

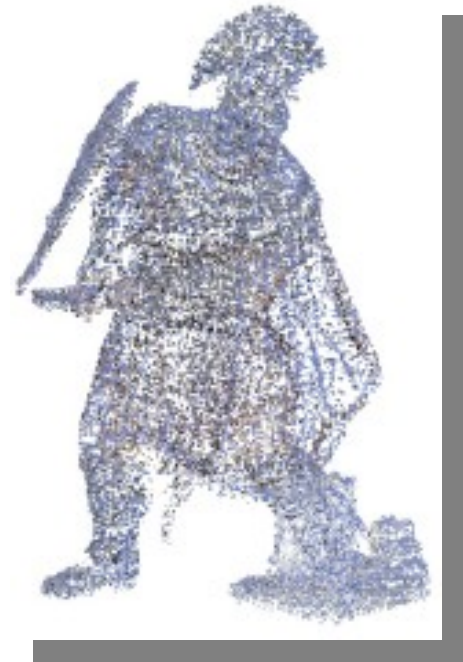
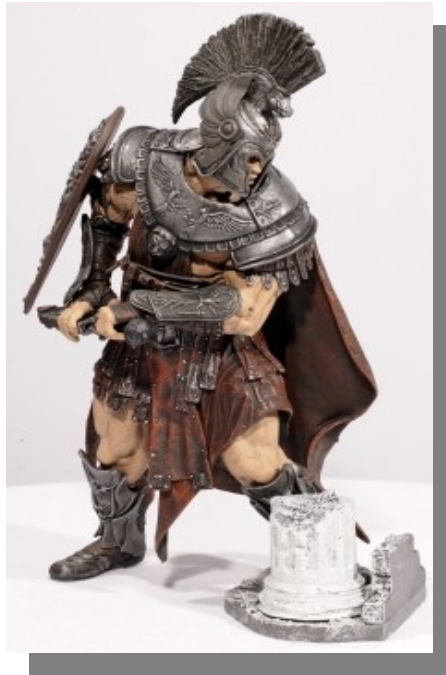
Multi-view Stereo

Ivo Boyadzhiev

CS7670: September 13, 2011

What is stereo vision?

Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape



General MVS Techniques

- Multibaseline Stereo



- Space carving



- Shape from silhouettes

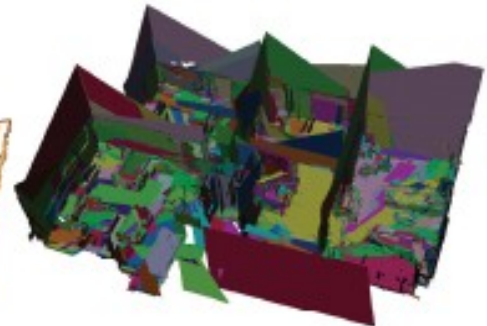
- Patched-based MVS



Specific MVS Techniques

- **Manhattan world stereo**
- **Piecewise-planar stereo**

Of particular interest for **architectural scenes** and **man-made objects**.



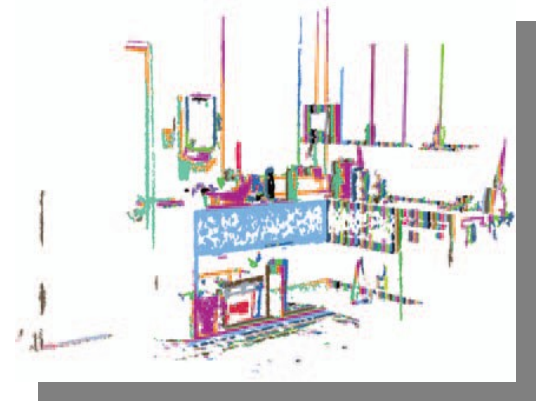
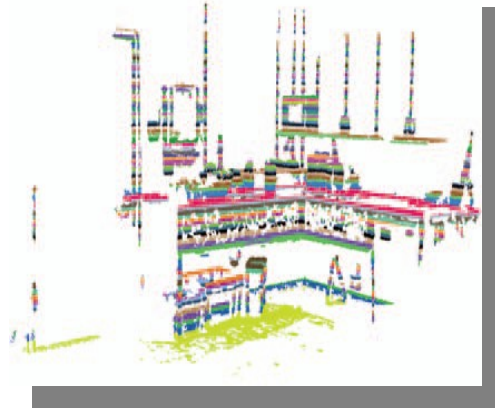
Challenges for architectural scenes

- Texture-poor surfaces
- Non-Lambertian surfaces
- Complicated visibility (occlusions)

- Point cloud complexity
 - We might want simple model representation for real time interactive visualization of a large scene (e.g. a house)

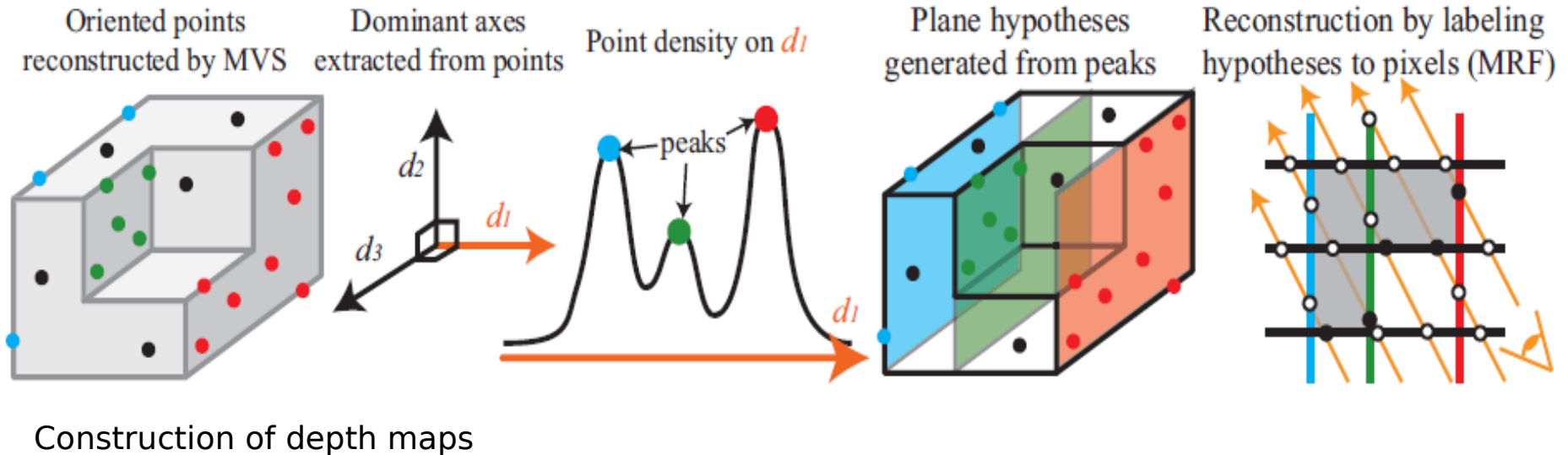
Manhattan-world Stereo

by
Yasutka Furukawa
[CVPR 2009]



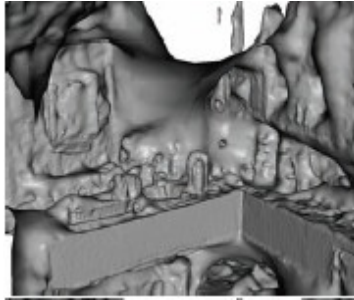
MWS Algorithm

Assume that all surfaces in the world are aligned with three dominant directions.

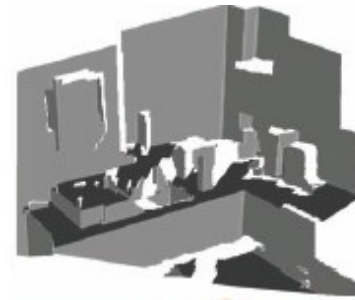


Construction of depth maps

MWS Results



PMVS + Poisson
surface reconstruction
model



Manhattan-world
Stereo model
(**single view**)

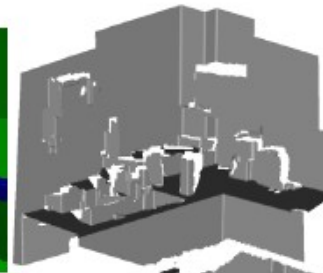
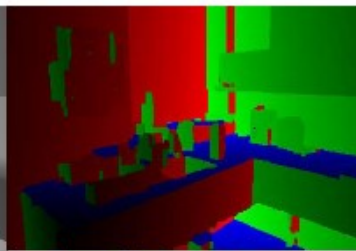
Target image

Depth map

Depth normal map

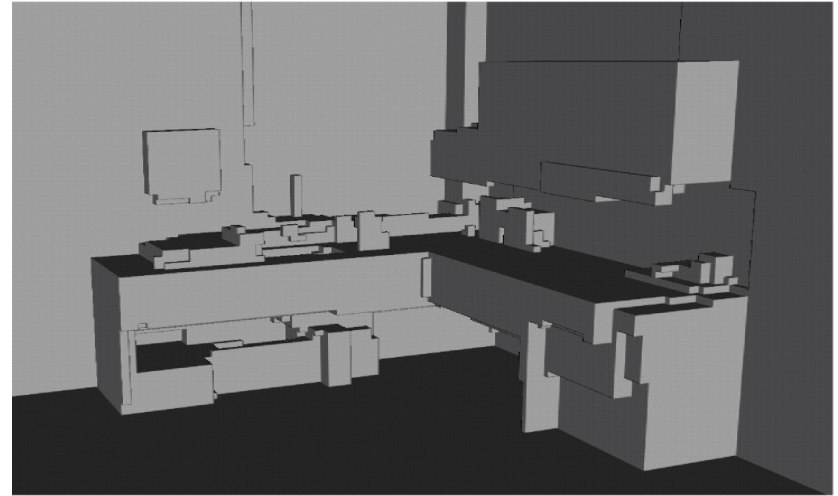
Mesh model

Texture mapped
mesh model



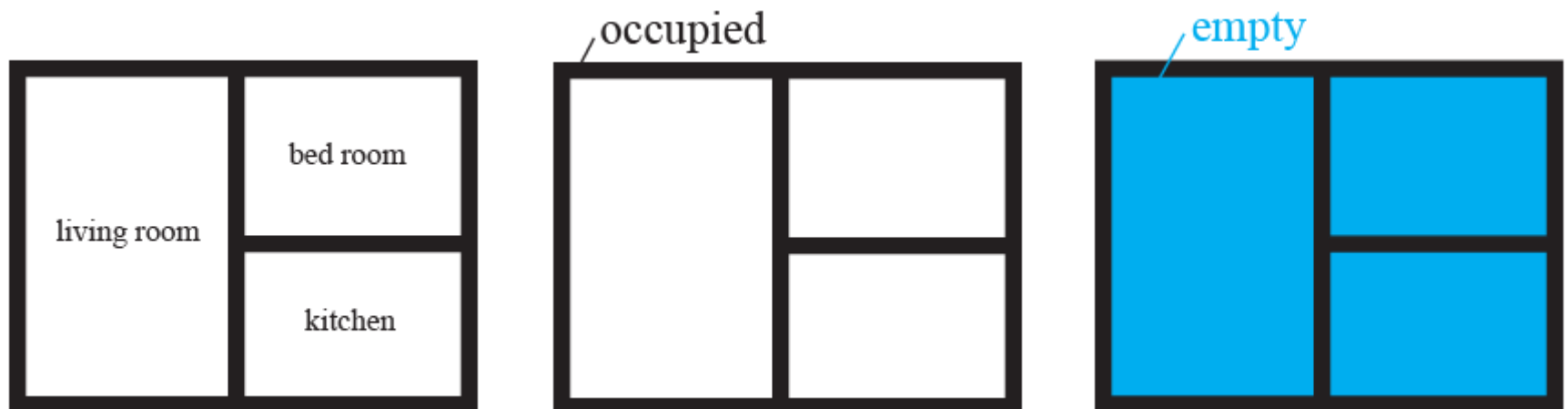
Reconstructing Building Interiors from Images

by
Yasutaka Furukawa
[ICCV 2009]



Axis-aligned depth map merging

- **Basic idea is similar to volumetric MRF** [Vogiatzis 2005, Hernández 2007]
- Assign label $l(\mathbf{v})$ (empty or occupied) to each voxel \mathbf{v}



Axis-aligned depth map merging as an optimization problem

- Assign label $l(\mathbf{v})$ (empty or occupied) to each voxel \mathbf{v} , in order to minimize:

$$\min. \sum F_1(l(\mathbf{v})) + \sum \sum F_2(l(\mathbf{u}), l(\mathbf{v}))$$

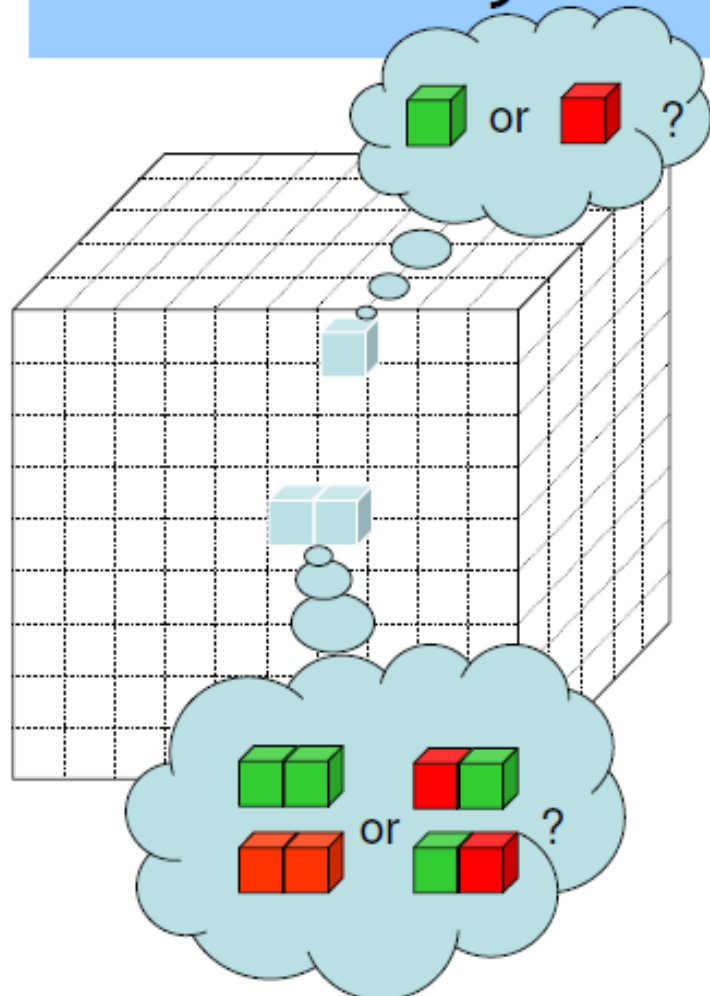
F_1 : unary penalty

F_2 : binary penalty for neighboring voxels

- Use graph cut to find the optimal label assignment [Boykov & Zabith 2001]

Typical volumetric MRF

3D binary labelling problem



Labelling cost:

- Every voxel has a certain preference for being **foreground** or **background**

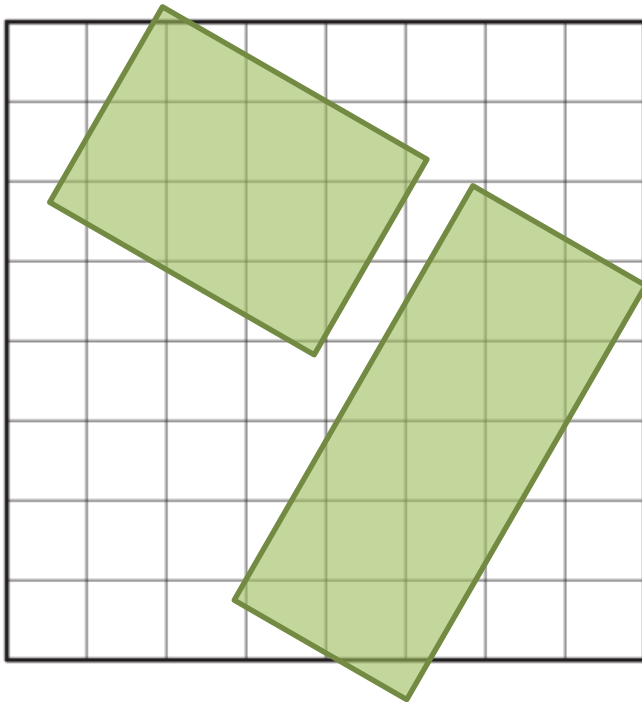
Compatibility cost:

- Every pair of neighbour voxels has a certain preference for being given the *same* or *opposite* labels
- Cost for opposite labels is greater than for same label (sub-modular)

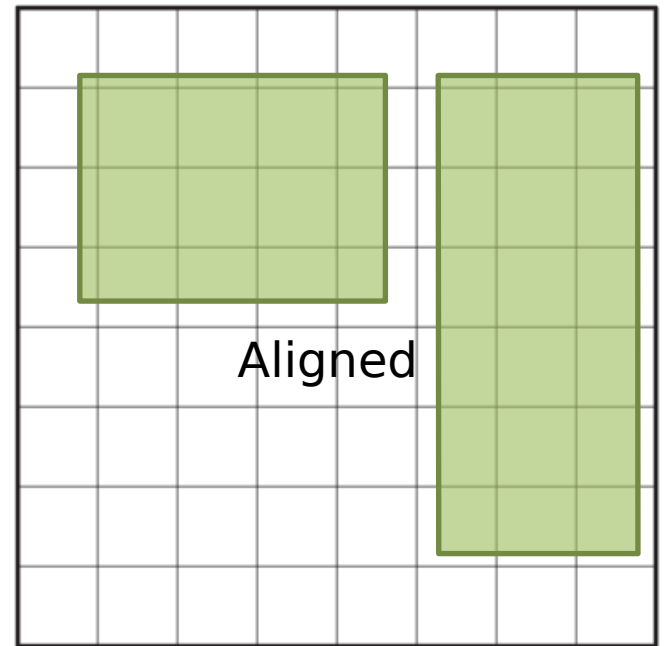
$$\text{Green-Green} + \text{Red-Red} \leq \text{Red-Green} + \text{Green-Red}$$

Align the voxel grid with the dominant axes

Voxel grid



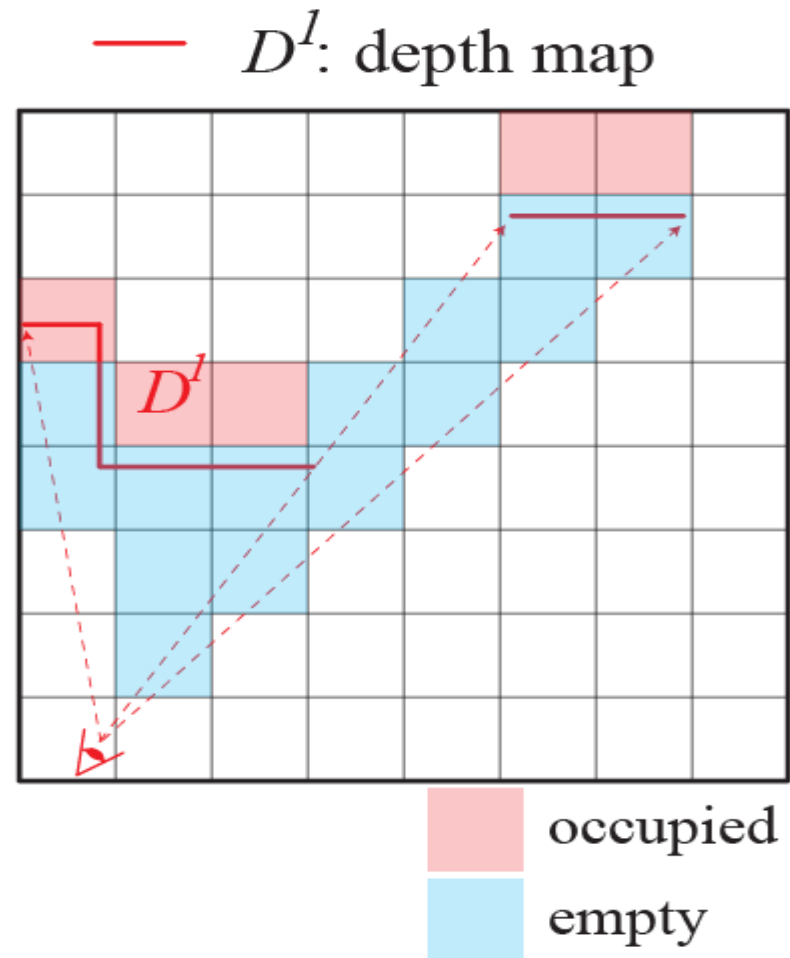
Voxel grid



Construction of the Unary terms

For **each camera**, and **every pixel** in the corresponding depth map, **traverse a ray** into the discretized scene

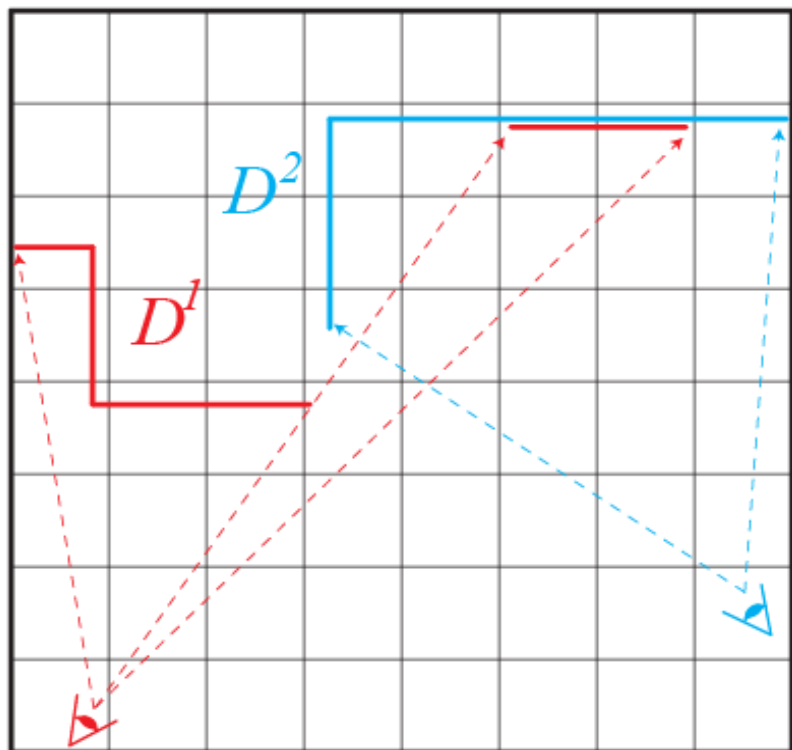
For each *occupied* (red) voxel v
// Decrease penalty for *occupied*
 $F_1(l(v) = \textit{occupied}) -= 1;$
For each *empty* (blue) voxel v
// Decrease penalty for *empty*
 $F_1(l(v) = \textit{empty}) -= 1;$



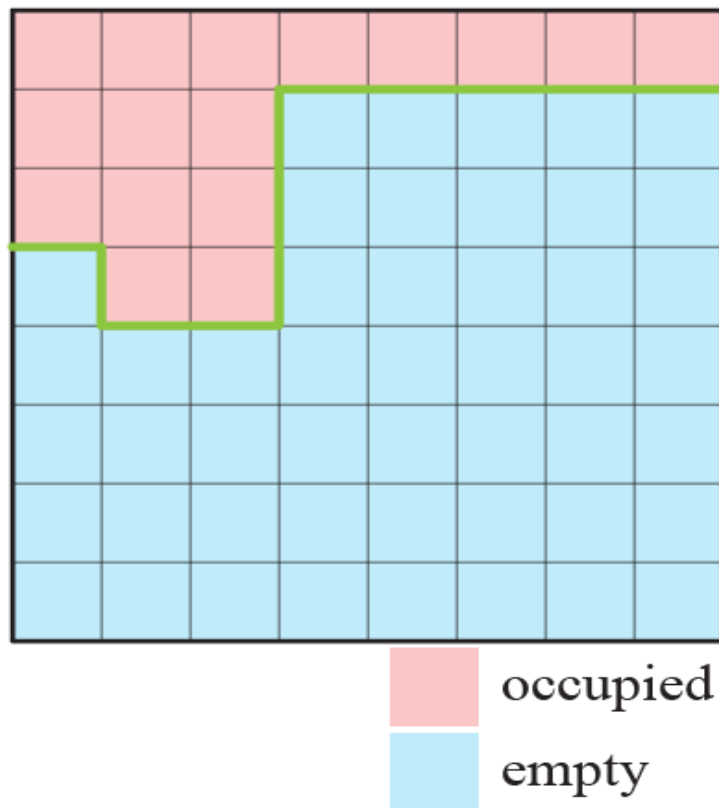
Unary encodes data

Are Unary terms just enough ?

— D^i : depth maps



— : reconstruction



If we have “perfect” depth maps, at the end of this stage the unary terms would encode a correct voxelized representation of the planar AA scene.

Are Unary terms just enough ?

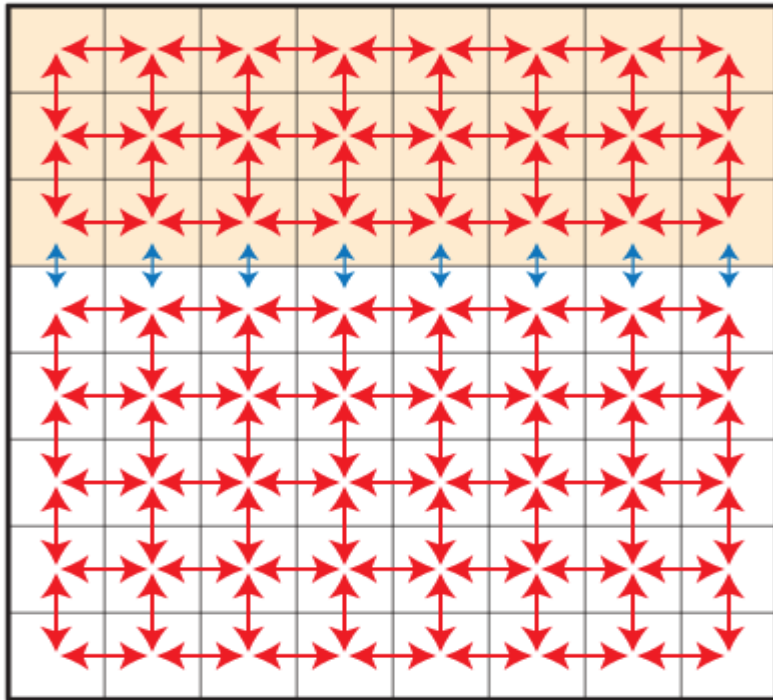
Due to problems, such as:

- Incomplete data
- Sensor errors
- Camera calibration errors
- Manhattan-world stereo errors

We need to define a binary term, that will smooth out the effect of those errors and will help us resolve ambiguities

Construction of the Binary terms

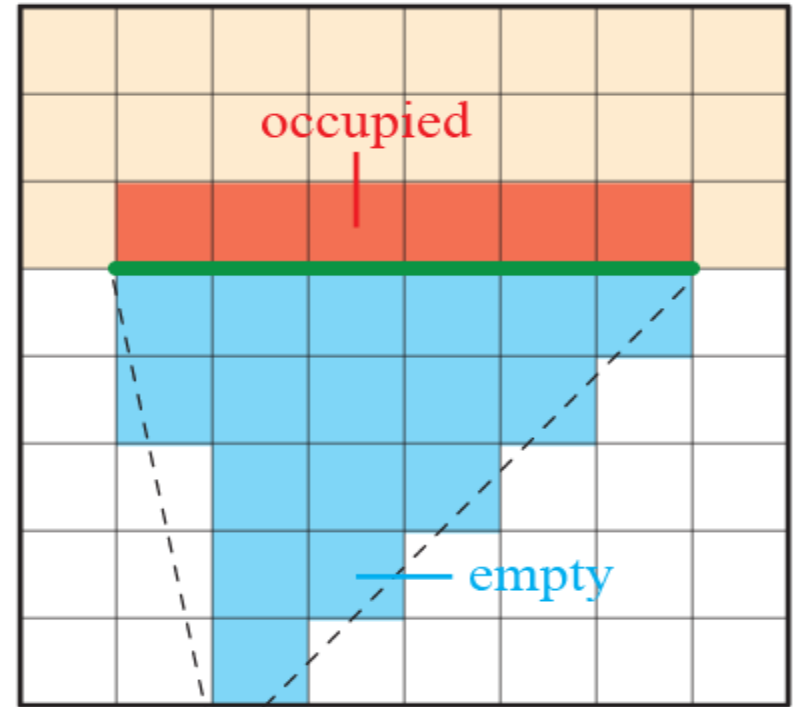
Typical volumetric MRF



Binary encodes smoothness & **data**

Unary is often **constant**

Proposed approach



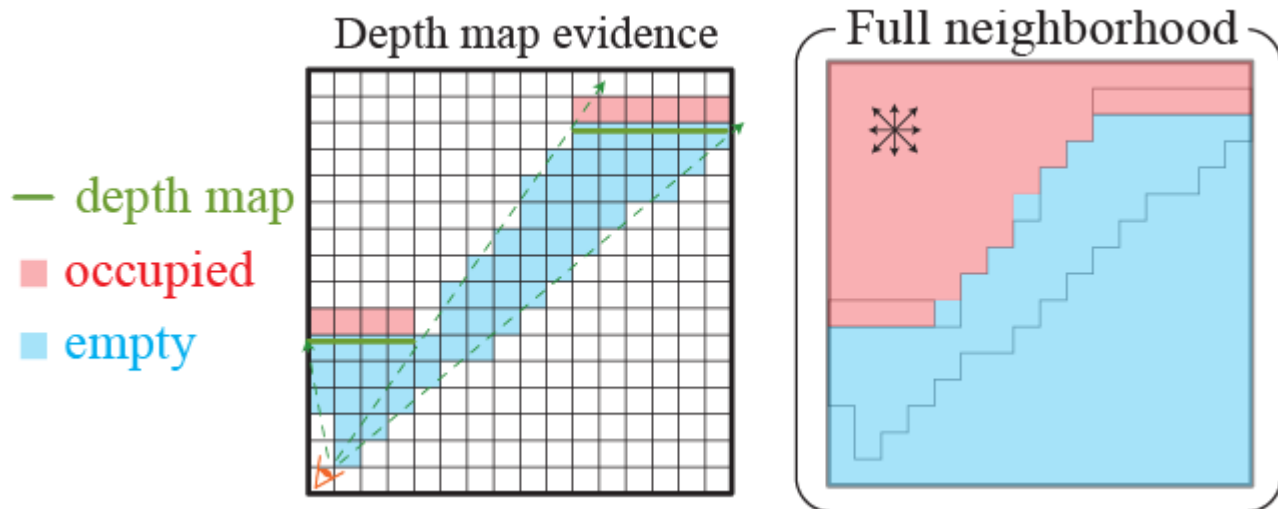
Binary is **smoothness** (the simple Potts model)

Unary encodes **data**

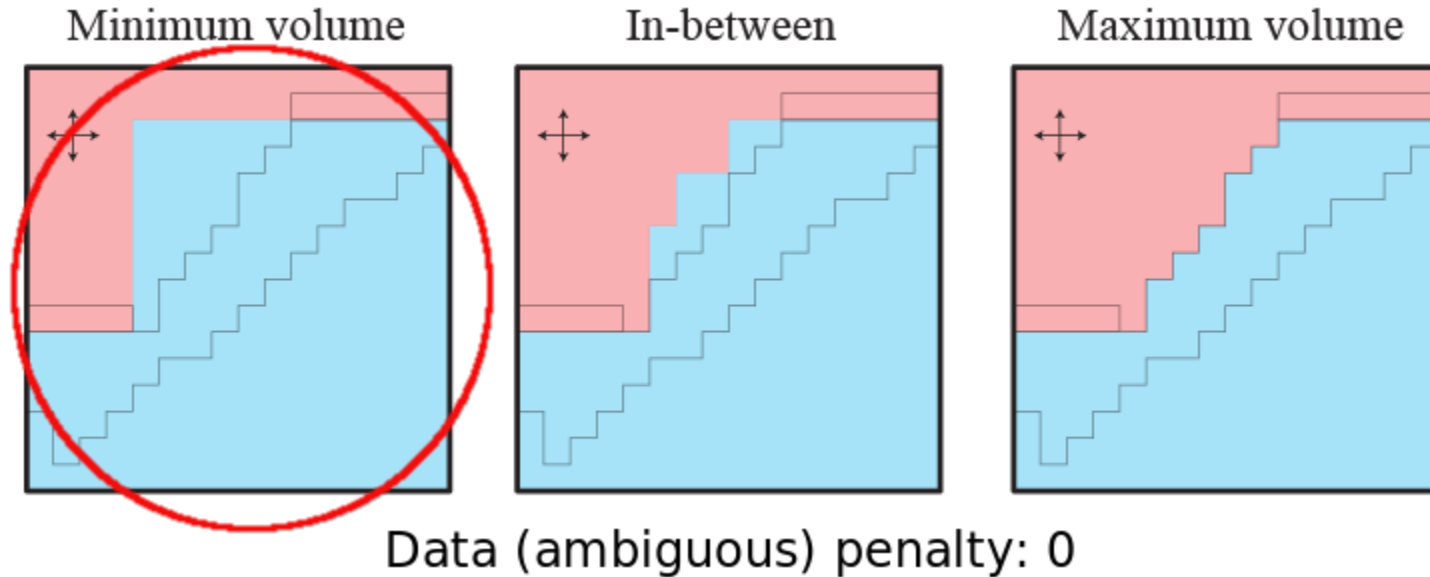
Regularization (1)

Typical volumetric MRFs bias to general minimal surface [Boykov & Kolmogorov, 2003]

In MWS, they want to bias to piece-wise planar axis-aligned models (for architectural scenes)



Regularization (2)



Axis-aligned neighborhood + Potts model
is **Ambiguous**



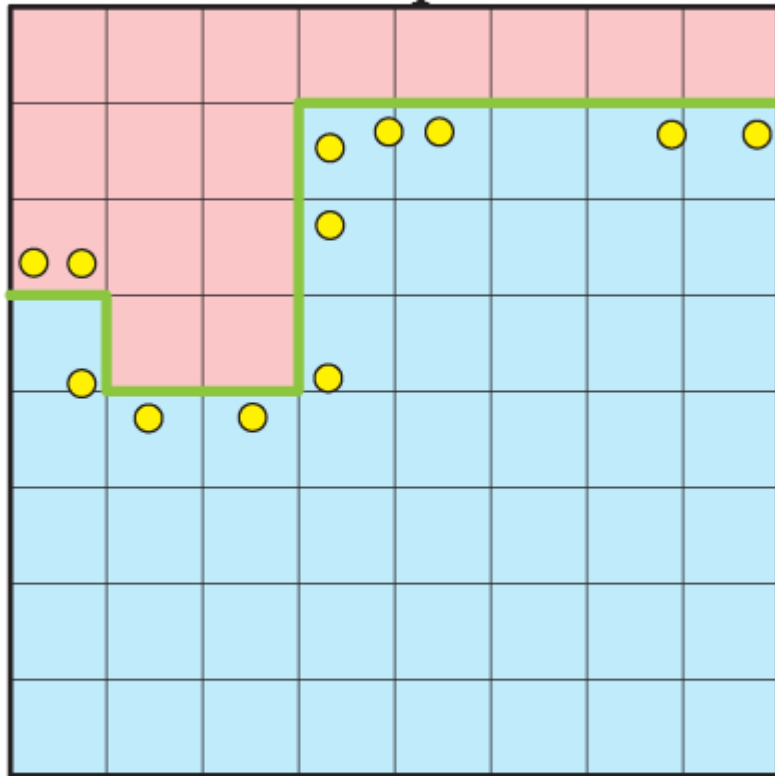
Break ties with the **minimum-volume solution**



Piece-wise planar **axis-aligned model!**

Mesh Extraction (1)

- : reconstruction
- : mvs point

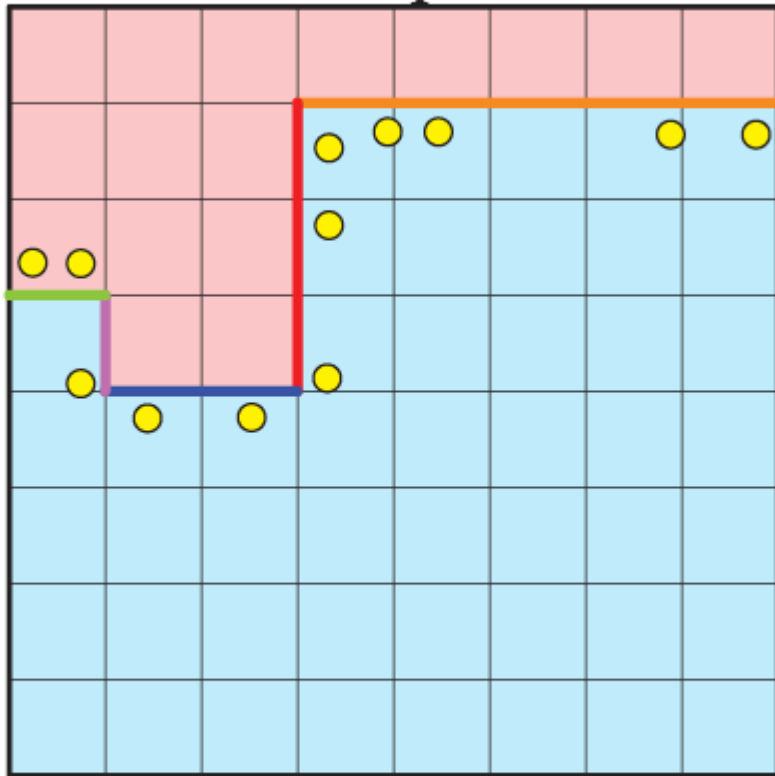


- occupied
- empty

Triangulate the boundary between interior and exterior voxels

Mesh Extraction (2)

- : reconstruction
- : mvs point

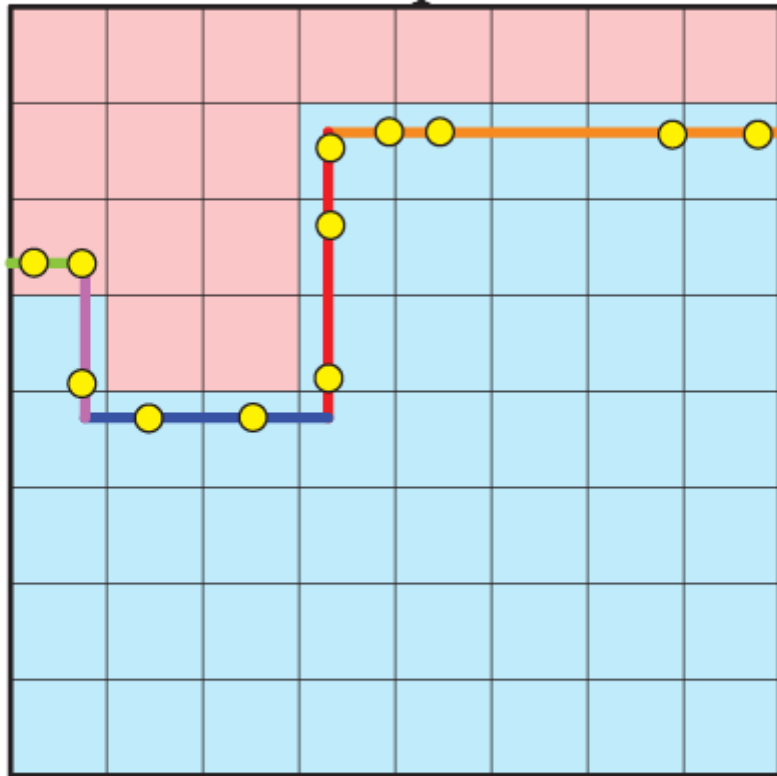


occupied
empty

Boundary is defined
as a set of voxel
faces whose
incident voxels
have different
labels

Mesh Extraction (3)

- : reconstruction
- : mvs point

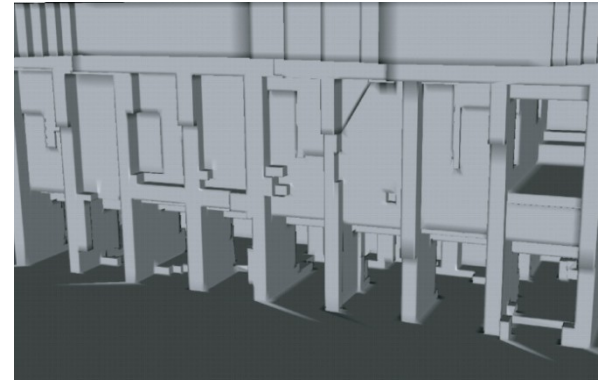


- occupied
- empty

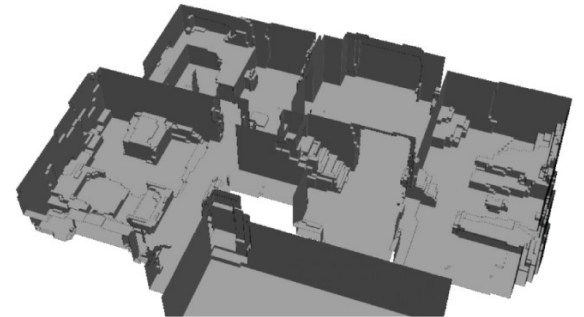
Compute a sub-voxel offset along the normal direction to each component C, based on the PMVS points

Results

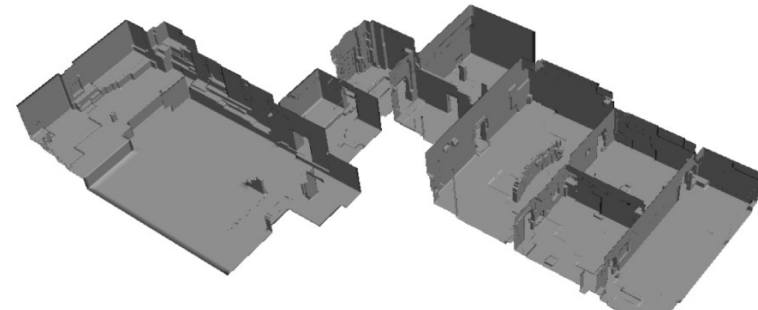
Hall - 97
images



House - 148
images



Gallery - 492
images



Pros and Cons of the Strong Planarity Assumption

- Pros
 - Deals with poorly textured surfaces
 - Deals with non-Lambertian surfaces
 - Produces low complexity 3D models for rendering, storage and transmission

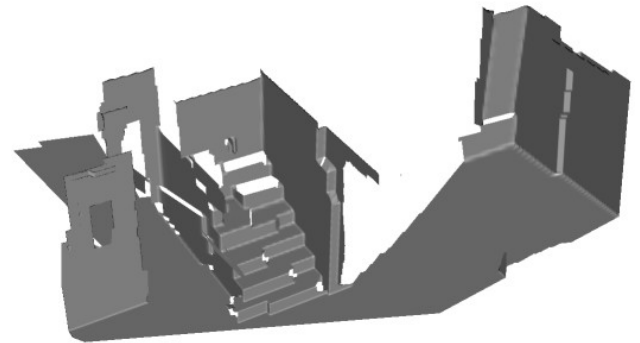
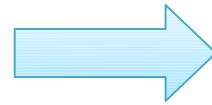
Pros and Cons of the Strong Planarity Assumption

- Cons
 - Performs poorly in the presence of non-planar objects:
 - Produces staircase appearance
 - Whole objects can be flattened to nearby planes

MWS Reconstruction Error



Image from the data set



The table is flattened



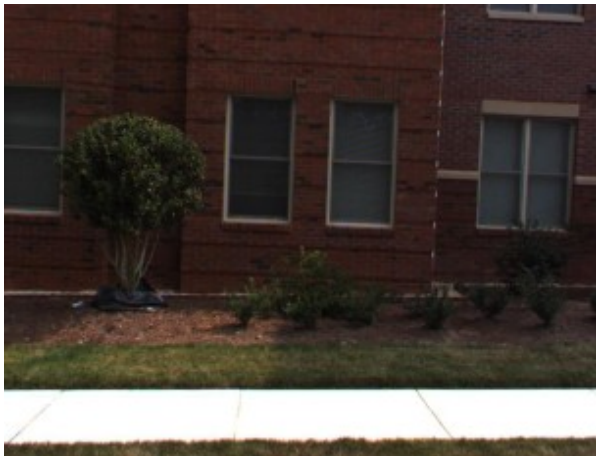
Textured model

Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction

by David Gallup
[CVPR 2010]



Piecewise Planar Stereo



Sources of Error

- **specular surfaces**
- **depth uncertainty**

Priors

- **planarity**
- **appearance**

3D Reconstruction



Sources of Error

- **specular surfaces**
- **depth uncertainty**

Priors

- **planarity**
- **appearance**

Plane Hypothesis Generation (1)




- Use RANSAC to obtain plane hypotheses for every image
- Link planes, that are seen in nearby views
- Formulate pixel to plane assignment as MRF energy minimization
 - Use **photo-consistency** measure for the **unary term**
 - Use **image gradient** and 3D points distance for the **smoothness term**

Plane Hypothesis Generation (2)

Video



Labels = $\{\pi_1, \dots, \pi_N, \pi_\infty, non\text{-}plane, discard\}$

 planes  non-plane  discard



Real-Time Stereo



Plane Detection



Planar and Non-Planar Segmentation

[slide credit: D. Gallup]

Appearance Prior



Based on **color** and **texture** appearance, distinguish between planar and non-planar surface.

Classification (1)

- Divide image into regular grid
- Compute RGB, HSV, edge orientation histogram for each patch
- 5000 human-labeled examples (5 images)
- Use k-nearest-neighbor classifier

Classification (2)



non-planar

planar

Assign probability to every new patch of being planar vs non-planar.

Effect of Classifier



non-planar  planar

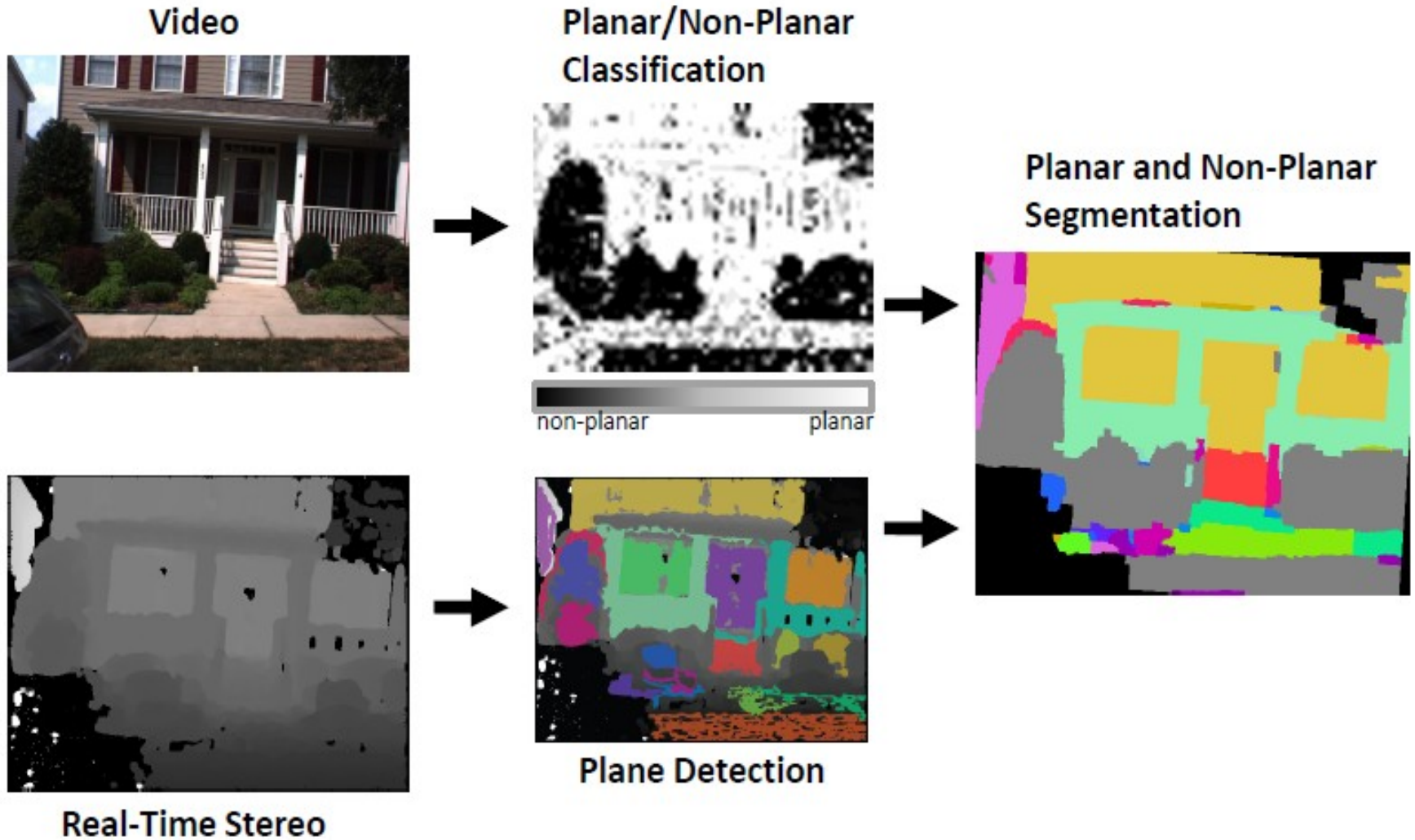


Without planar
class likelihood



With planar
class likelihood

Full Algorithm



Results - Intermediate Stages



(a)



(b)



(c)



(d)

Results - 3D Reconstruction



Conclusions and open issues (1)

- Can we apply the planar, non-planar mixture paper for interiors ?
 - Much more occlusions, light variation and variety of non-planar surfaces
 - Harder to learn a planar model there
- Apparently, piece-wise planar assumptions work well for big parts of architectural scenes, can we think of a more general approach to detect boundaries between planar and non-planar surfaces ?

Conclusions and open issues (2)

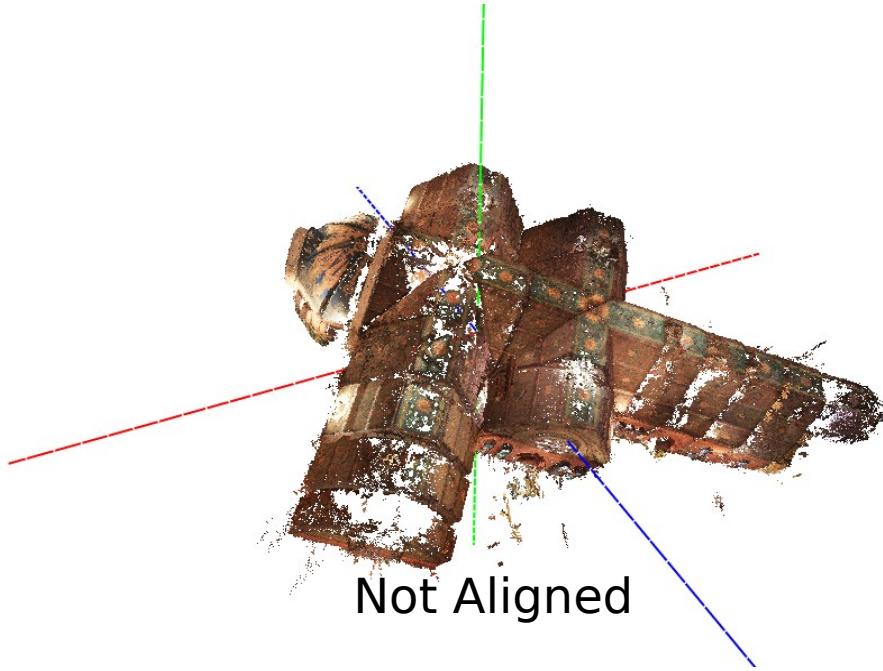
- Apply more priors for architectural scenes
 - Where do we expect furniture, windows, walls, etc.
- Detect common interior objects, using database of models (Google Sketchup)
- Put, intelligently, a person into the loop

My Experiments

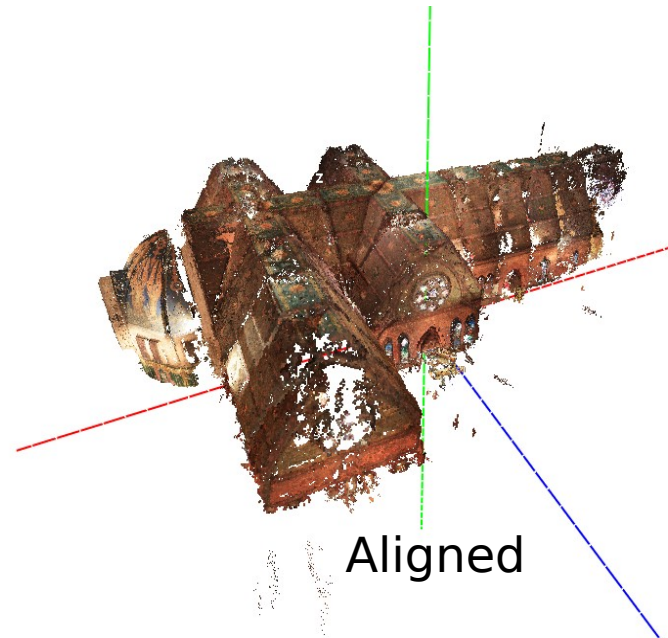
- PMVS reconstruction of Sage Chapel
 - ~450 fish-eye images
 - ~1,500,000 reconstructed points



Dominant axes detection

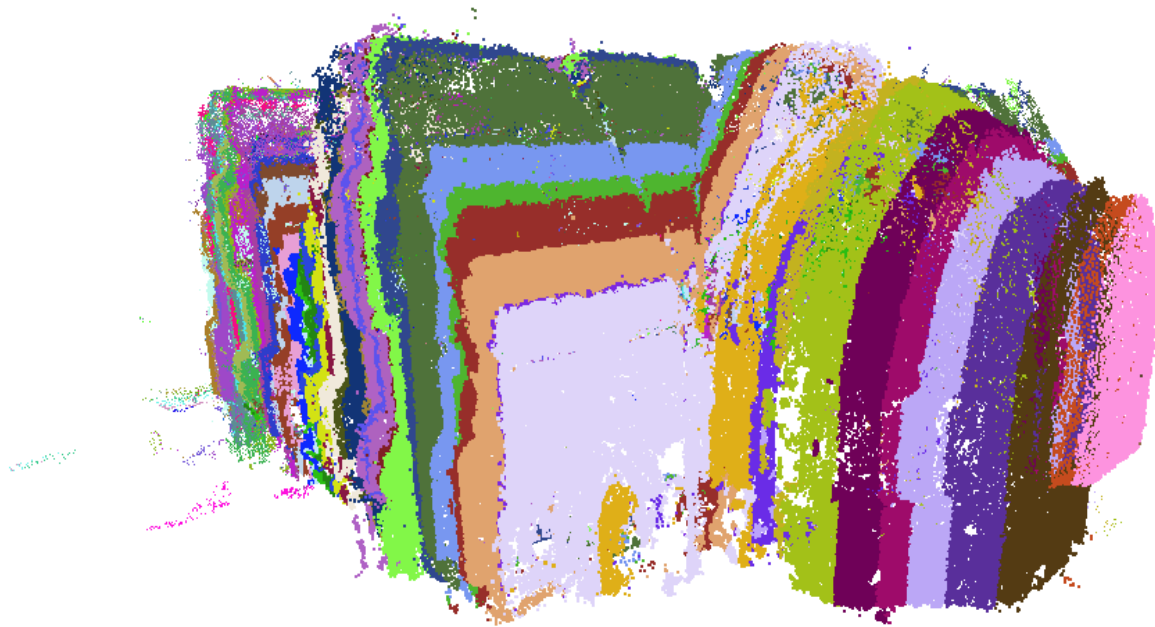


Not Aligned



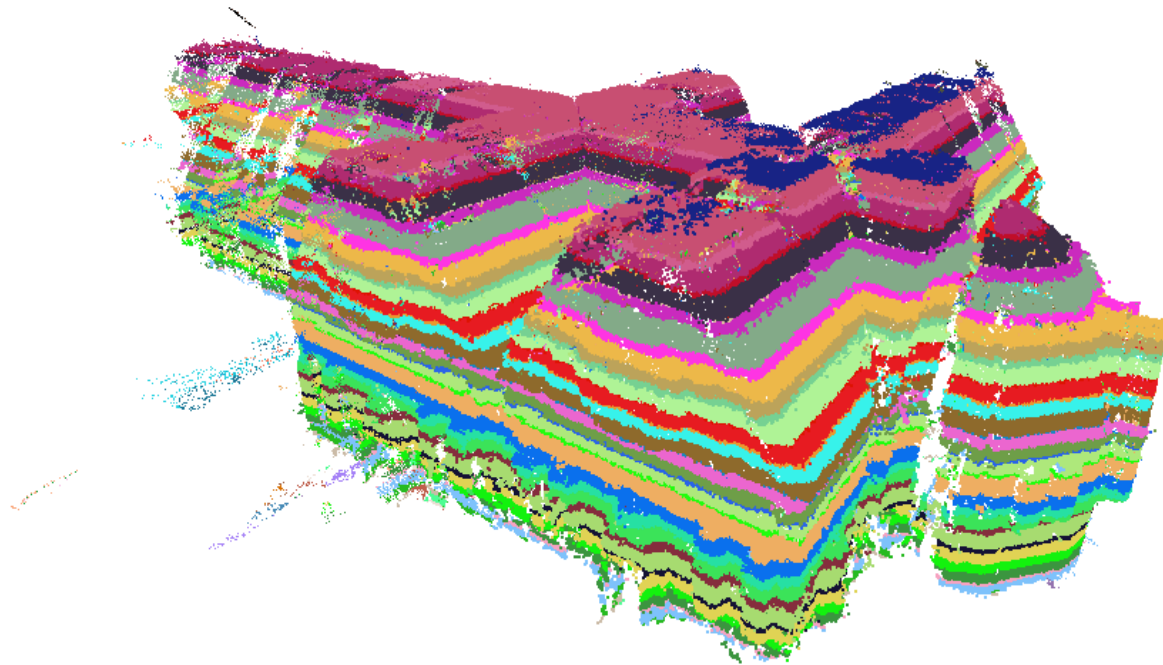
Aligned

Manhattan planes detection



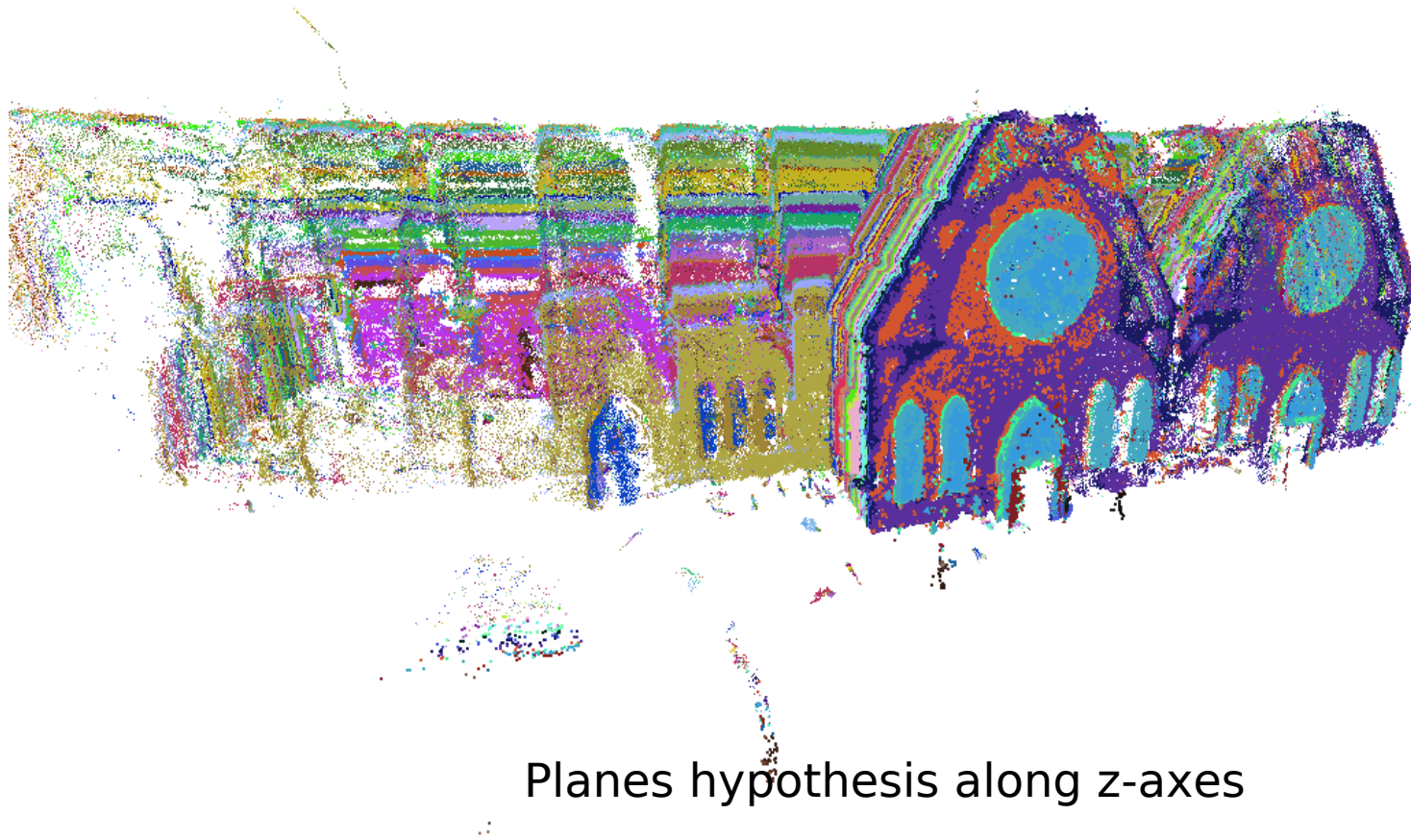
Planes hypothesis along x-axes

Manhattan planes detection



Planes hypothesis along y-axes

Manhattan planes detection



Planes hypothesis along z-axes

Thank You!



Voxelized + Textured Model