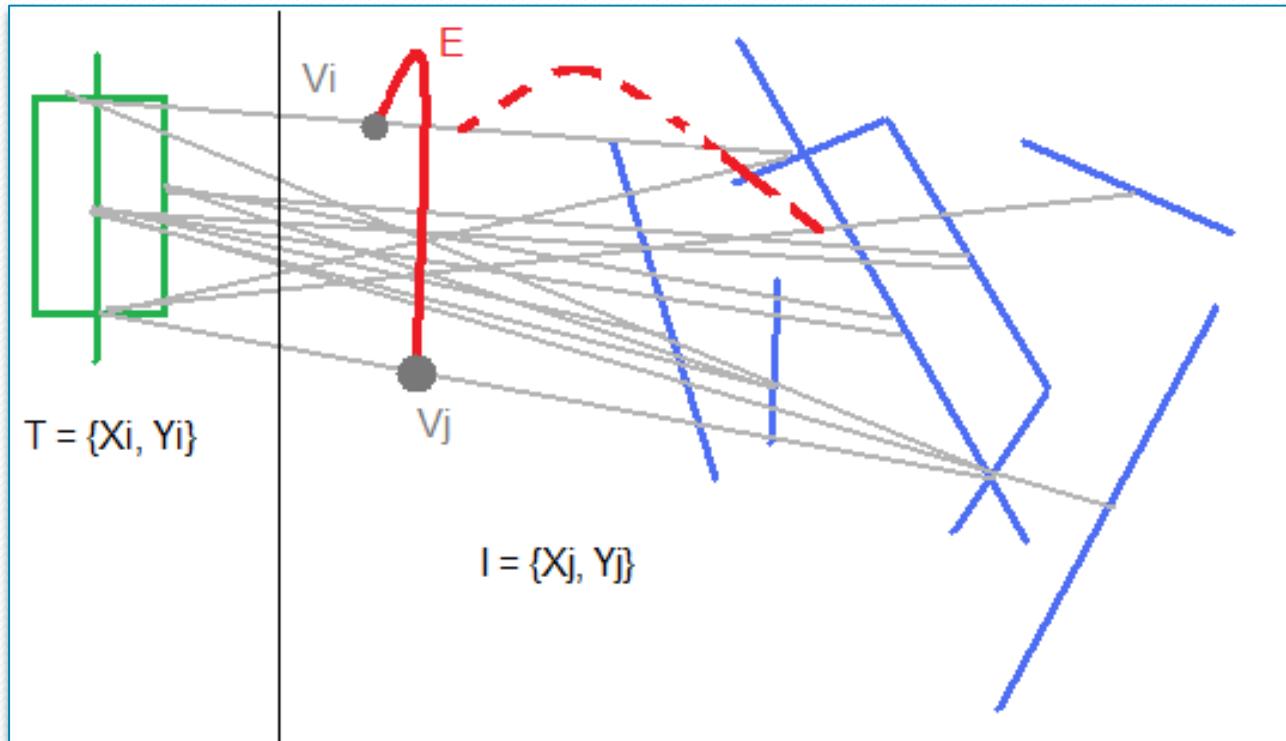


- Пусть каждое такое сопоставление формирует вершину  $V$  некоторого графа  $G$ .

# Поиск объекта с помощью нахождения максимальной клики.

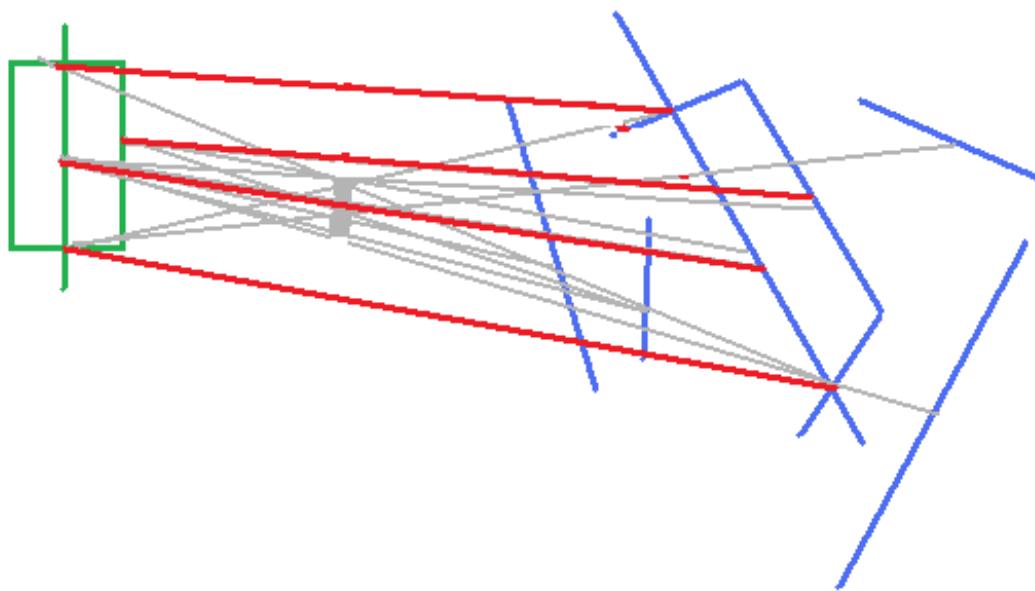
- Две вершины из  $G$  соединяются ребром  $E_k(V_i, V_j)$  если они согласованы по изменению масштаба и повороту.

Задаются допустимые вариации изменения масштаба и поворота (т.е. насколько части объектов могут отличаться друг от друга на шаблоне и на изображении)



# Поиск объекта с помощью нахождения максимальной клики.

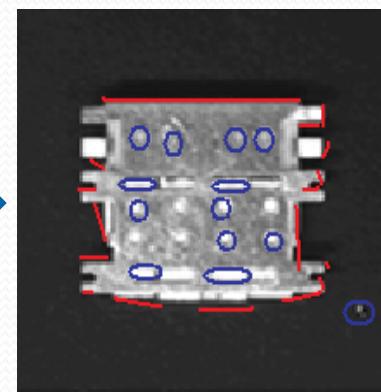
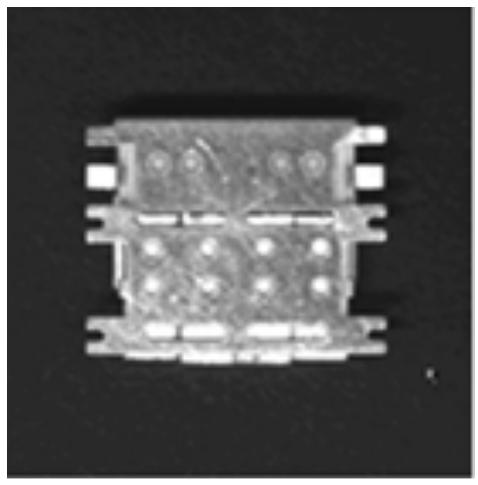
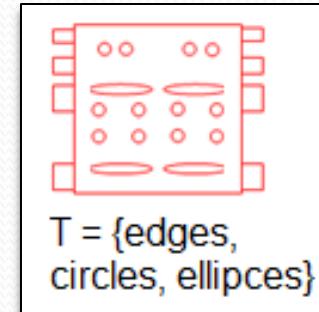
- Таким образом поиск похожей конфигурации можно свести к поиску максимальной клики в графе  $G(V,E)$ .



- Для поиска клики можно использовать генетические алгоритмы.

# Поиск объекта с помощью нахождения максимальной клики.

- Части объекта могут быть разного типа: отрезки, круги, прямоугольники...



Извлечение  
примитивов

По-  
строен  
ие  
 $G(V,G)$   
(с  
учётом  
типа  
зл-та)

Max  
Clique

# Детектирование пешеходов с

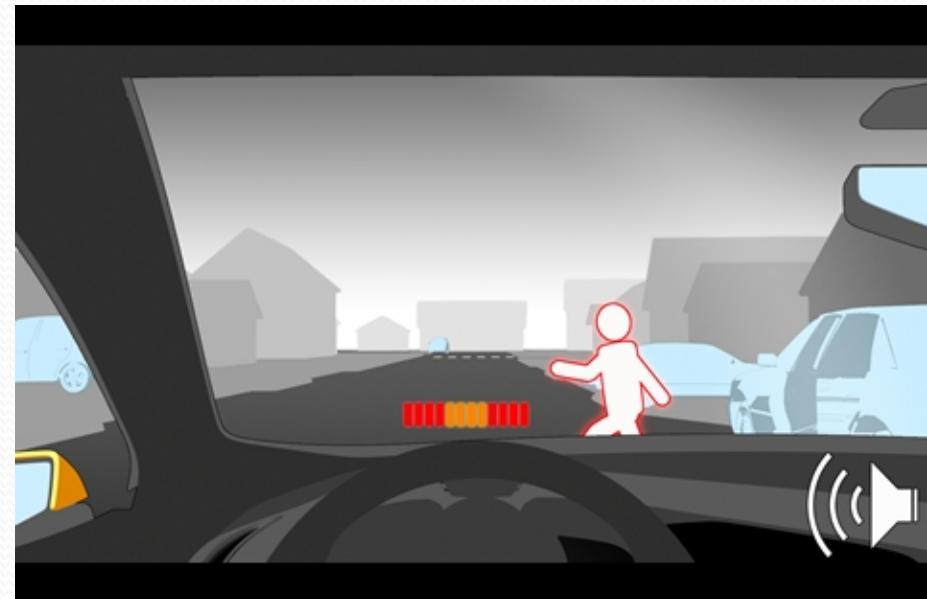
## помощью гистограммы

## ориентаций градиентов (НоГ)

People detection  
is required for:

- **Driver-assistant systems.**

According to Techno Systems Research report the percentage of new vehicles with integrated cameras is projected to increase from ~20 percent in 2008 to nearly 70 percent in 2012.

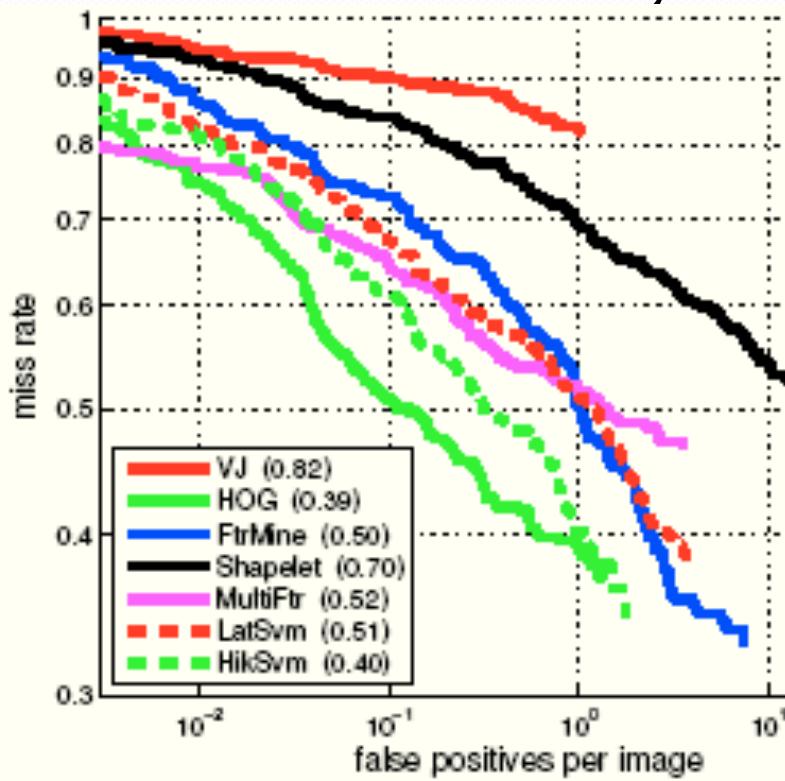


- **Robotics.**

- **Intelligent video surveillance systems.**

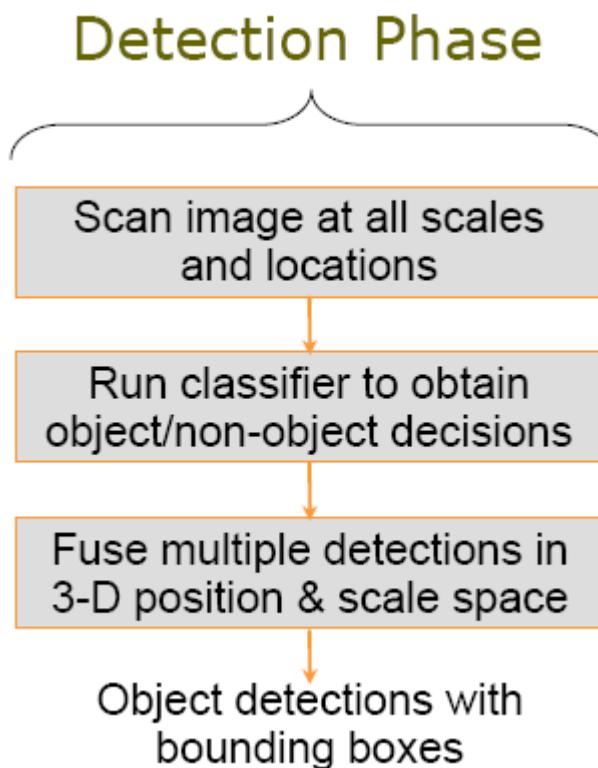
# Pedestrian detection based on Histograms of Oriented Gradients

- Pedestrian Detection: A Benchmark (CVPR' 2009)  
*HOG tends to outperform the other methods surveyed.*

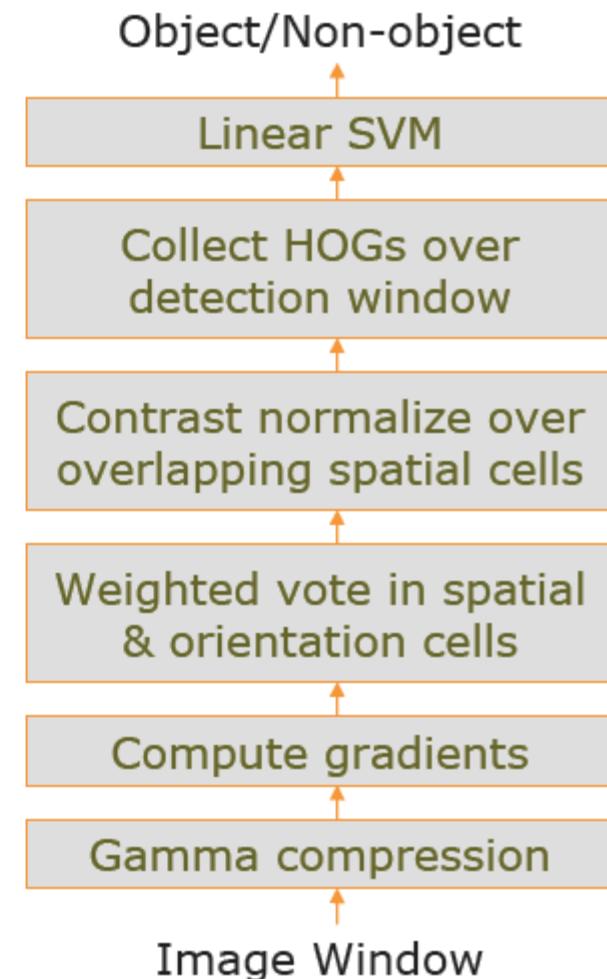
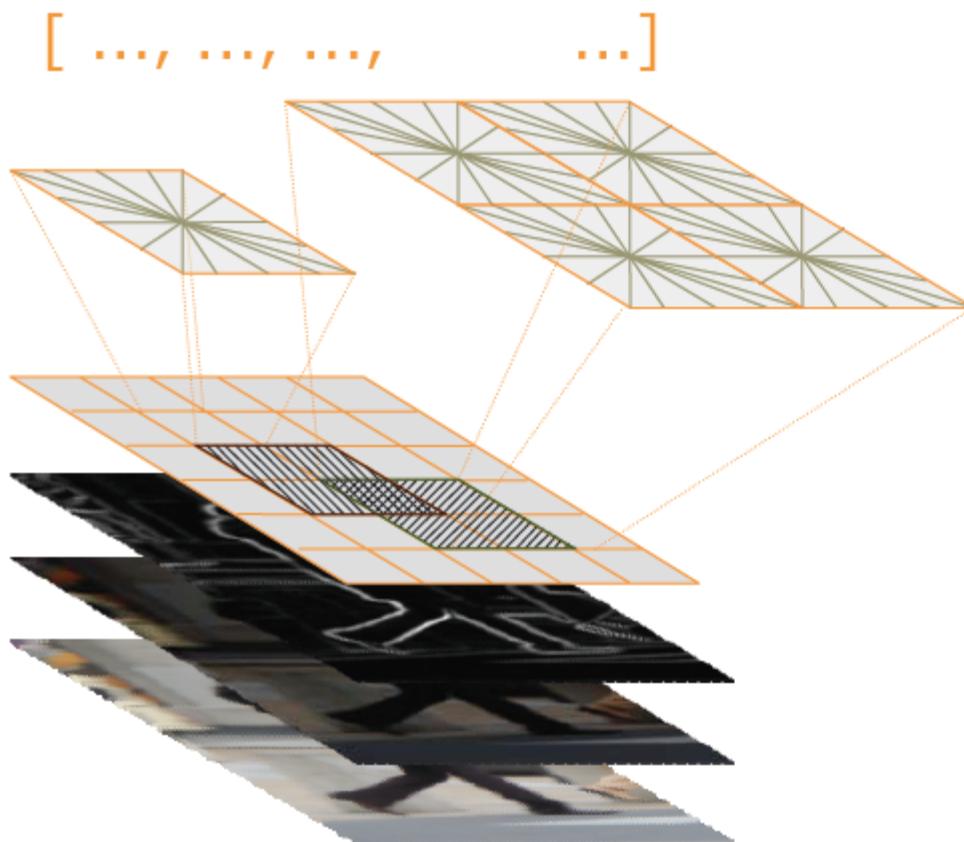


total frames	~1000K
labeled frames	~250K
frames w peds.	~132K
# bounding boxes	~350K
# occluded BB	~126K
# unique peds.	~2300
ave ped. duration	~5s
labeling effort	~400h

# HOG overview



# HOG overview





# Вопросы?

# Быстрое детектирование лиц.

- Нужно найти позицию и масштаб фронтальных лиц на изображении.

- Применение:  
Распознавание лиц;  
Видеонаблюдение;  
Digital Signage;  
Получение и обработка фотографий.



## Rapid Object Detection using a Boosted Cascade of Simple Features

Paul Viola

[viola@merl.com](mailto:viola@merl.com)

Mitsubishi Electric Research Labs  
201 Broadway, 8th FL  
Cambridge, MA 02139

Michael Jones

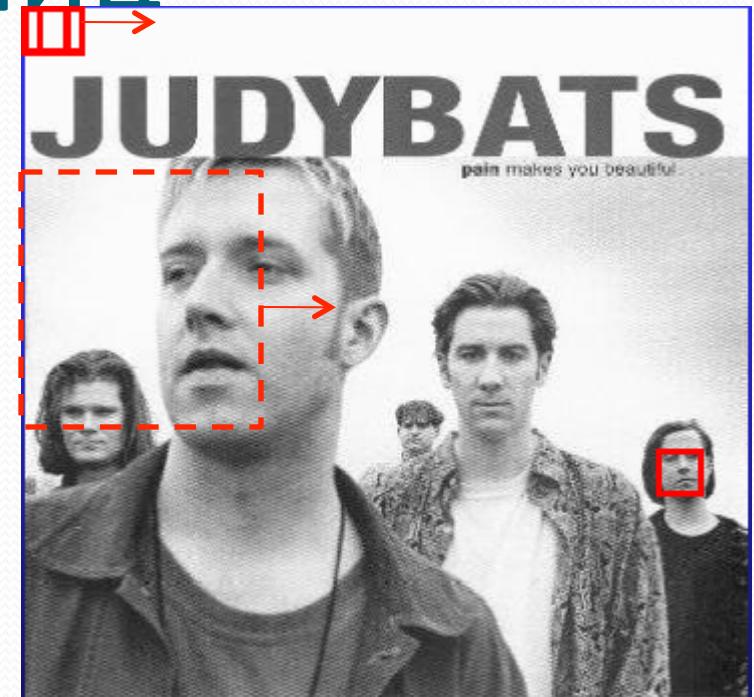
[mjones@crl.dec.com](mailto:mjones@crl.dec.com)

Compaq CRL  
One Cambridge Center  
Cambridge, MA 02142

# Детектирование лиц

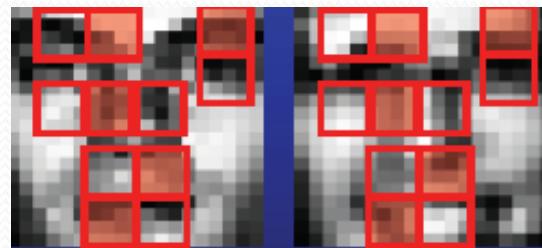
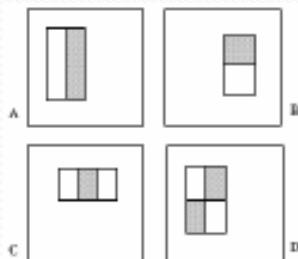
## Генерация гипотез

- Цикл по масштабу и по координатам гипотезы
- Фильтруются заведомо ложные гипотезы:
  - Проверяется наличие рёбер/градиентов (они должны присутствовать)
  - Средняя интенсивность в лице не может быть очень большой или очень малой.
- По оставшимся вариантам строится вектор признаков и происходит классификация методом adaboost.



# Вектор признаков

- Для классификации используются Haar вейвлеты (требование к признакам: должны быстро вычисляться в любом положении и любом масштабе).



- Для быстрого вычисления используется интегральное

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

A	B	
	1	2
C	3	D

The sum within D  
 $4 + 1 - (2 + 3)$ .

# “Слабый” классификатор

- Всего признаков > 100000 .
- По любому признаку можно построить “слабый” классификатор

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

где  $x$  – вектор размерности 24 x 24,  
соответствующий текущему куску изображения.

$f_j$  - j-ый признак

$\theta_j$  - порог

$p_j$  - знак

# Выбор признаков и классификация

- Дано обучающее мн-во изображений  $(x_1, y_1), \dots, (x_n, y_n)$  где  $y_i = 0, 1$ .  
0 – не лица, 1 – изображения лица.
- Инициализируется массив весов  $w_{0,i} = \{1/(2m) \text{ or } 1/(2l)\}$  для каждого изображения  $i$ , где  $m$  и  $l$  число негативных или позитивных примеров
- For  $t = 1 \dots T$ 
  - 1) Веса  $w_t$  нормализуются (пл-ть вер-ти)  $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$
  - 2) Для каждого признака  $j$  тренируется 'слабый' классификатор  $h_j$  и вычисляется ошибка классификации  $\varepsilon_j$  с весом  $w_t$ .  $\varepsilon_j = \sum_i w_i |h_j(x_i) - y_i|$ .
  - 3) Выбирается классификатор  $h_j$  с наименьшей ошибкой.
  - 4) Веса обновляются:

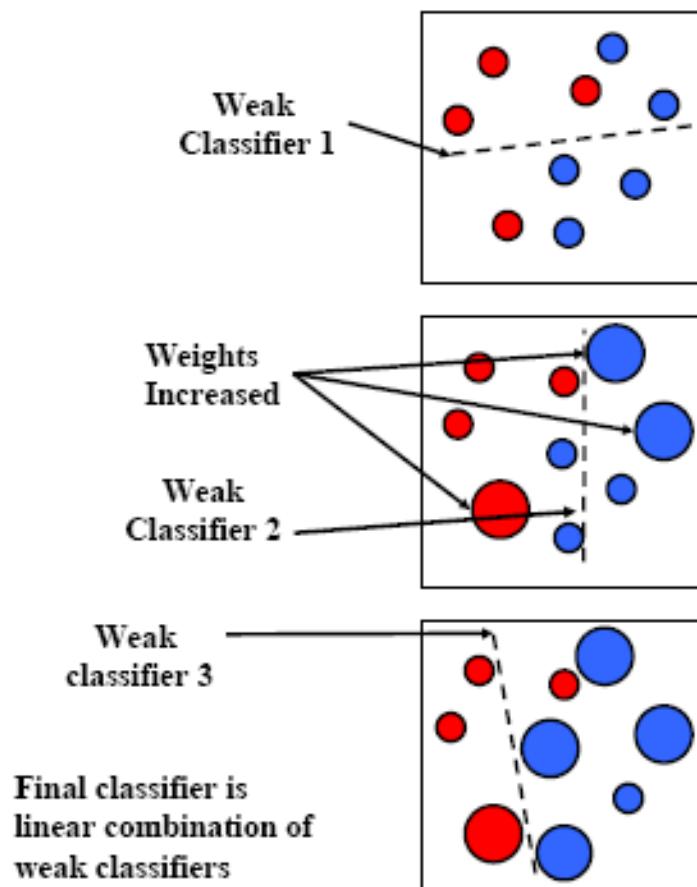
$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

где  $e_i = 0$  если  $x_i$  классифицировалось корректно, 1 в противном случае

- В итоге получим  
'сильный' классификатор:

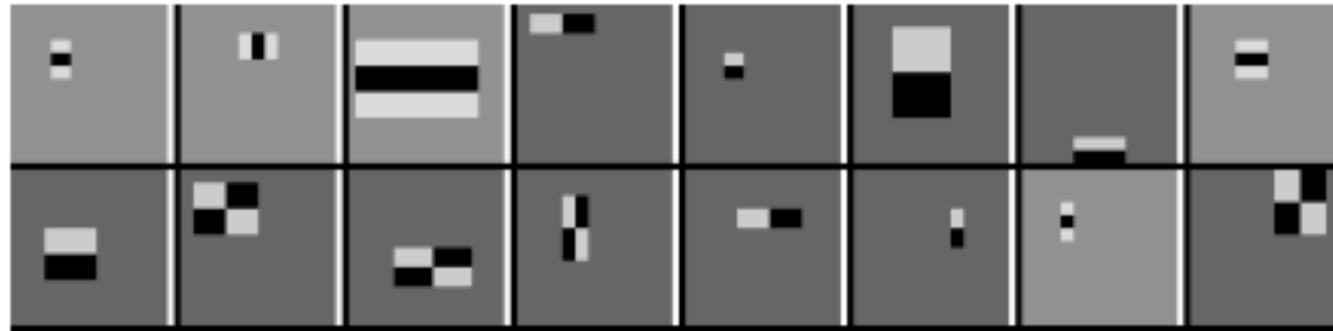
$$h_f(x) = \begin{cases} 1 & \sum_{t=1}^r \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^r \alpha_t \\ 0 & \text{otherwise} \end{cases}, \alpha_t = \log \frac{1}{\beta_t}$$

AdaBoost является адаптивным в том смысле, что каждый следующий классификатор строится по объектам неверно классифицированным предыдущими классификаторами



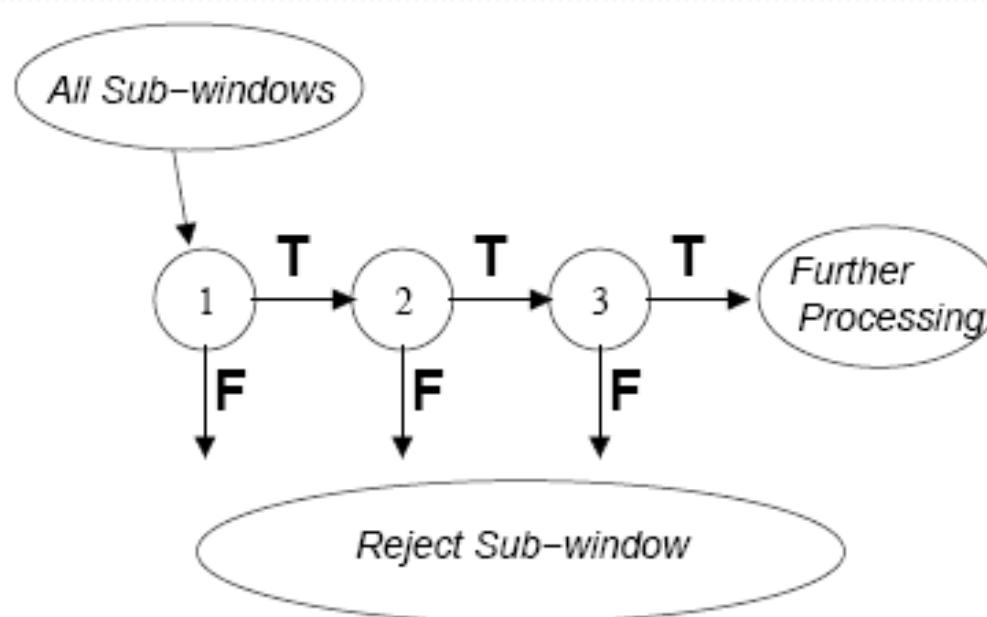
# Детектирование лиц

- Ошибка классификации уменьшается с ростом Т.



# Использование каскада классификаторов

- Подсчёт большого числа признаков и их проверка может быть медленной.
- Используя малое кол-во итераций при обучении можно получить простой классификатор, который тем не менее будет способен отфильтровывать заведомо ложные лица;



# Детектирование лиц

## Тренировка каскада.

- Для каждого классификатора в каскаде используется порядка 10000 примеров лиц и столько же не лиц.
- Тренировка останавливается когда достигается ошибка по частоте ложных срабатываний порядка 50% и правильной классификации лиц 100%.
- Для следующего сильного классификатора набираются другие 10000 негативных примеров, на которых не правильно работают предыдущие ‘сильные’ классификаторы.

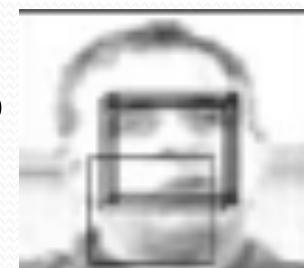


# Детектирование лиц

- Тренировка каскада
- Example: 32 stage cascade classifier
  - ◆ 2-feature classifier in the first stage → rejecting 60% non-faces while detecting 100% faces
  - ◆ 5-feature classifier in the second stage → rejecting 80% non-faces while detecting 100 % faces
  - ◆ 20-feature classifier in stages 3, 4, and 5
  - ◆ 50-feature classifier in stages 6 and 7
  - ◆ 100-feature classifier in stages 8 to 12
  - ◆ 200-feature classifier in stage 13 to 32

# Детектирование лиц

- Каждая гипотеза о положении и масштабе лица оценивается каскадом классификаторов.
- В районе лица получается ‘кластер’ гипотез положения и масштаба лица.
- Формируется среднее положение искомого объекта.
- Число элементов в кластере отражает меру уверенности в этой гипотезе.



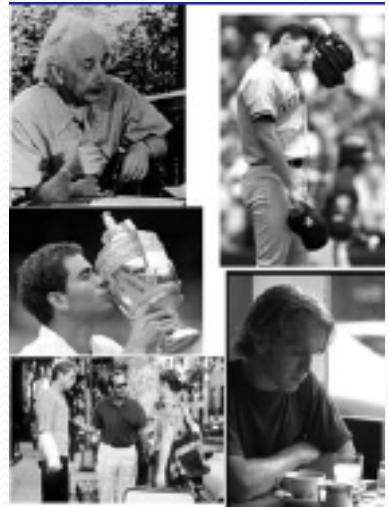
# Детектирование лиц.

## Проблемы

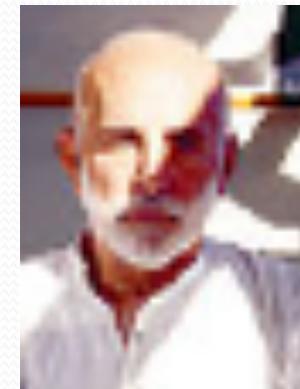
- Ищет только фронтальные лица.



- Метод требователен к освещенности.

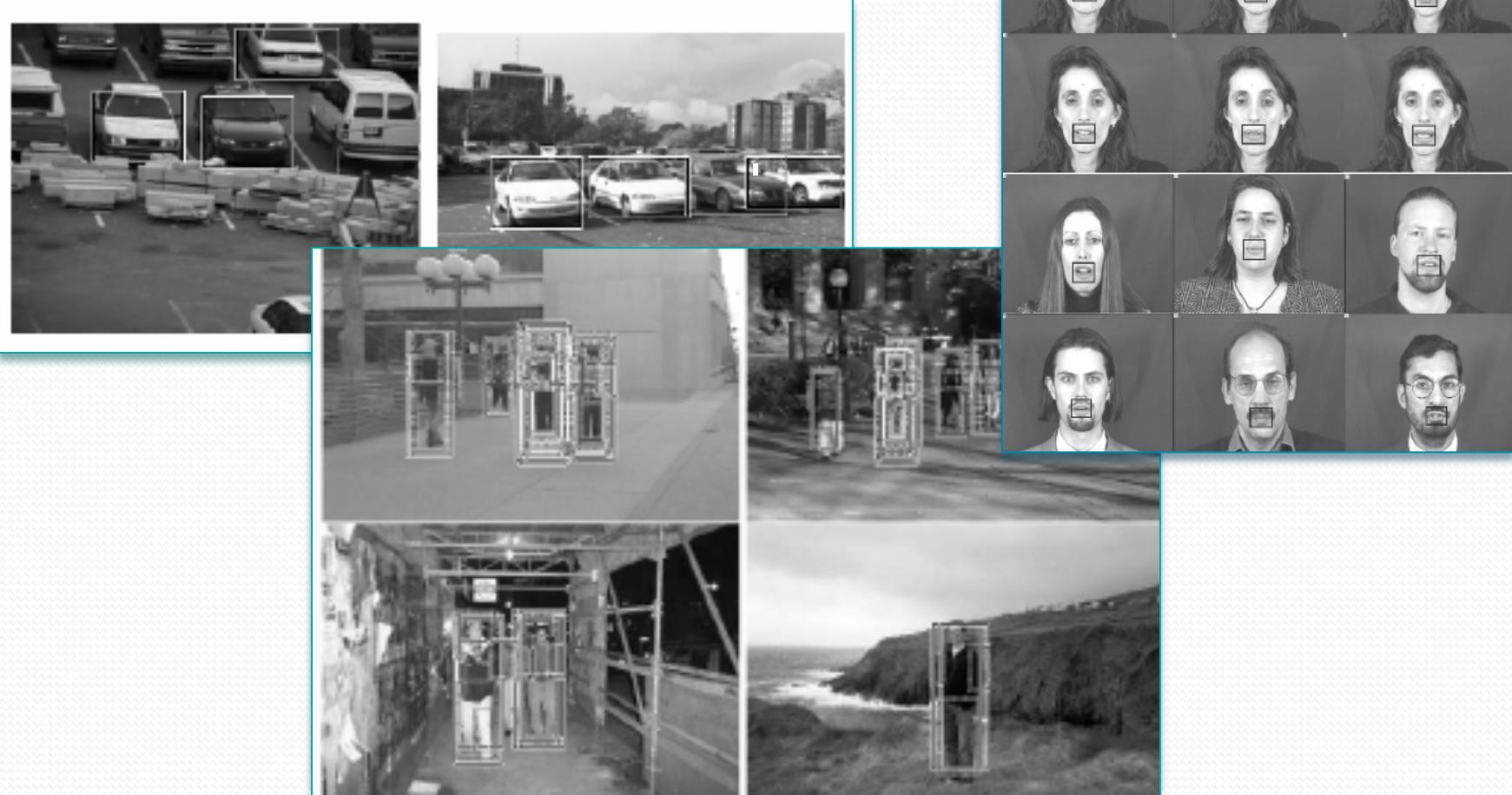


- Производительность ( $640 \times 480$  меньше 25FPS).



# Детектирование объектов

- Вариации



# Обучение каскада

- Детектирование: cvLoad() + cvHaarDetectObjects()
- Обучение:

Для обучения надо тысячи примеров изображения объекта и “не объекта”. Нужны различные вариации объекта (масштабы, повороты, освещение)

Где взять?
- Разметить своими руками
- Разметить чужими руками Amazon’s “mechanical turk” (<http://www.mturk.com/mturk/welcome>).
- Сгенерировать искусственные вариации.

# Обучение каскада

- Для тренировки формируется файл описывающий позитивные примеры

```
<path>/img_name_1 count_1 x11 y11 w11 h11 x12 y12 . . .
<path>/img_name_2 count_2 x21 y21 w21 h21 x22 y22 . . .
    . . .
```

- Утилита createsamples используется для создания бинарного файла содержащего позитивные примеры

```
createsamples -vec faces.vec -info faces.idx -w 30 -h 40
```

- С помощью createsamples можно сгенерировать вариации позитивного примера

```
createsamples.exe -vec 1.vec -img 1.bmp -w 25 -h 25 -maxzangle 0.5 -maxxangle 0.5 -maxyangle 0.5 -show -num 500
```

# Обучение каскада

- Формируется массив изображений из которых случайным образом извлекаются негативные примеры

`data/vacations/beach.jpg`

`data/nonfaces/img_043.bmp`

`data/nonfaces/257-5799_IMG.JPG`

- Для тренировки используется утилита haartraining

```
Haartraining /  
-data face_classifier_take_3 /  
-vec faces.vec -w 30 -h 40 /  
-bg backgrounds.idx /  
-nstages 20 /  
-nsplits 1 /  
[-nonsym] /  
-minhitrate 0.998 /  
-maxfalsealarm 0.5
```

# Расстояние Хаусдорфа. Поиск сравнением границ.

- Находятся границы объектов с помощью сегментации изображения (meanshift) или с помощью детектора краёв (Canny).
- Метод скользящего окна используется для генерации гипотез.
- Для каждой гипотезы вычисляется расстояние Хаусдорфа
  - Hausdorff distance (HD)

For two sets of points  $A = a_1, \dots, a_m$  and  $B = b_1, \dots, b_n$

$$H(A, B) = \max(h(A, B), h(B, A))$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

- Modified Hausdorff distance (MHD)

$h(\cdot)$  –  
несимметричная!

$$h(A, B) = \frac{1}{m} \sum_{a \in A} \min_{b \in B} \|a - b\|$$

- Выбирается гипотеза с минимальным расстоянием между шаблоном и гипотезой.
- Для ускорения вычислений используют Distance Transform.

# Домашнее задание

- Реализовать поиск любого шаблонного объекта на изображении с помощью модифицированного расстояния Hausdorff и distance transform используя библиотеку opencv.