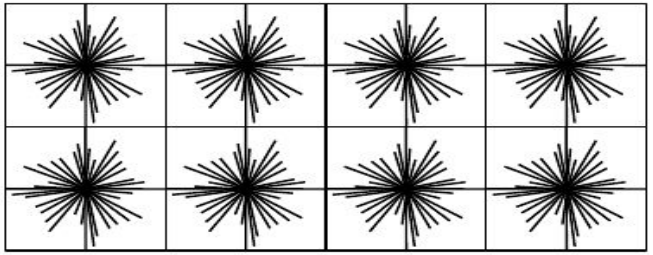
HISTOGRAMS OF ORIENTATION GRADIENTS

Histograms of Orientation Gradients

- Objective: object recognition
- Basic idea
  - Local shape information often well described by the distribution of intensity gradients or edge directions even without precise information about the location of the edges themselves.



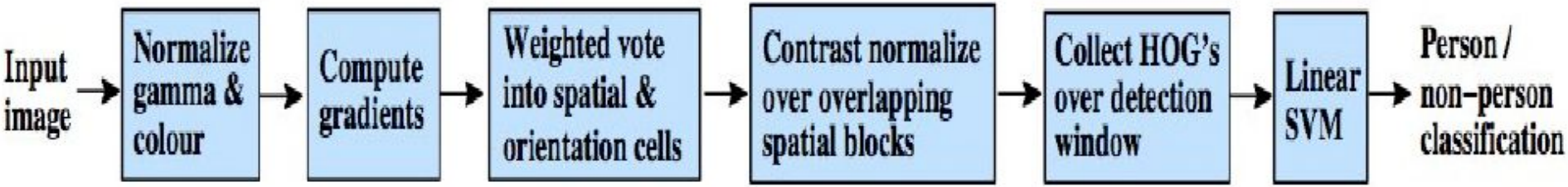
Algorithm Overview

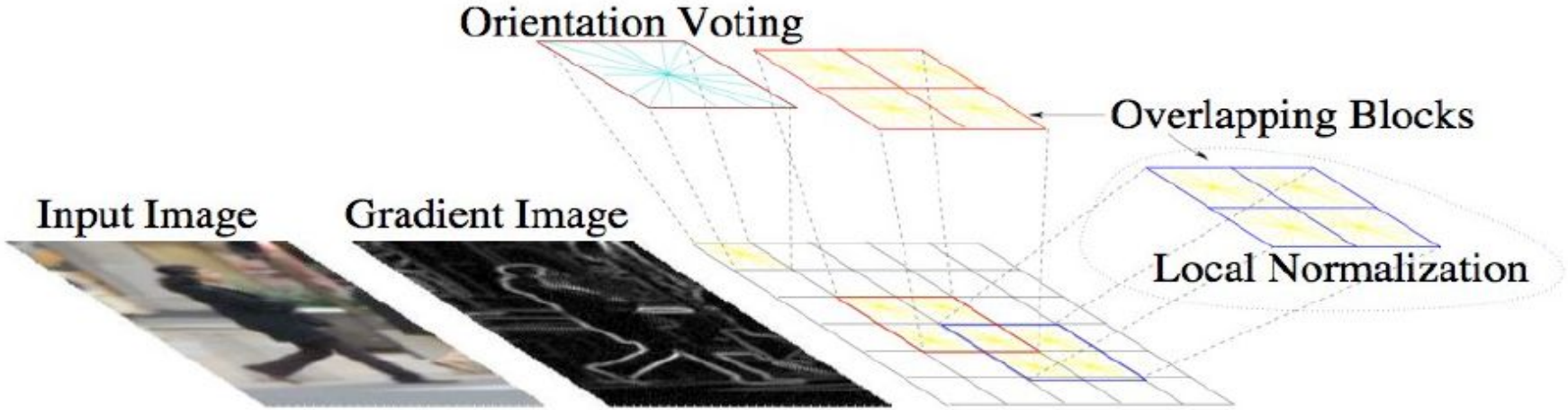
- Divide image into small sub-images: “cells”
  - Cells can be rectangular (R-HOG) or circular (C-HOG)
- Accumulate a histogram of edge orientations within that cell
- The combined histogram entries are used as the feature vector describing the object
- To provide better illumination invariance (lighting, shadows, etc.) normalize the cells across larger regions incorporating multiple cells: “blocks”

Why HOG?

- Capture edge or gradient structure that is very characteristic of local shape
- Relatively invariant to local geometric and photometric transformations
  - Within cell rotations and translations do not affect the HOG values
  - Illumination invariance achieved through normalization
- The spatial and orientation sampling densities can be tuned for different applications
- For human detection (Dalal and Triggs) coarse spatial sampling and fine orientation sampling works best
- For hand gesture recognition (Fernandez-Llorca and Lacey) finer spatial sampling and orientation sampling is required

A worked example: Dalal and Triggs





Colour Normalisation

- RGB and LAB colour spaces equivalent
- Gamma correction (gain) no major affect on results
- Greyscale only small negative effect (-1.5%) on results

Computing the Gradient

- Several gradient detectors tried
  - [1,-1], [1,0,-1], [1,-8,0,8,-1], Sobel Unfiltered and Pre-filtered with Gaussian smoothing
  - Simplest [1,0,-1] proved best
  - Gaussian smoothing affected results negatively
- For colour images Compute each channel separately
- Choose the largest value as the gradient for that pixel

Orientation Binning

- Each pixel votes for an orientation according to the closest bin in the range
  - 0 to 180 (ignore negative edge directions)
  - 0 to 360 Bilinear smoothing to reduce aliasing effects
- The “vote” is weighted by the gradient magnitude
  - Magnitude, Magnitude^2, edge presence absence, etc

Normalization

- Gradient is affected by illumination changes
  - normalization needed
  - Normalization takes the maximum range of a signal and stretches it to take up the maximum possible range
  - E.g.:- A simple normalization scheme: if the range of pixel values is 50-180 out of a total of 0-255 then min=50 and max =180, normalization > (old_pix - min)*((max-min)/255)= new_pix
- Other normalization schemes can be used
- Group cells into larger blocks
  - Normalize within the blocks This ensures that low contrast regions are stretched
- Overlapping blocks
  - This ensures consistency

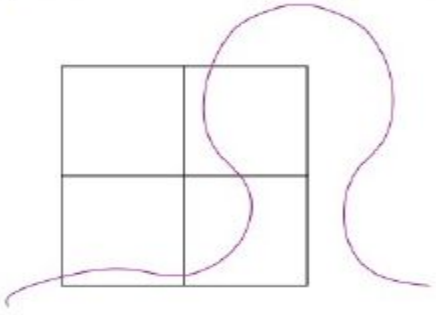
Dalal and Triggs results (demo)

input image

weighted pos wts

weighted neg wts

avg. grad



outside in block

- The most important cues are head, shoulder, leg silhouettes
- Vertical gradients inside the person count as negative
- Overlapping blocks those just outside the contour are the most important

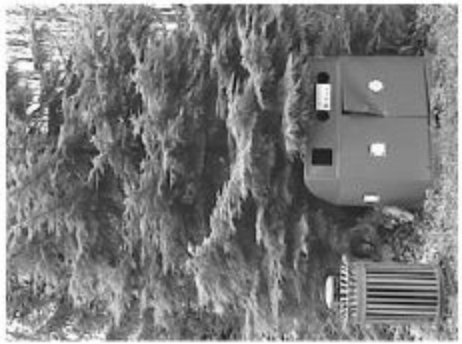

SIFT

SIFT: Introduction

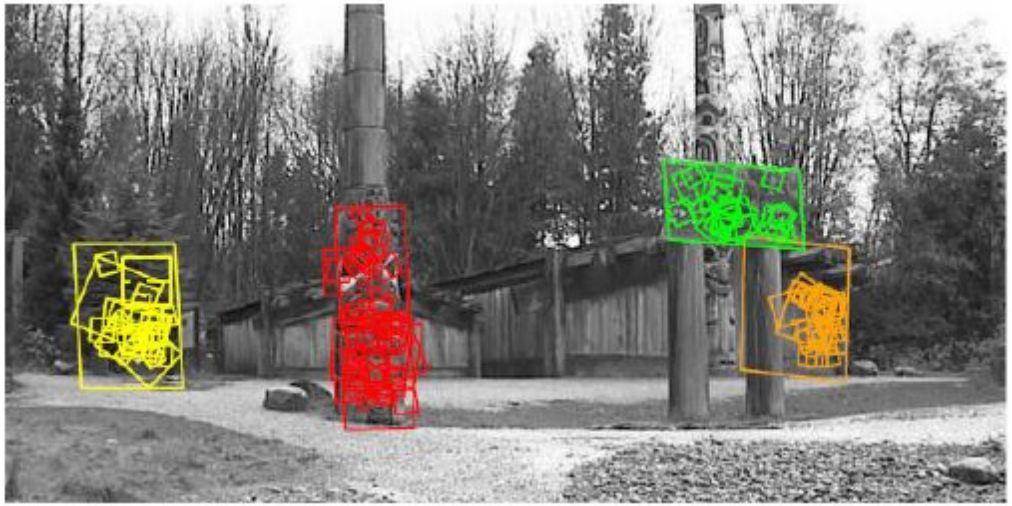
- Matching features across different images in a common problem in computer vision. When all images are similar in nature (same scale, orientation, etc) simple corner detectors can work. But when you have images of different scales and rotations, you need to use the Scale Invariant Feature Transform.

Why care about SIFT?

- SIFT isn't just scale invariant. You can change the following, and still get good results:
  - Scale (duh)
  - Rotation
  - Illumination
  - Viewpoint

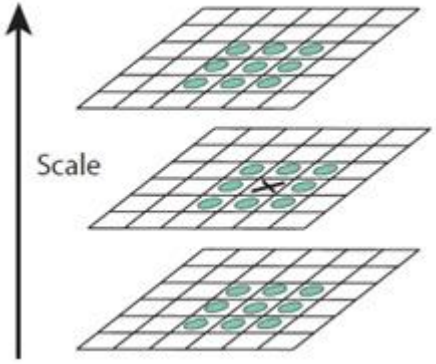


Here's the result:



Locate maxima/minima in DoG images (demo)

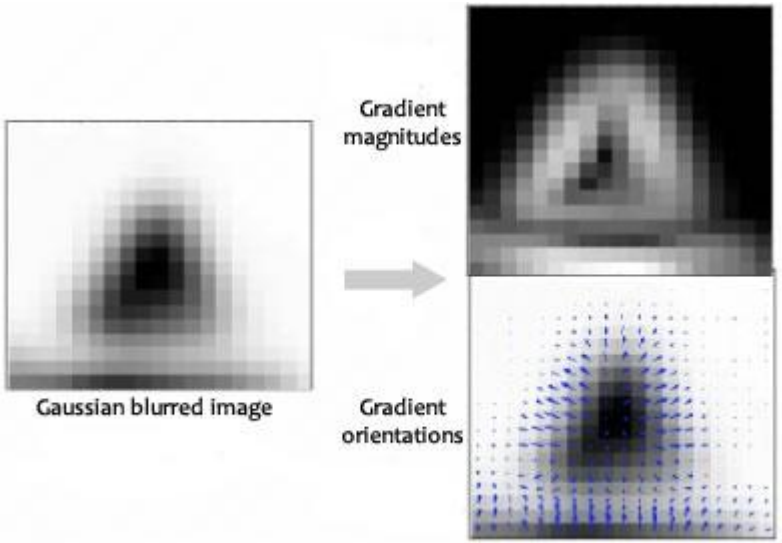
- The first step is to coarsely locate the maxima and minima.
- You iterate through each pixel and check all it's neighbours.
- The check is done within the current image, and also the one above and below it.
- X marks the current pixel. The green circles mark the neighbours. This wa made. X is marked as a "key point" if it is the greatest or least of all 26 neighbours.



SIFT: Keypoint orientations

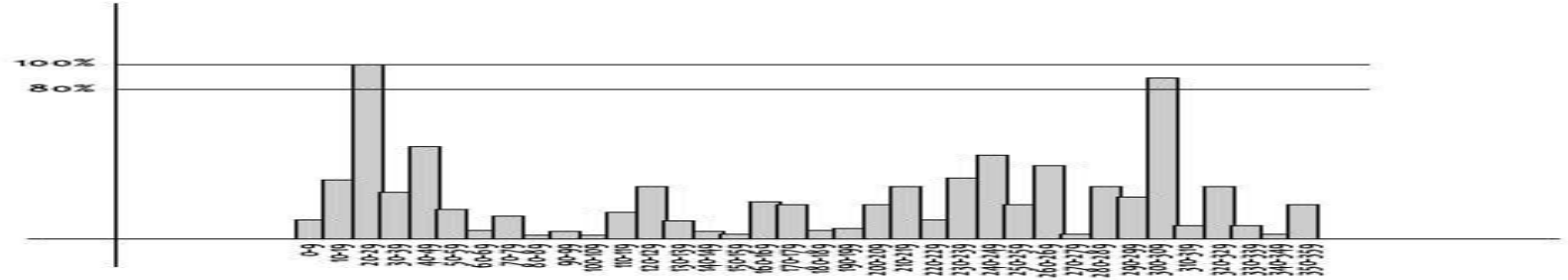
The idea

- The idea is to collect gradient directions and magnitudes a prominent orientation(s) in that region. And we assign this orientation(s) to the keypoint.
- Any later calculations are done relative to this orientation. This ensures rotation invariance.



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

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



The magnitude and orientation is calculated for all pixels around the keypoint. Then, A histogram is created for this.

In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). Lets say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, then it will go into the 10-19 degree bin. And the "amount" that is added to the bin is proportional to the magnitude of gradient at that point.

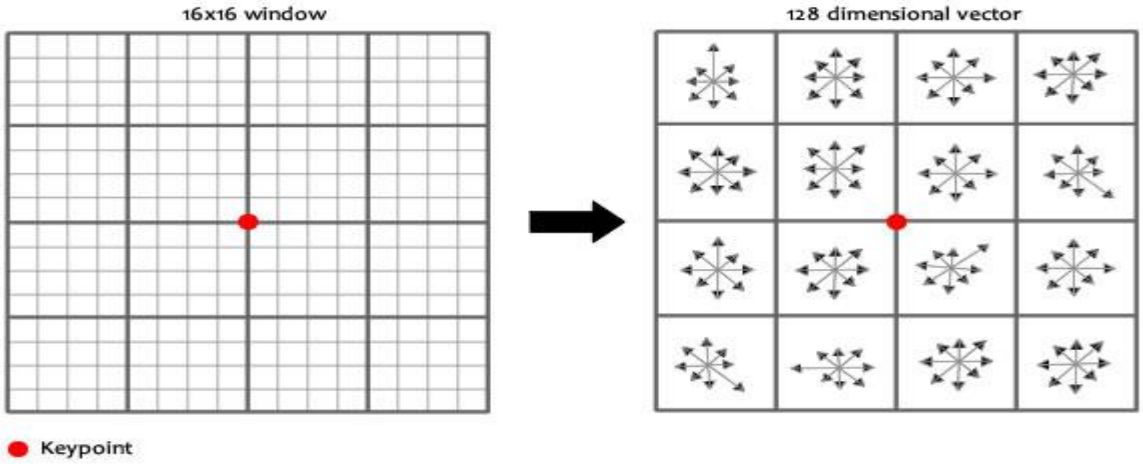
Once you've done this for all pixels around the keypoint, the histogram will have a peak at some point.

Above, you see the histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin)

Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But it's orientation is equal to the other peak.

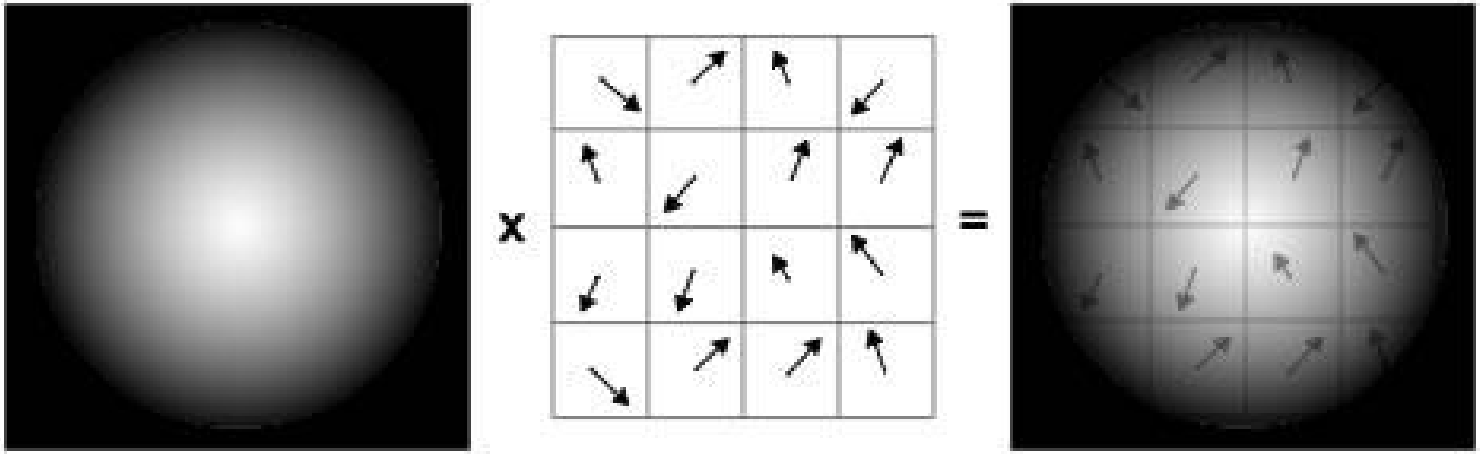
So, orientation can split up one keypoint into multiple keypoints.

SIFT



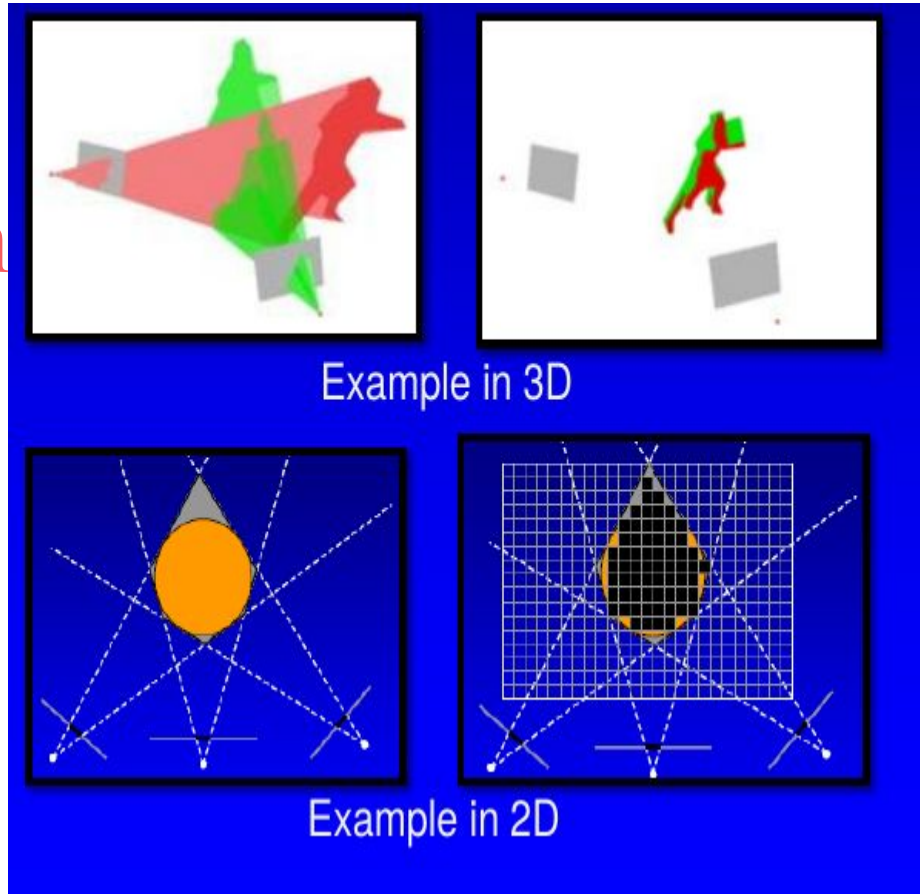
- We want to generate a very unique fingerprint for the keypoint. It should be easy to calculate. We also want it to be relatively lenient when it is being compared against other keypoints. Things are never EXACTLY same when comparing two different images.
- To do this, a 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows.

Gaussian Blur!

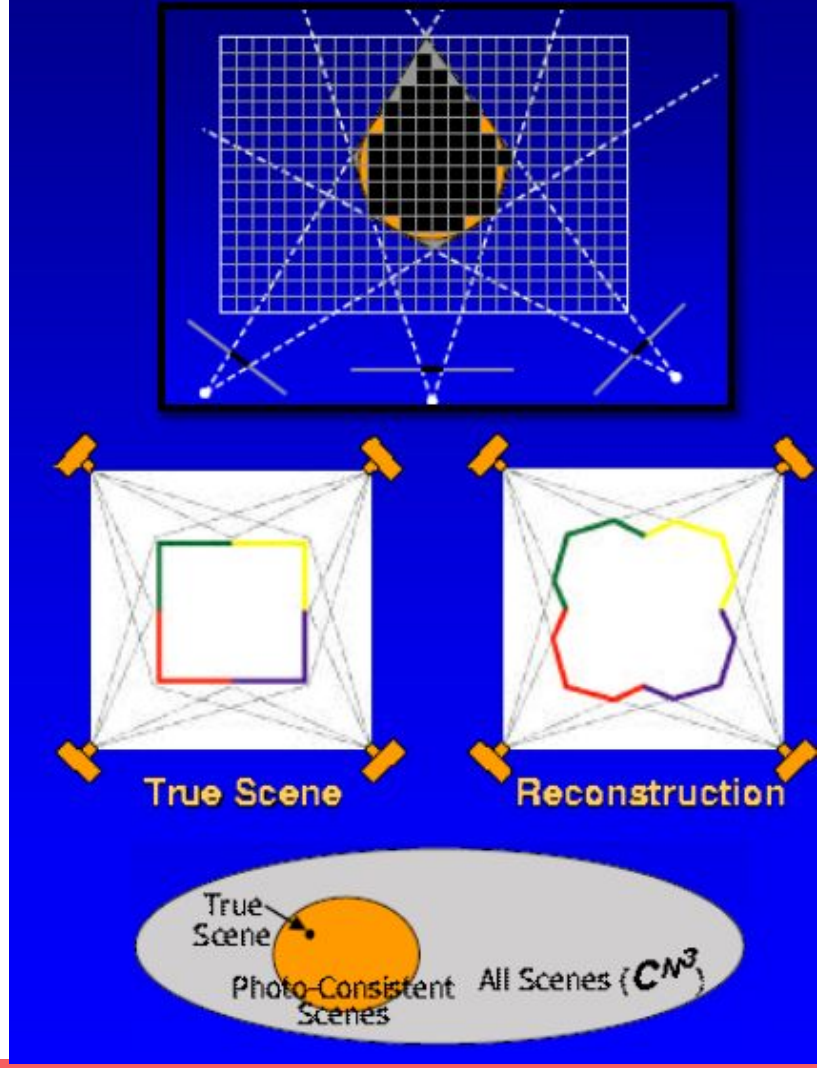


A Space Carving Approach to Surface Reconstruction

- Basic Idea: Volume Intersection
  - Back-project each silhouettes
  - Intersect back-projected volumes

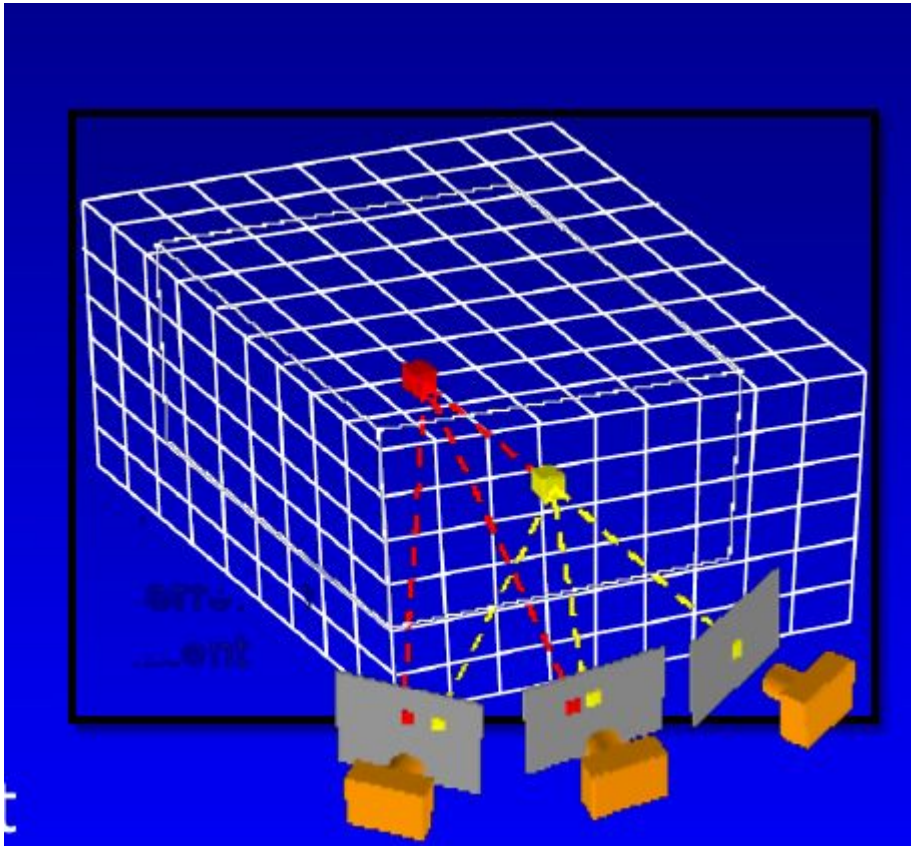


- Given N calibrated views/images, recover the 3D scene
    - Problem posed as a constraint satisfaction task
    - Photo-consistent shape recovery
    - Practical Algorithm to perform ‘Space Carving’



Algorithm

- Initialize a volume V containing the true Scene
- For each voxel on current surface
  - Project it to all visible input images
  - if not photo-consistent, carve (remove) it
- Stop if no photo-consistent voxel found



Visibility Ordering

- Basic Idea: Visit Occluders first
- More Generally, Plane Sweep Algo:-
  - Sweep Plane in each of 6 principle directions
  - Consider cameras on only one side of plane
  - Repeat until convergence