Approximate Scene Geometry From a Single View through Learning and Optimization

Olga Veksler

University of Western Ontario

joint work with  Xiaoqing Liu

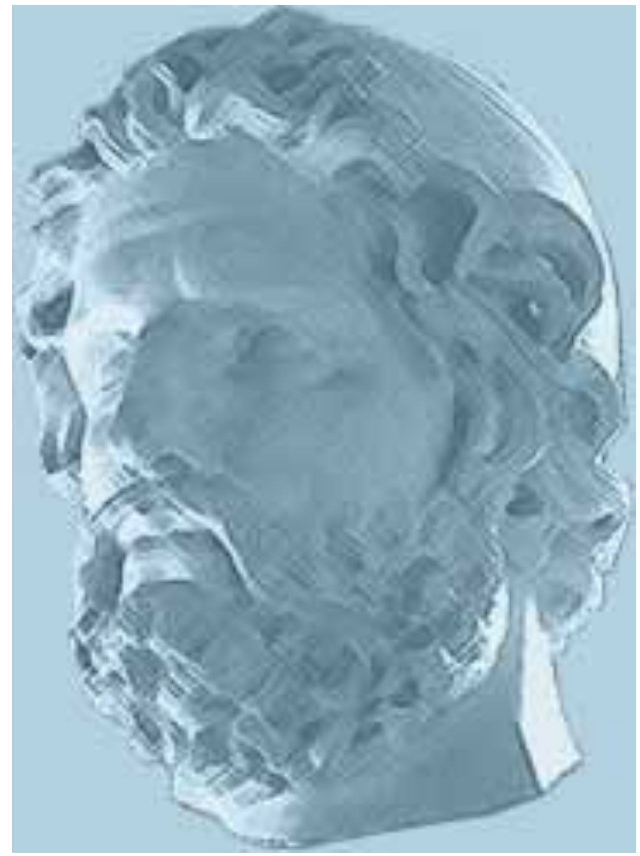
Some slides are from Alexei Efros, Deric Hoiem, Steven Seitz

Western

Outline

- How do we extract 3D information from 2D images?
  - Cues from two (or multiple) views
  - Single view cues
- *Geometric Class Scene Labelling* by Hoiem, Efros, and Hebert
  - first automatic single-view reconstruction method applicable to general scenes
- Our work: optimization methods for improving Geometric Class Scene Labeling

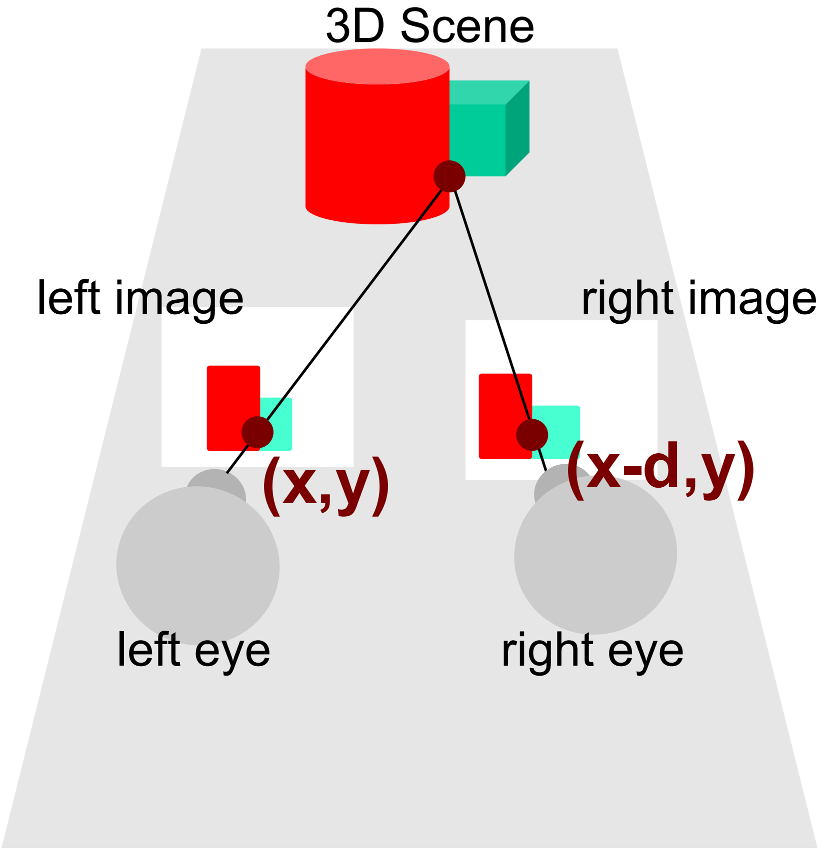
Why do we have two eyes?



Cyclope **vs.** **Odysseus**

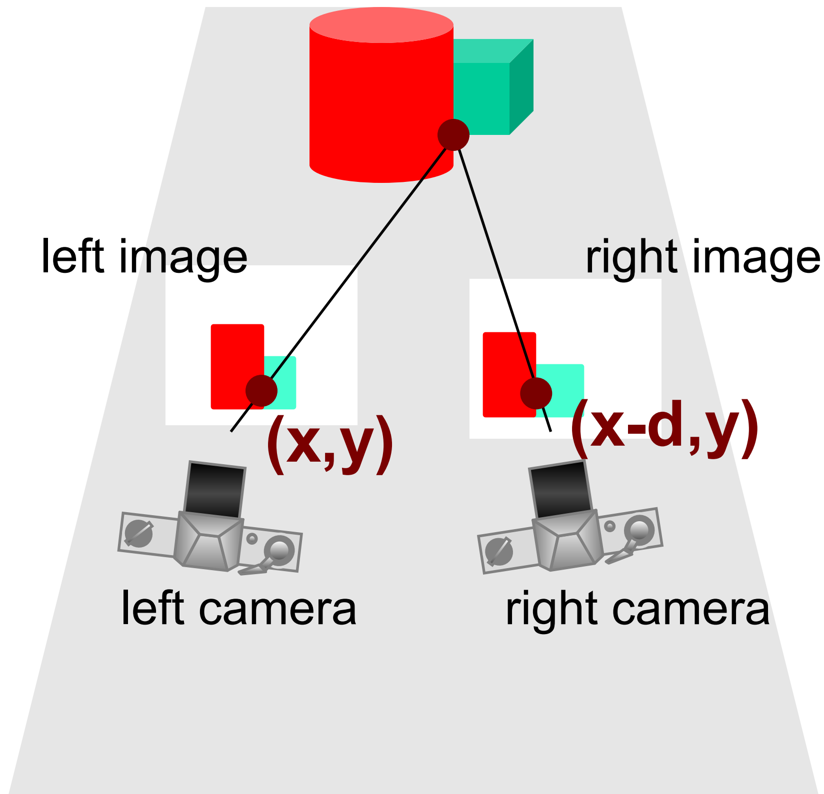
Why do we have two eyes?

- For stereopsis, first described by *Charles Wheatstone* in 1838



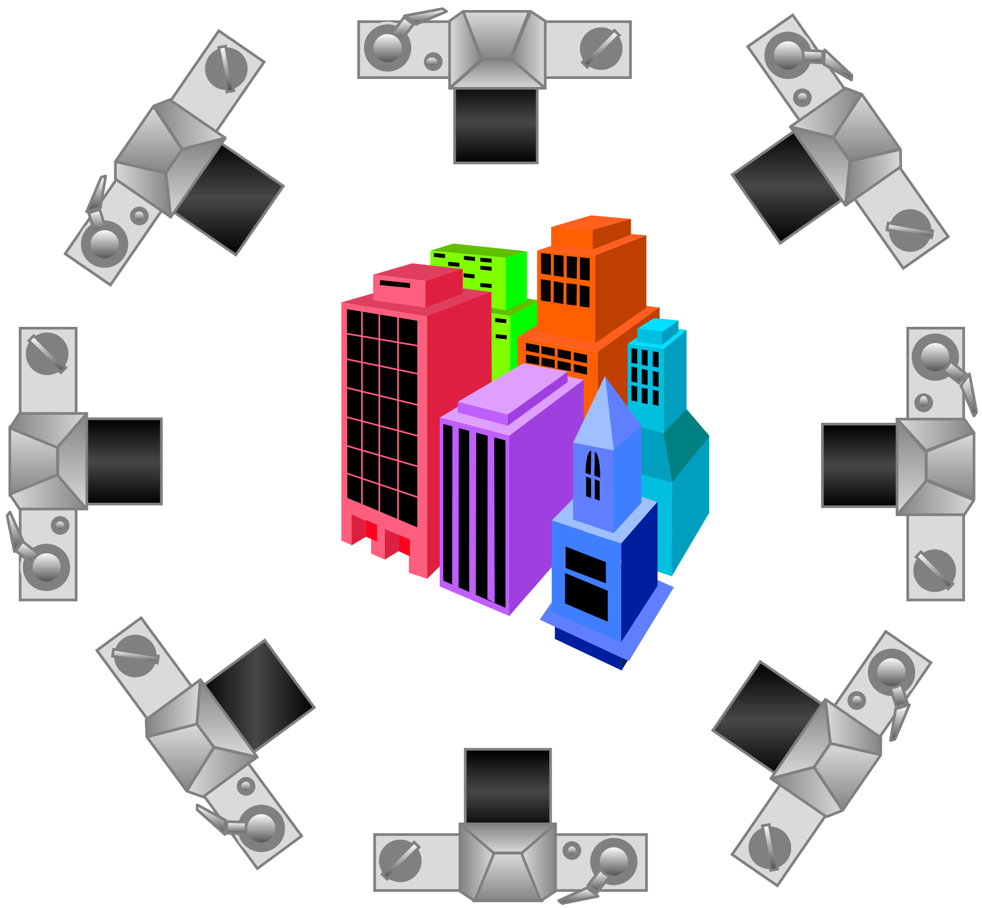
Artificial Pair of eyes

- Depth (or the third coordinate) can be extracted from the disparity between the x coordinates of the *corresponding points* in the two views
- Finding the *corresponding points* is a notoriously difficult problem in vision

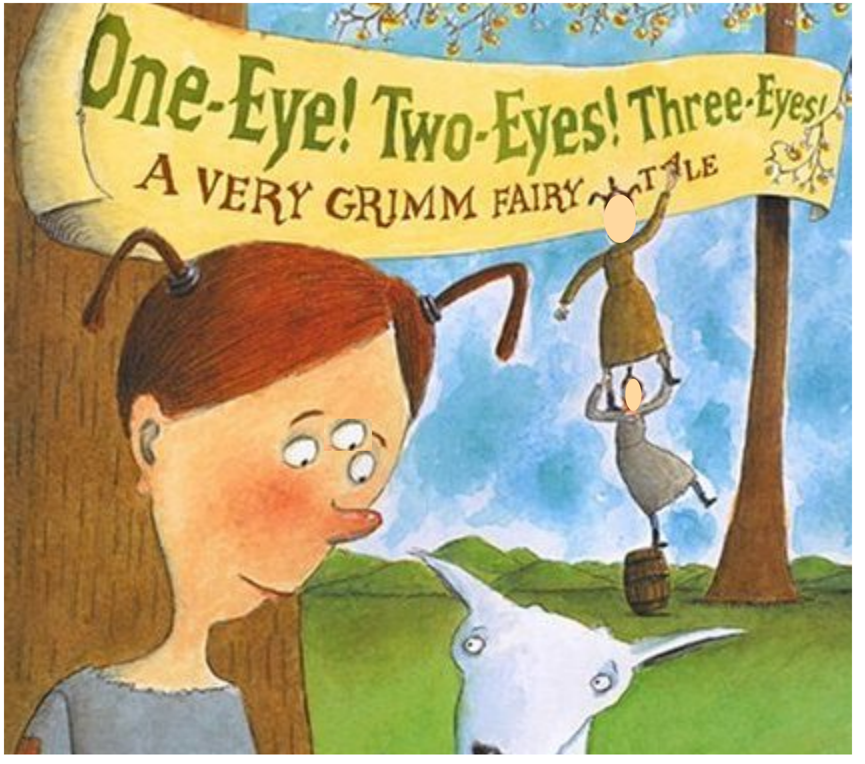


Multiple Artificial Eyes

- Two eyes are better than one eye….therefore many eyes are better than two eyes



Common Folk New that Already

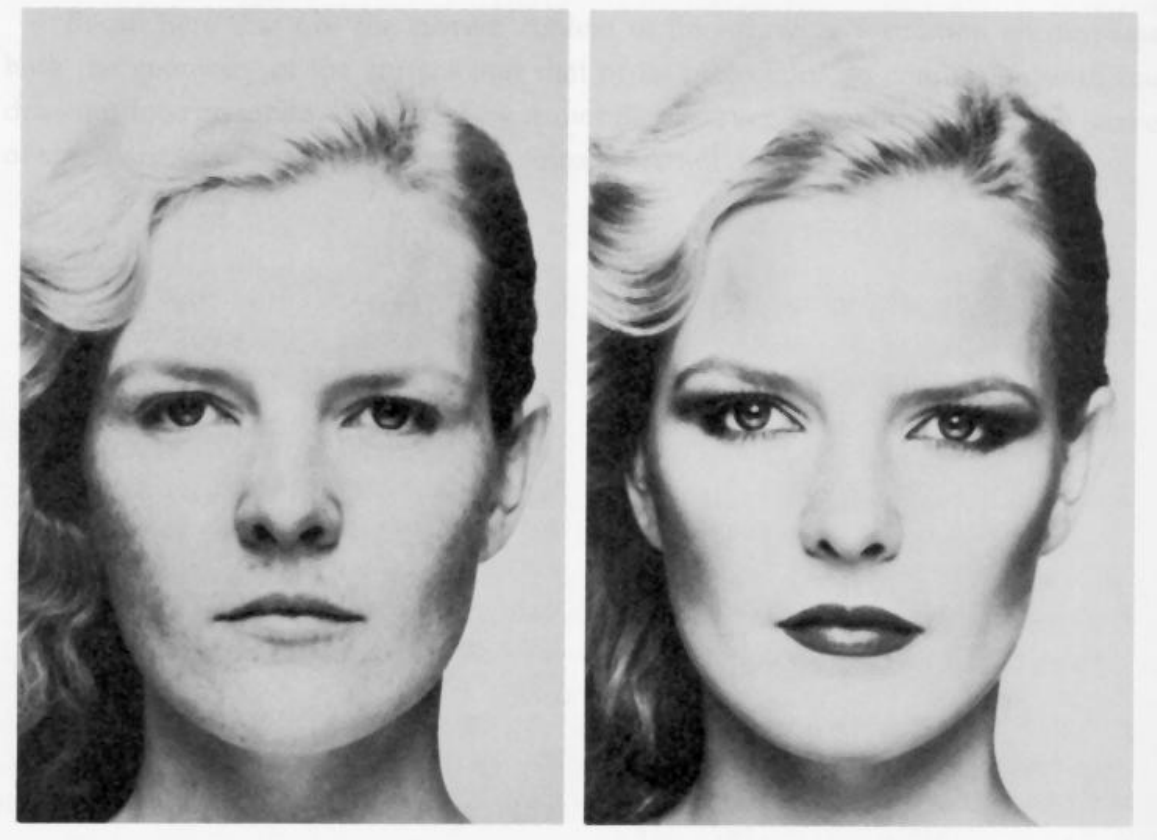


Multiple Artificial Eyes

- Do people with one eye perceive the world as one dimensional?
- No, there are multiple 3D cues from a single image
- In fact, sometimes single view cues are stronger than multiple view cues, leading to funny illusions

Single Image 3D Cues: Shading

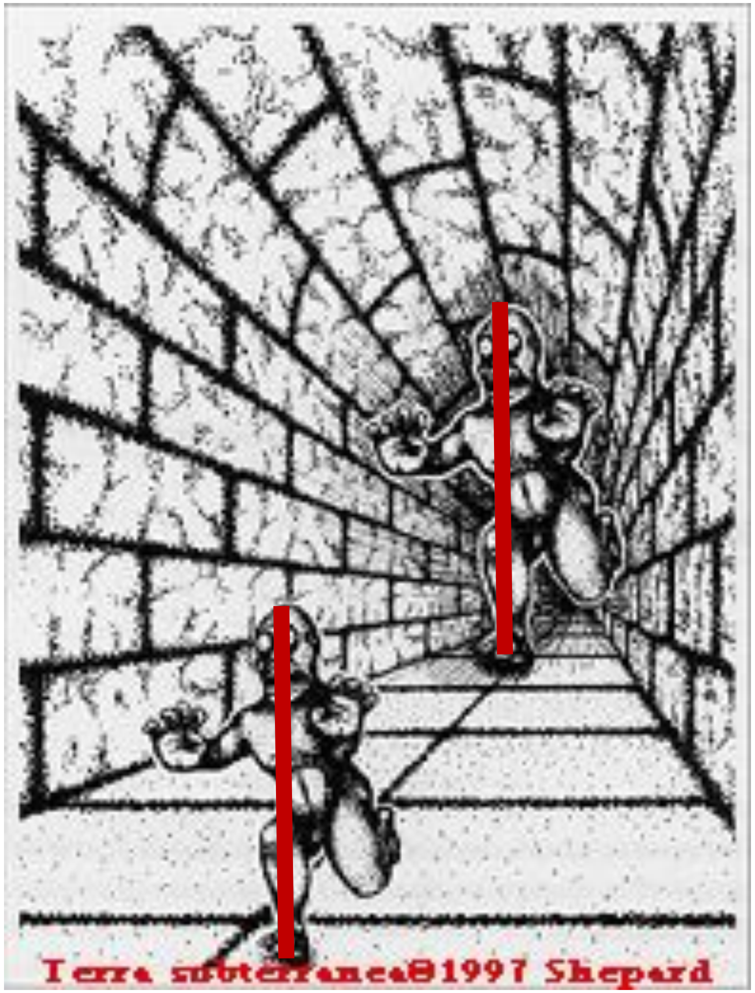
Pixels covered by shadow are perceived to be further away



Merle Norman Cosmetics, Los Angeles

Single Image 3D Cues: Linear Perspective

- The further away are parallel lines, the closer they come together



Single Image 3D Cues: Texture

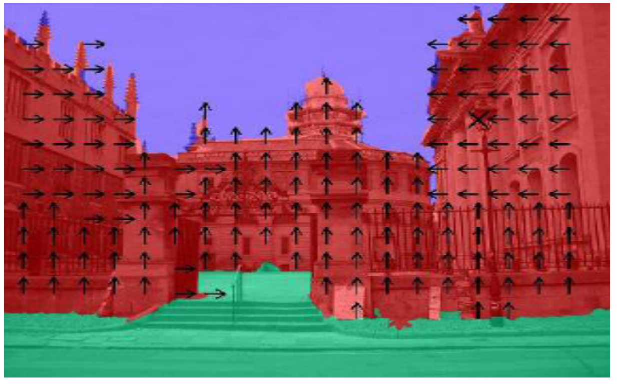
- The further away the texture is, the finer it becomes

Single Image Reconstructions

- There are single-view reconstruction algorithms based on
  - Shading
  - Texture
  - Linear perspective
- but they
  - work when there is no texture in the scene (shape-from-shading)
  - or work when there is texture in the scene (shape-from-texture)
  - or require intensive human interaction (shape from linear perspective)

Geometric Class Scene Labelling

by Hoiem, Efros, Hebert



- First automatic single-view approach applicable to any scene (reconstruction is non-metric)
- Goal: label each image pixel with one out of 7 *Geometric Classes*:
  - **Support (ground)**
  - **Vertical**
    - Planar: facing Left (←), Center (↑), Right (→)
    - Non-planar: Solid (X), Porous or wiry (O)
  - Sky

Geometric Class Scene Labelling [HEH]

- Geometric class labelling gives a rough 3D structure

[HEH] Main Idea: Learn from Labeled Data

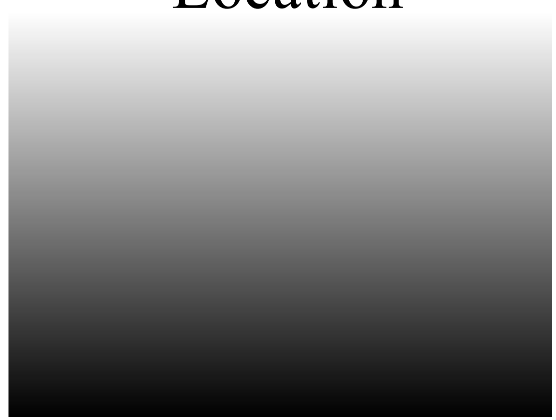
- Learn appearance-based models of geometry

[HEH] Geometric Cues

Color

Texture

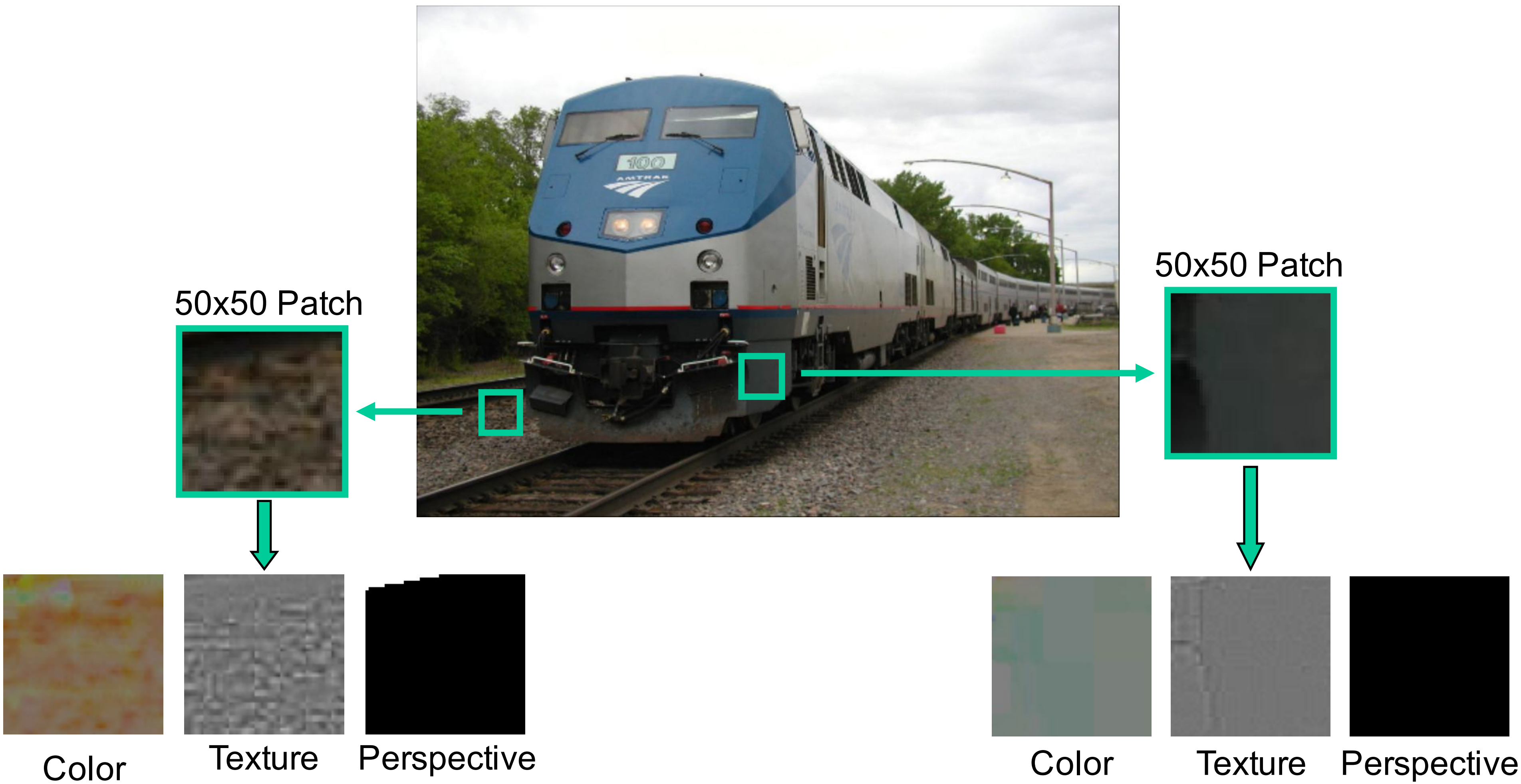
Location



Perspective

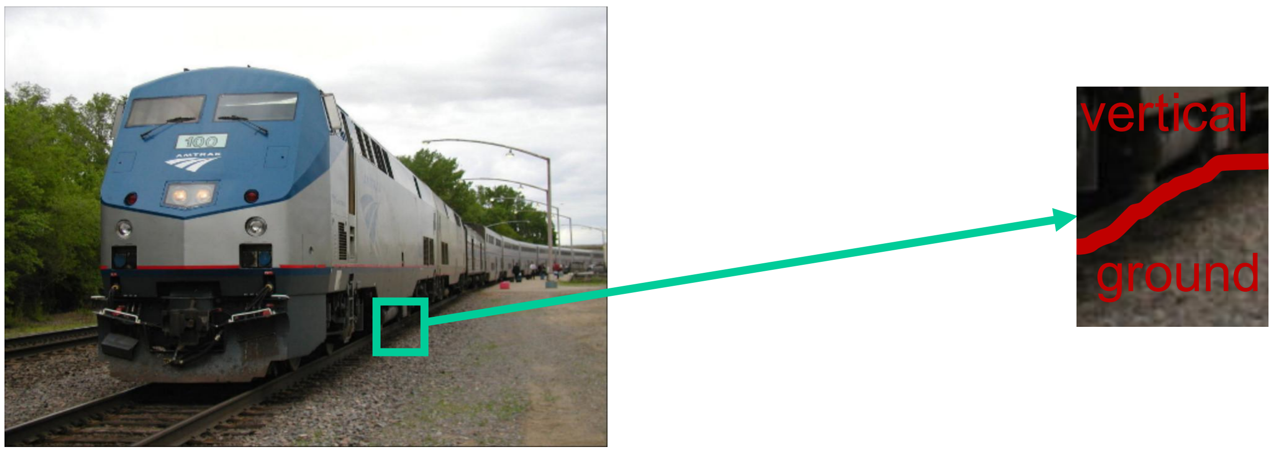
[HEH] Need Spatial Support

- Most geometric cues are properties of an image patch, not a single pixel!
- Could compute geometric cues in a square image patch



[HEH] Problem with Square Patches

- Many square patches contain pixels from different geometric surfaces



- Half of the patch contains very distinct “gravel” texture, and should be easy to classify as “ground”
- However, if texture statistics is computed inside the whole patch, “gravel” texture statistics becomes contaminated by the pixels that belong to the train
- It is not possible to classify this patch correctly

[HEH] Better Spatial Support

RGB Pixels

Superpixels

[Felzenszwalb and Huttenlocher 2004]

- Segment each training image into “superpixels”
- Superpixels are less likely to be in more than one geometric class
- Extract colour, texture, location, and perspective features from each segment
- How do we assign each superpixel its geometric class?
- Through *supervised machine learning*!

Digression into Supervised Machine Learning

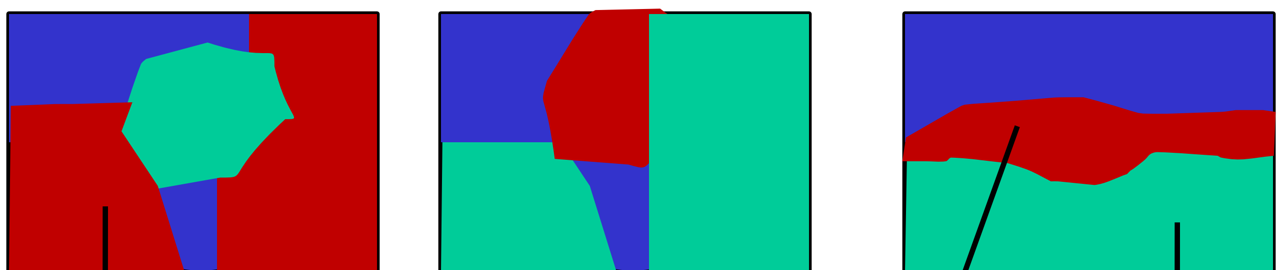
- Collect *training examples* or *training data*: $x^1, x^2, \ldots x^n$
- Each example is typically multi-dimensional, each feature corresponds to a dimension
  - $x^i = [x^i_1, x^i_2, \ldots, x^i_d]$ =[feature *1*, feature 2,…,feature *d*]
- Know desired output or "label" for each example: $y^1, y^2, \ldots, y^n$
  - How? Usually assign label by hand. Takes a bit of time.
- Now develop a *classifier* $f(x,w)$ s.t.

$$f(x,w) = \text{true label of } x$$

  - the goal is, of course, that $f(x,w)$ gives the correct label even if x is not in the training data
  - classifier that performs well on unseen (not training) data is said to generalize well

Digression into Supervised Machine Learning

- Training data for Geometric Class Labeling:
  1. Get natural images, segment them in superpixels and label them (by hand ;)



  2. Extract color, texture, perspective, and location cues from each superpixel

[*1.2, 7.3, 1.0,1.2*] [*1.5, 2.6, 1.8,3.9*] [*0.5, 3.6, 3.8,0.9*]

Digression into Supervised Machine Learning

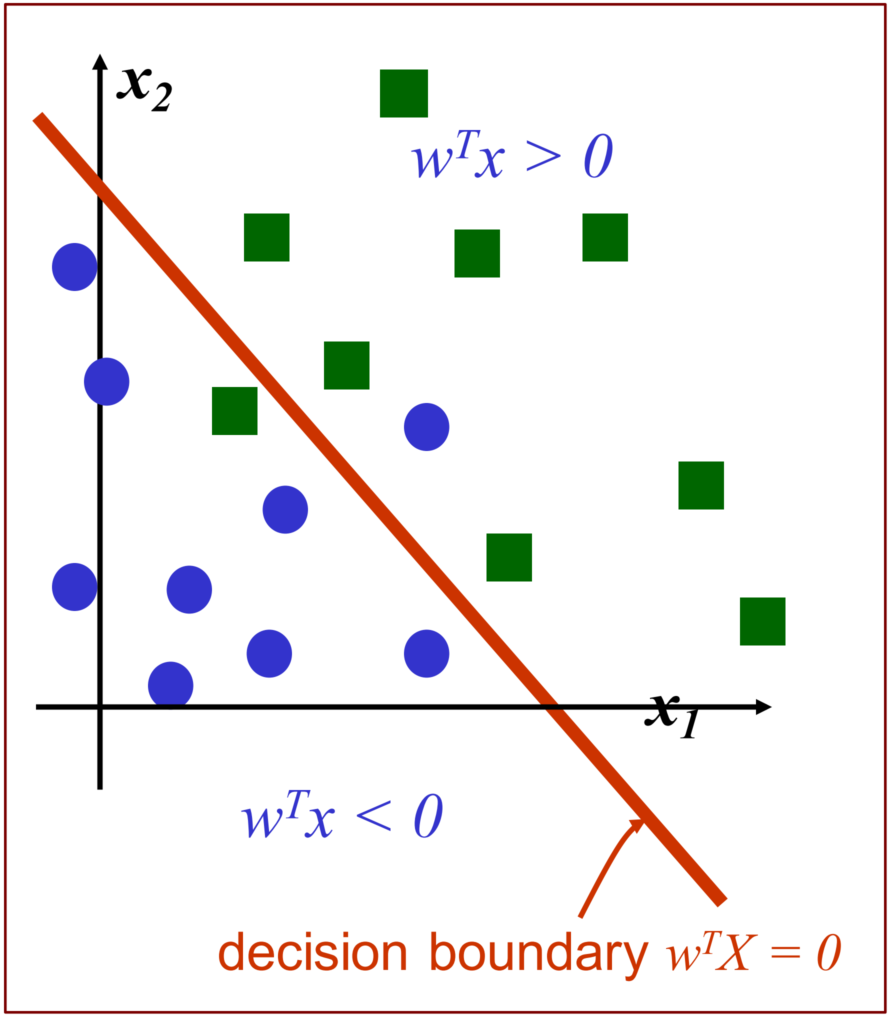
- Wish to develop a classifier

  $f(x,w)$ s.t. $f(x,w)$ = true label of x

  - How do we choose f? A particular choice of f corresponds to making implicit assumptions about the problem
- $w = [w_1, w_2, \ldots w_k]$ is typically a multidimensional vector of weights (also called parameters) which enables the classifier to "learn" or be "trained"
- Training is just finding a vector w such that for training examples x, $f(x,w)$ = true label of x "as much as possible"
  - "as much as possible" can be defined precisely
  - after training, we have learned "good" set of weights w^t
- The hope is that after training, $f(x,w^t)$ = true label of x for all, not just training, examples x

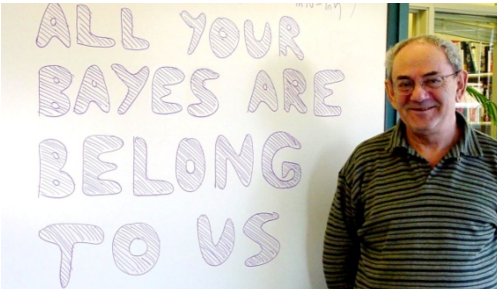
Digression into Supervised Machine Learning

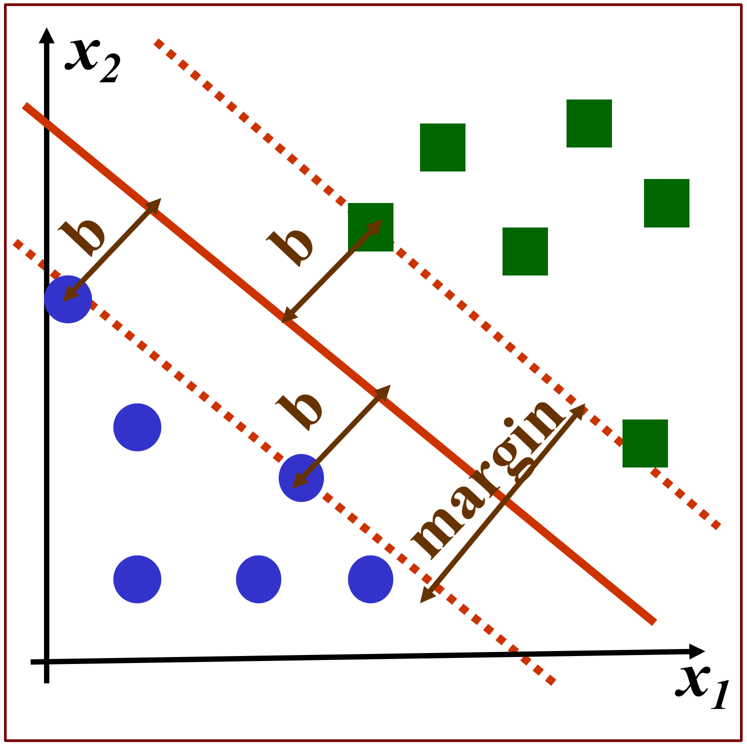
- Linear classifier is one of the simplest classifiers
- In two class case:
- $f(x,w) = sign(w_0+\Sigma_{i=1,2,...d}\, w_i x_i\,)$
  - $sign$(positive) = 1, $sign$(negative) = -1
  - w_0 is called bias
- In vector form, if we let $x = (1,x_1,x_2,\ldots,x_d)$ then $f(x,w) = sign(w^T x)$



Digression into Supervised Machine Learning

- We use Support Vector Machines (SVM) for classification (HEH use a different classifier)
- Developed by Vapnik and others



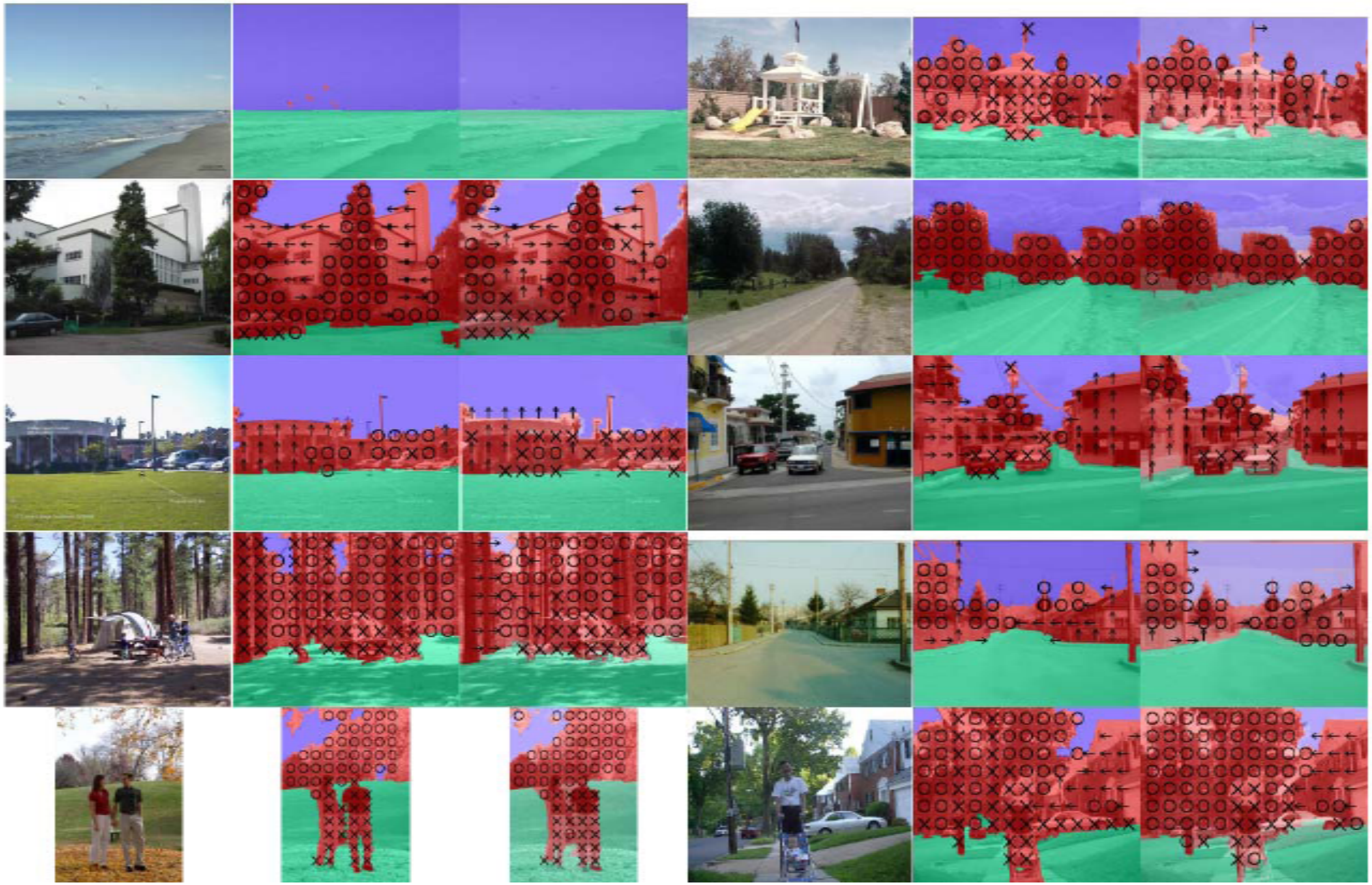


- SVM: maximize the ***margin***
  - ***margin*** is twice the absolute value of distance ***b*** of the closest example to the separating hyperplane
- Better generalization (performance on the unseen data)
  - in practice and in theory
- SVM can be generalized to non-linearly separable data

Back to [HEH]

- Geometric Class Labelling Summary
- Training phase:
  1. collect training images
  2. segment images in superpixels
  3. Extract texture, colour, location, perspective features from each segment
  4. label each segment with its geometric label
     - ground, vertical, sky, etc.
  5. train a classifier on the collected feature vectors
- Application phase, given a new image
  - segment image in superpixels
  - extract the features
  - apply the classifier to superpixels

Good Results [HEH]



Drawbacks of [HEH]

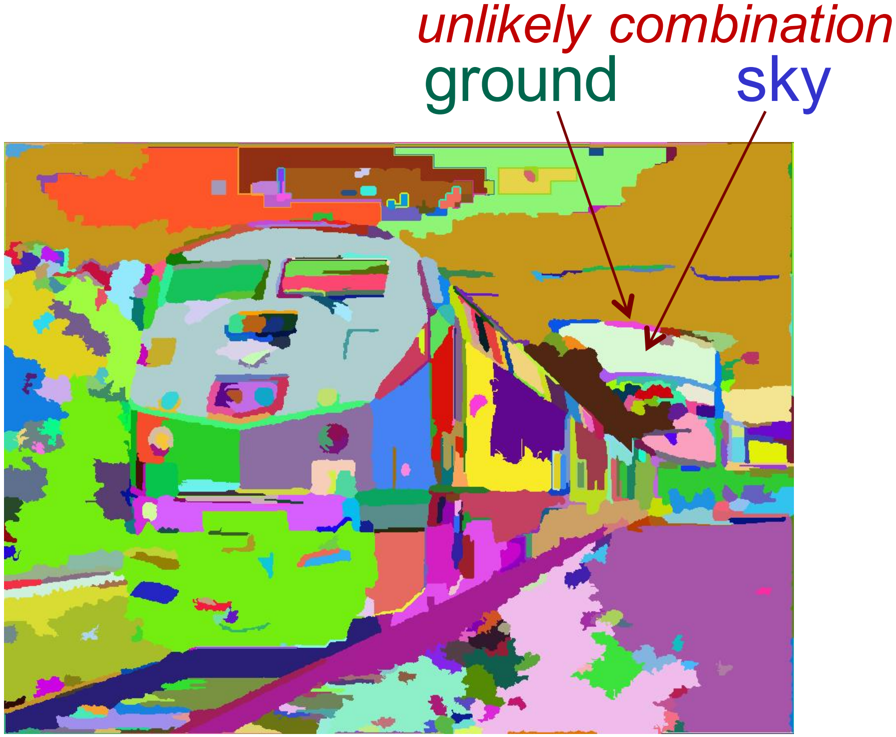
- Labelling is based on superpixels
- If a superpixel has pixels that should have different geometric labels, a mistake will be made
- [HEH] have some ad-hoc method to alleviate the severity of the problem
- but it is nicer to label each pixel individually

Drawbacks of [HEH]

- No coherence between superpixels is assumed
- But in fact, most nearby superpixels should have the same label
- Can help especially if there are many small superpixels

Drawbacks of [HEH]

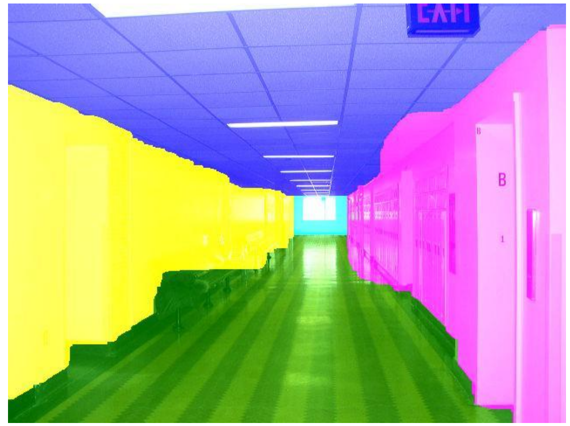
- Does not prohibit physically unlikely assignments of labels

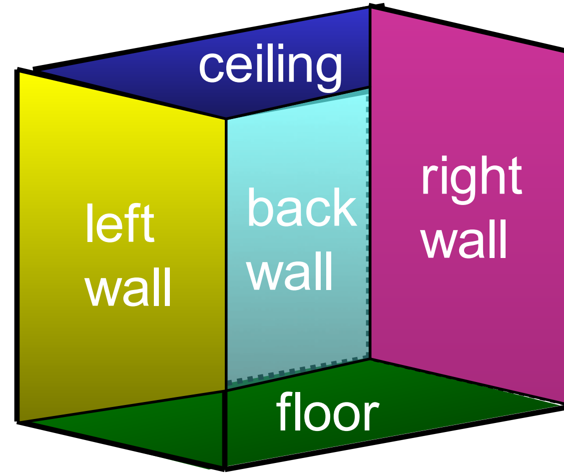


Our Work

- We address all the three drawbacks of [HEH]
  1. Label pixels individually
  2. Assume coherence between labels of nearby pixels
  3. Prohibit physically unlikely configurations
- We address (1,2,3) in a global optimization framework
- In addition, we develop an optimization algorithm, which works better than a standard one
- DRAWBACK: we deal only with a simpler model than [HEH]

Our Model



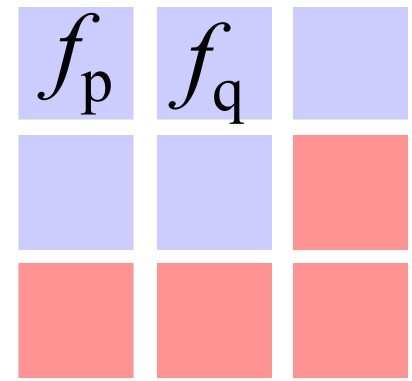


- We assume the scene consists of five parts
- Labels are “left”, “right”, “top”, “bottom”, “center”
- Natural set of restrictions:
  - “left” has to be to the left of “center” and “right”
  - “top” has to be above “center” and “bottom”
  - etc.

Global Optimization Approach

- Energy function:

labelling f



data term　　　　smoothness term

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q)$$

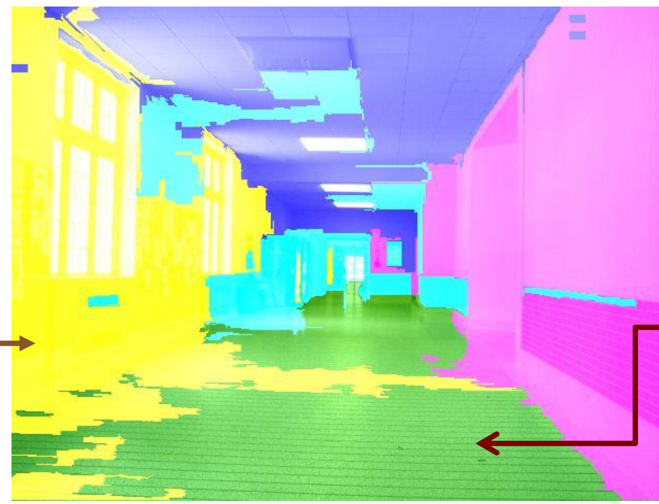
- f_p is the geometric label assigned to pixel p
- $f_p \in$ {"left", "right" ,"center" ,"top", "bottom"}
- f is the collection of all label-pixel assignments
- Have to find f that minimizes the energy above

Data Term D

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q)$$

- $D(f_p)$ is small if pixel p likes label f_p, and large otherwise
- we get $D(f_p)$ though learning, like [HEH]

small D(“left”)



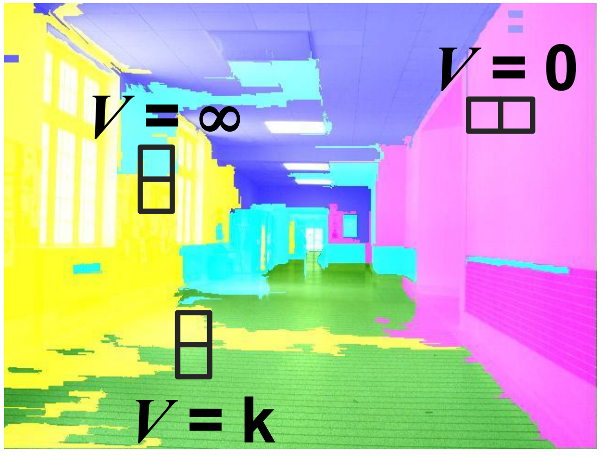
small D(“floor”)

- shows most likely label for each pixel
- this would be the optimal assignment if there was no smoothness term V in the energy function

Global Optimization Approach

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q)$$

- $$V_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q \\ \infty & \text{if} \quad (f_p, f_q) \text{ is prohibited} \\ k & \text{otherwise} \end{cases}$$

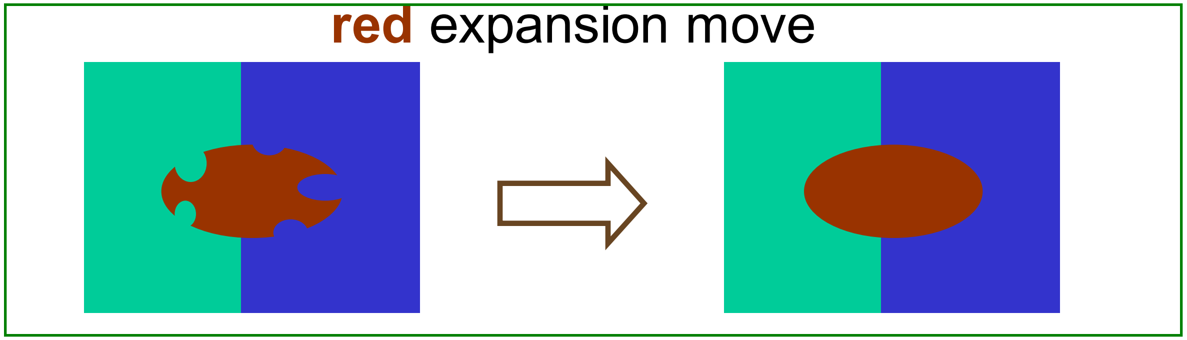


smallest energy labelling f we found

Global Optimization Approach

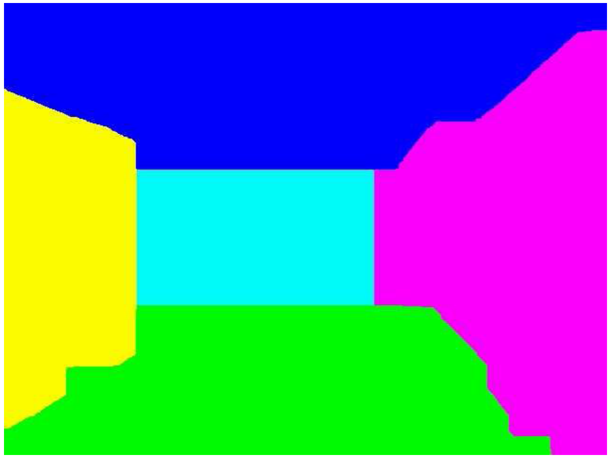
$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q)$$

- Energy is NP-hard to optimize, in general
- Have to use approximate methods
- Expansion algorithm [Boykov, Veksler, Zabih'2001]

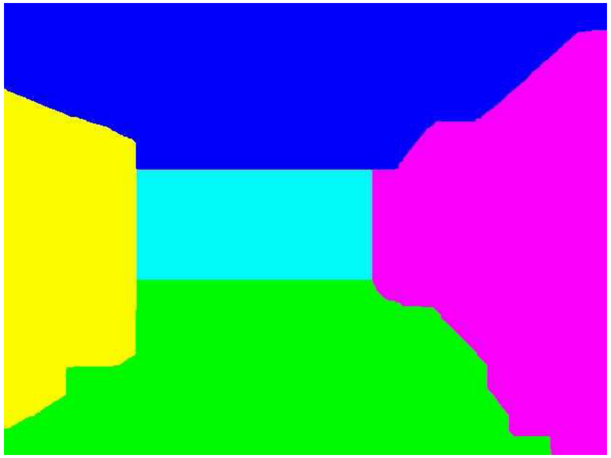


- Optimal expansion move is computed with a min cut
- Red, green, blue expansion moves are repeated until no progress can be made
- Works provably well for some energies

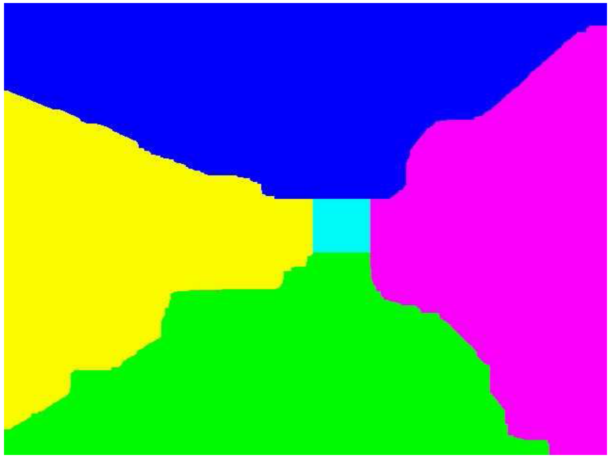
Problems with Expansion Algorithm



expansion after one iteration



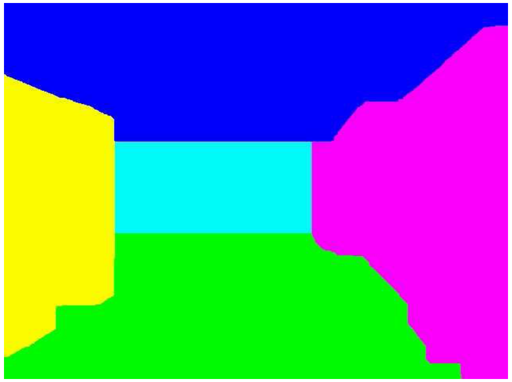
expansion after at convergence



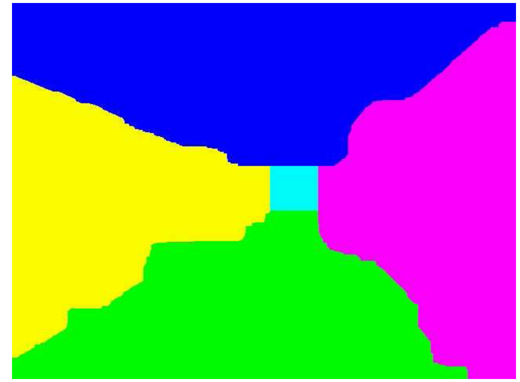
lower energy configuration

- Expansion algorithm is easier to get stuck in a local minima when prohibited assignments are present

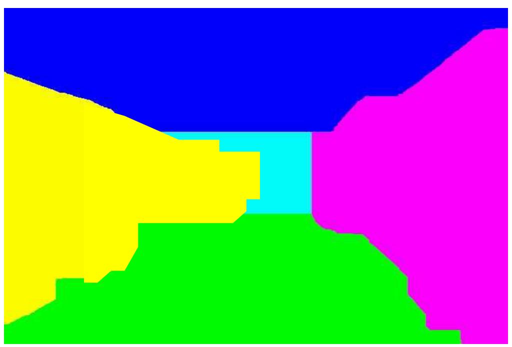
Problems with Expansion Algorithm



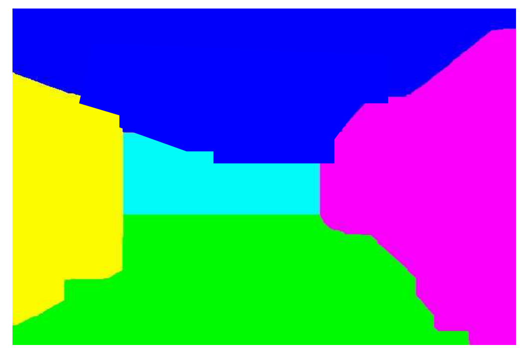
expansion after at convergence



lower energy configuration



this "yellow" expansion has infinite cost

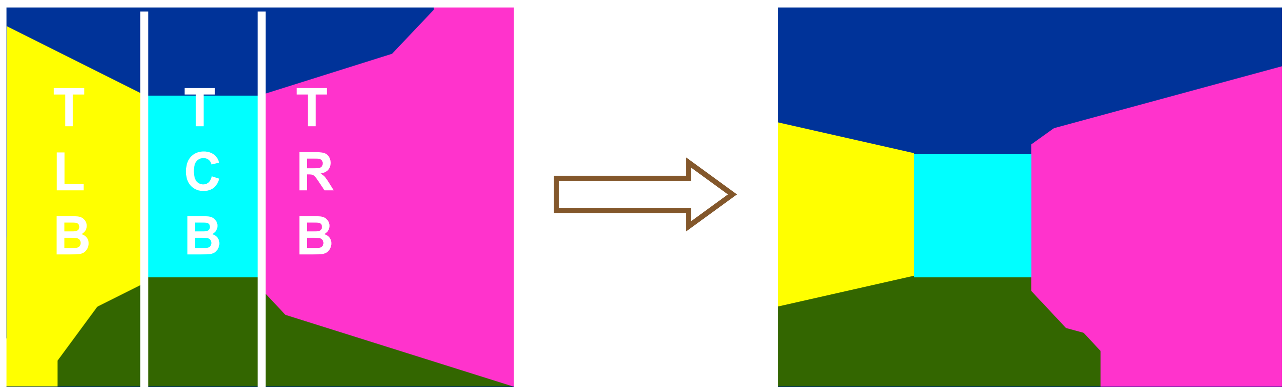


this "blue" expansion has infinite cost

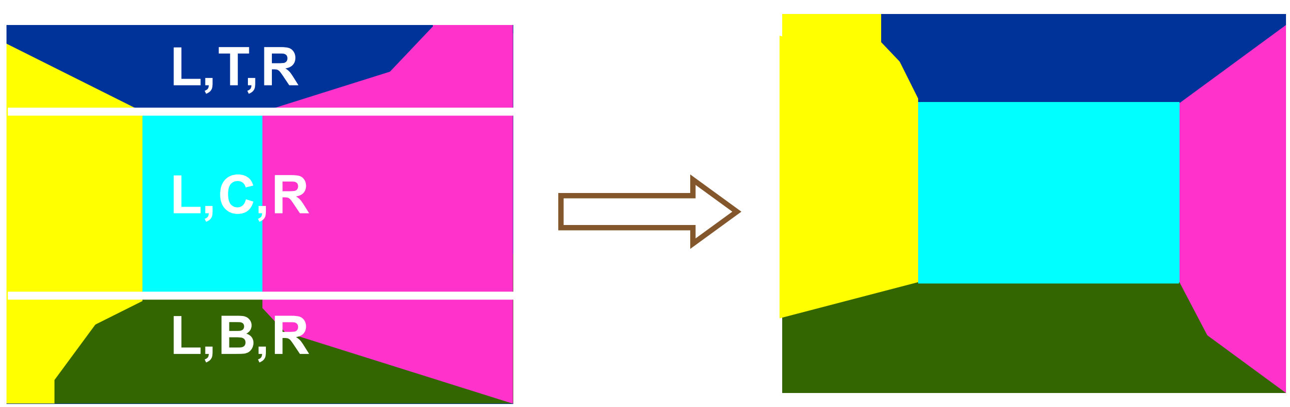
- Need moves that allow more than one label to "expand"

Order Preserving Moves

- 2 types of order-preserving moves: “vertical” and “horizontal”
  - horizontal move, width of the “center” is preserved

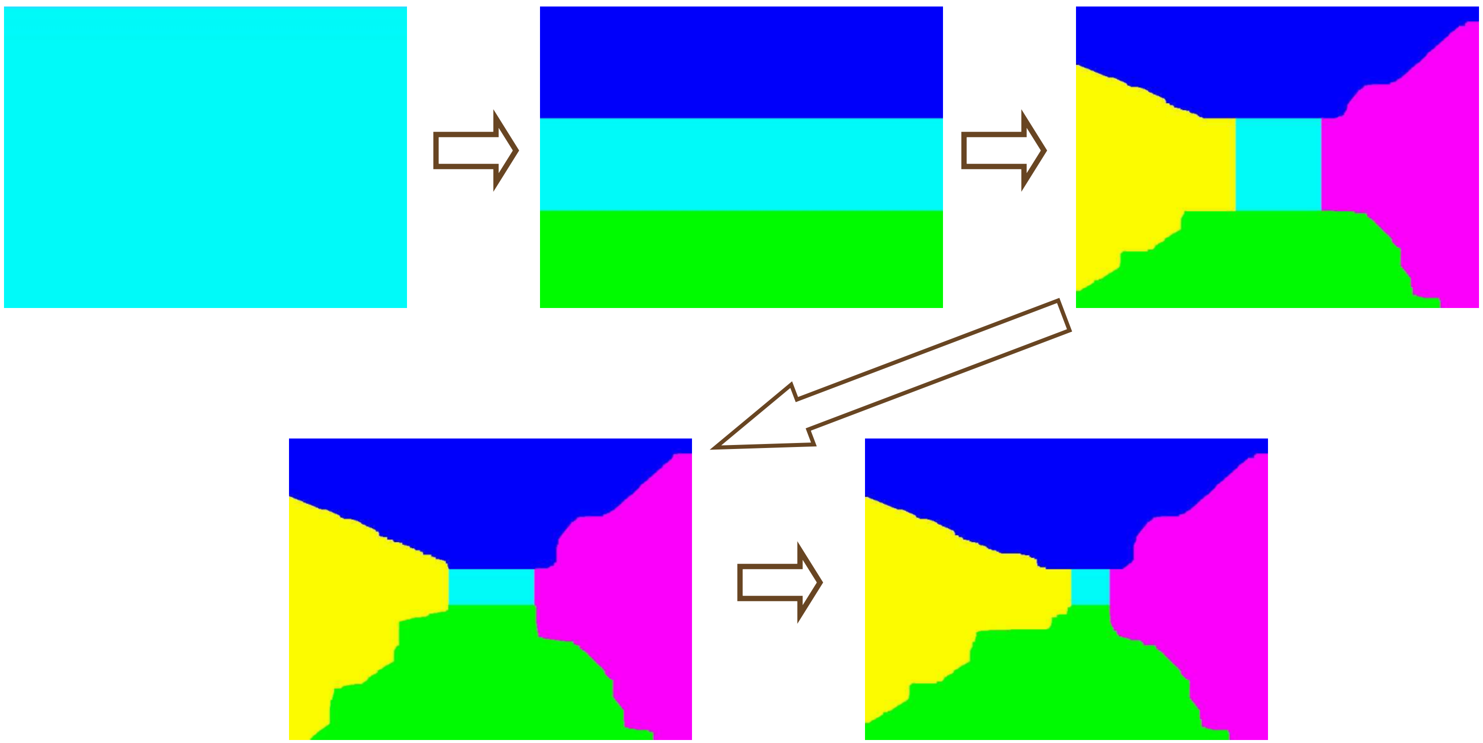


  - vertical move, height of the “center” is preserved

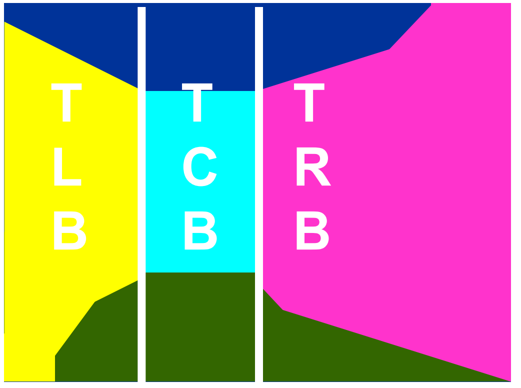


Order Preserving Moves

- Vertical and horizontal moves are alternated until convergence
- Start with labelling "all center"

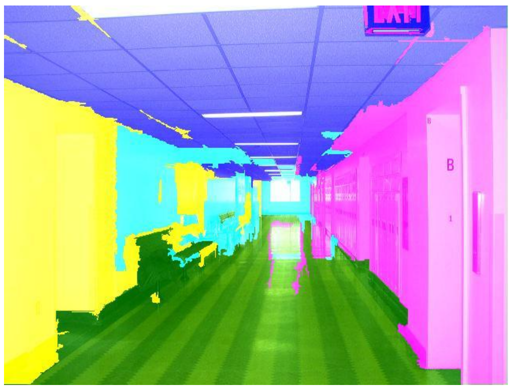


Horizontal Move

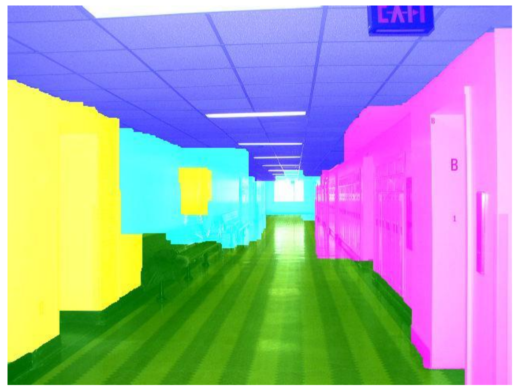


- All pixel participate in a move
- Each pixel has a choice of 3 labels, including its old label
  - The expansion algorithm allows a choice of only 2 labels
- Optimal horizontal move can be found exactly with a minimum graph cut algorithm
- Similar comments apply to the horizontal move

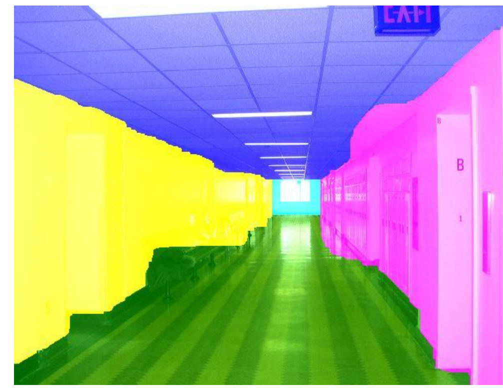
Order Preserving Moves



independent labelling, i.e. only data term is used



V_{pq} for smoothness only

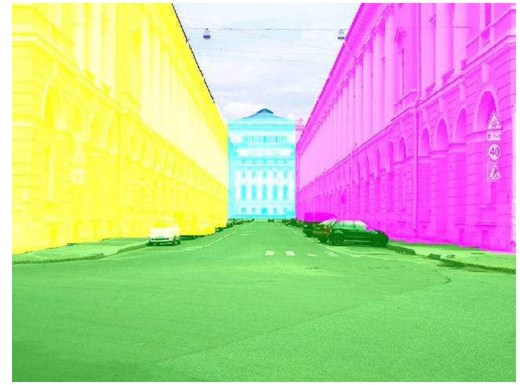


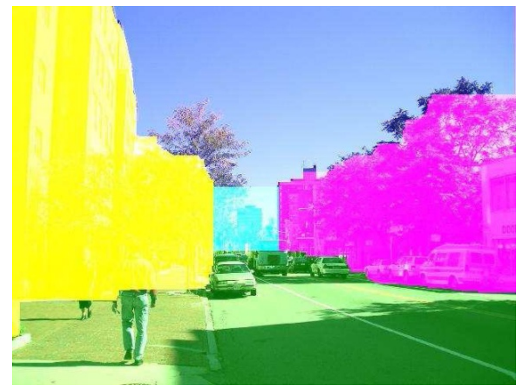
V_{pq} for smoothness and order preservation

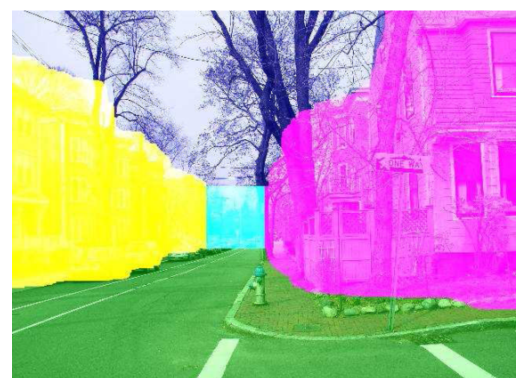
Order Preserving Moves: Results

- On 300 images, the energy, on average is about 30% smaller with order-preserving moves, compared to the expansion algorithm
- Classification results are also significantly better

Order Preserving Moves: Results





Generating Novel Views

original image

walk forward

look left

look right

look up

look down

Generating Novel Views

Conclusions?

- Learn and optimize!