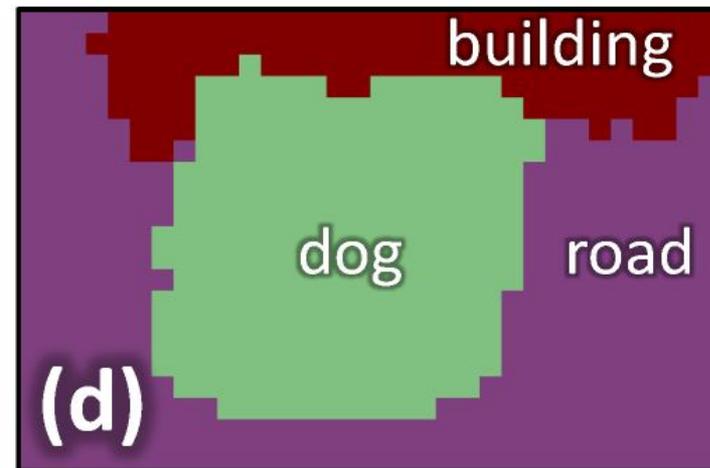


Семантическая сегментация



Наивная классификация



- Нужно классифицировать каждый пиксель
 - 1 МП на картинку!
- Что можно сказать про 1 пиксел?
 - Классификация окрестности пиксела

Наивная классификация

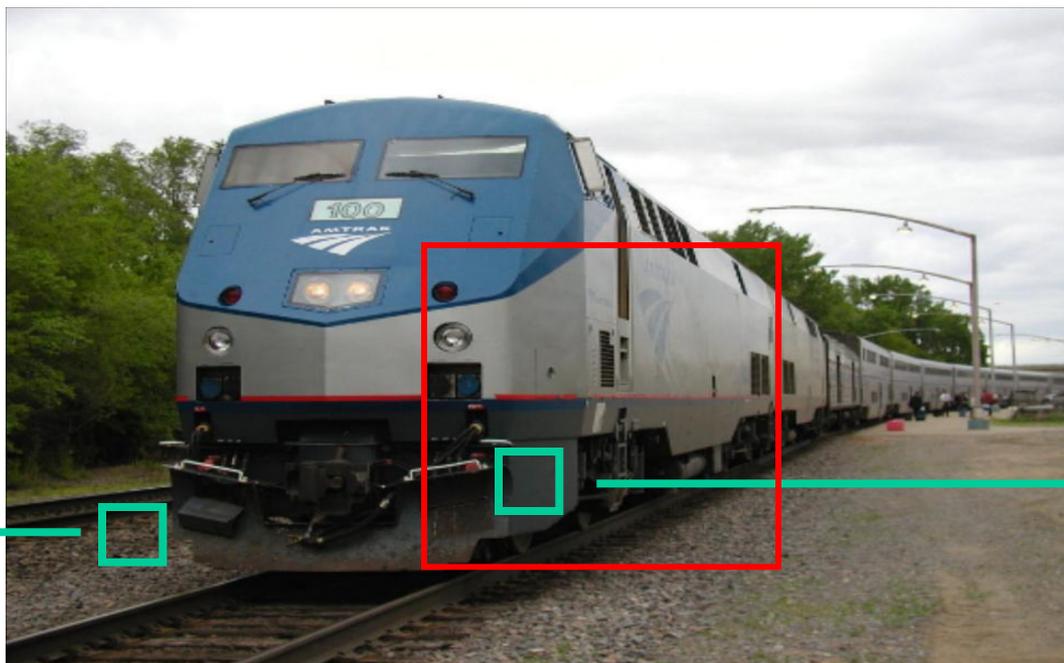


Классифицируем каждый пиксель по окрестности

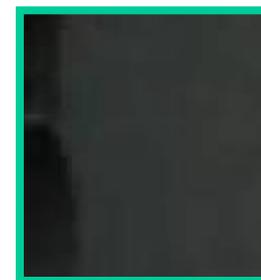


Сегментируем картинку, затем классифицируем сегменты

Пространственная поддержка



50x50 Patch



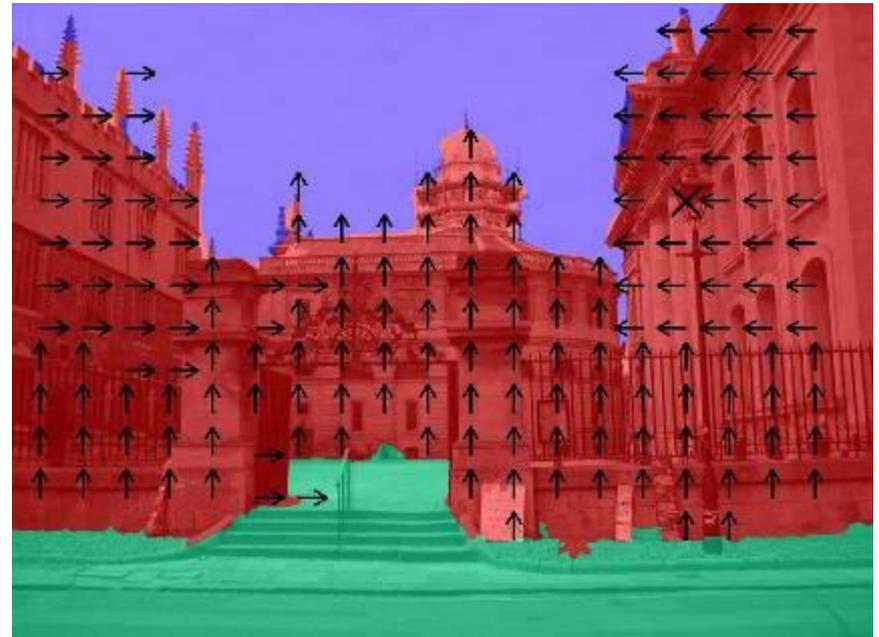
50x50 Patch

- По небольшой окрестности зачастую невозможно правильно определить метку
- Пространственная поддержка
- Необходимо каким-то образом задавать метки для всех пикселей в совокупности

Построение решения

- Задача / Данные
- Элемент
 - Пиксель
 - Сегмент
- Классификация пикселей / регионов
 - Признаки для классификации
 - Метод классификации (бустинг, лес, SVM)
- Расширение пространственной поддержки
 - Множественные сегментации
 - Случайные поля

Уличные изображения



Цель: 7 геометрических классов

- Земля
- Вертикальные стены
 - Плоскости: смотрящие влево (\square), Прямо (), Направо (\square)
 - Другое: Твердые (X), Дырявые (O)
- Небо

Размеченные данные

- 300 изображений из гугла



...



Признаки



Цвет



Текстура



Положение



Перспектива



Сегментация изображений

- Использование нескольких вариантов сегментации (с разными параметрами)
- Решение, какие сегменты хорошие, откладывается на потом

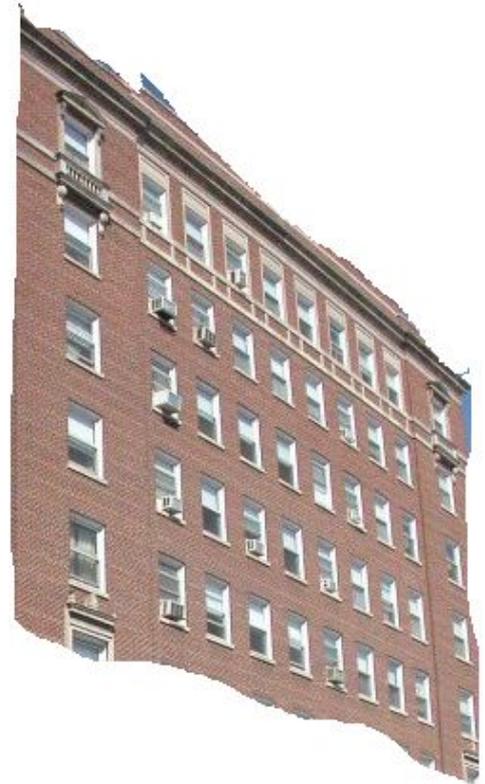


...



Классификация областей

- Что мы хотим узнать:
 - Хороший ли это сегмент?
 $P(\text{good segment} \mid \text{data})$
 - Если сегмент хороший, то какая у него метка?
 $P(\text{label} \mid \text{good segment}, \text{data})$
- Обучаем модель по размеченным данным
 - Бустинг на решающих деревьях



Классификация



Для каждого сегмента вычисляется:

- $P(\text{good segment} \mid \text{data}) P(\text{label} \mid \text{good segment}, \text{data})$

Разметка изображений

Размеченные сегментации

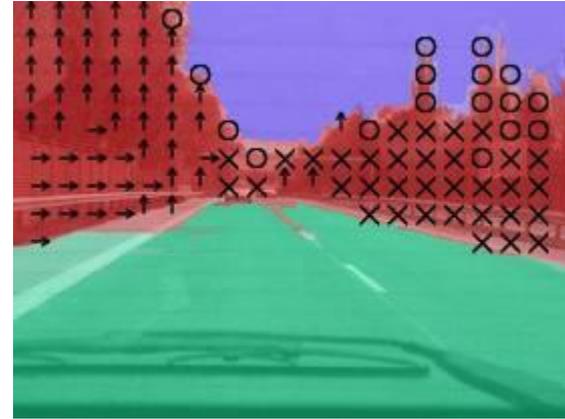


Размеченные пиксели

Вероятностная разметка



Support



Vertical

Sky



V-Left



V-Center



V-Right

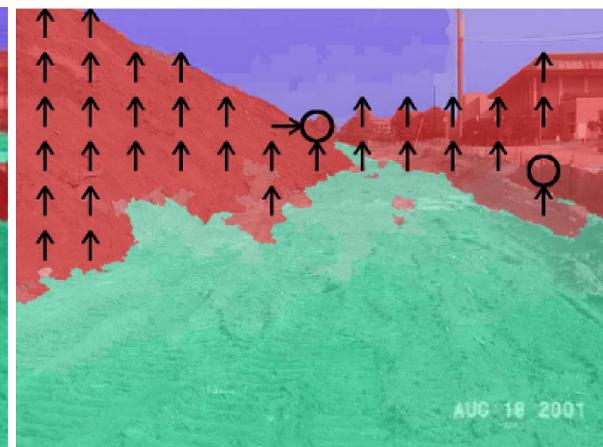
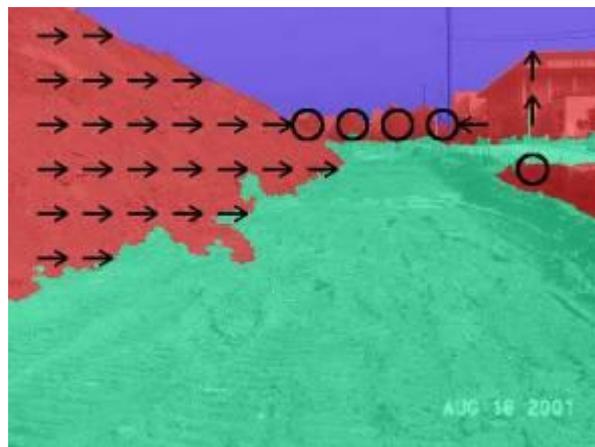


V-Porous



V-Solid

Результат

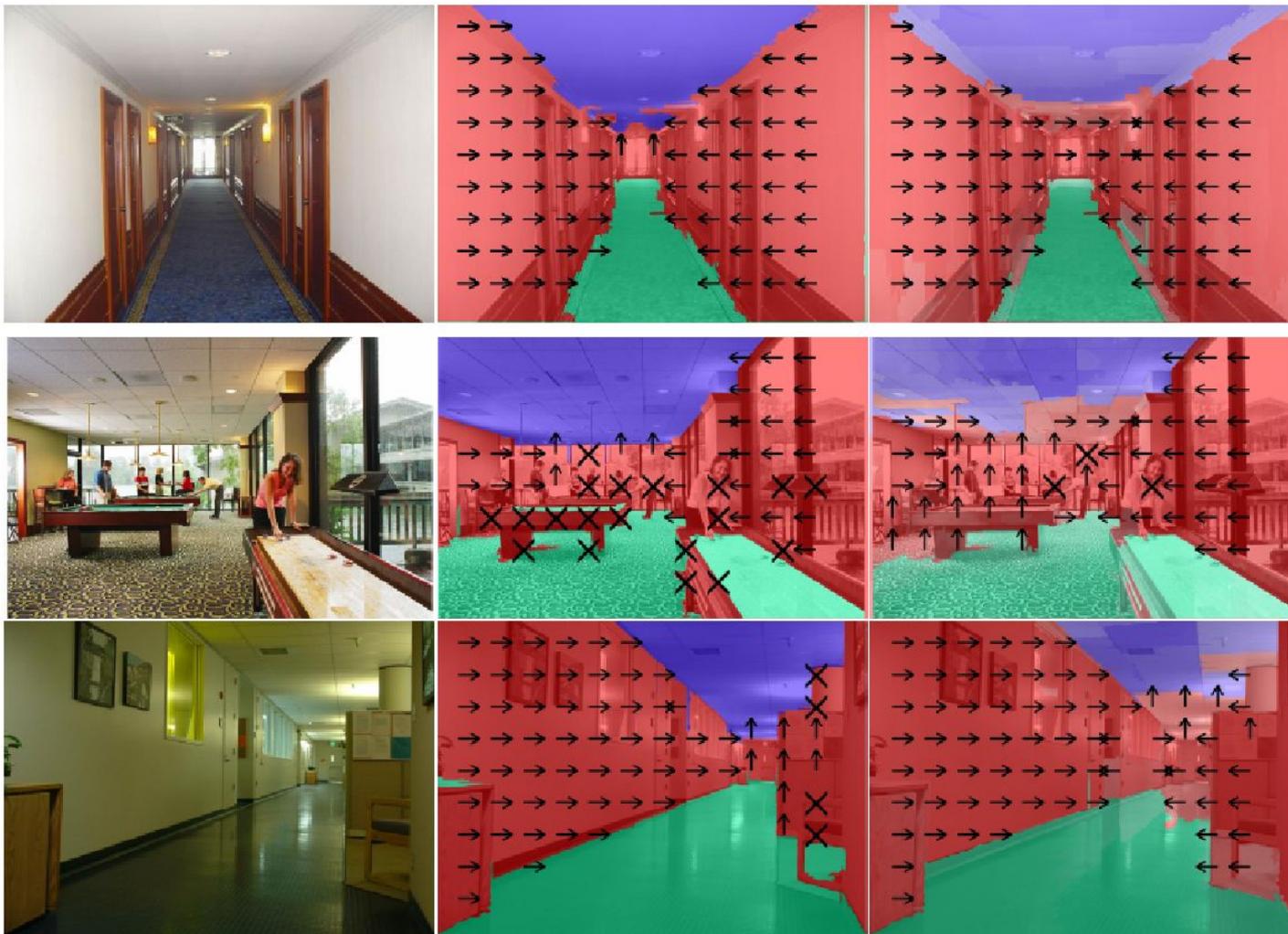


Вход

Ручная разметка

Результат алгоритма

Изображения из помещений

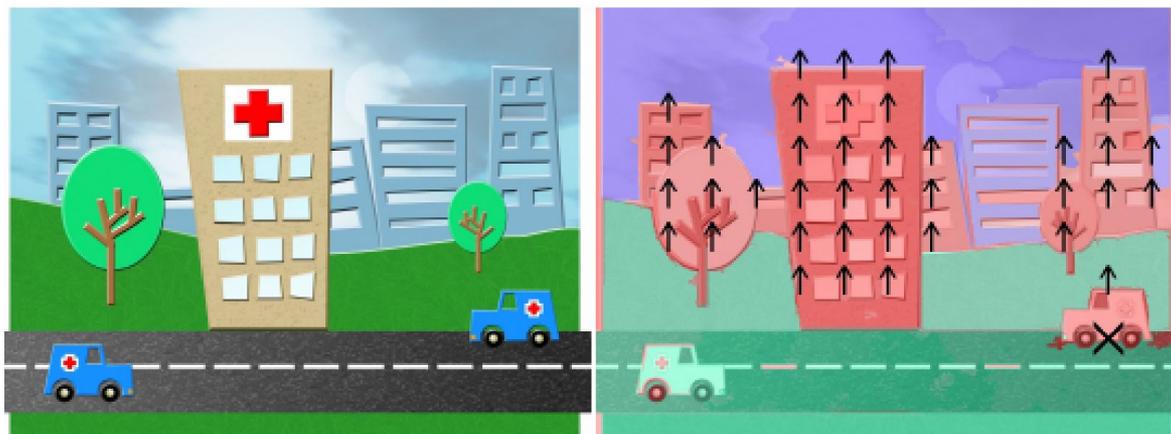


Вход

Ручная разметка

Результат

Рисунки



Вход

Результат

Приложение: Automatic Photo Pop-up (SIGGRAPH'05)



Изображение



Метки



Нижняя линия



Раскладка



Новый вид

Automatic Photo Pop-up

Automatic Photo Pop-up

D. Hoiem A.A. Efros M. Hebert
Carnegie Mellon University

TextonBoost



- J. Shotton, J. Winn, C. Rother, A. Criminisi, *TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation*, ECCV 2006

Data and Classes

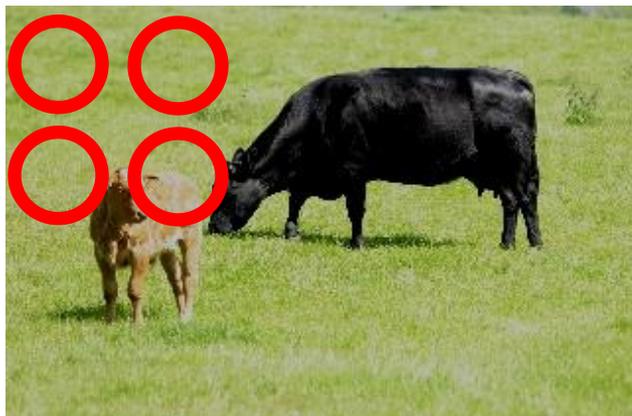
- Goal: assign every pixel to a label

<i>Object classes</i>	Building	Grass	Tree	Cow	Sheep	Sky	Aeroplane	Water	Face	Car
Bike	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat

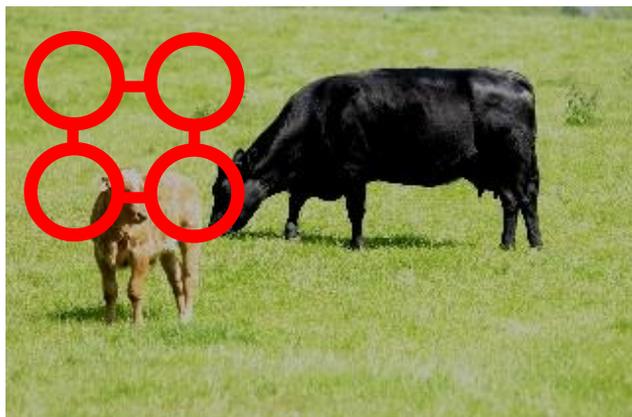
- MSRC-21



Марковские Случайные Поля



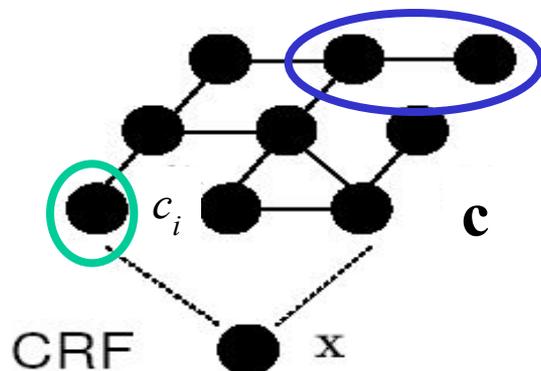
- Независимая классификация
 - Применяем обычный метод классификации (SVM, бустинг и т.д.)



- Схема Марковских Случайных Полей (MRF) для совместной классификации
 - Каждый пиксел – вершина неориентированного графа
 - Связи между пикселями задаются ребрами графа

Условные случайные поля

- МСП для совместной оценки разметки случайных переменных (\mathbf{c}), при условии всех данных (\mathbf{x})



- Модель совместного распределения

$$P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \in V} \Psi_i^{(1)}(c_i, \mathbf{x}; \boldsymbol{\theta}) \prod_{(i,j) \in E} \Psi_{i,j}^{(2)}(c_i, c_j, \mathbf{x}; \boldsymbol{\theta}) \quad (1)$$

- $\Psi^{(1)}$ – модель локальной оценки качества метки
- $\Psi^{(2)}$ – модель попарной оценки качества разметки

Вывод (Inference)

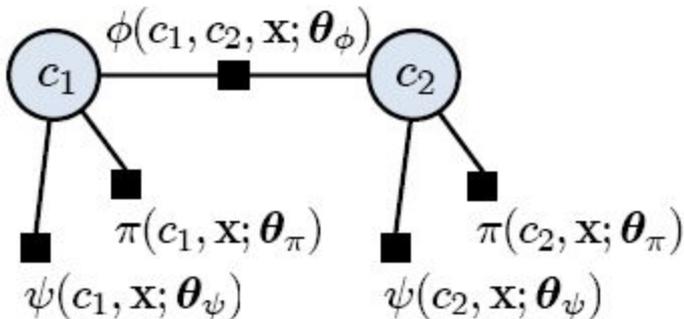
- Вывод = поиск наилучшей совместной разметки
 - NP-полная задача в общем случае
- Argmax-разметка
 - Парные потенциалы должны удовлетворять условию субмодулярности
 - Разрезы графов (GraphCuts)
 - Не-субмодулярные потенциалы
 - Quadratic Pseudo-Boolean Optimization (QPBO)
- Разметка с оценкой достоверности
 - Belief Propagation, TRW
 - Приближенное решение при наличии циклов
 - Сложность экспоненциально зависит от размера клики
 - Поэтому в основном рассматриваются модели с кликой не выше 2 (парные)

Обзор метода

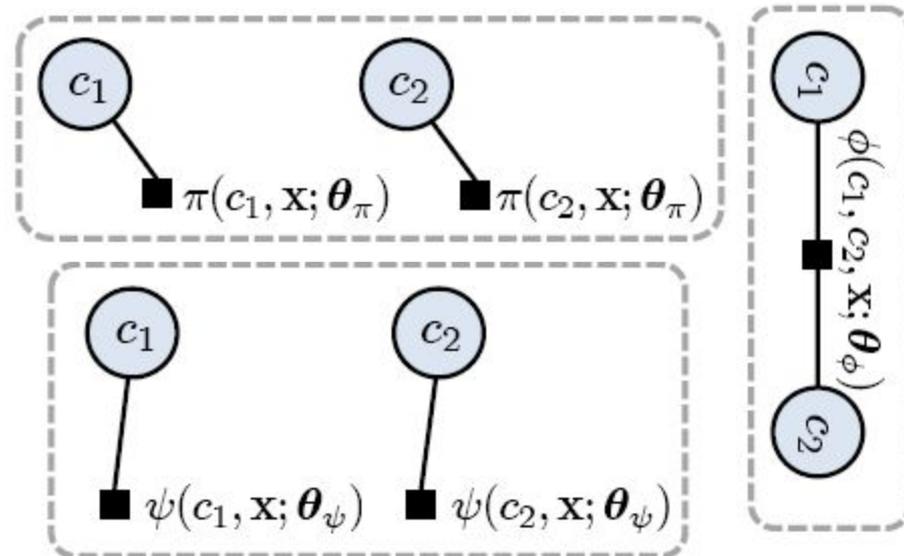
- Модель TextonBoost на основе CRF

$$\log P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \overbrace{\pi(c_i, \mathbf{x}_i; \boldsymbol{\theta}_\pi)}^{\text{color}} + \overbrace{\lambda(c_i, i; \boldsymbol{\theta}_\lambda)}^{\text{location}} \\ + \sum_{(i,j) \in \mathcal{E}} \overbrace{\phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi)}^{\text{edge}} - \log Z(\boldsymbol{\theta}, \mathbf{x})$$

- 4-х связанные окрестности
- Параметры обучаются независимо
- Вывод GraphCut



VS

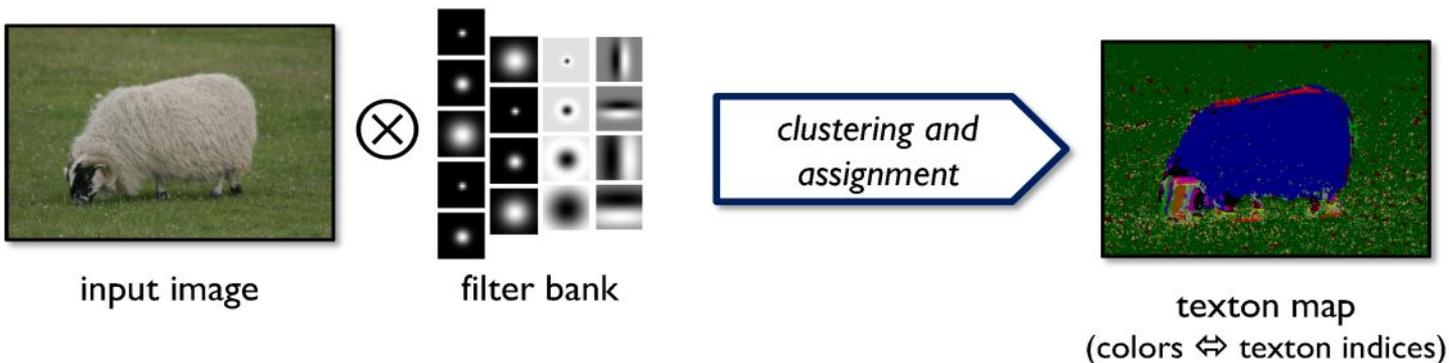


Форма и текстура (Shape & Texture)

- Первая и главная компонента модели

$$\log P(\mathbf{c}|\mathbf{x}, \theta) = \sum_i \overbrace{\psi_i(c_i, \mathbf{x}; \theta_\psi)}^{\text{shape-texture}}.$$

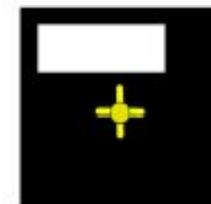
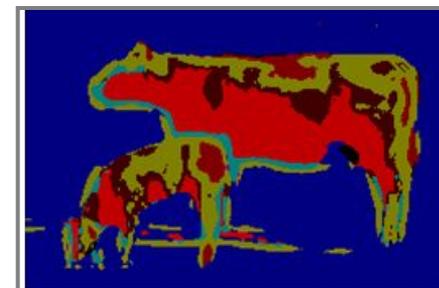
- Текстоны



- Фильтруем изображение банком фильтров (17 фильтров)
- Каждый фильтр – вычисление определенной характеристики/статистики окрестности точки
- Получаем 17 признаков для каждого пиксела (вектор-признаки)
- Кластеризуем список всех вектор-признаков (400 кластеров)
- Каждый кластер – «текстон»
- Квантуем каждый пиксель к ближайшему текстону (карта текстонов)

Моделирование формы

- Шаг 1: получили карту текстонов
- Шаг 2: Фильтры формы (Shape Filters)
 - Для каждого текстона t
 - Вход
 - » Карта текстонов
 - » (Прямоугольная маска r , текстон t)
 - » Положение пикселя i
 - Выход
 - » Площадь в маске r , отвечающая t
 - Результат – гистограмма откликов по окрестностям



rectangle r

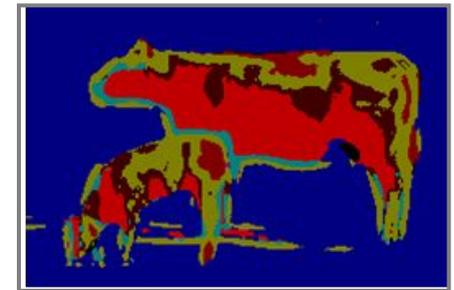


texton t



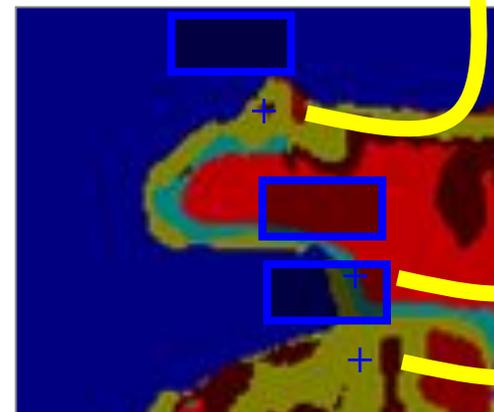
Фильтры формы

- Пара: $\left(\begin{array}{c} \text{up to 200 pixels} \\ \text{rectangle } r \\ + \\ \text{texton } t \end{array} \right)$



$$v(i, r, t) = a$$

- Отклики $v(i, r, t)$
- Большие области обеспечивают большую пространственную поддержку
- Расчет через интегральные изображения

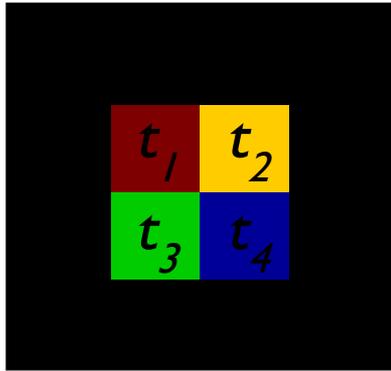


$$v(i_2, r, t) = 0$$

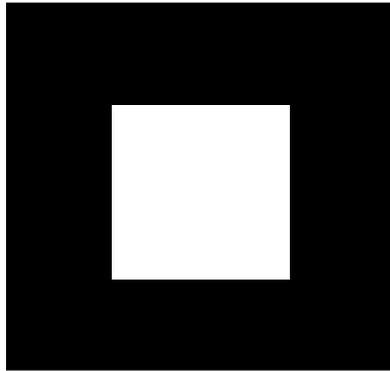
$$v(i_3, r, t) = a/2$$

appearance context

Форма задается положением текстонов

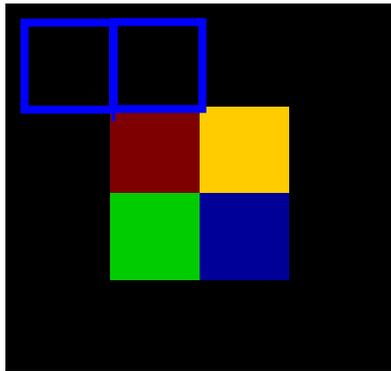


texton map

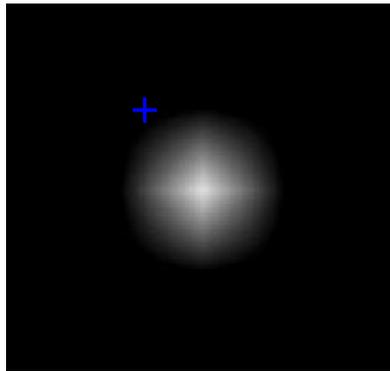


ground truth

$$(r_1, t_1) = \left(\begin{array}{c} \text{[Black square with white square and blue border]} \\ \text{[Red square]} \end{array} \right)$$
$$(r_2, t_2) = \left(\begin{array}{c} \text{[Black square with white square and blue border]} \\ \text{[Yellow square]} \end{array} \right)$$



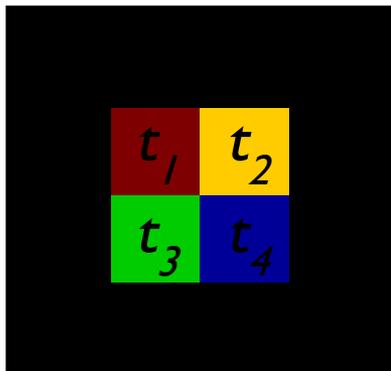
texton map



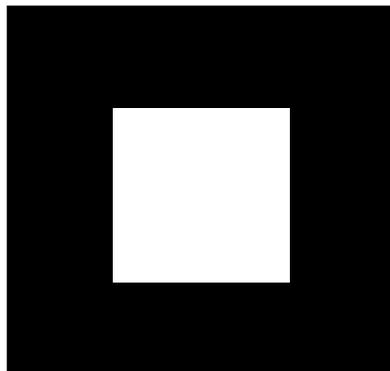
feature response image

$$v(i, r_2, t_2)$$

Форма задается положением текстонов

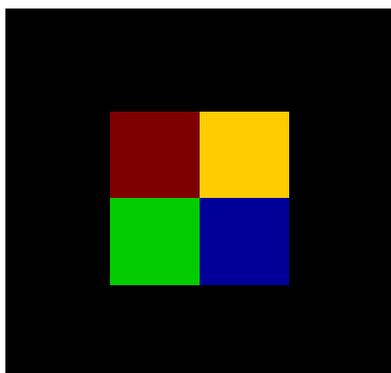


texton map

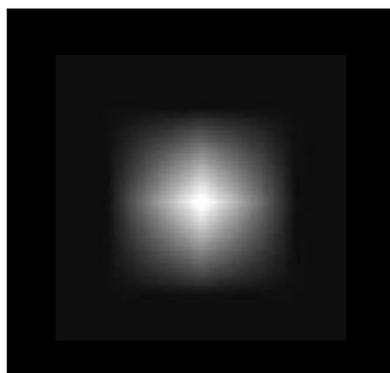


ground truth

$$(r_1, t_1) = \left(\begin{array}{c} \text{[Black image with a white square and a blue bounding box]} \\ \text{[Red square]} \end{array} \right)$$
$$(r_2, t_2) = \left(\begin{array}{c} \text{[Black image with a white square and a blue bounding box]} \\ \text{[Yellow square]} \end{array} \right)$$

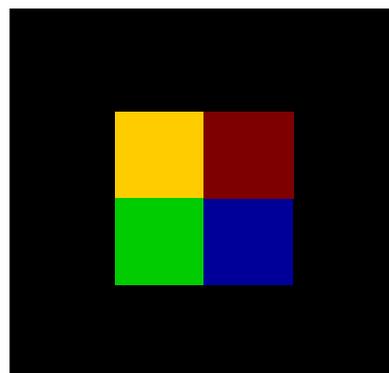


texton map

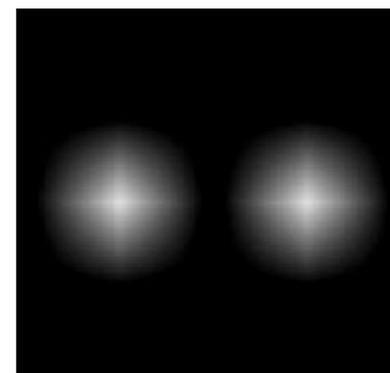


summed response images

$$v(i, r_1, t_1) + v(i, r_2, t_2)$$



texton map



summed response images

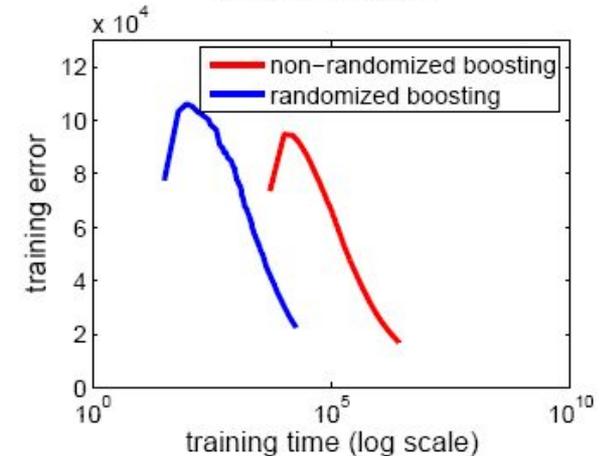
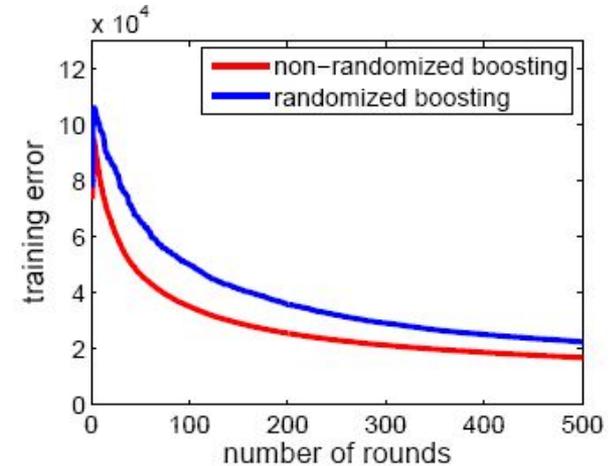
$$v(i, r_1, t_1) + v(i, r_2, t_2)$$

Обучение

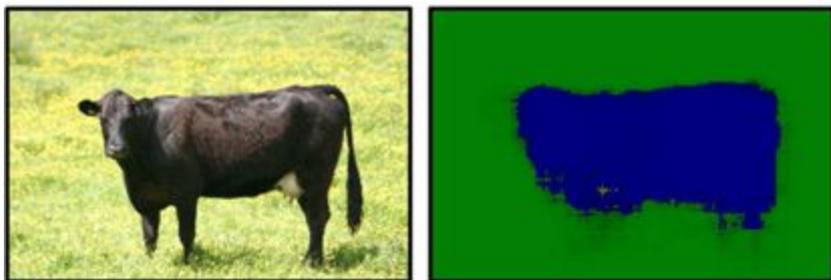
- Используется бустинг
- Обычный бустинг
 - Для каждого пикселя
 - Для каждой возможной маски
 - » Для каждого текстона
 - » Считаем признак

$$h(c_i) = \begin{cases} a\delta(v(i, r, t) > \theta) + b & \text{if } c_i \in N \\ k_{c_i} & \text{otherwise} \end{cases}$$

- Ускоренная версия
 - Для каждого пикселя в уменьшенном изображении
 - Для 10 случайных масок
 - » Для каждого текстона (K=400)
 - » Считаем признаки
- 42 часа на 276 изображениях



Первый результат



shape-texture

Только форма и текстура:

69.6%

Точность
попиксельной
сегментации

Уточняем разметку

- Добавляем границы

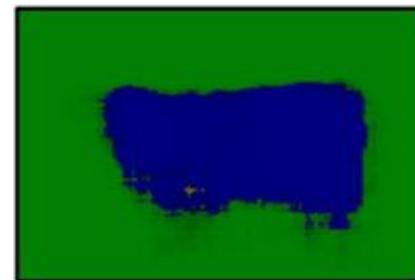
$$\log P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \sum_{(i,j) \in \mathcal{E}} \overbrace{\phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi)}^{\text{edge}}$$

- **Потенциал границ**

- Используем попарные потенциалы для определения и

$$\Gamma \phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi) = -\boldsymbol{\theta}_\phi^T \mathbf{g}_{ij}(\mathbf{x}) \delta(c_i \neq c_j).$$

$$\mathbf{g}_{ij} = [\exp(-\beta \|x_i - x_j\|^2), 1]^T$$

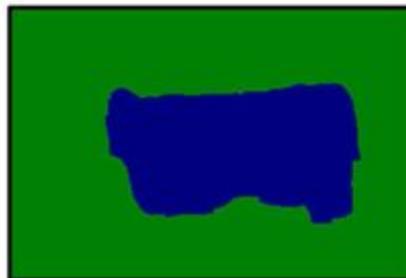
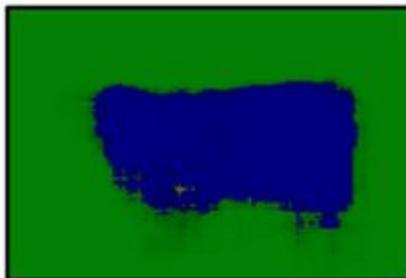


- **Идея:**

- Если метки одинаковые, разница пикселей должна быть маленькой
- Если метки разные, разница пикселей должна быть большой

- Модель Поттса, допускает разрезы графов

Точность



shape-texture

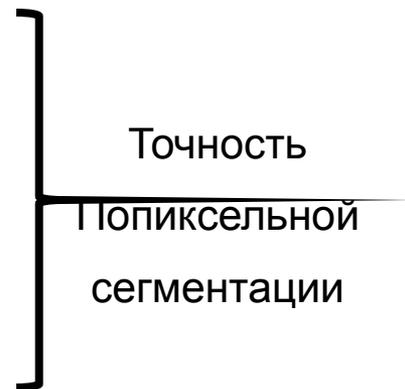
+ edge

Форма-текстура:

69.6%

+ границы:

70.3%



Положение объектов

- **Положение**

$$\lambda_i(c_i, i; \theta_\lambda) = \log \theta_\lambda(c_i, \hat{i})$$

- Нормализуем координаты по всем изображениям

- Посчитываем частоту появления объектов в данной точке изображения;

$$\theta_\lambda(c_i, \hat{i}) = \left(\frac{N_{c_i, \hat{i}} + \alpha_\lambda}{N_{\hat{i}} + \alpha_\lambda} \right)^{w_\lambda}$$

Think Naïve Bayes

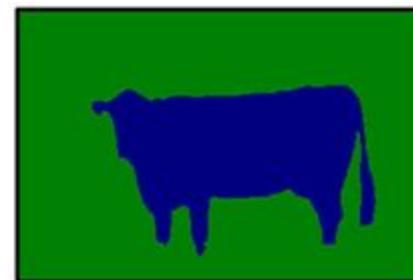
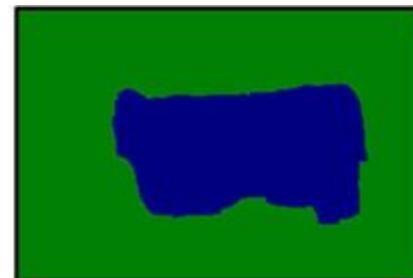
Prevent overfit (tuned)



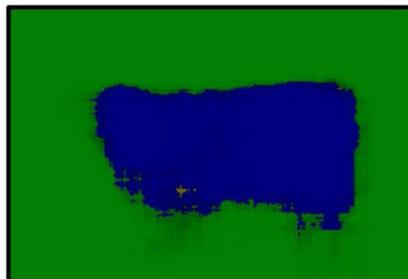
- $N_{\text{cow}, \hat{i}} = 1, N_{\hat{i}} = 3$

Моделирование цвета

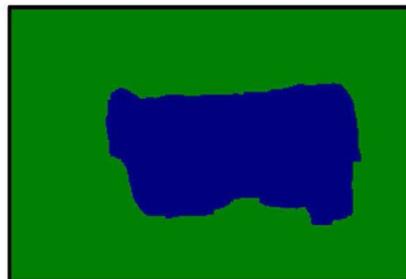
- **Цвет** $\overbrace{\pi(c_i, x_i; \theta_\pi)}^{\text{color}}$
- Обучаем модель цвета только по изображению
- **Идея**
 - Используем классификацию по другим признакам как исходные параметры
 - Обучаем модель смеси гауссиан (кластеризация цветов)
 - Каждый класс – свои веса смеси
 - Обучаем веса итеративным EM-алгоритмом



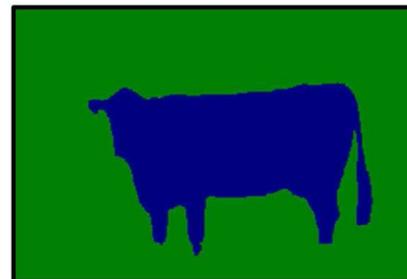
Общий результат



shape-texture



+ edge



+ colour & location

Форма и текстура:

+ границ:

+ цвет:

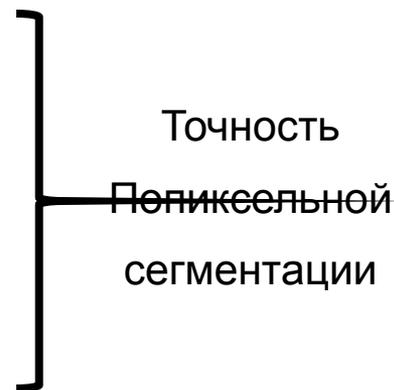
+ положение:

69.6%

70.3%

72.0%

72.2%



Ошибки



Завтра

Сказ о том, как Алеша Ефрос жульничал и грубил в компьютерном зрении за счет Гугла, Яндекса и их пользователей, и на этом прославился

