

# Structured 3D Reconstruction: From geometric sensing to perception

Yasutaka Furukawa



SIMON FRASER  
UNIVERSITY

# Evolution of 3D Reconstruction Techniques

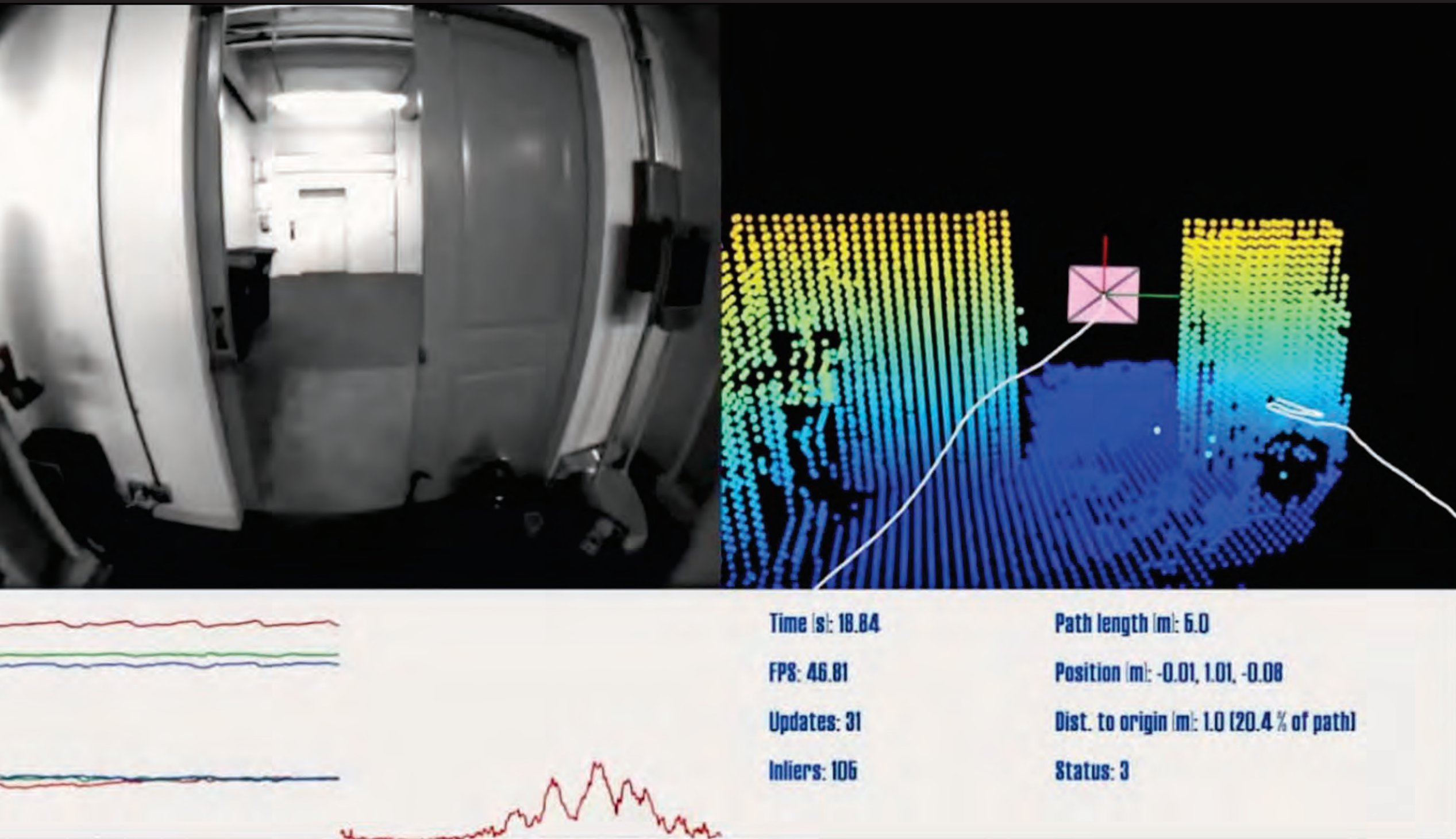


**Sensing**



**Perception**

# Sensing

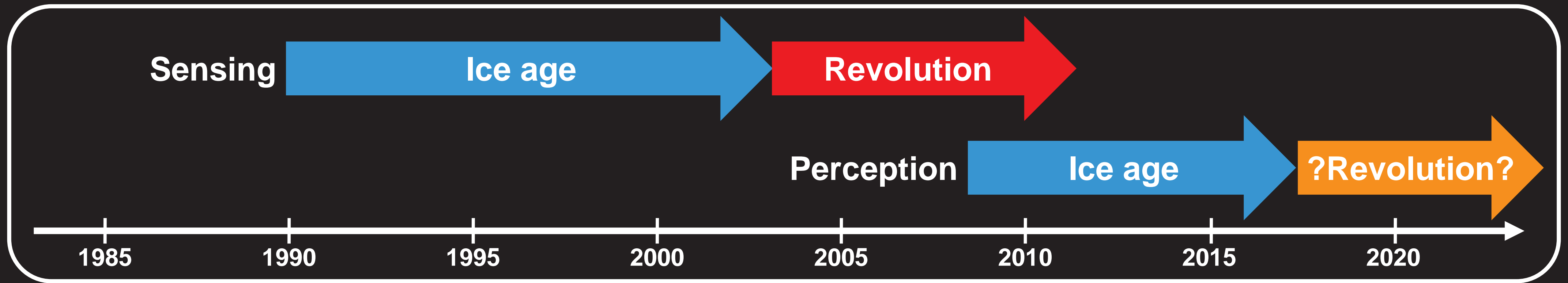


[ Google Project Tango ]

# Perception

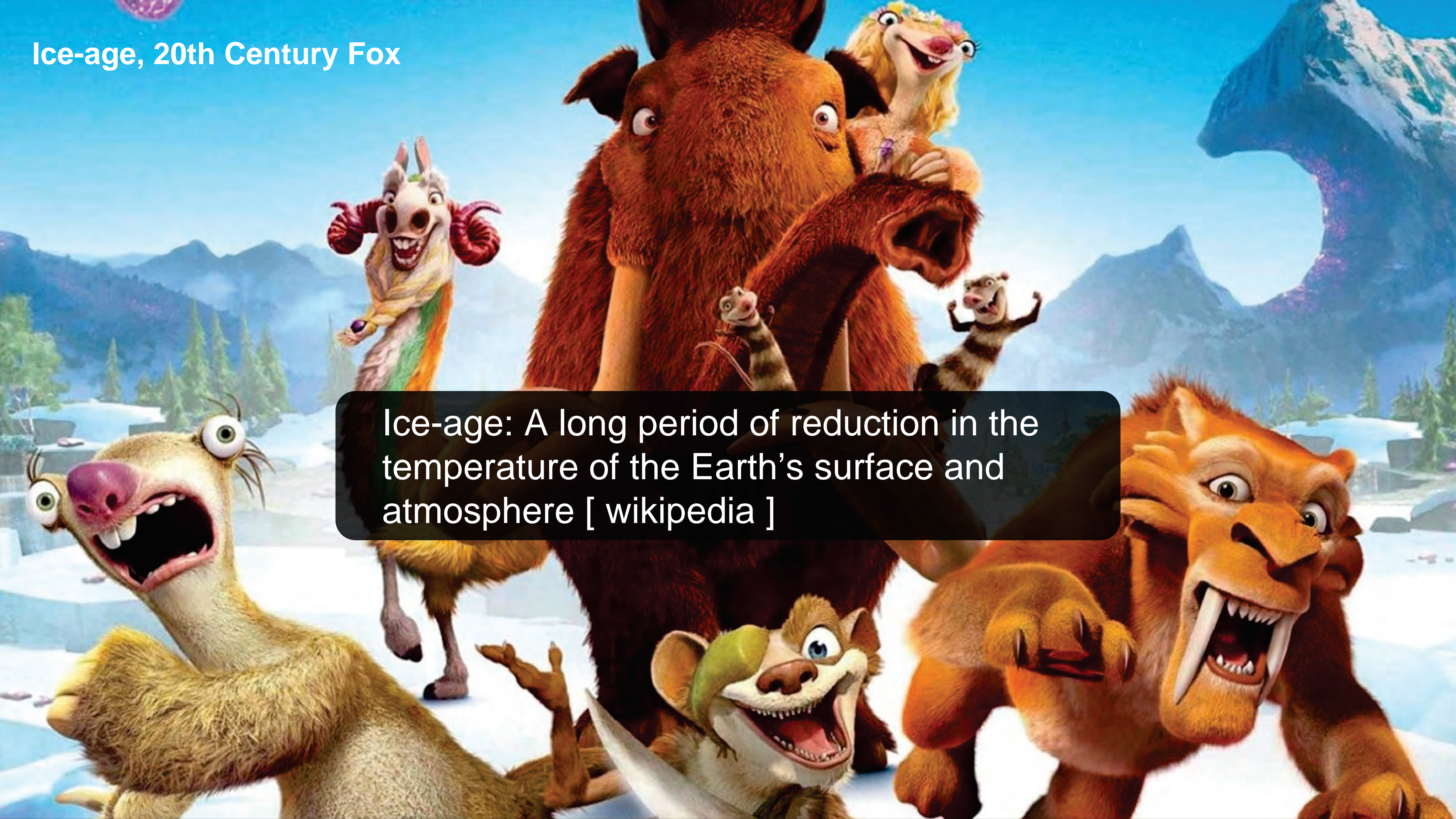


[ CANVAS by Occipital (<https://canvas.io>) ]




Ice-age, 20th Century Fox

Ice-age: A long period of reduction in the temperature of the Earth's surface and atmosphere [ wikipedia ]

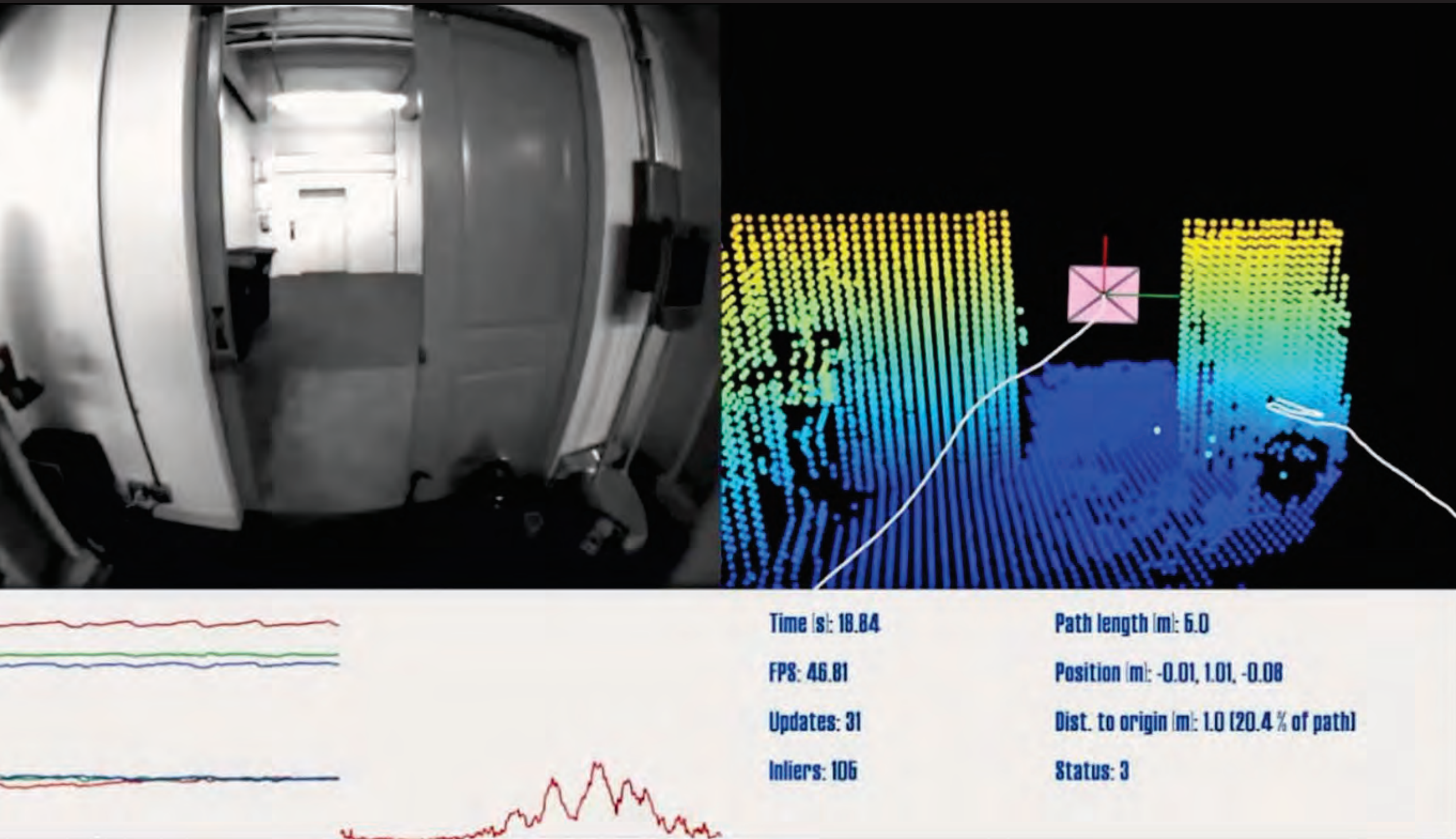


Ice-age, 20th Century Fox

A vibrant, colorful illustration of the Ice Age movie cast. In the center, Manny the mammoth looks surprised. To his left, Sid the sloth is laughing with his mouth wide open. To his right, Diego the saber-toothed tiger is roaring. Other characters like the reindeer, the squirrel, and the rabbit are scattered around. The background shows a snowy mountain range under a blue sky.

Ice-age: A long period of reduction in the impact to our society

# Sensing

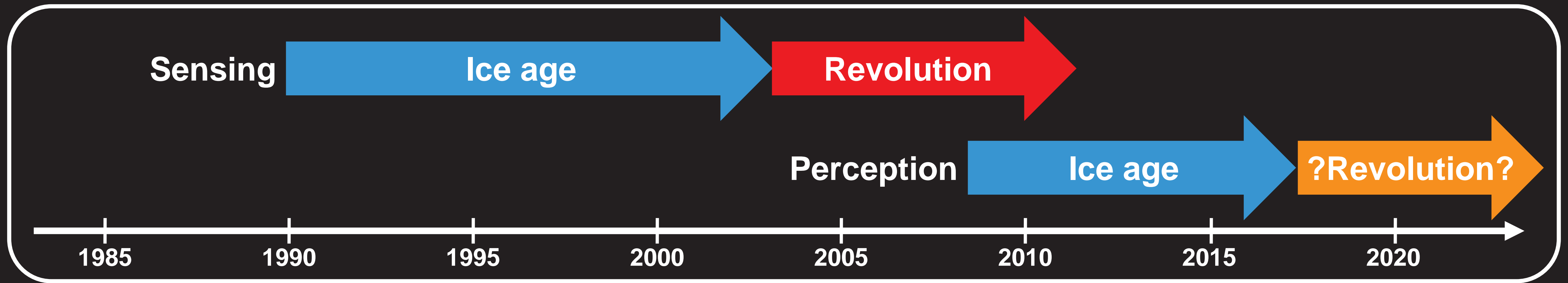


[ Google Project Tango ]

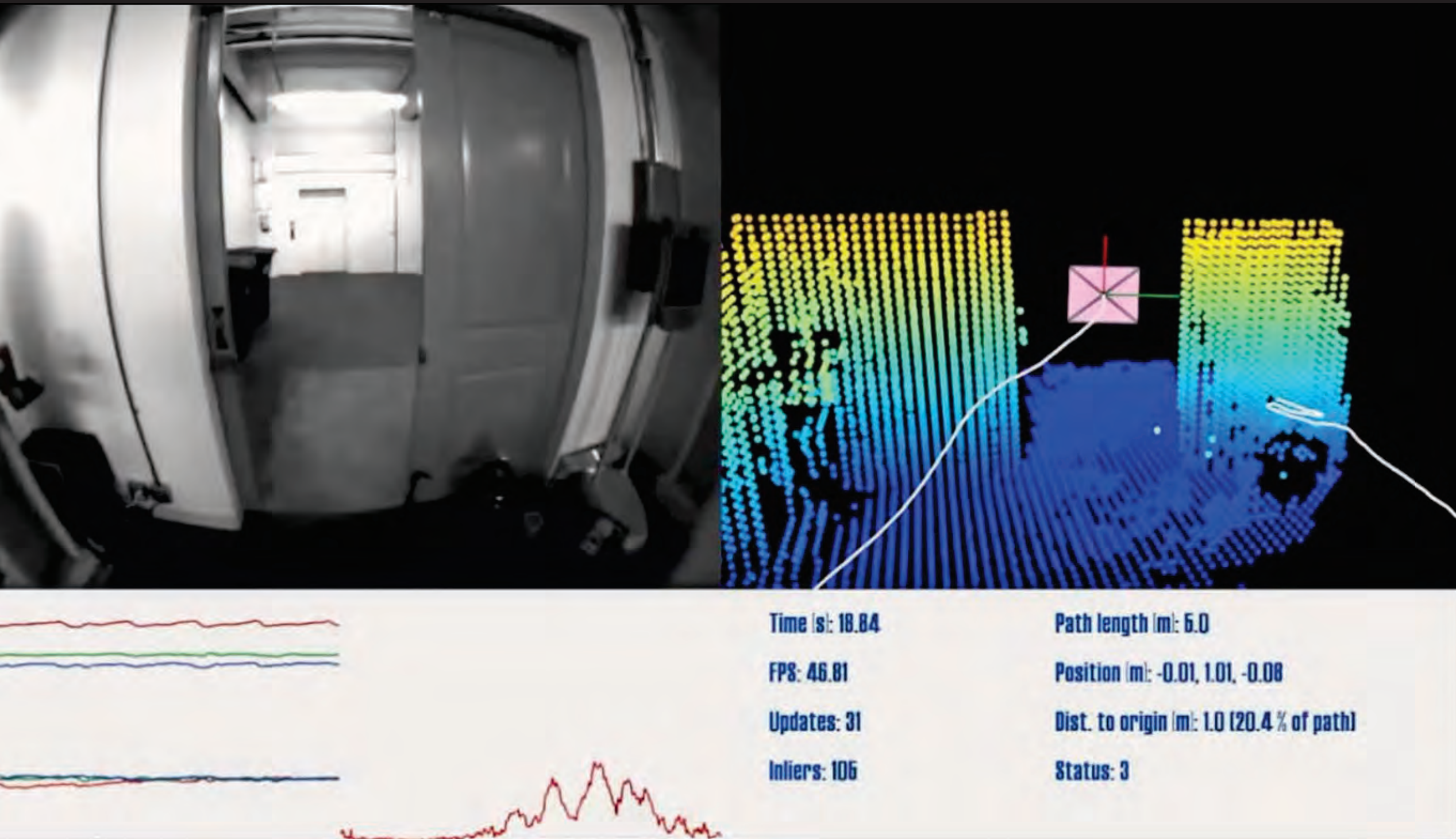
# Perception



[ CANVAS by Occipital (<https://canvas.io>) ]



# Sensing

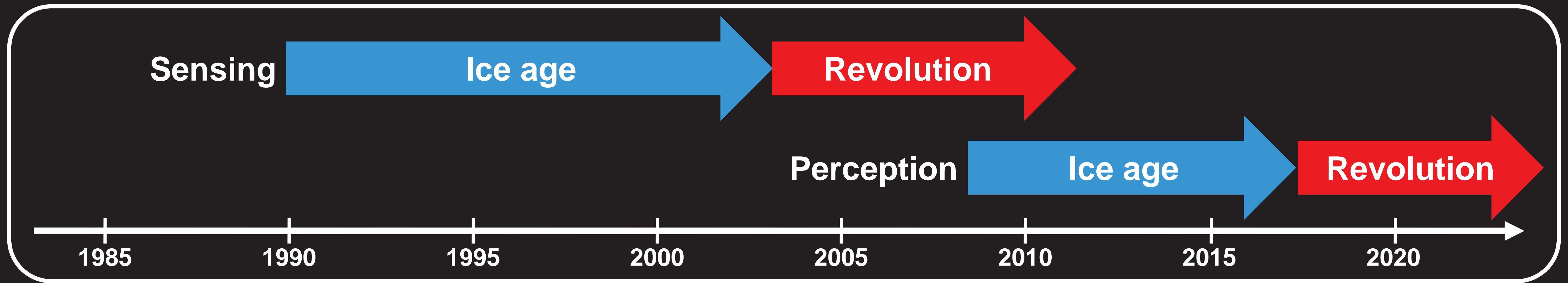


[ Google Project Tango ]

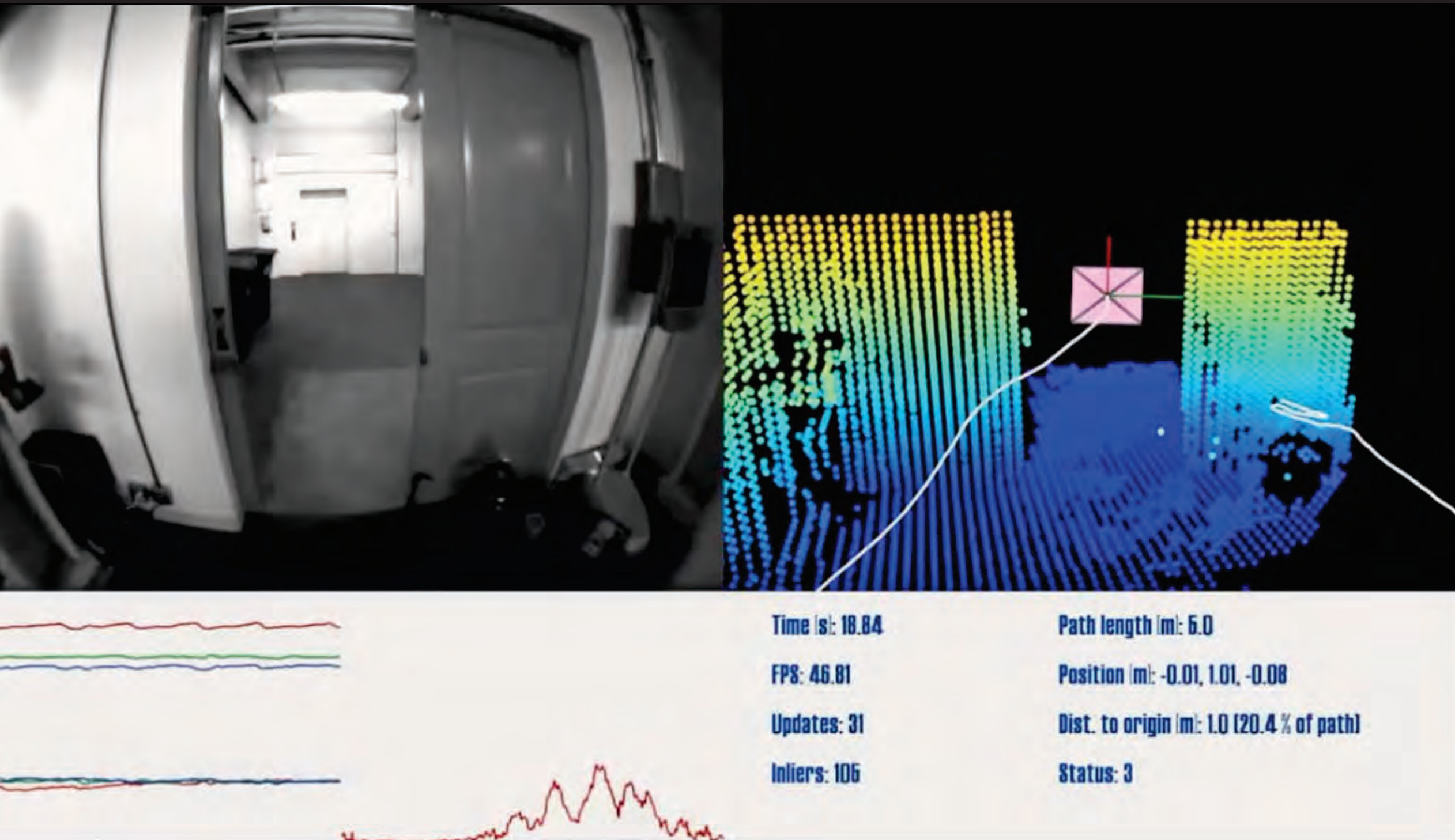
# Perception



[ CANVAS by Occipital (<https://canvas.io>) ]



# Sensing

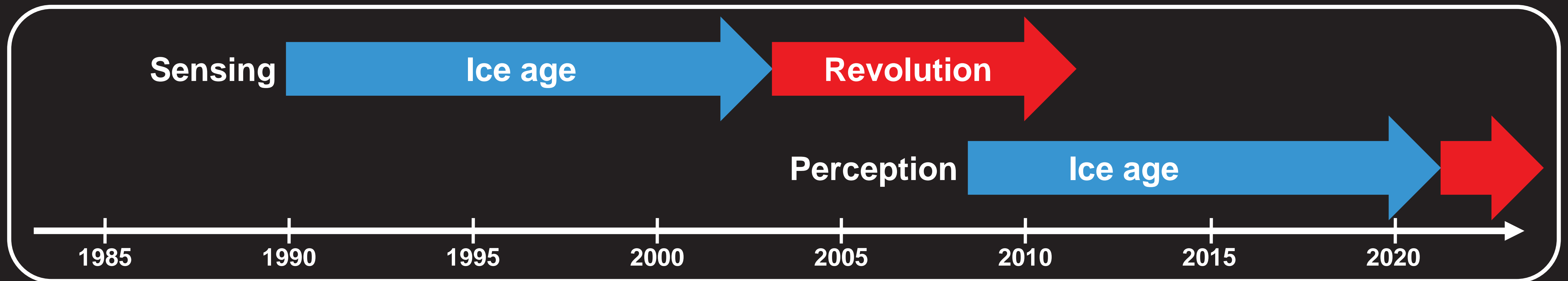


[ Google Project Tango ]

# Perception



[ CANVAS by Occipital (<https://canvas.io>) ]





# Sensing revolution



2007



2009



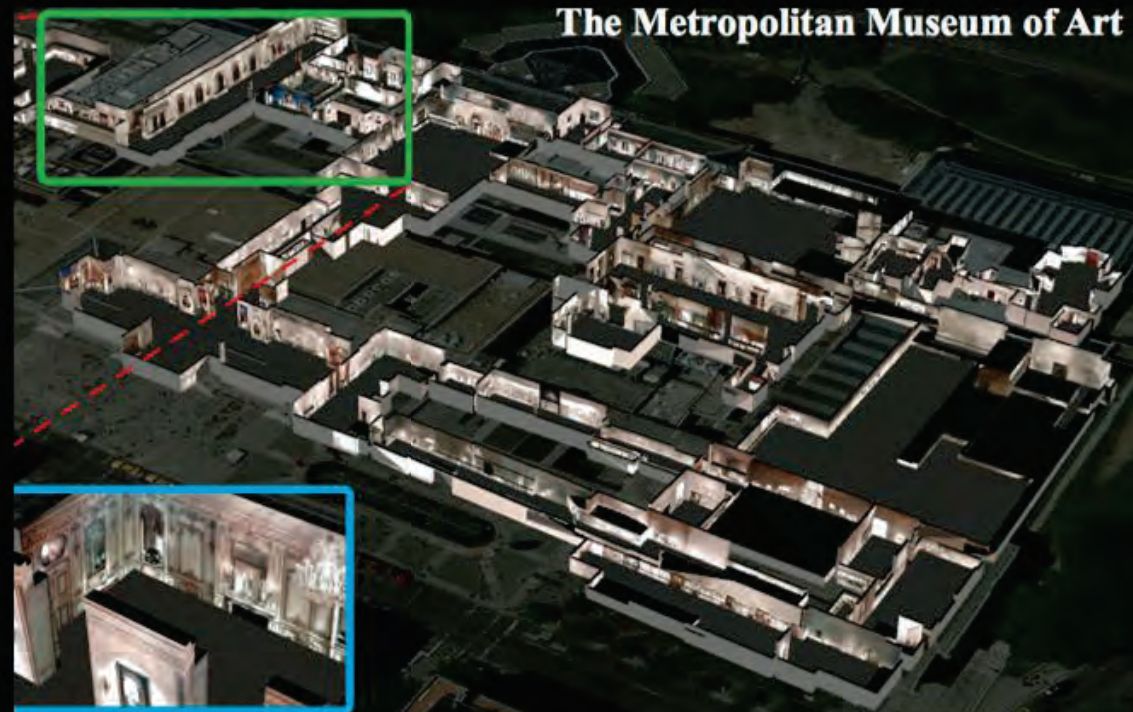
2010



2011



2012

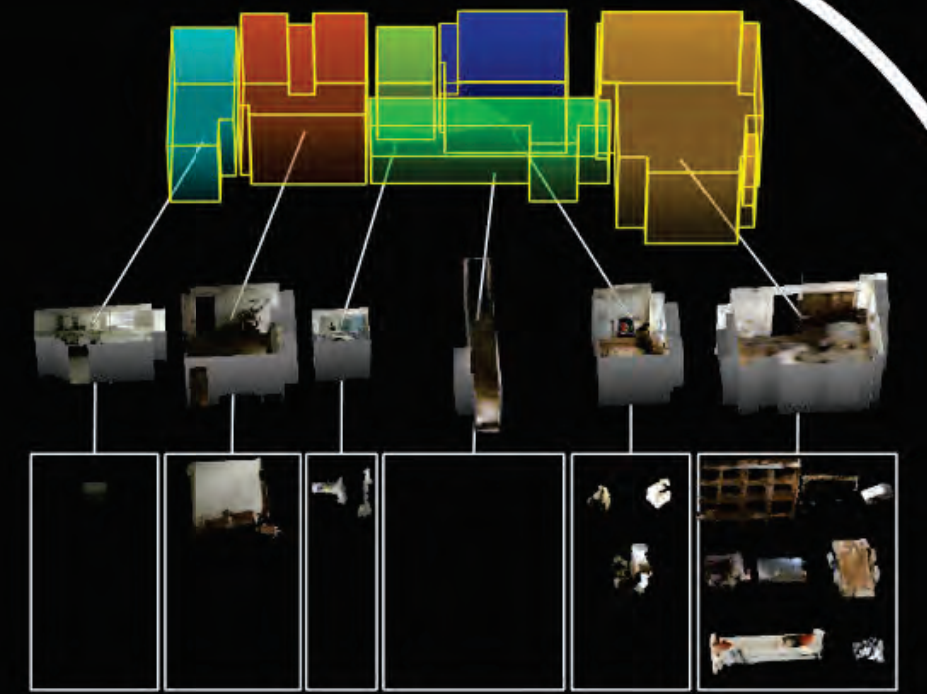


2012

# Perception - ice-age/revolution?



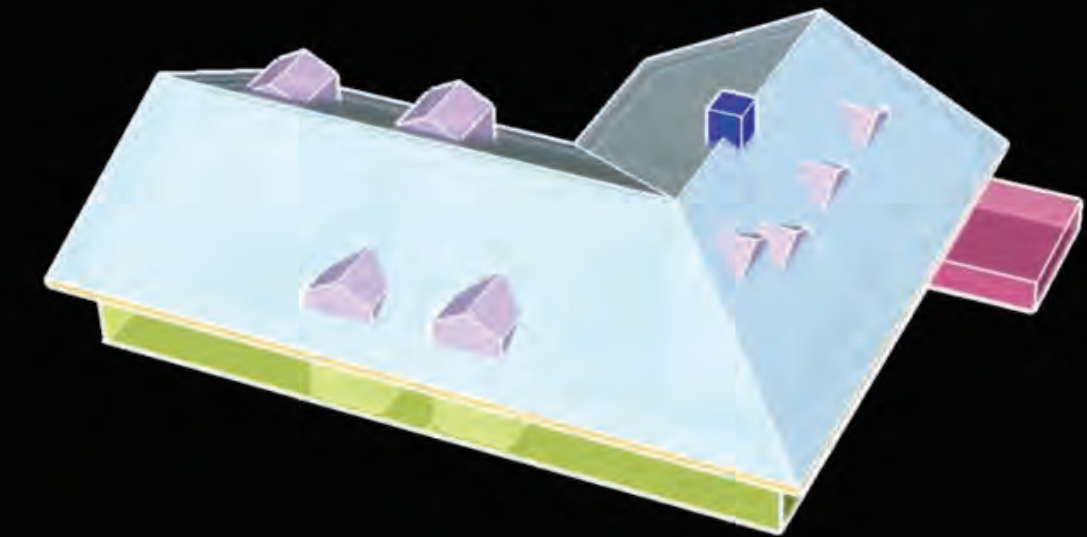
2014



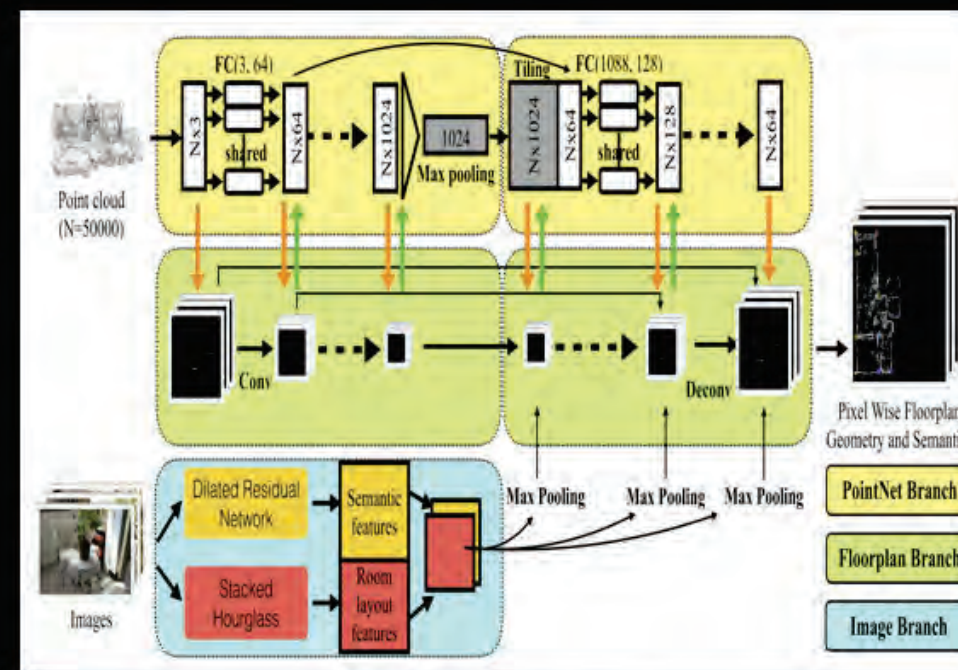
2015



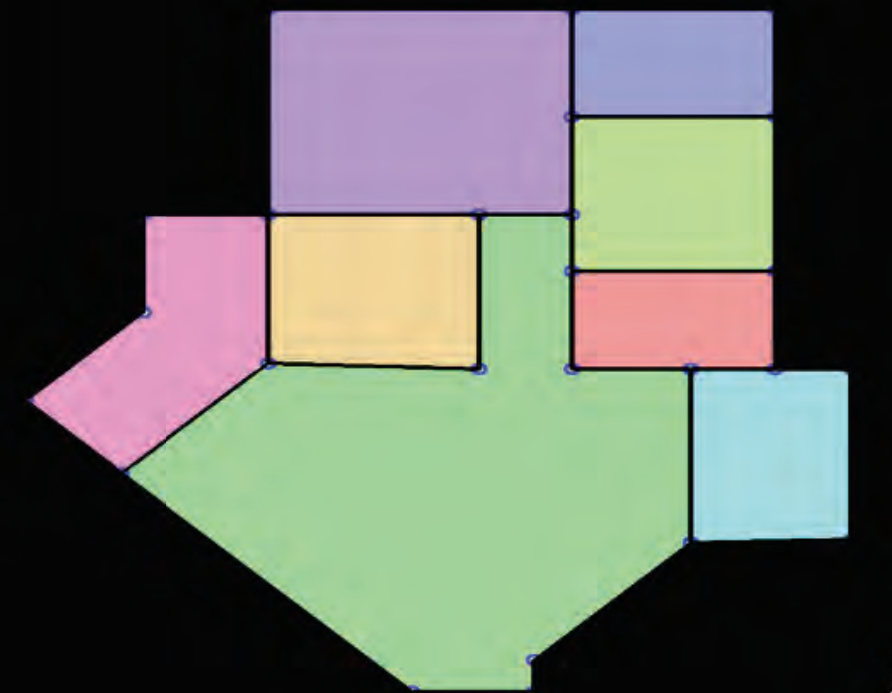
2017



2018



2018



2019

# Sensing ice age



2000

# Sensing revolution



2007



2009



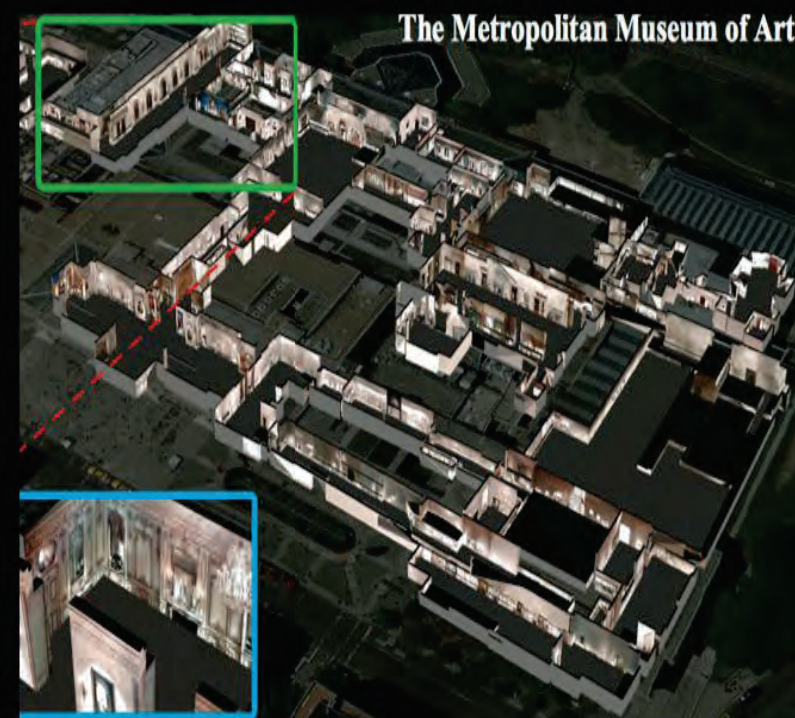
2010



2011



2012

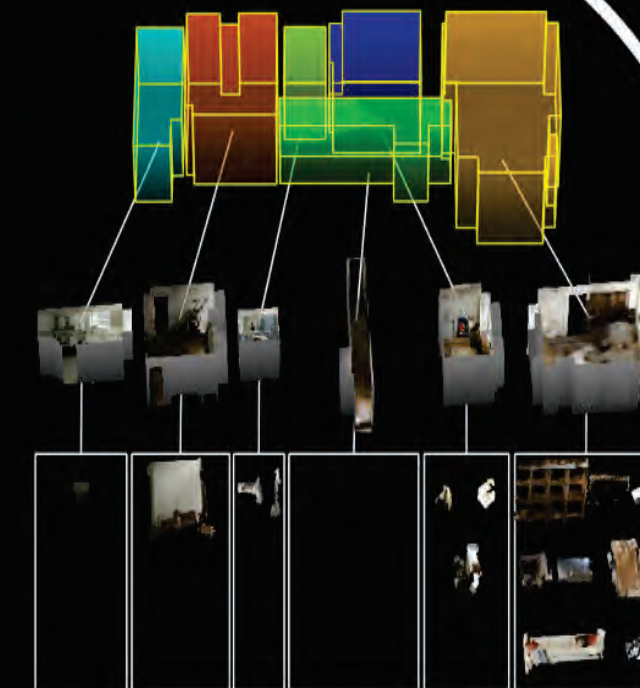


2012

# Perception - ice-age/revolution?



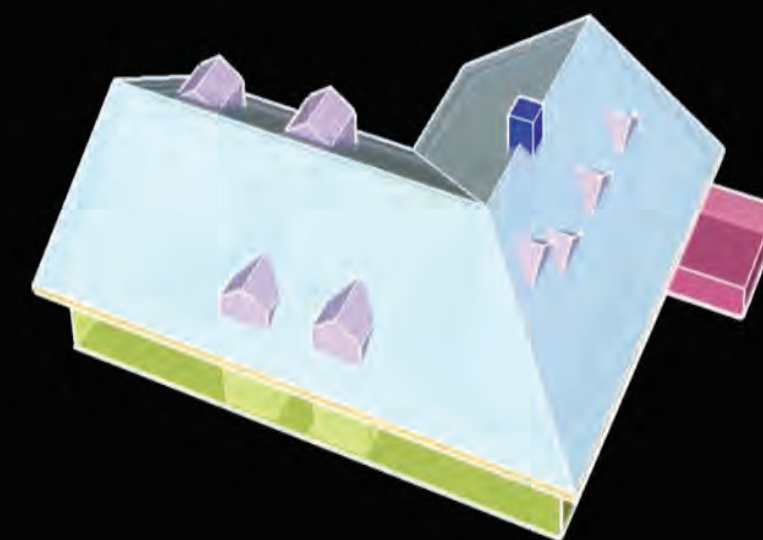
2014



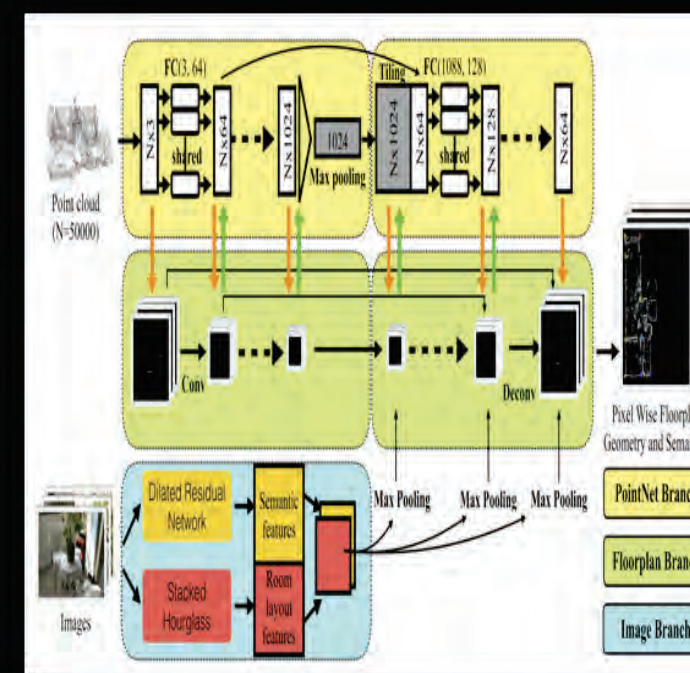
2015



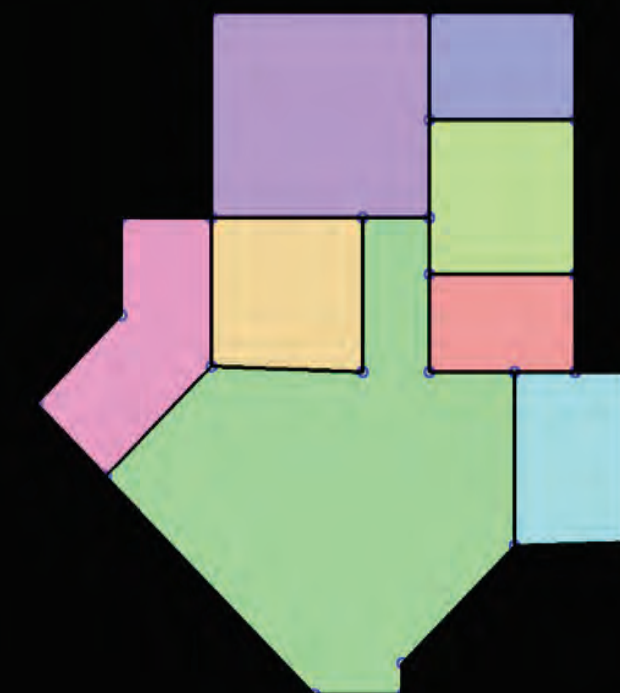
2017



2018



2018



2019

# Problem setting

- Input
  - Visual sensor data (e.g., image, depth-images, point-clouds, ...)
  - Calibration and camera poses (e.g., intrinsic/extrinsic camera params)
- Output
  - Geometric model (e.g., point-clouds, meshes, ...)

# Problem example: 3D reconstruction from images

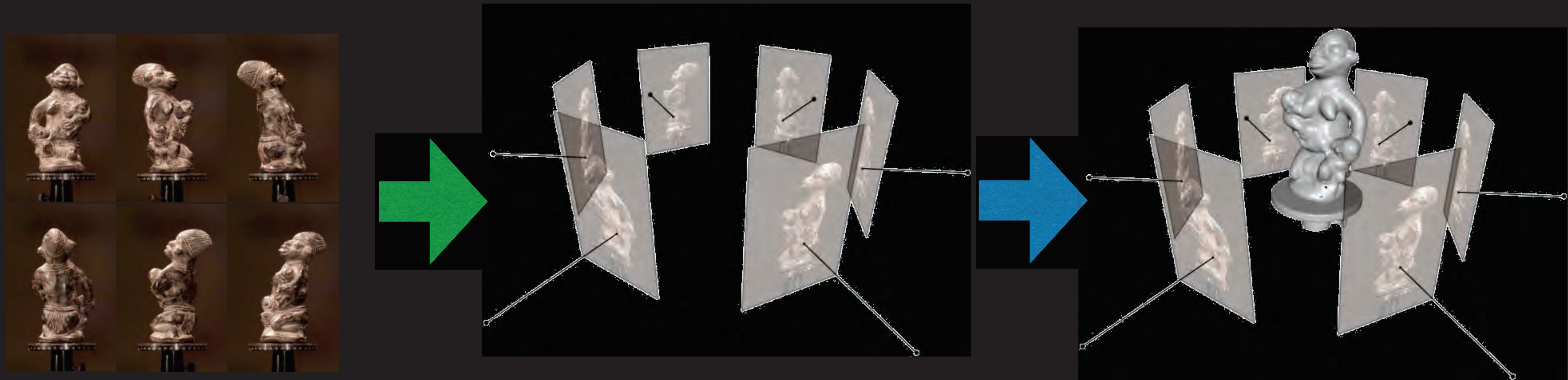


Image acquisition

Camera pose

3d reconstruction

- Structure from Motion (SfM)
- Simultaneous Localization & Mapping (SLAM)
- Match-moving
- Visual Odometry

Multi-view Stereo (MVS)

# Problem setting

- Input
  - Visual sensor data (e.g., image, depth-images, point-clouds, ...)
  - Calibration and camera poses (e.g., intrinsic/extrinsic camera params)
- Output
  - Geometric model (e.g., point-clouds, meshes, ...)



Refer to “Structure-from-Motion” (SfM) and  
“Simultaneous Localization & Mapping” (SLAM) literatures.

# Sensing ice age



2000

# Sensing revolution



2007



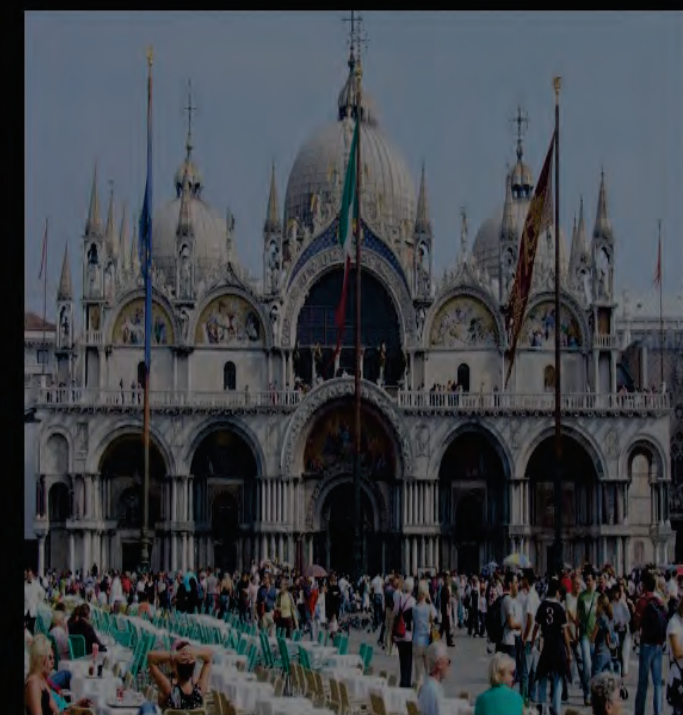
2009



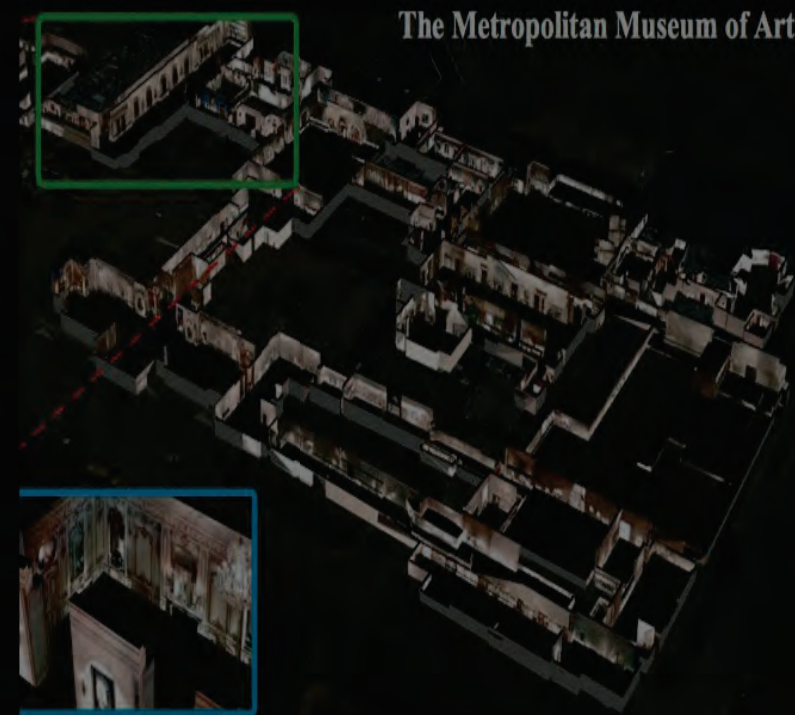
2010



2011



2012

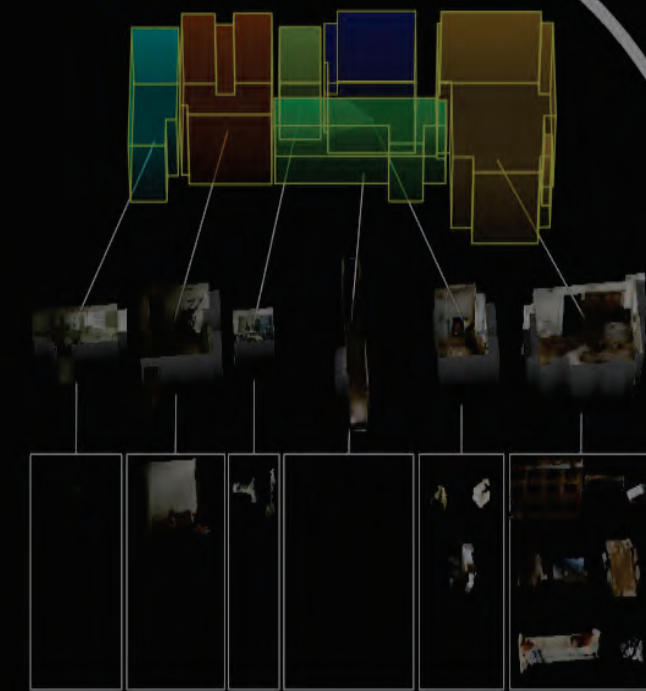


2012

# Perception - ice-age/revolution?



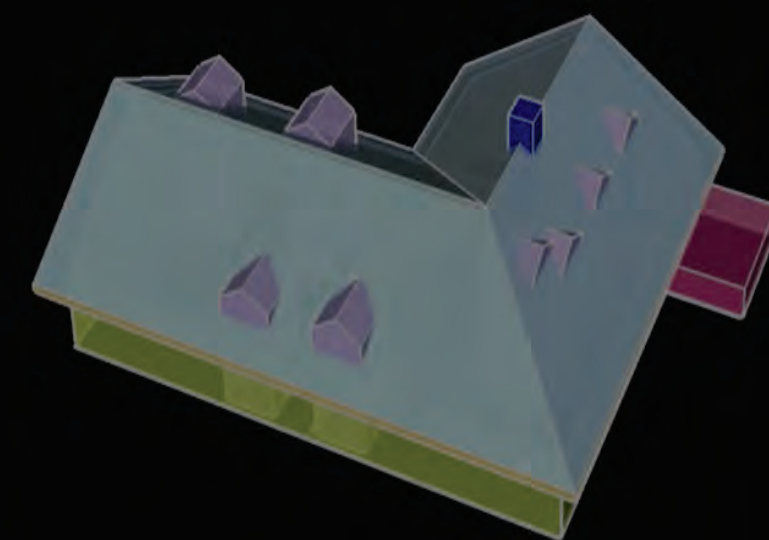
2014



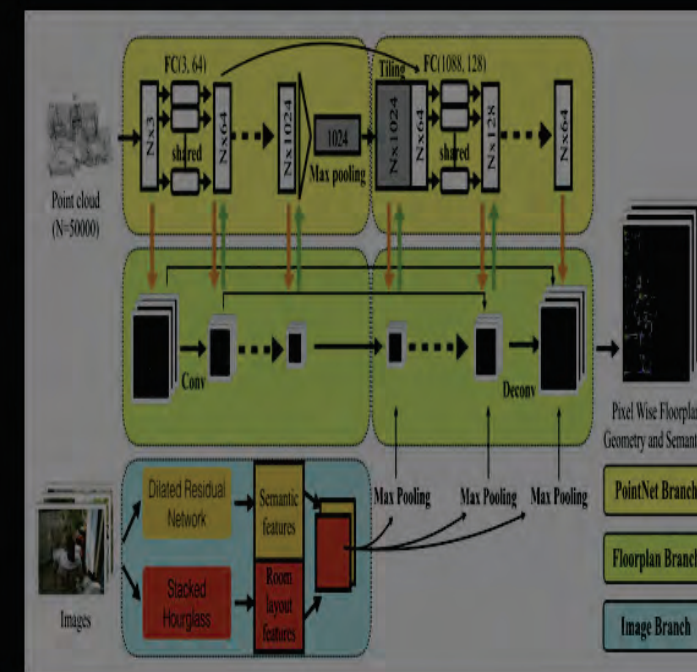
2015



2017



2018



2018



2019

# 3 mistakes in “sensing ice-age”

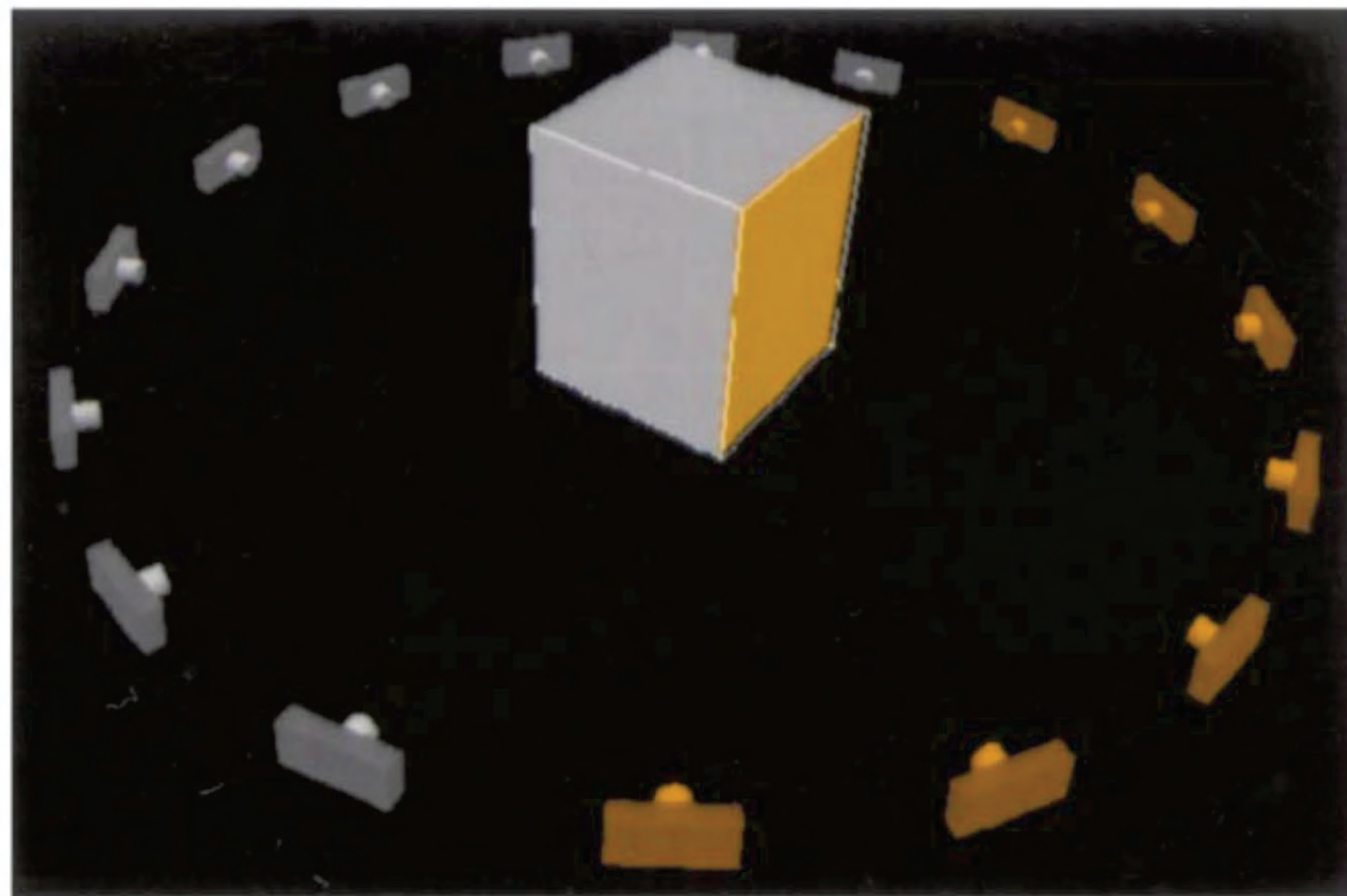
- Point-wise photo consistency
- Global optimization dilemma
- Occlusion dilemma



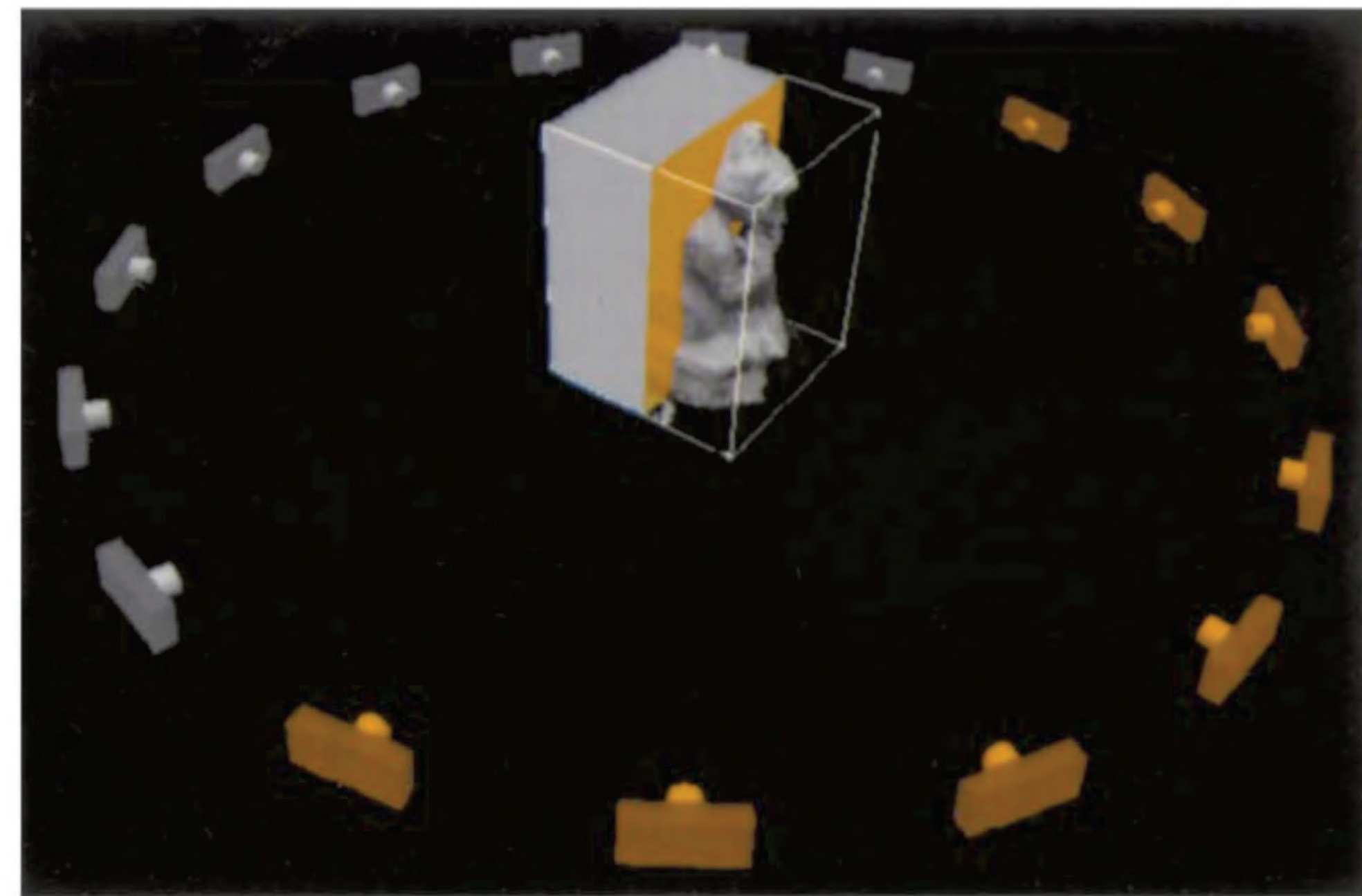
# A Theory of Shape by Space Carving

Steve Seitz and Kyros Kutulakos

ICCV 1999, Best Paper Award (Marr Prize)



(a)



(b)

*Figure 5.* Plane-Sweep Visibility. (a) The plane-sweep algorithm ensures that voxels are visited in order of visibility with respect to all active cameras. The current plane and active set of cameras is shown in orange. (b) The shape evolves and new cameras become active as the plane moves through the scene volume.



# Space Carving: An algorithm

## *Plane Sweep Algorithm*

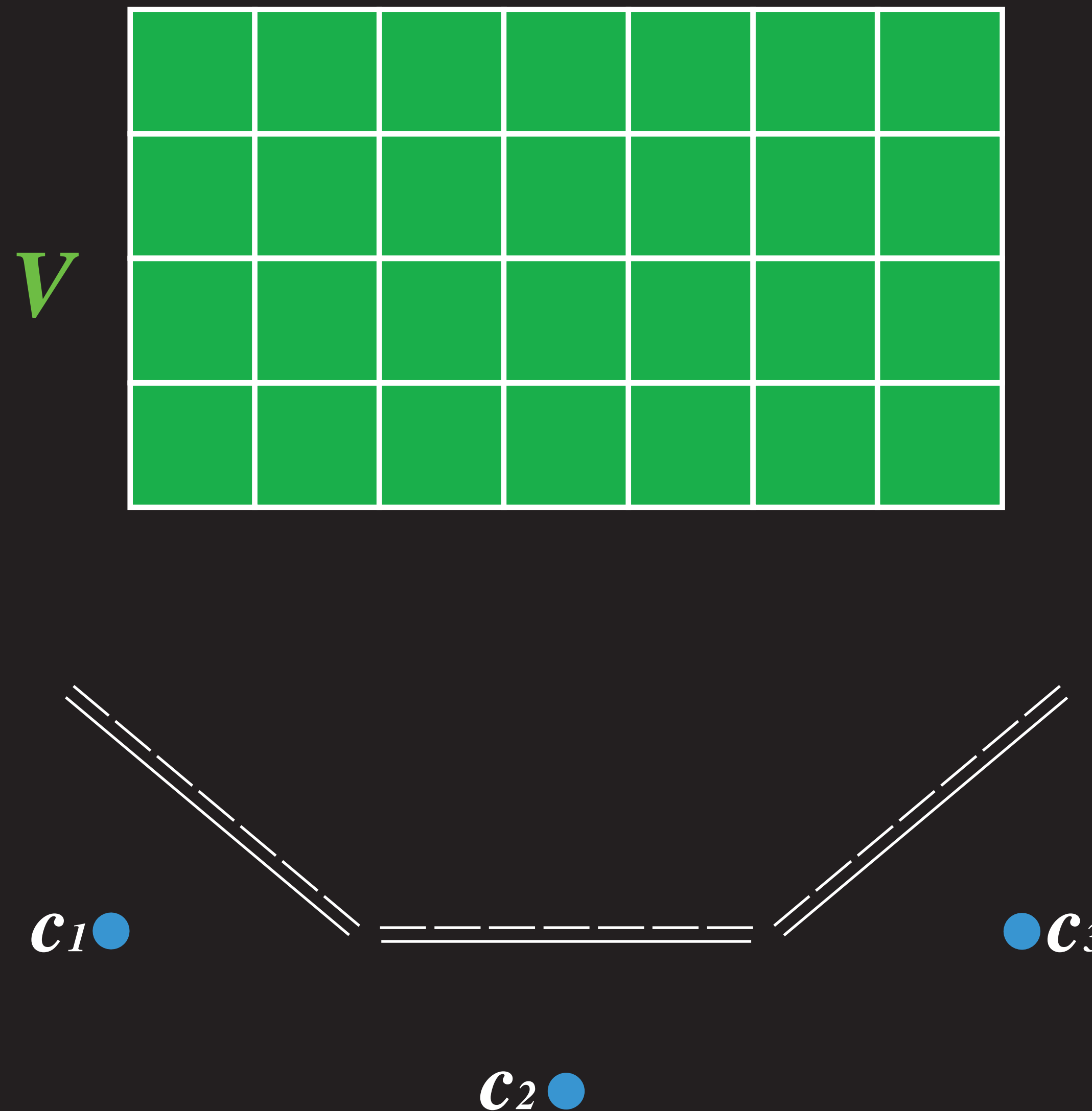
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## *Plane Sweep Algorithm*

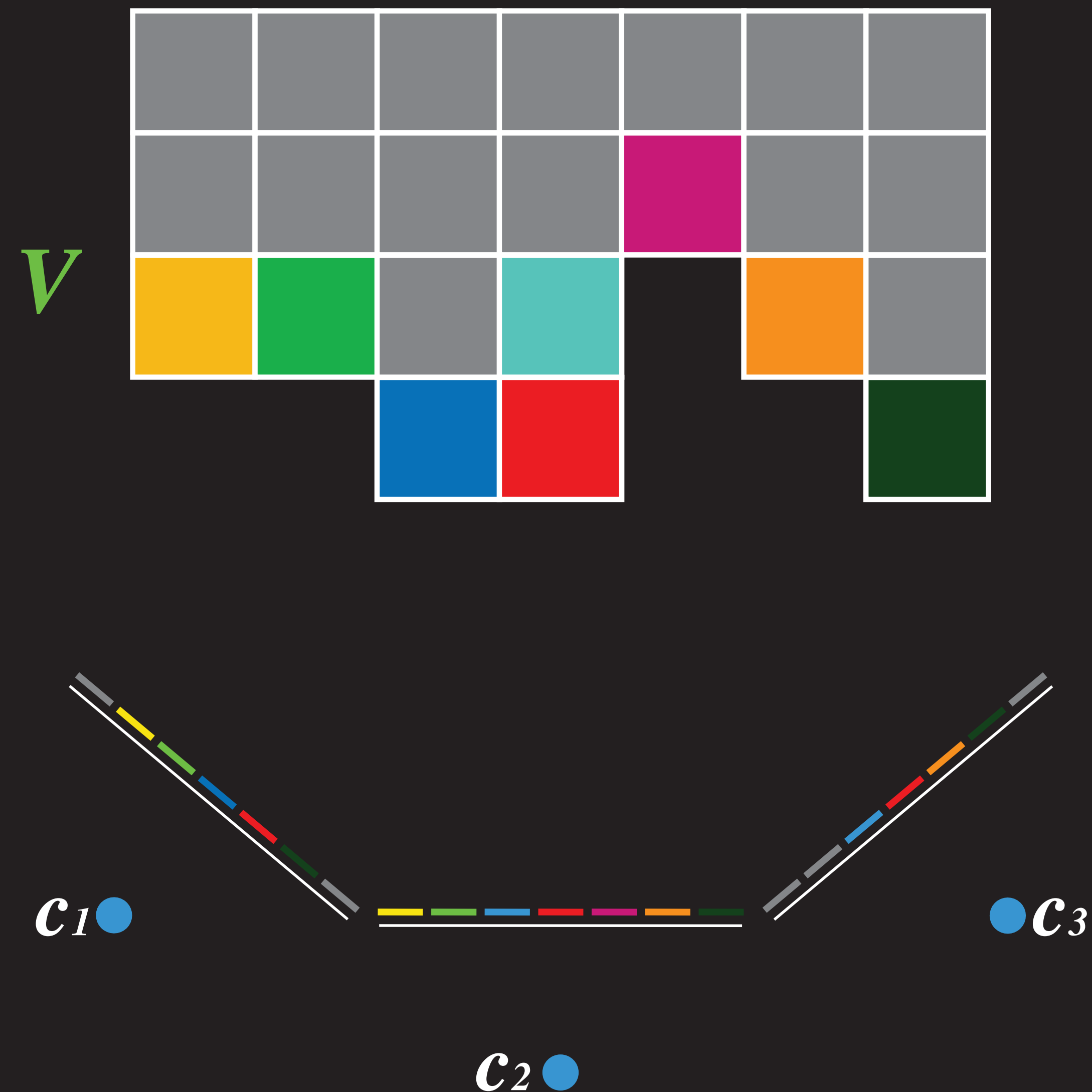
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## *Plane Sweep Algorithm*

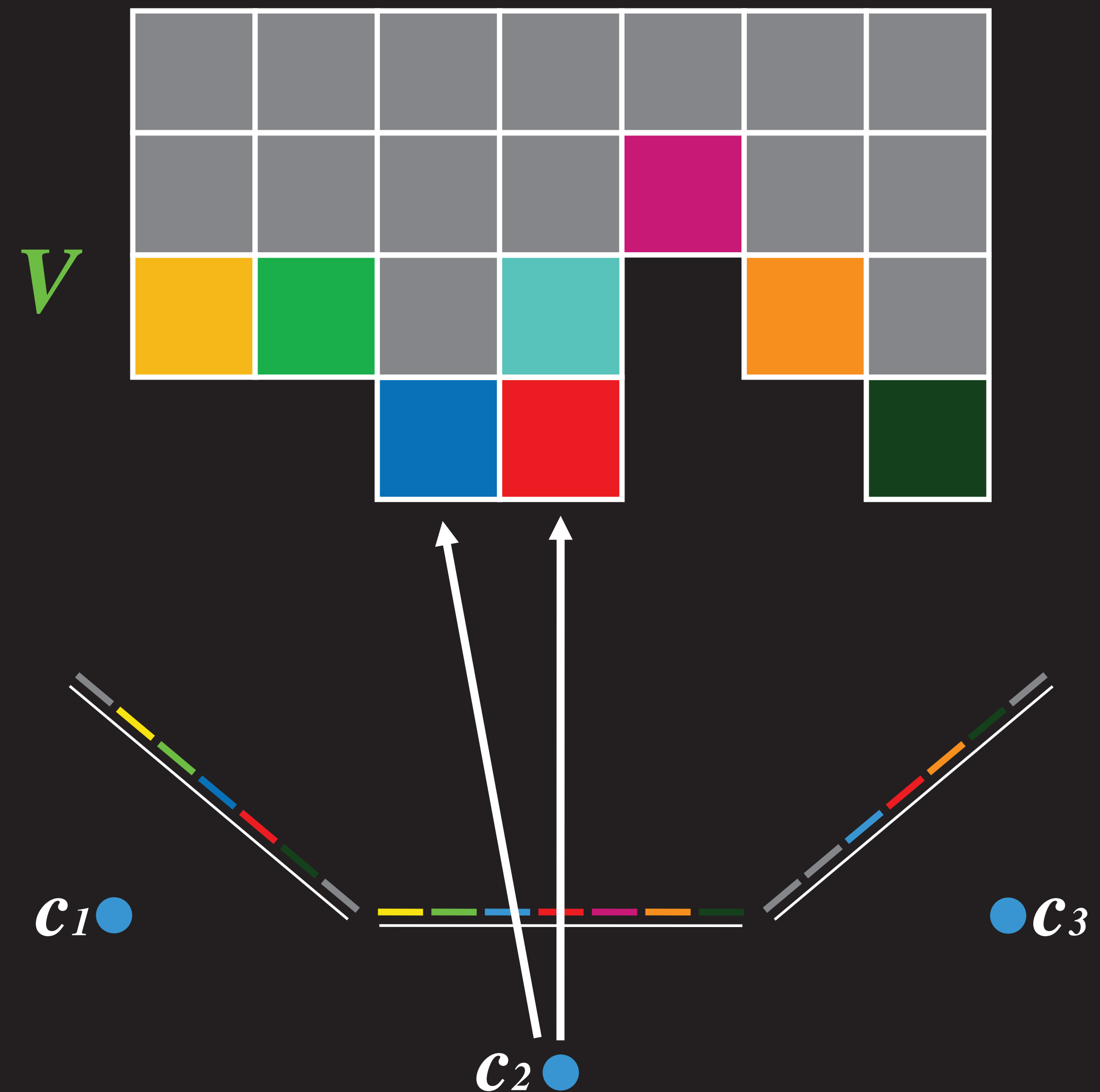
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## *Plane Sweep Algorithm*

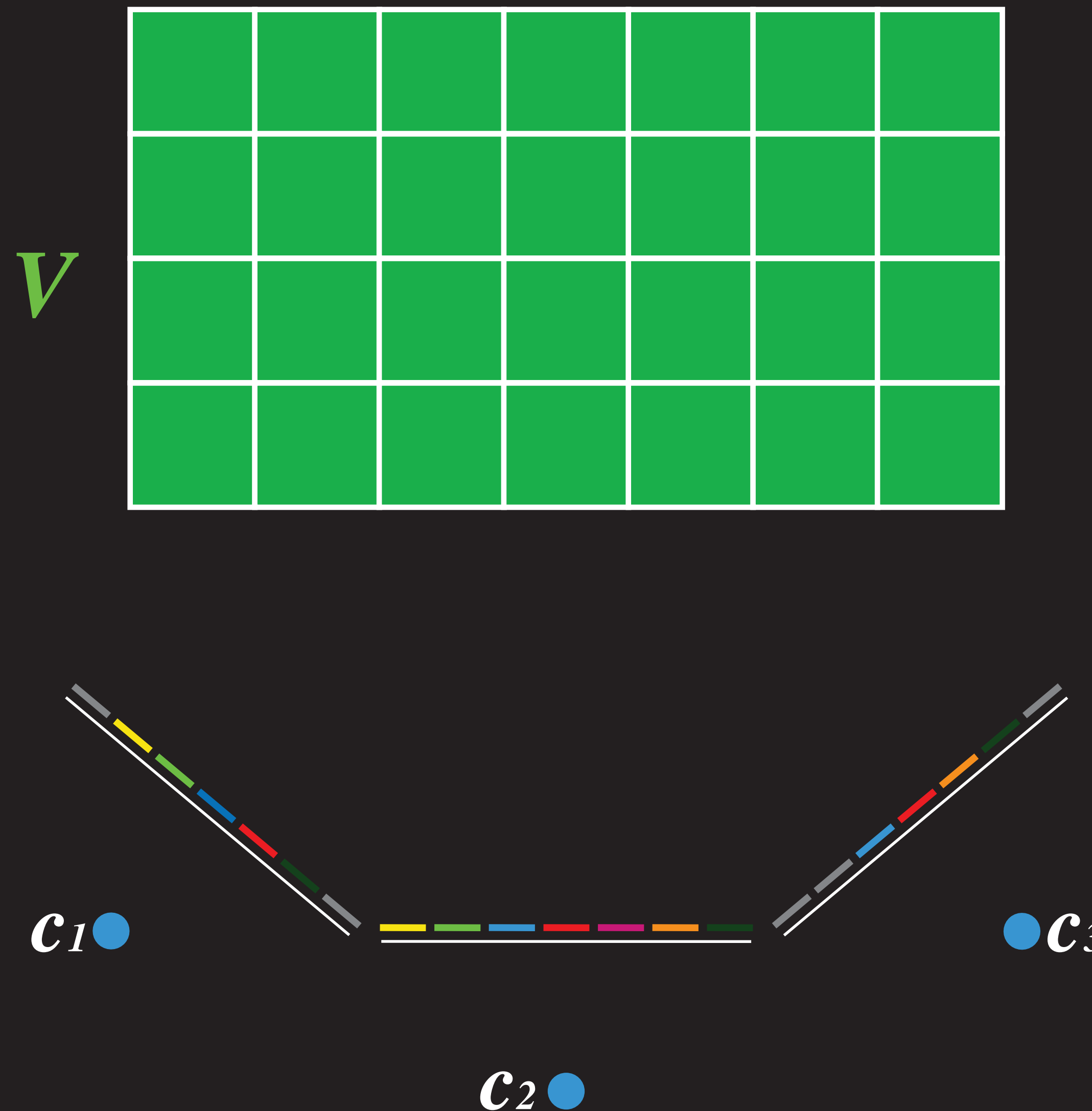
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

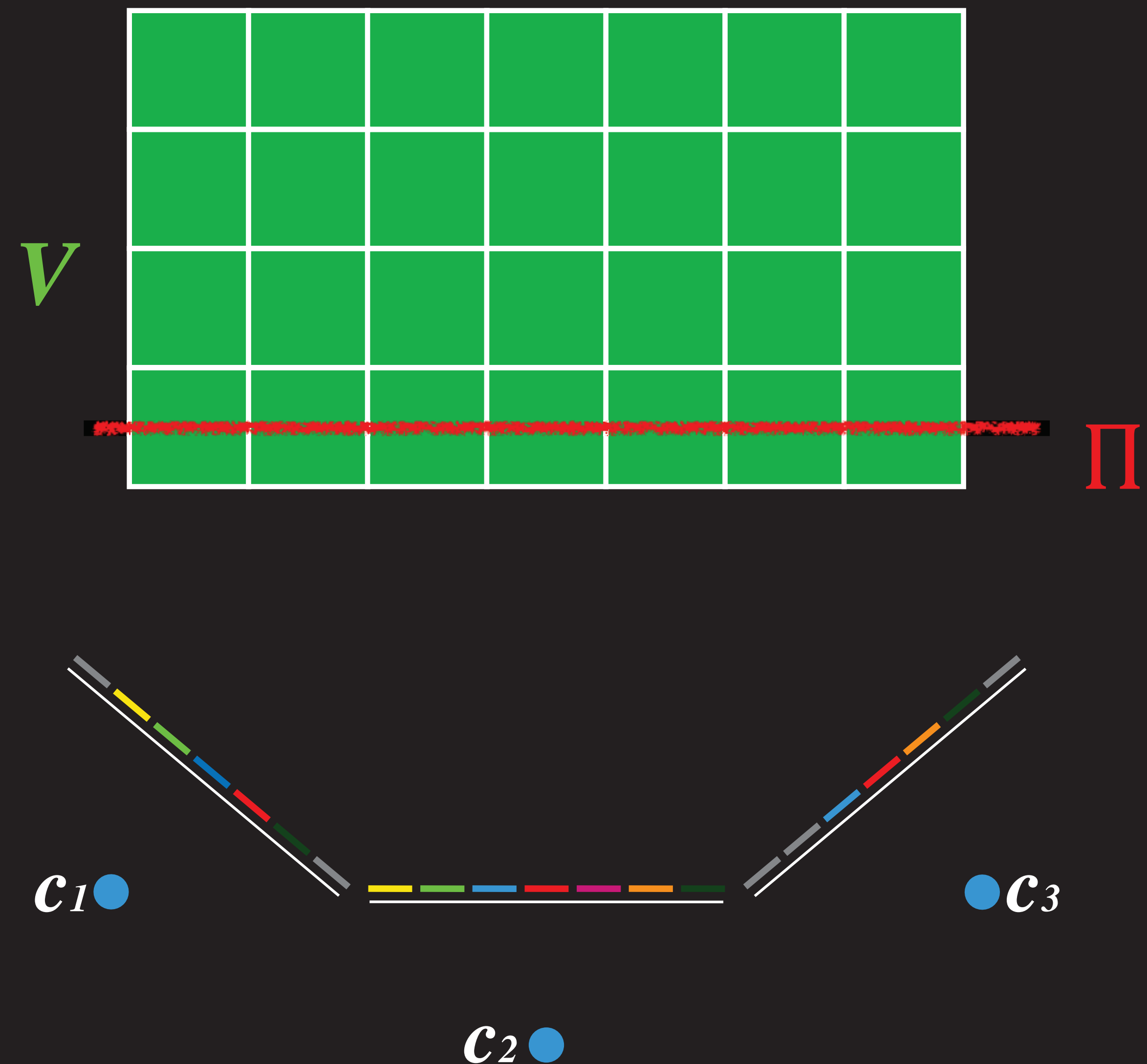
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

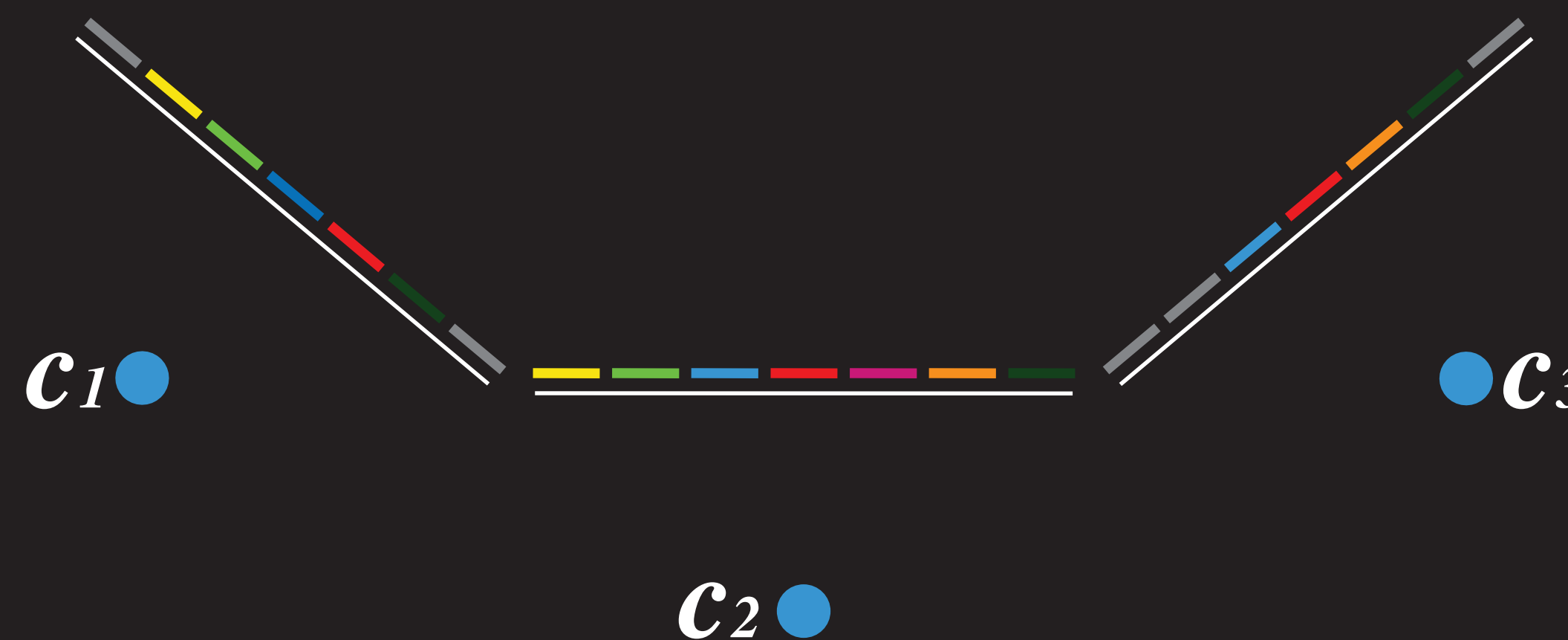
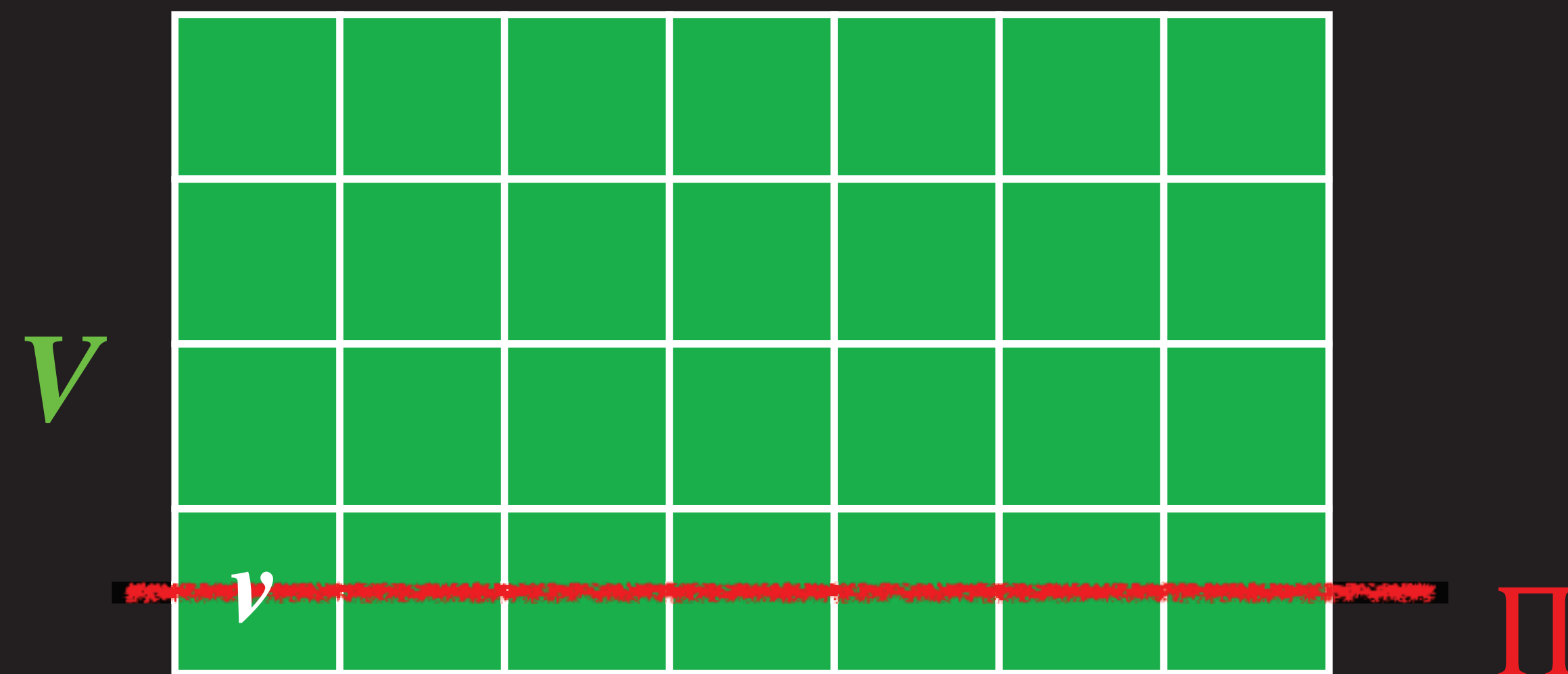
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

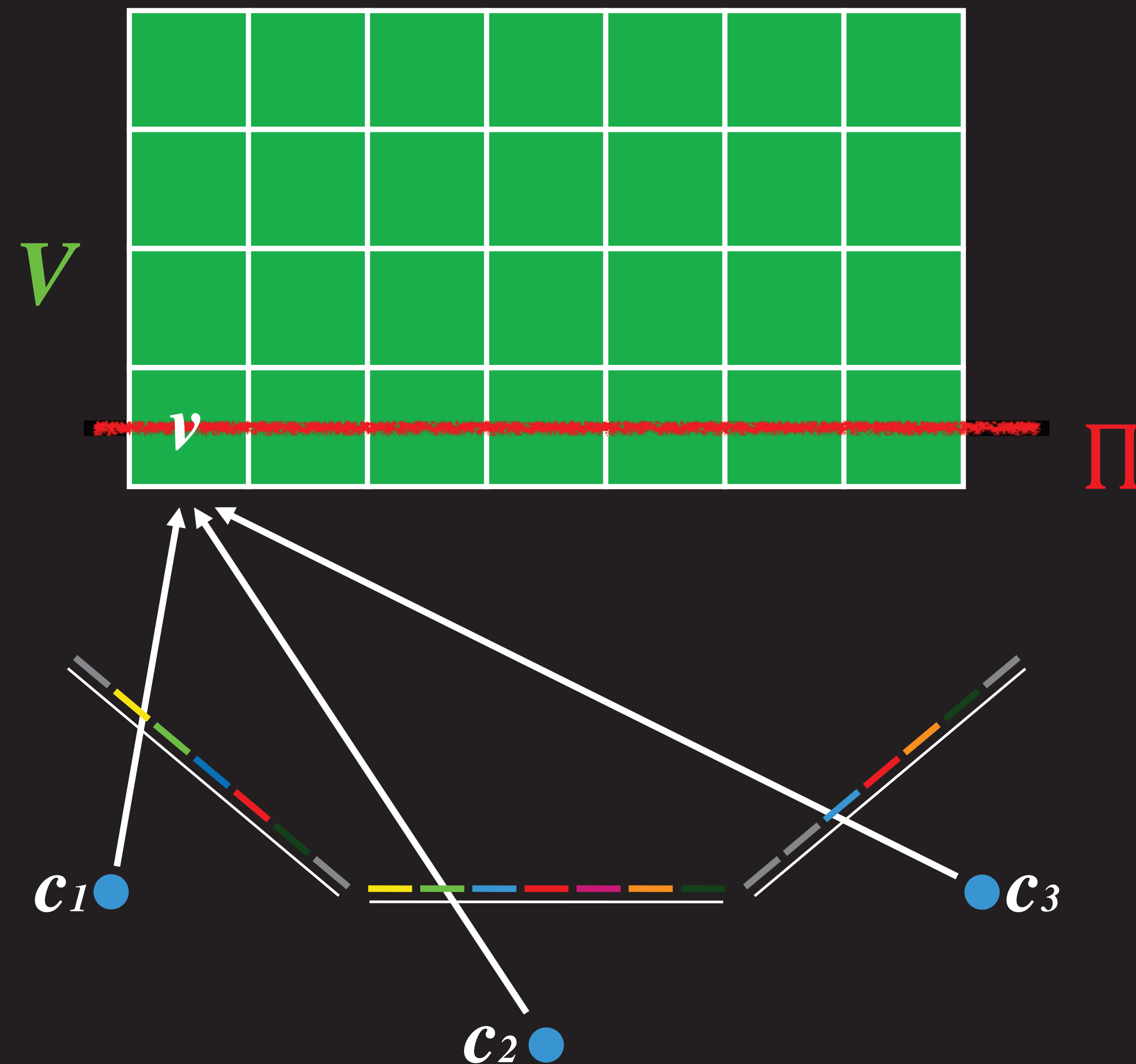
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

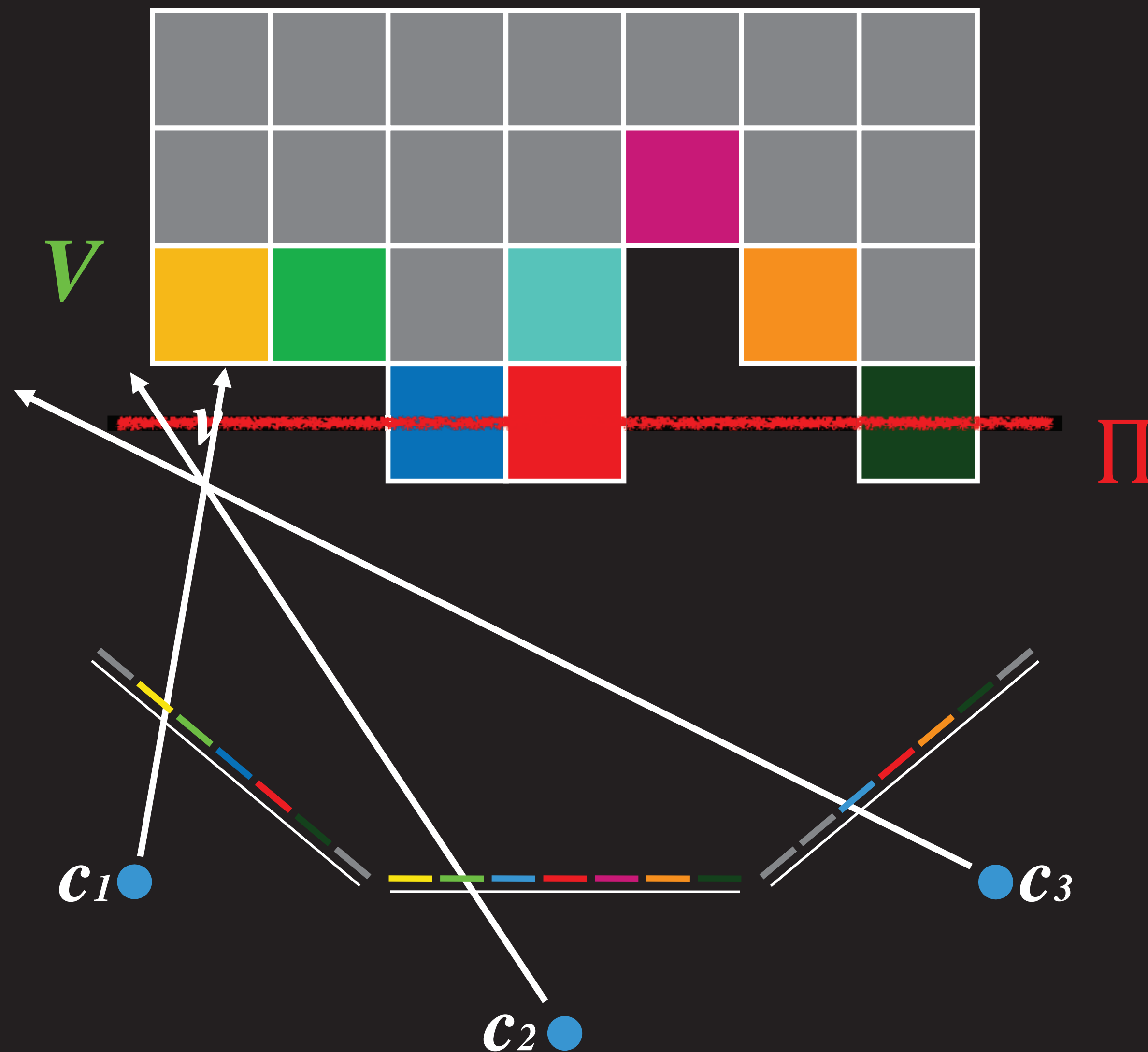
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .





# Space Carving: An algorithm

## Plane Sweep Algorithm

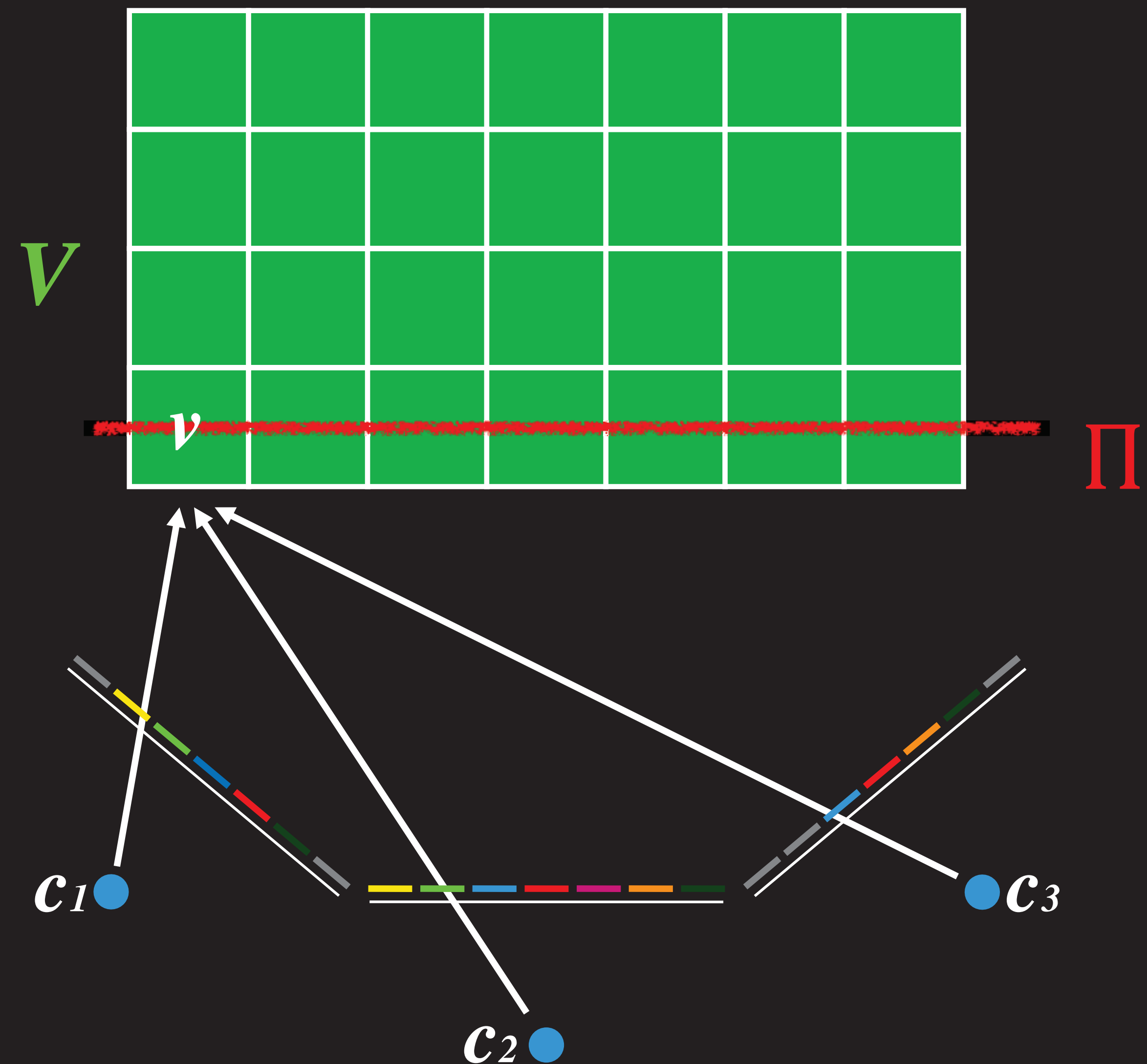
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

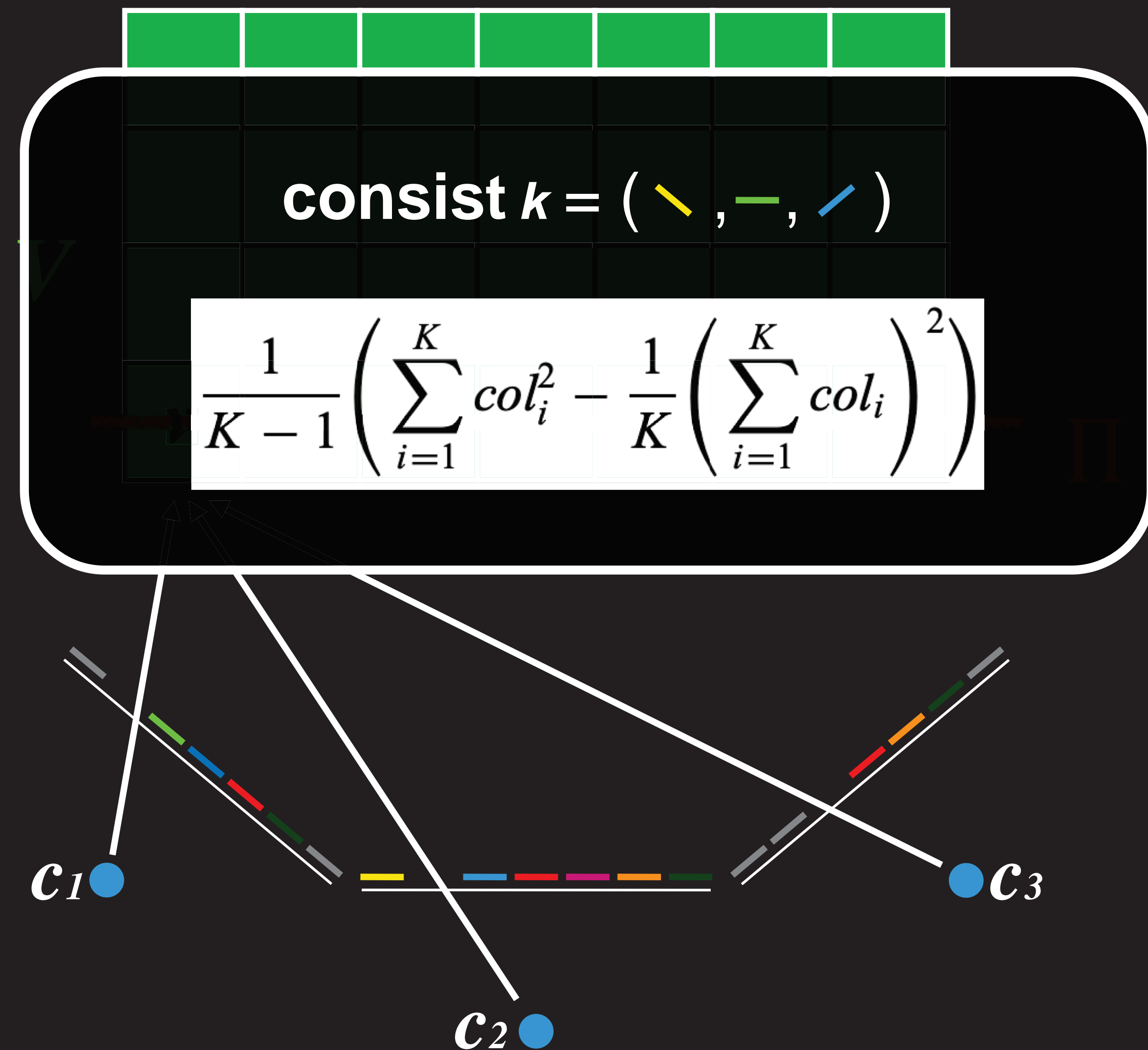
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

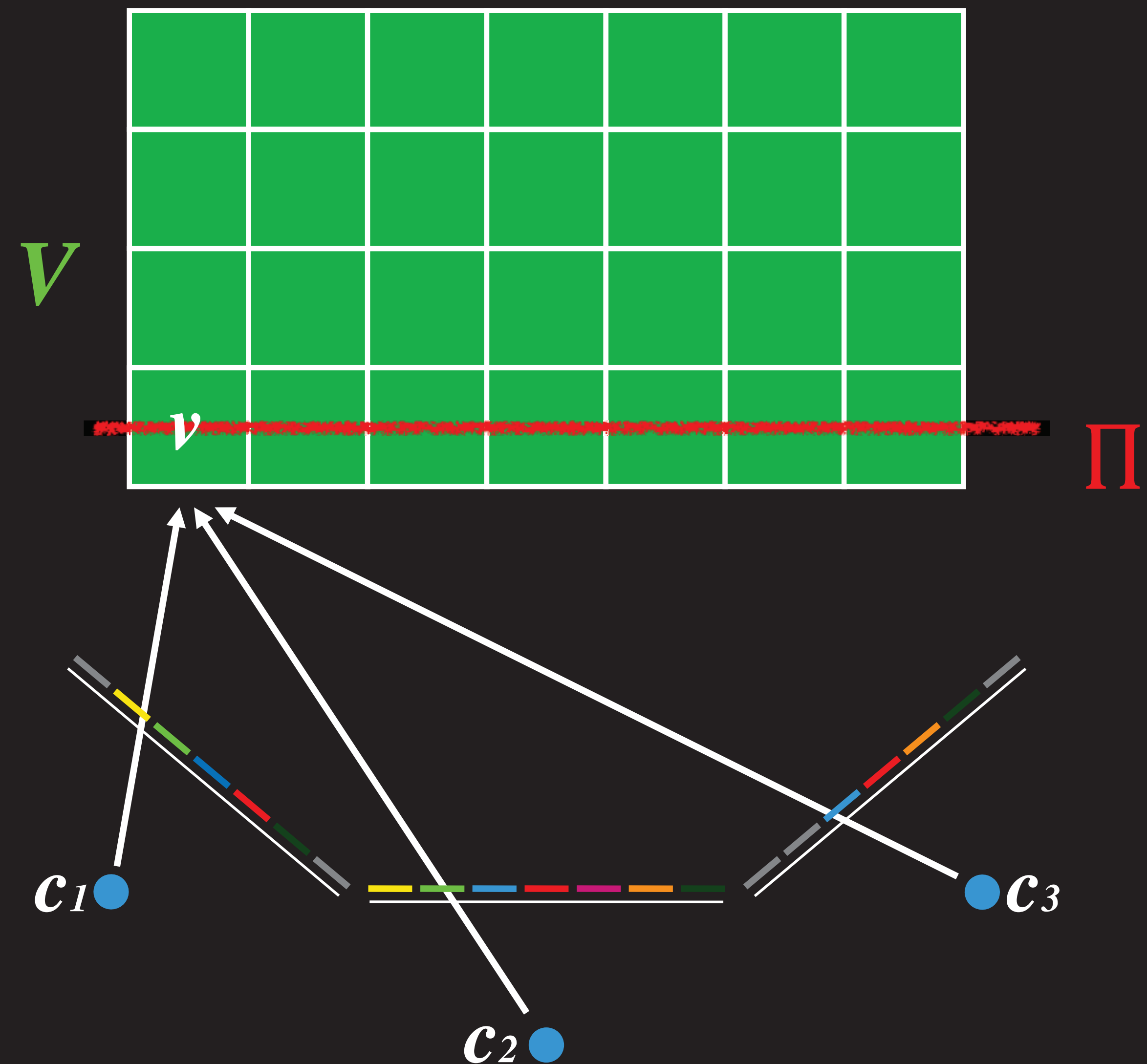
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

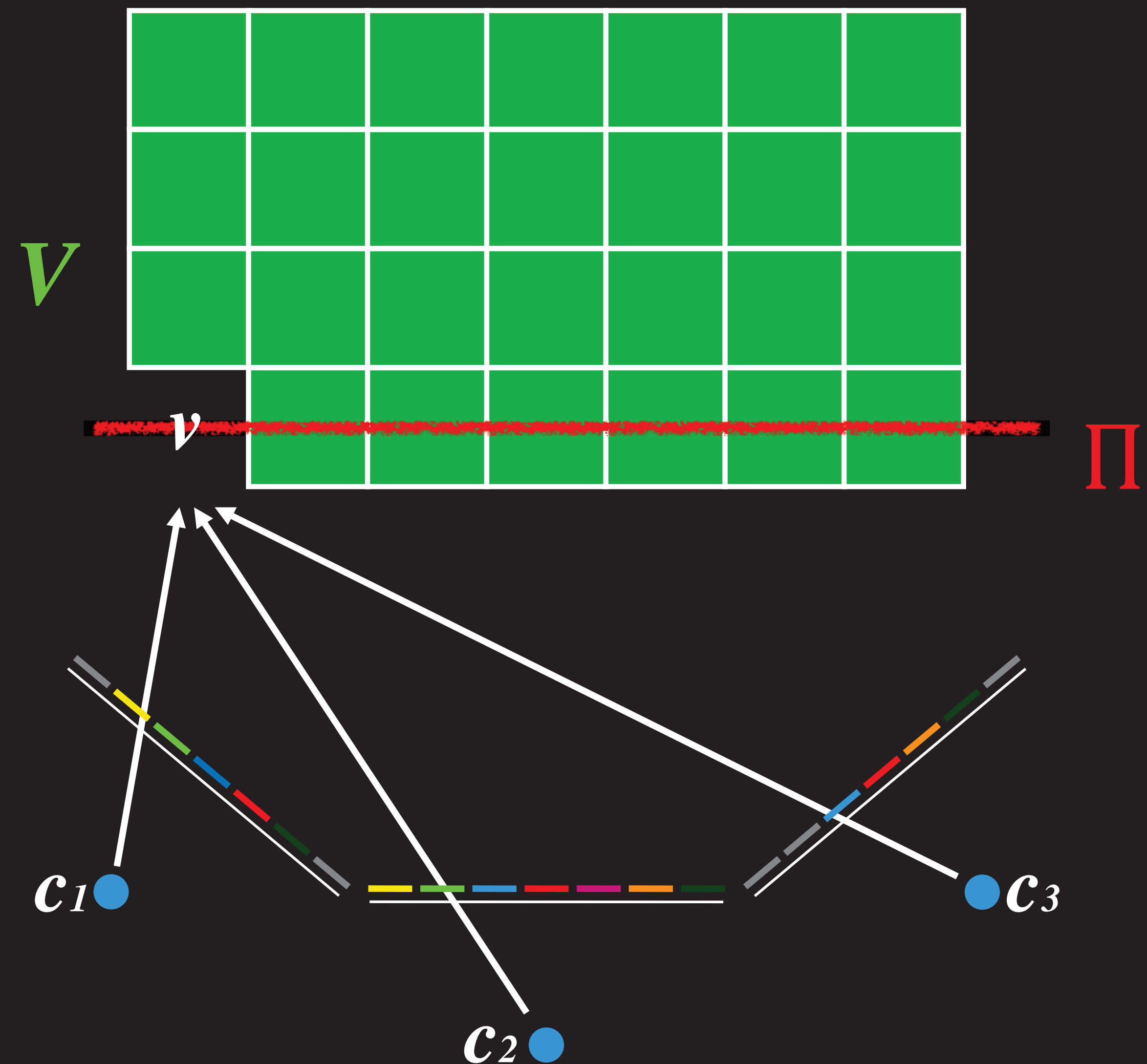
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

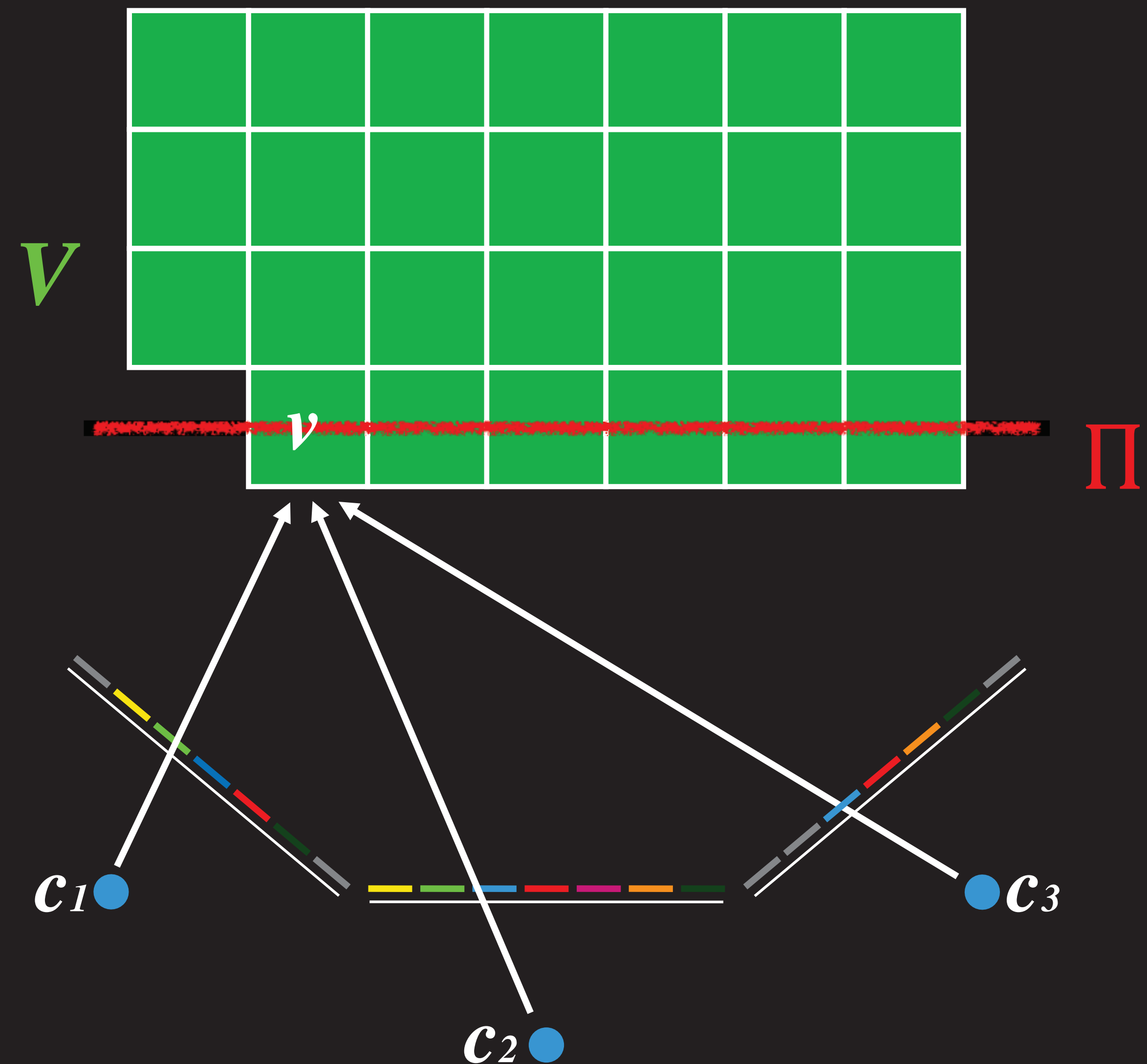
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

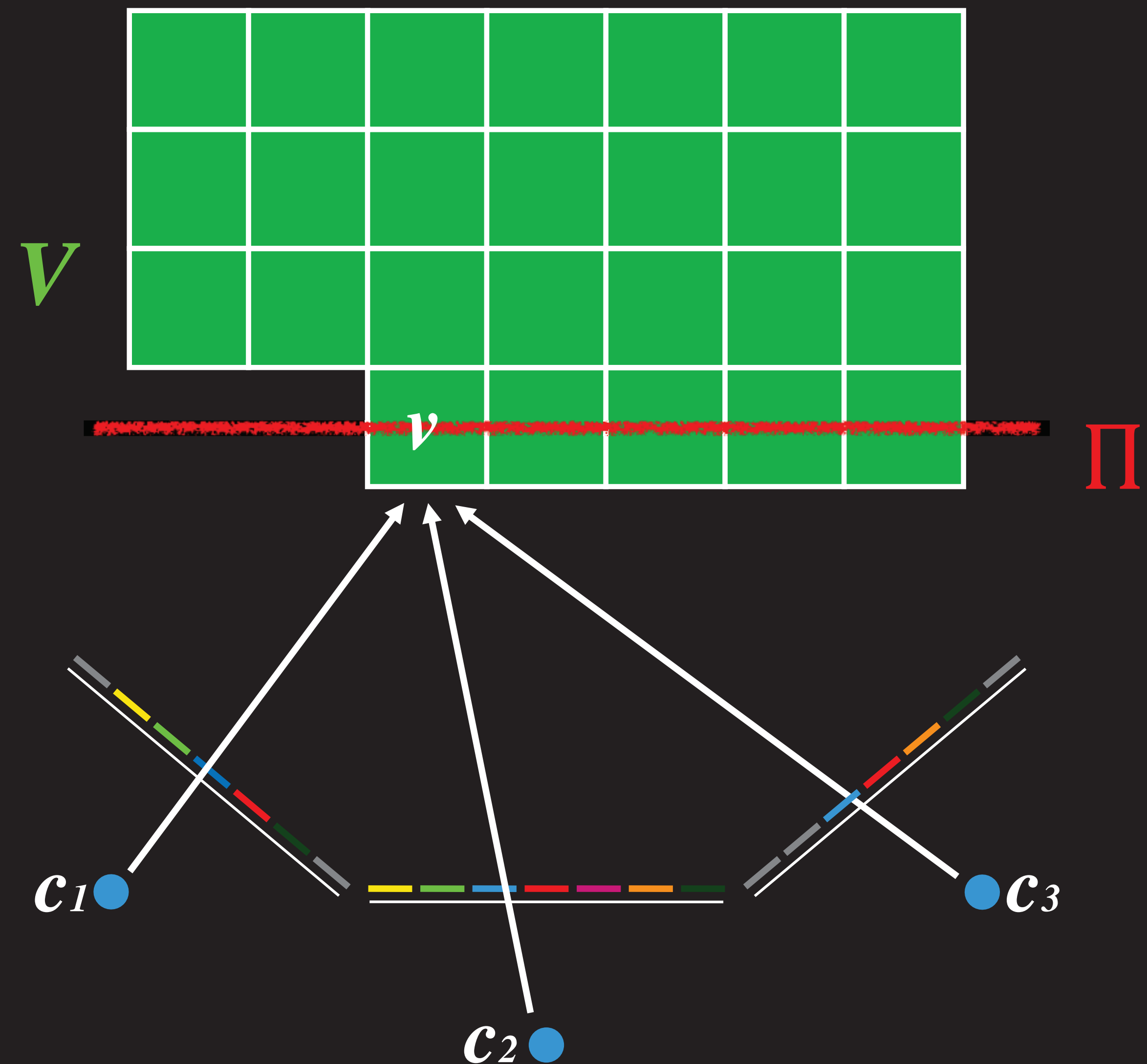
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

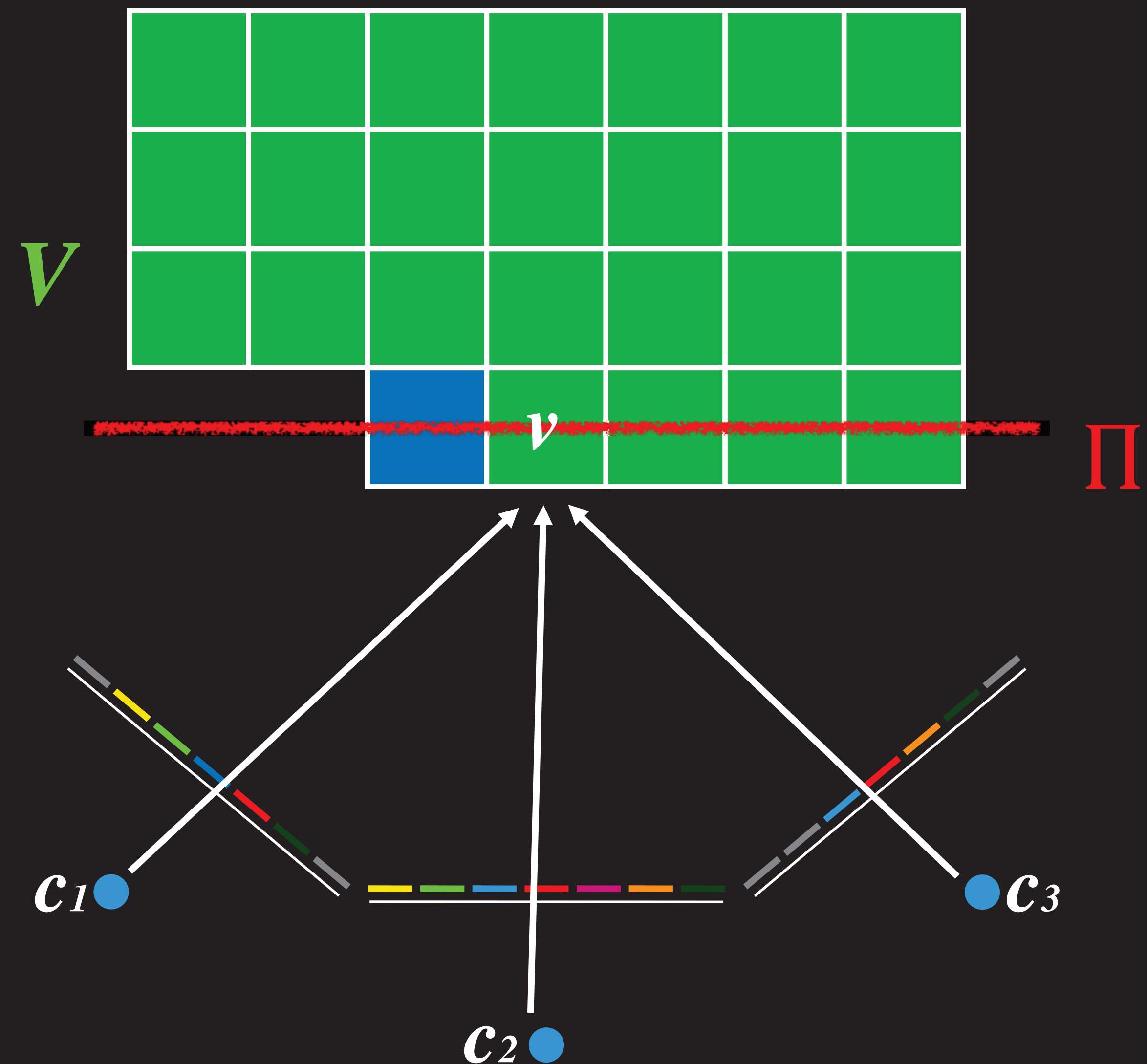
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

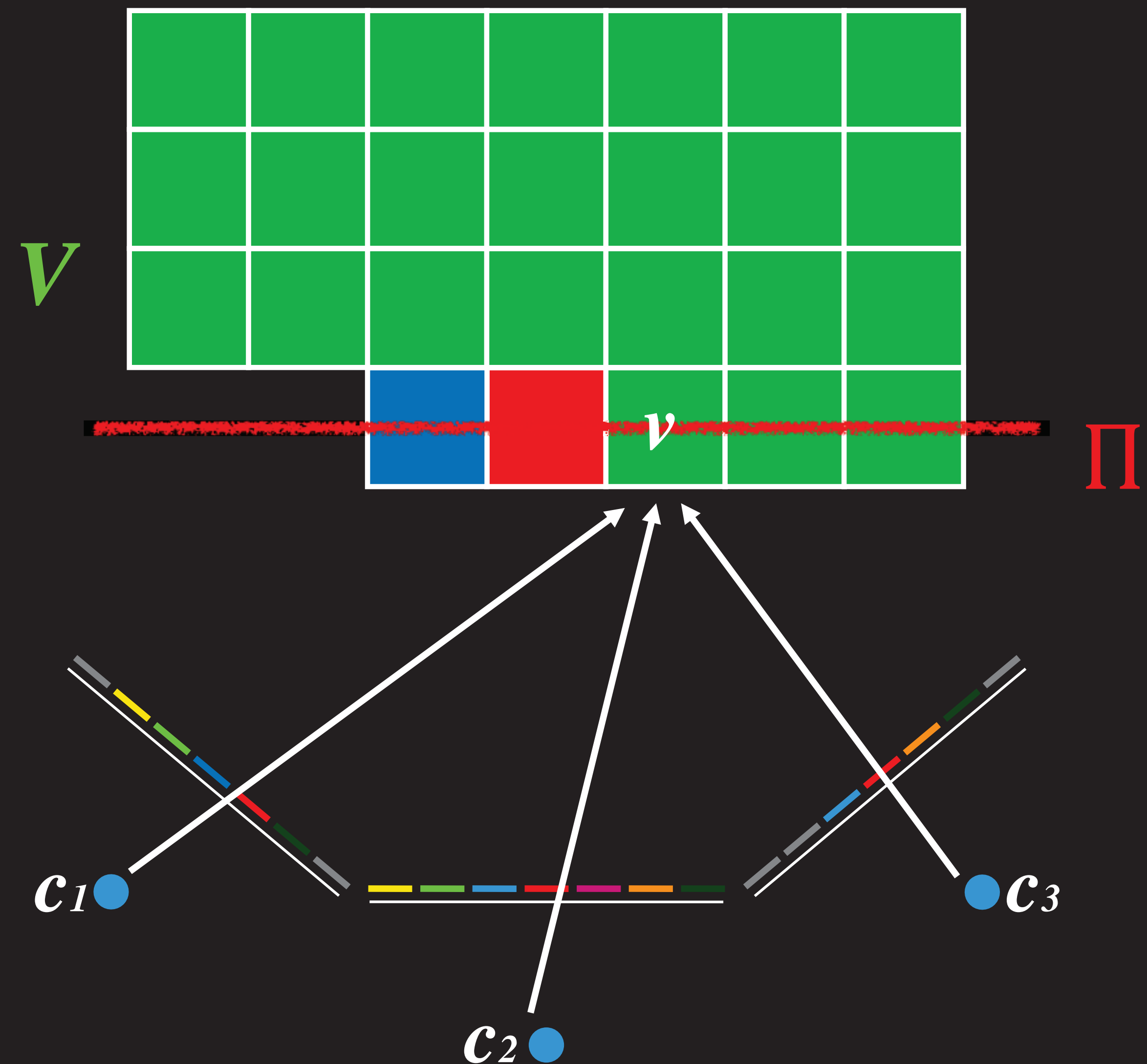
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .





# Space Carving: An algorithm

## Plane Sweep Algorithm

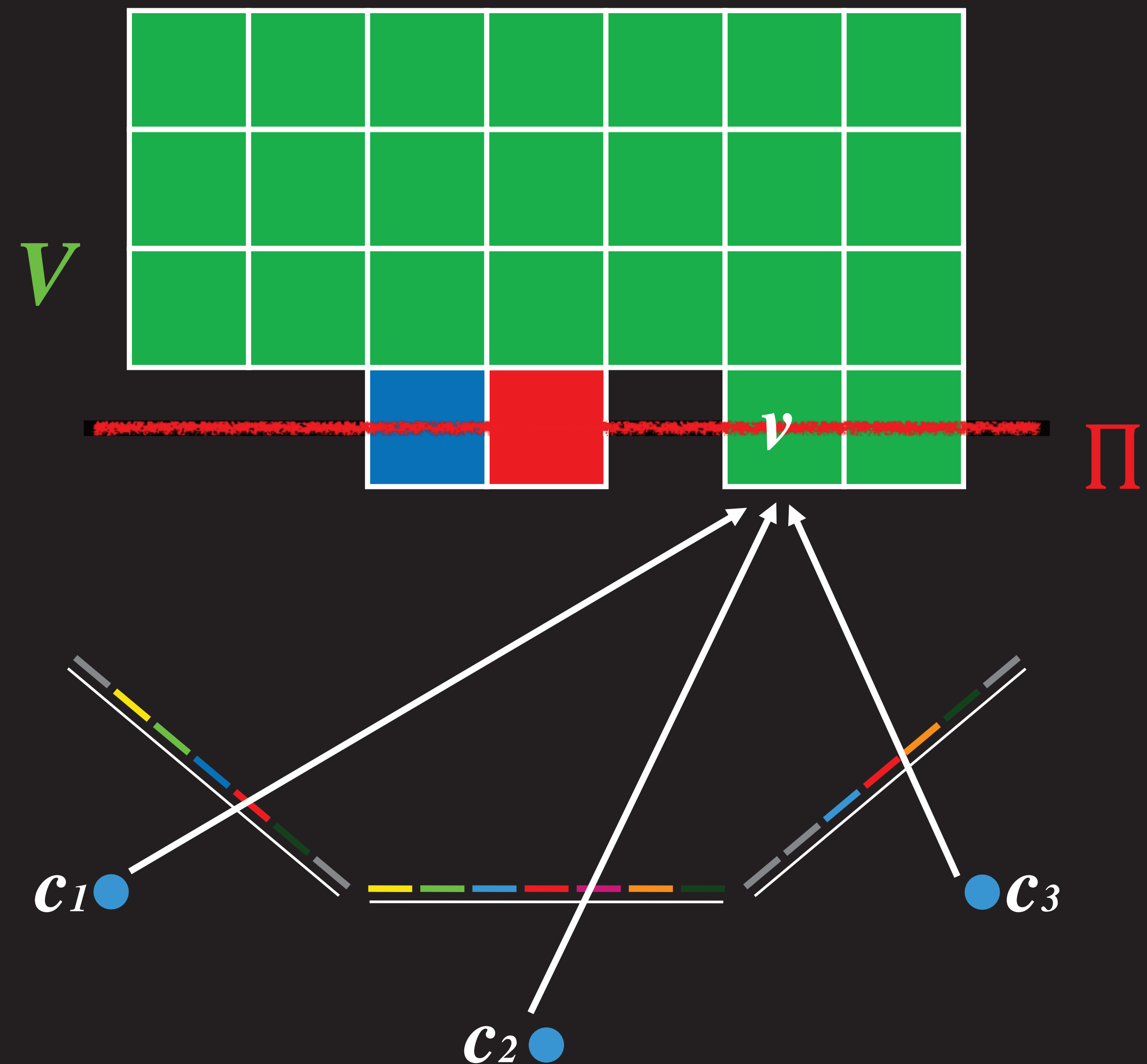
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

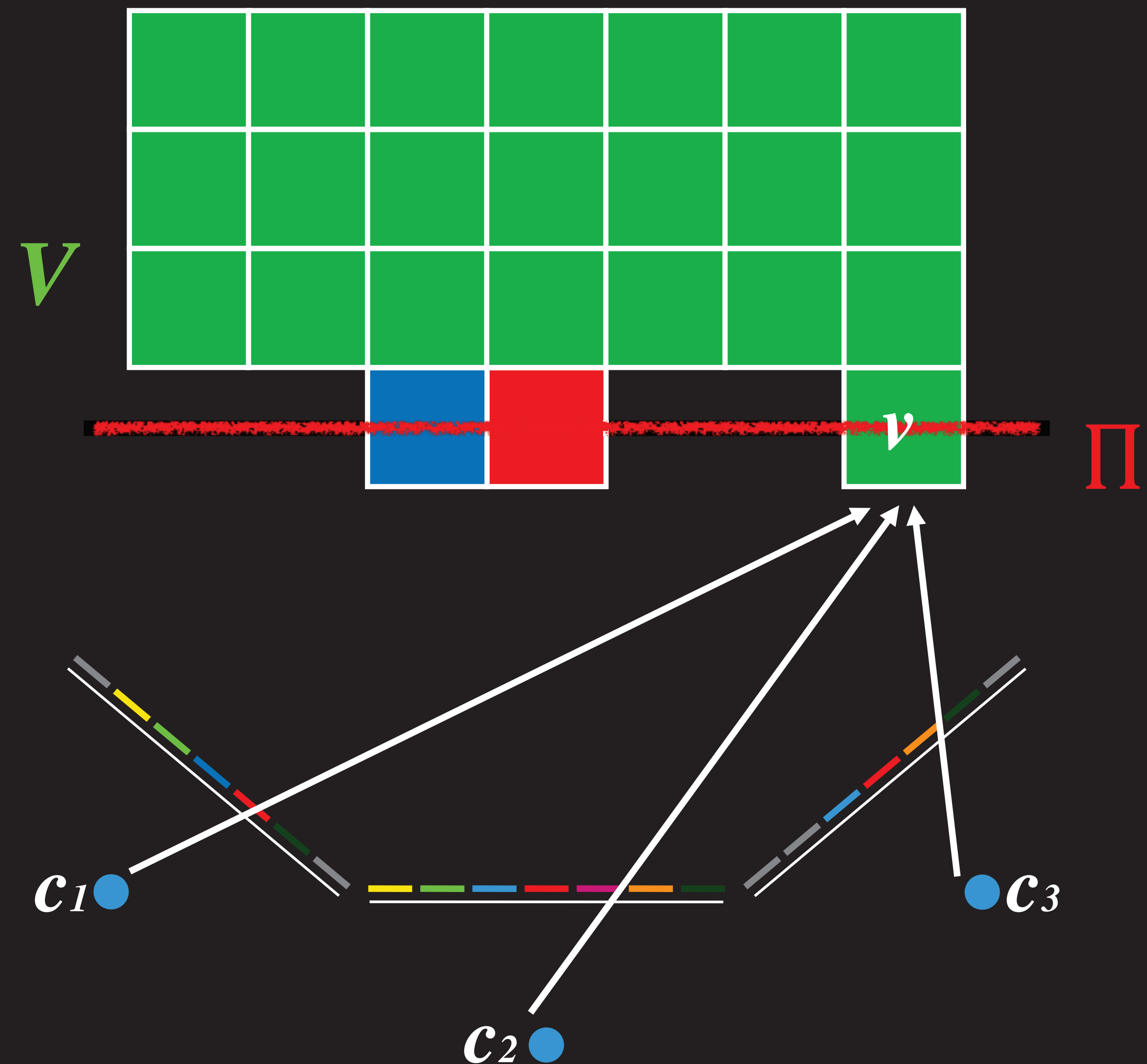
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

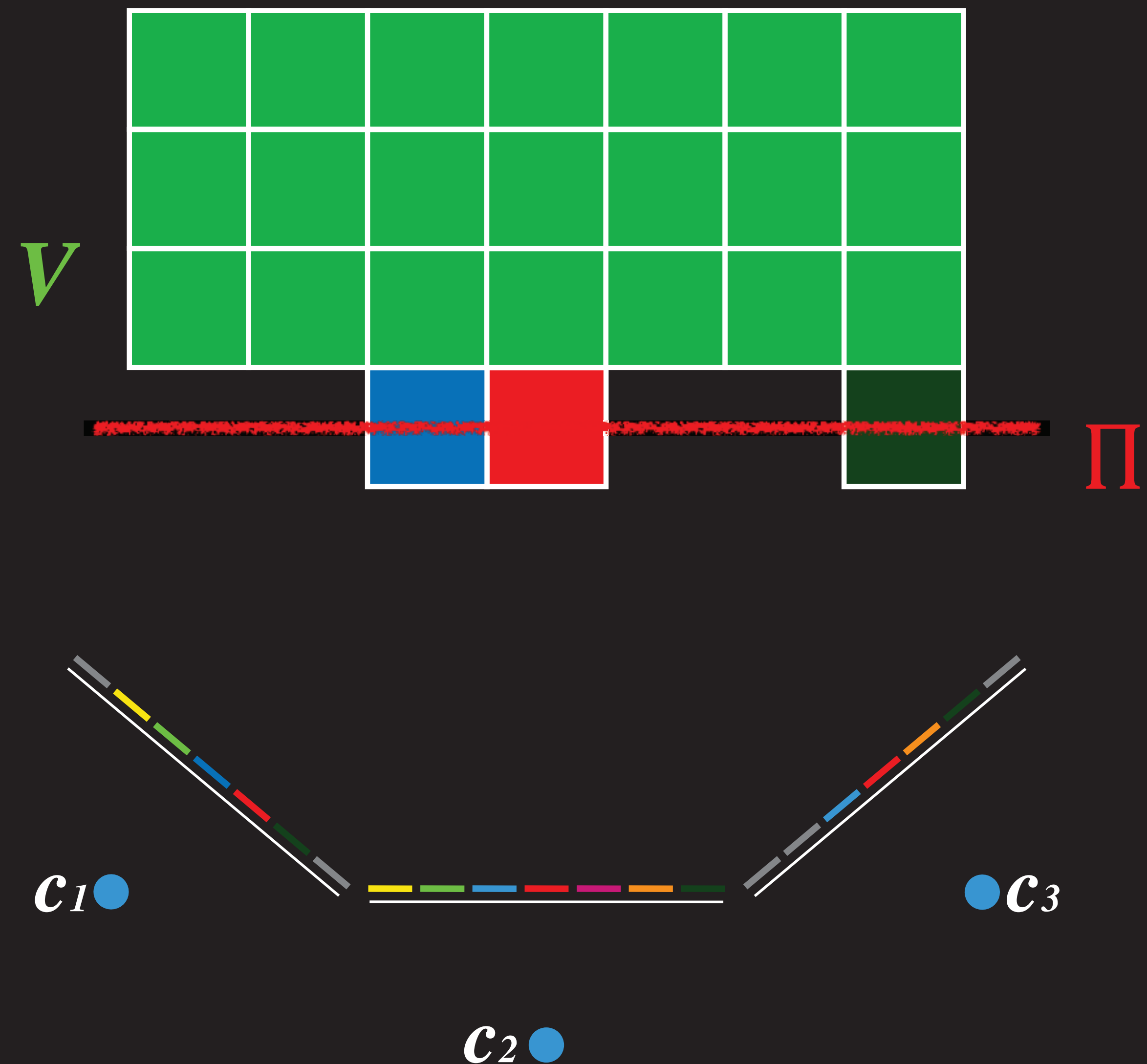
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

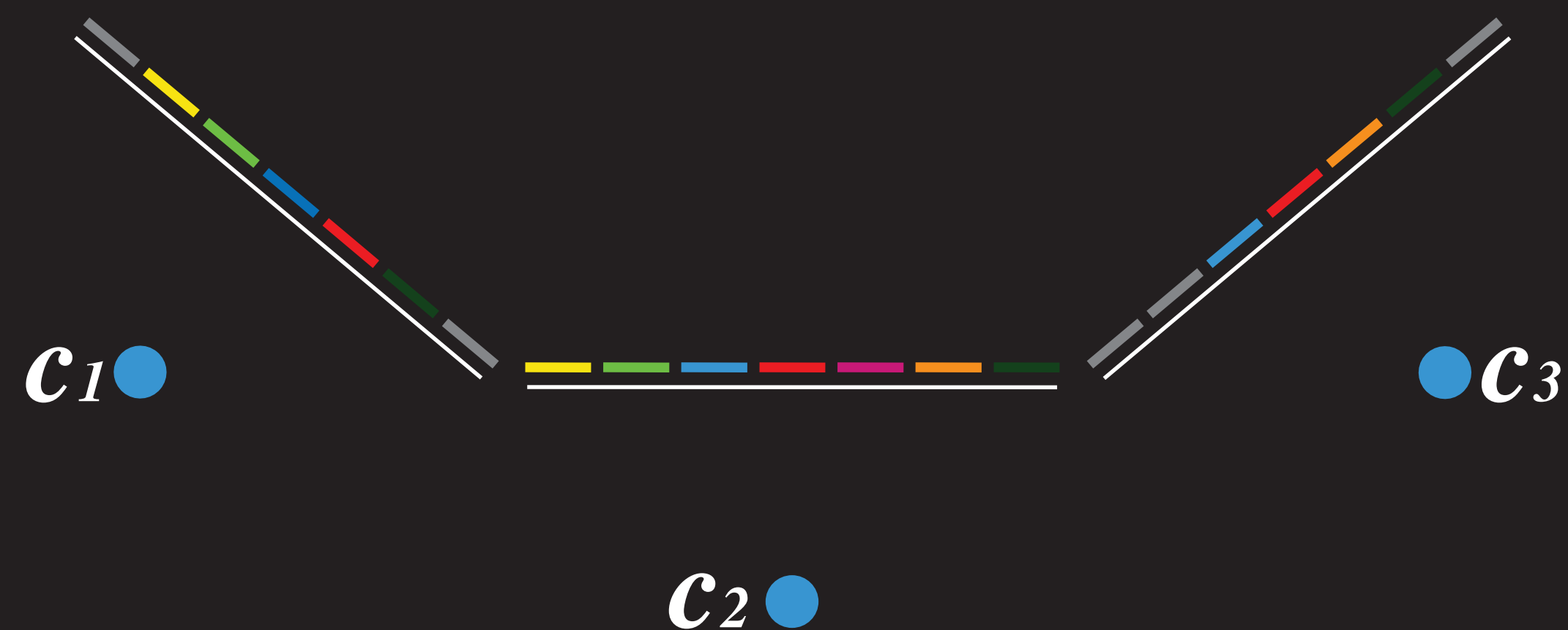
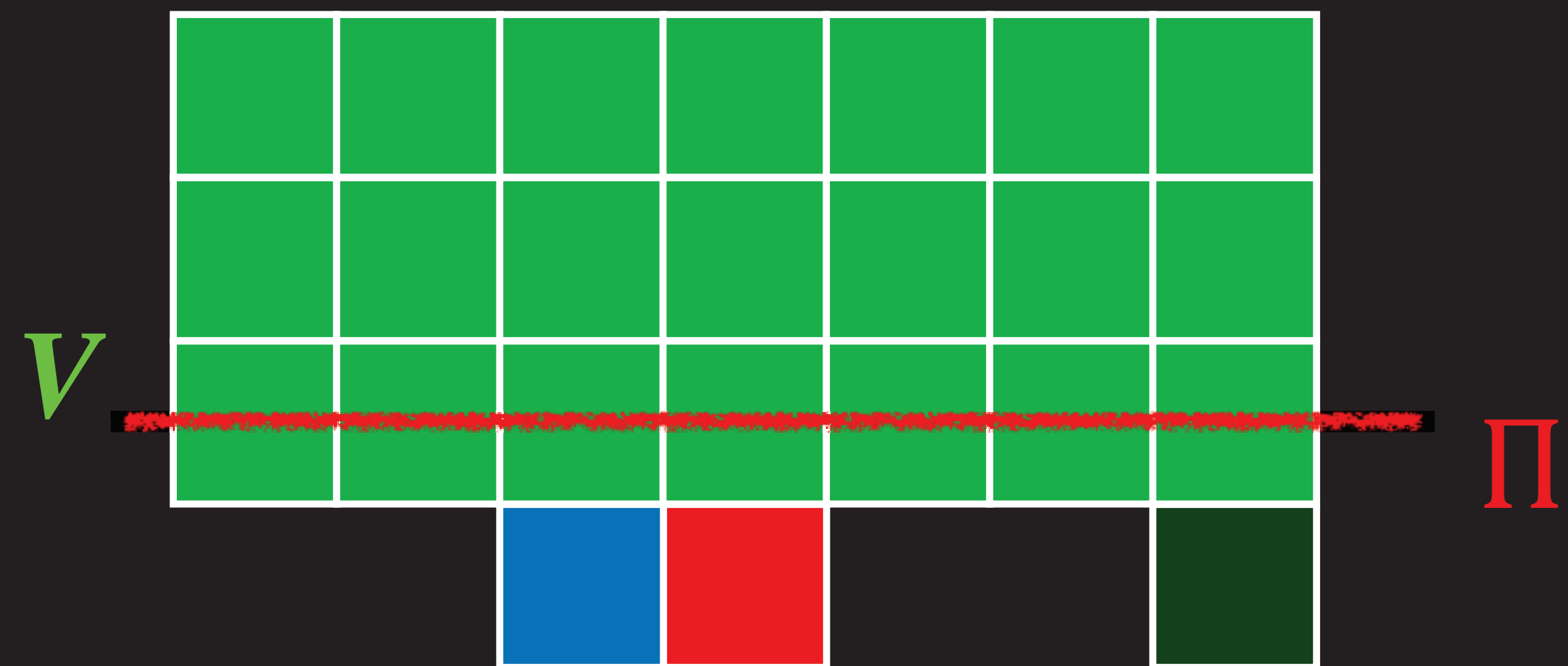
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

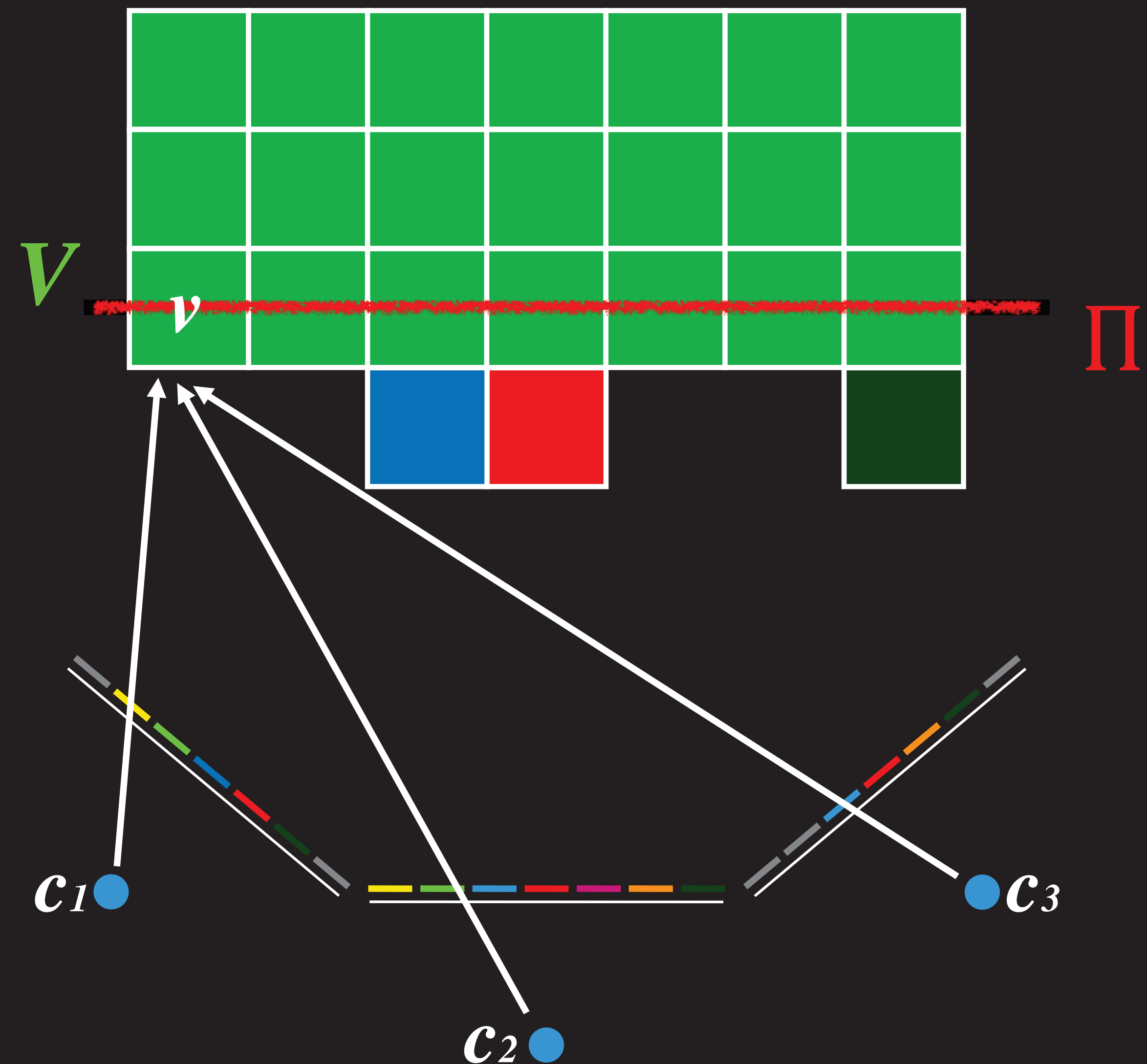
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

## Plane Sweep Algorithm

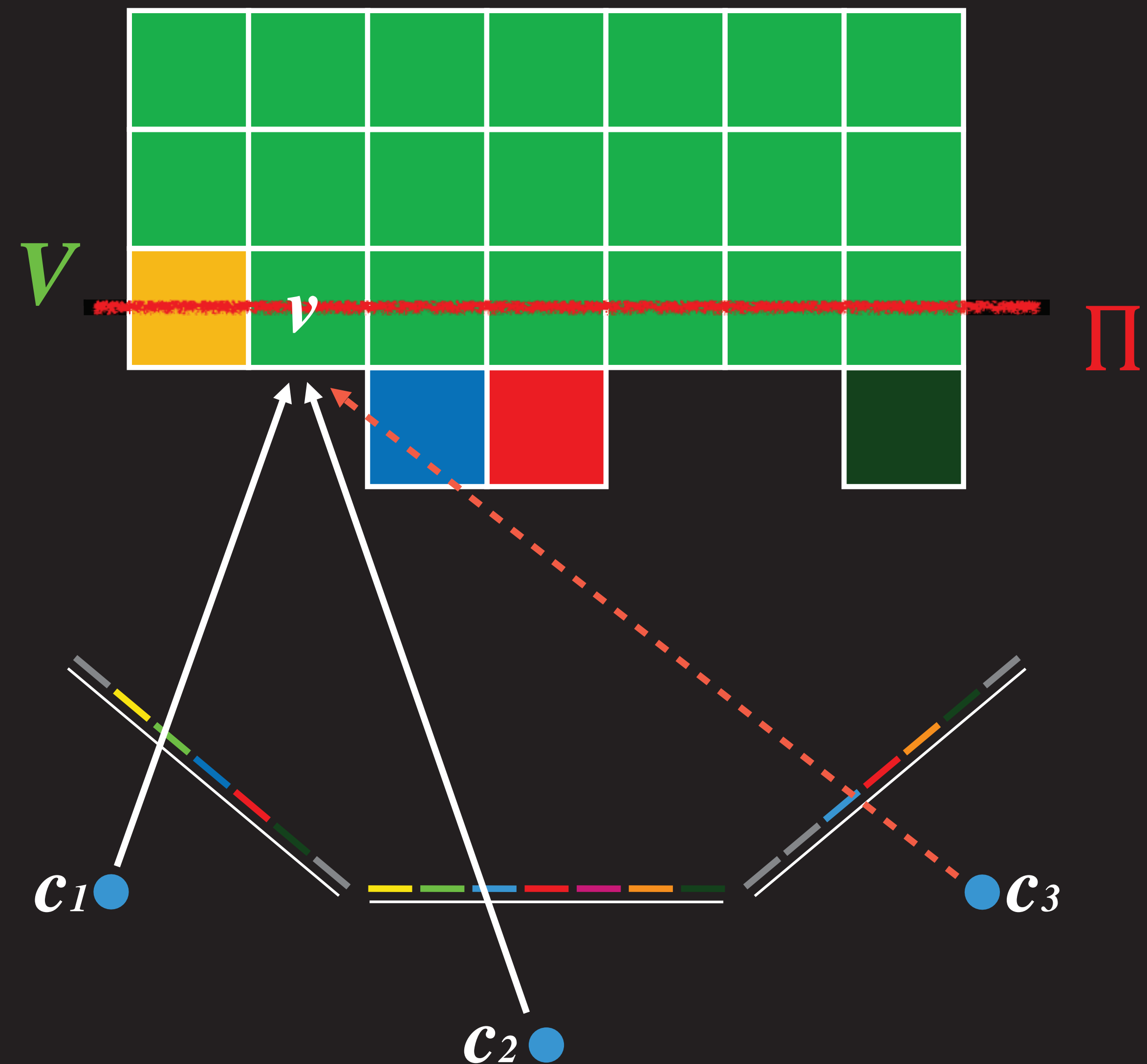
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



# Space Carving: An algorithm

Pla

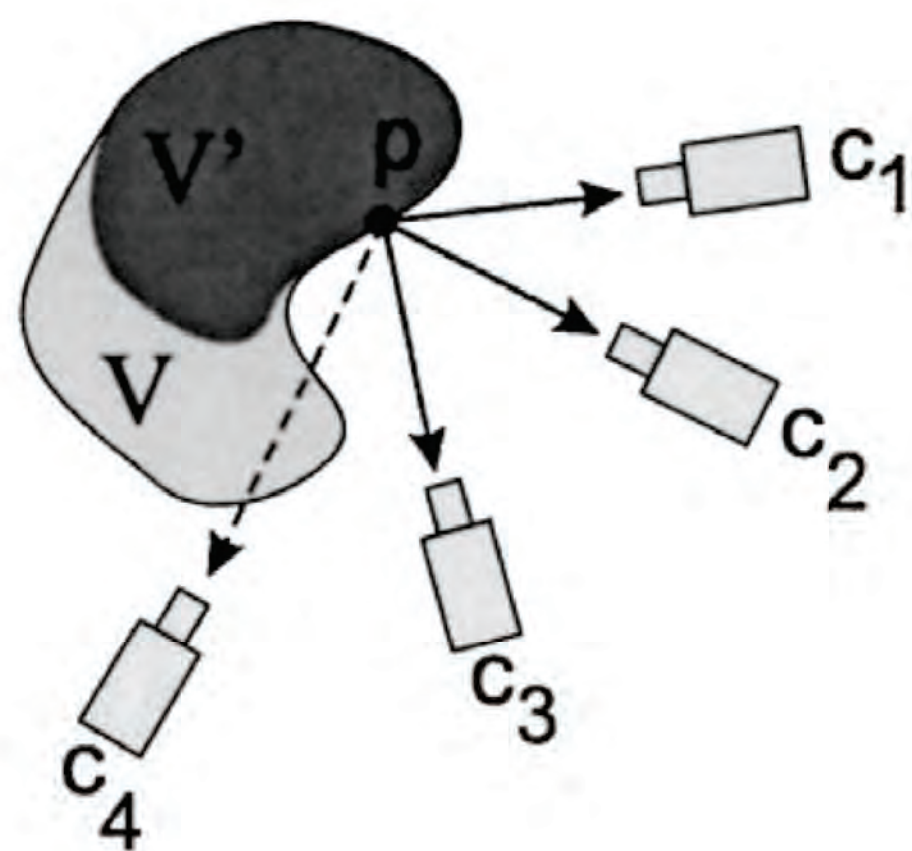
Ste

It

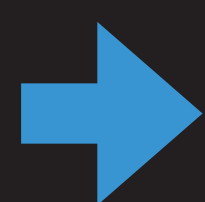
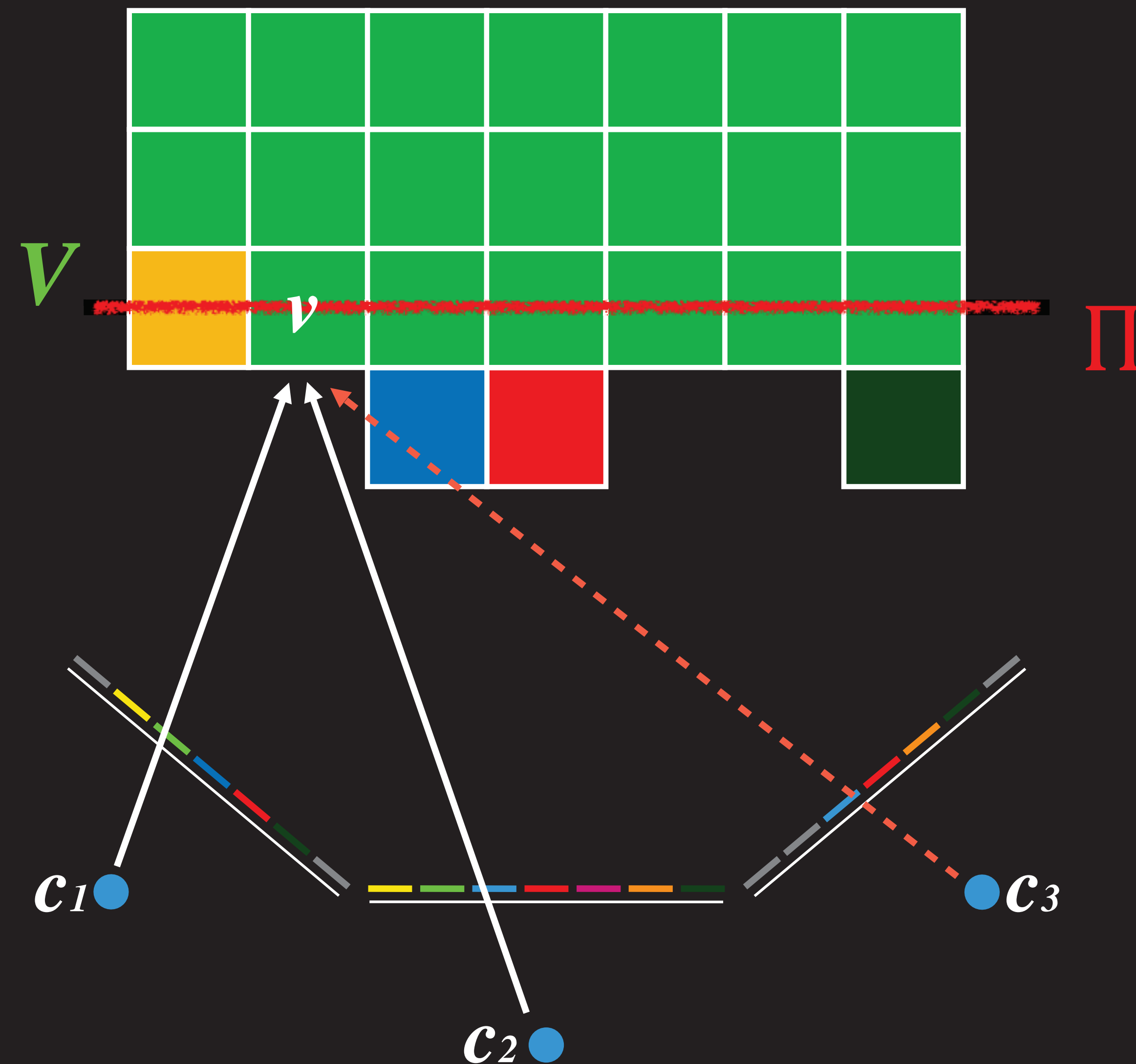
Ste

Ste

**Lemma 1 (Visibility Lemma).** Let  $p$  be a point on  $\mathcal{V}$ 's surface,  $\text{Surf}(\mathcal{V})$ , and let  $\text{Vis}_{\mathcal{V}}(p)$  be the collection of input photographs in which  $\mathcal{V}$  does not occlude  $p$ . If  $\mathcal{V}' \subset \mathcal{V}$  is a shape that also has  $p$  on its surface,  $\text{Vis}_{\mathcal{V}}(p) \subseteq \text{Vis}_{\mathcal{V}'}(p)$ .



*Figure 2.* Illustration of the Visibility and Non-Photo-Consistency Lemmas. If  $p$  is non-photo-consistent with the photographs at  $c_1, c_2, c_3$ , it is non-photo-consistent with the entire set  $\text{Vis}_{\mathcal{V}'}(p)$ , which also includes  $c_4$ .



Ste

S

# 3 mistakes in Ice age

- Point-wise photo consistency
- Global optimization dilemma
- Occlusion dilemma



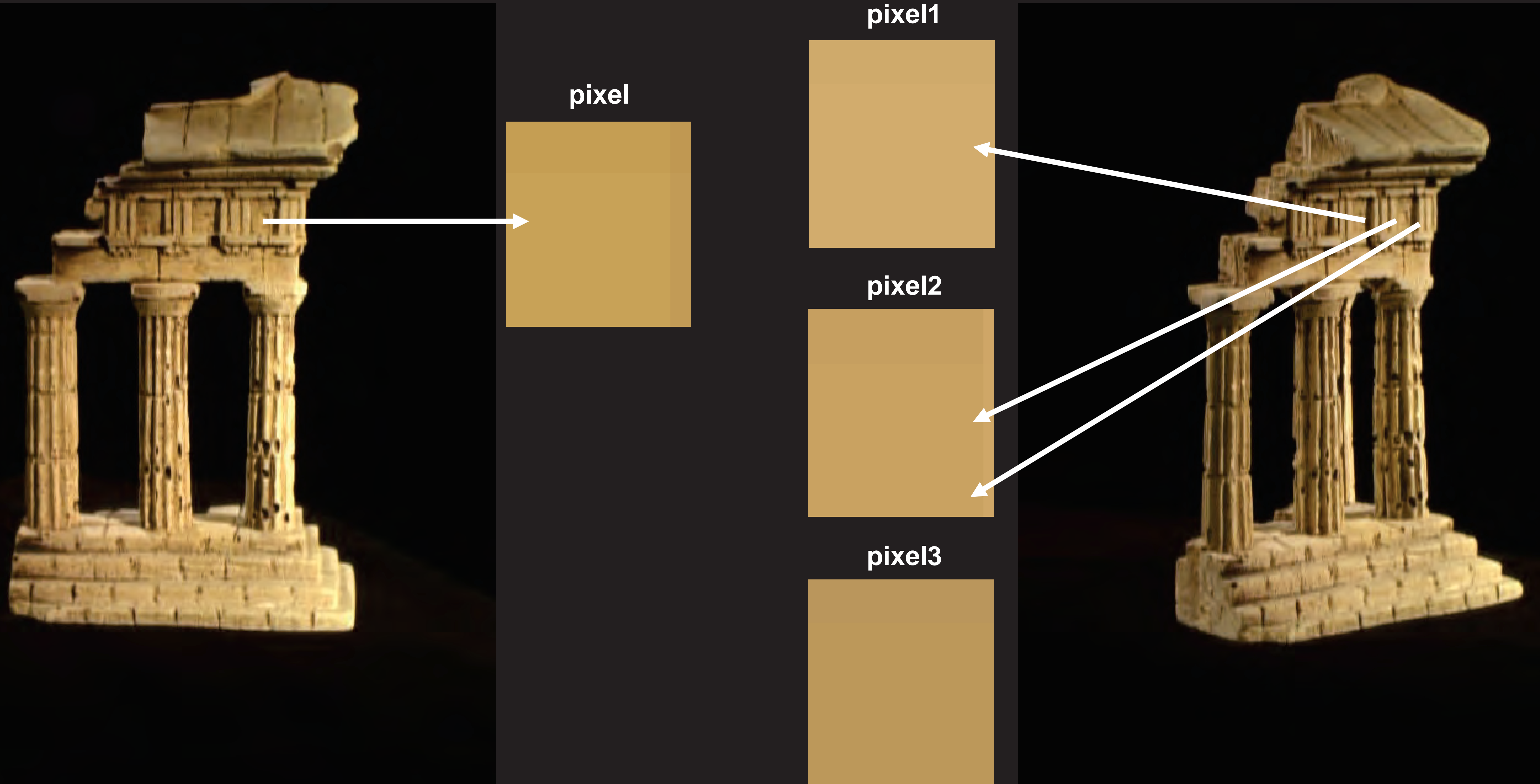


# Pixel vs. Patch



A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms  
[ Seitz, Curless, Diebel, Scharstein, and Szeliski, CVPR 2006 ]

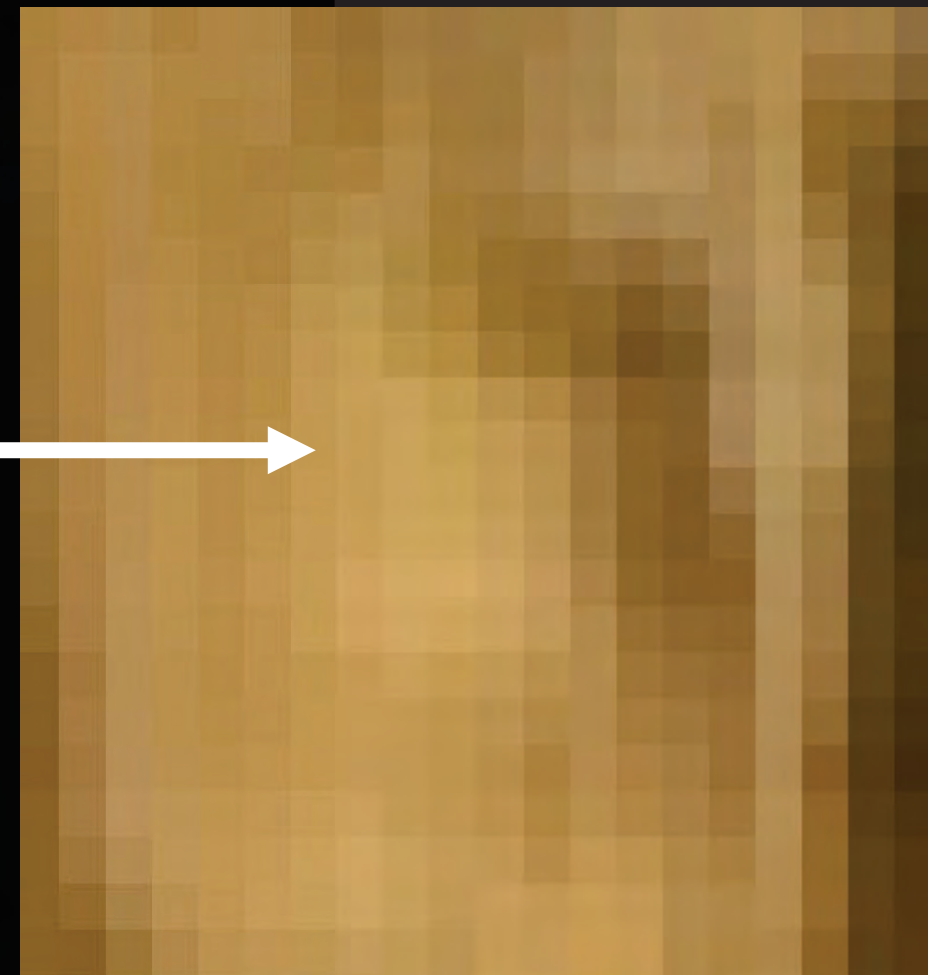
# Pixel vs. Patch



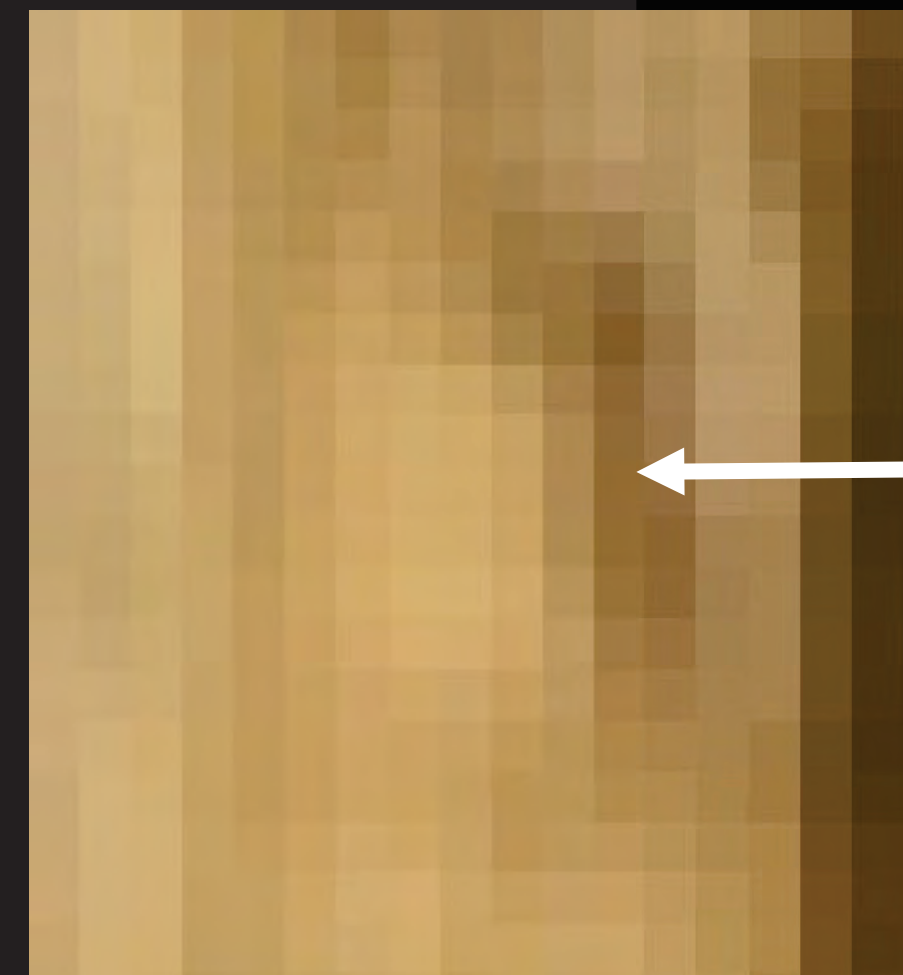
# Pixel vs. Patch



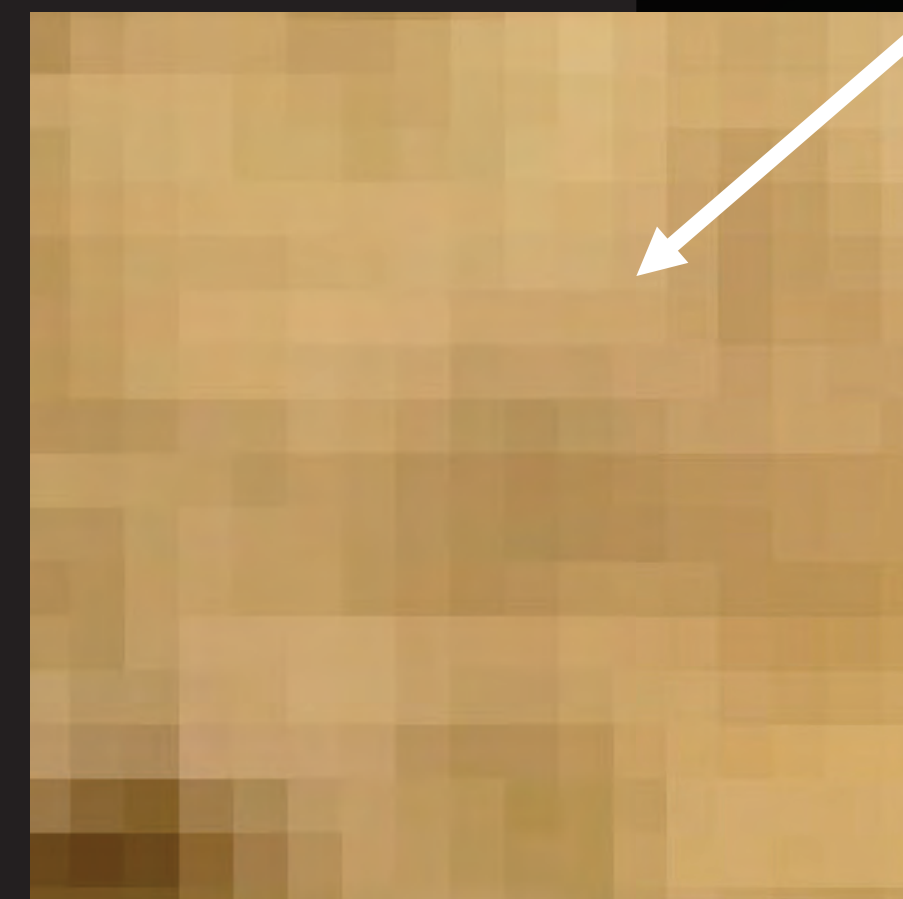
patch



patch1



patch2



# Space Carving: An algorithm

## Plane Sweep Algorithm

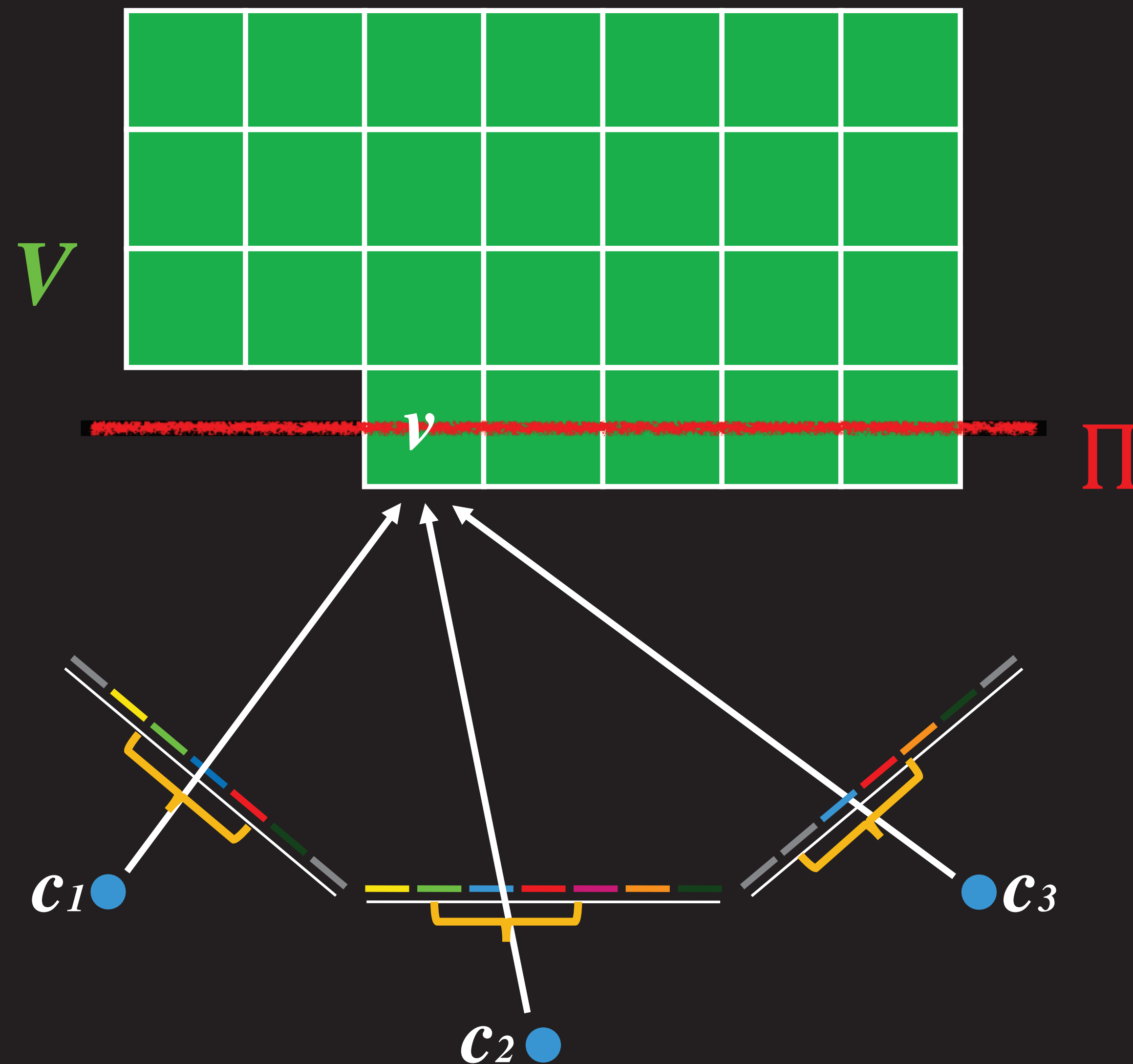
**Step 1:** Given an initial volume  $\mathcal{V}$ , initialize the sweep plane  $\Pi$  such that  $\mathcal{V}$  lies below  $\Pi$  (i.e.,  $\Pi$  is swept towards  $\mathcal{V}$ ).

**Step 2:** Intersect  $\Pi$  with the current shape  $\mathcal{V}$ .

**Step 3:** For each surface voxel  $v$  on  $\Pi$ :

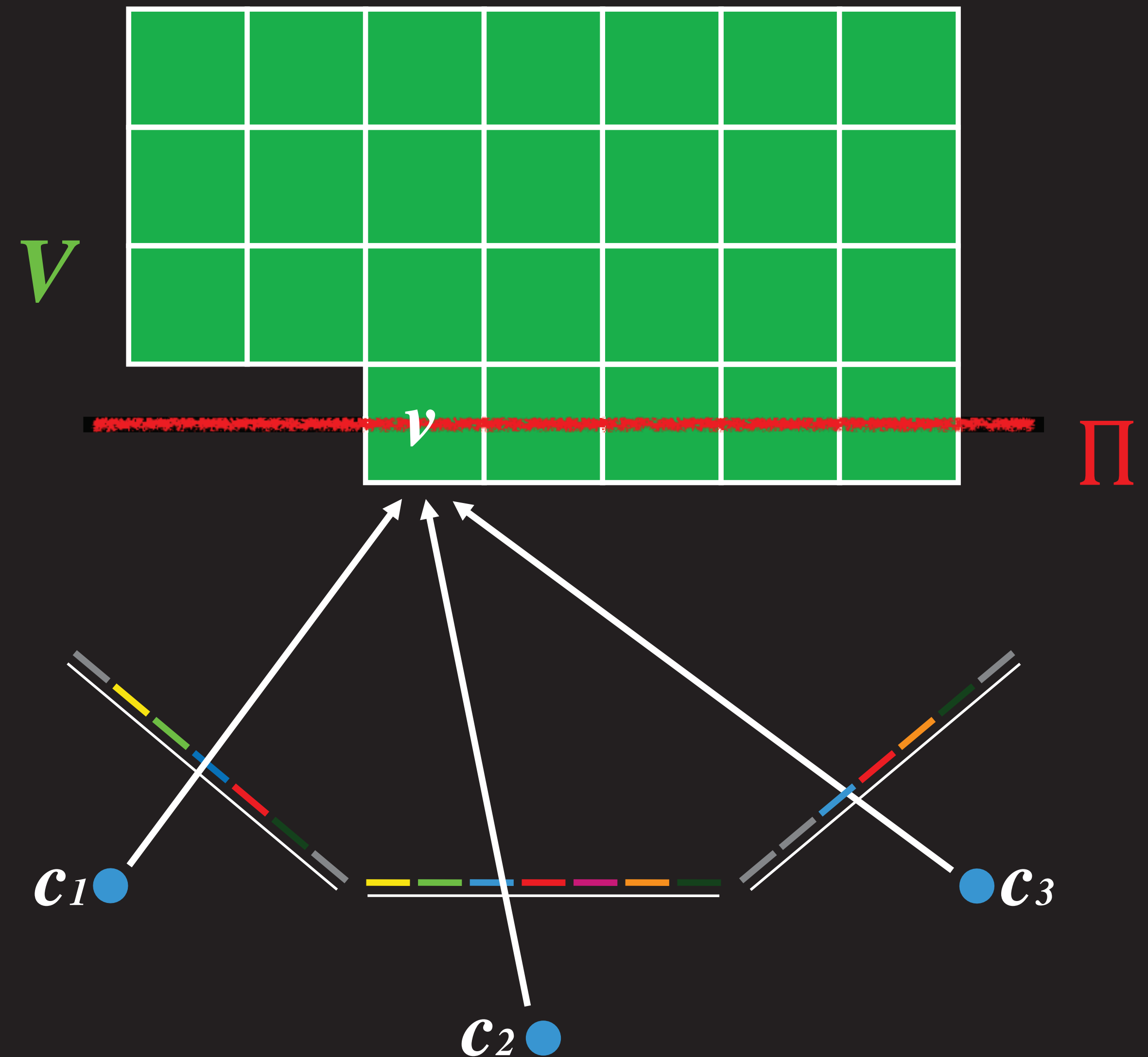
- let  $c_1, \dots, c_j$  be the cameras above  $\Pi$  for which  $v$  projects to an *unmarked* pixel;
- determine the photo-consistency of  $v$  using  $\text{consist}_K(\text{col}_1, \dots, \text{col}_j, \xi_1, \dots, \xi_j)$ ;
- if  $v$  is inconsistent then set  $\mathcal{V} = \mathcal{V} - \{v\}$ , otherwise mark the pixels to which  $v$  projects.

**Step 4:** Move  $\Pi$  downward one voxel width and repeat Step 2 until  $\mathcal{V}$  lies above  $\Pi$ .



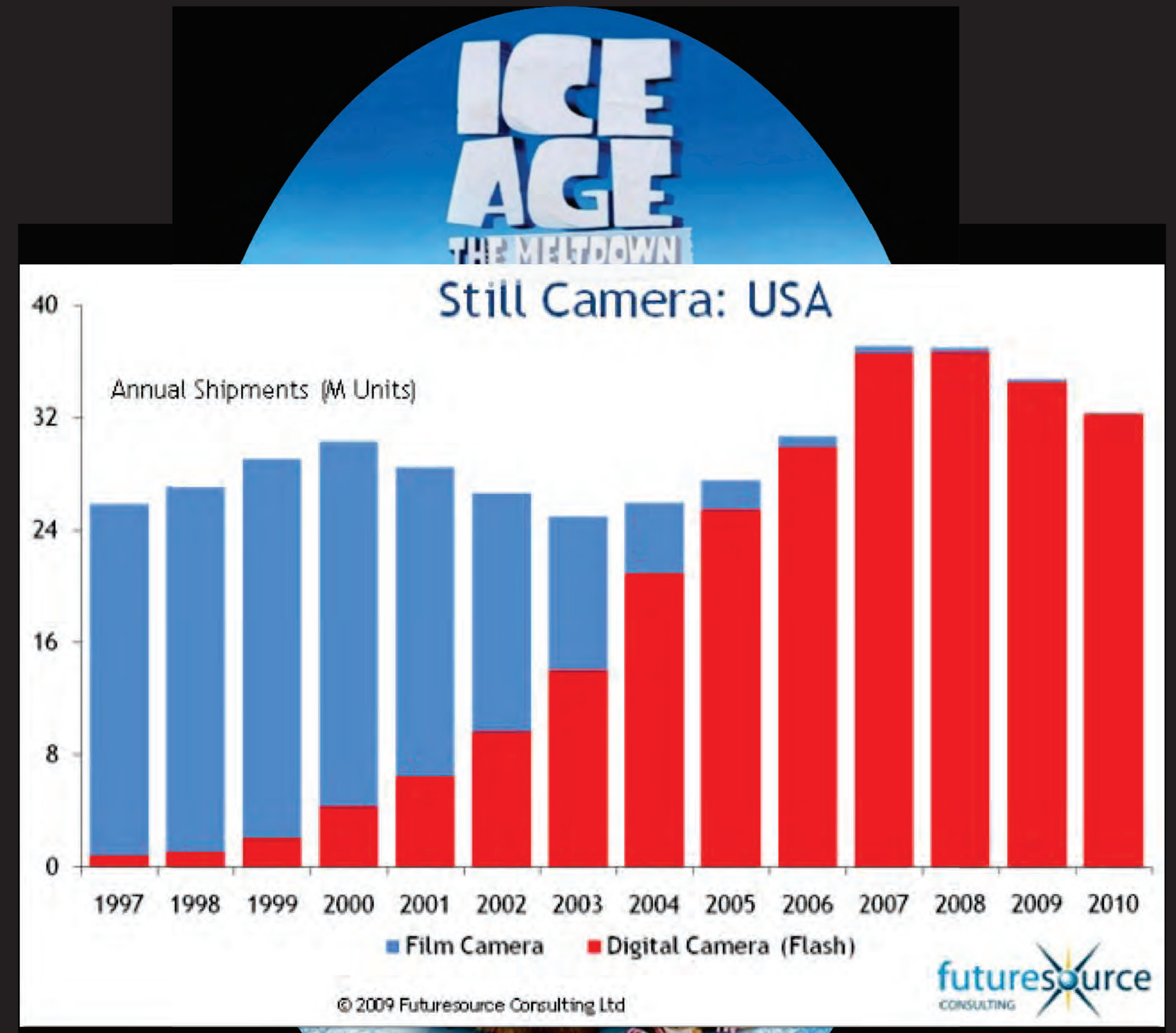
# Space Carving: An algorithm

- Point-wise photo consistency  
Use a patch instead of a pixel!
- Global optimization dilemma  
Local method works well!
- Occlusion dilemma  
Robust statistics overcome occlusions!



# 3 mistakes in Ice age

- Point-wise photo consistency  
**Use a patch instead of a pixel!**
- Global optimization dilemma  
**Local method works well!**
- Occlusion dilemma  
**Robust statistics overcome occlusions!**



# Ice age



ICE AGE  
THE MELTDOWN

2000



# Sensing revolution



2007



2009



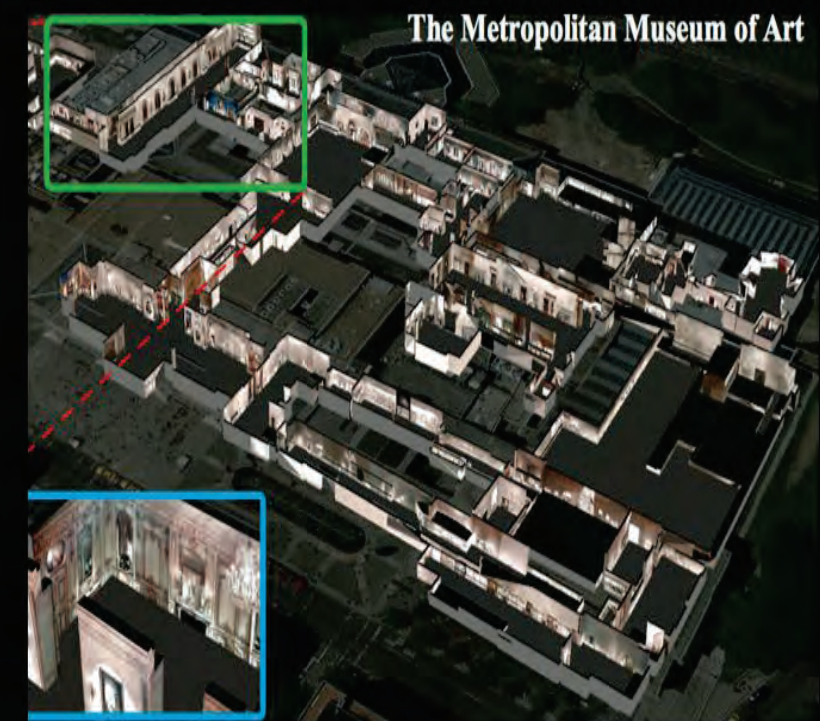
2010



2011



2012

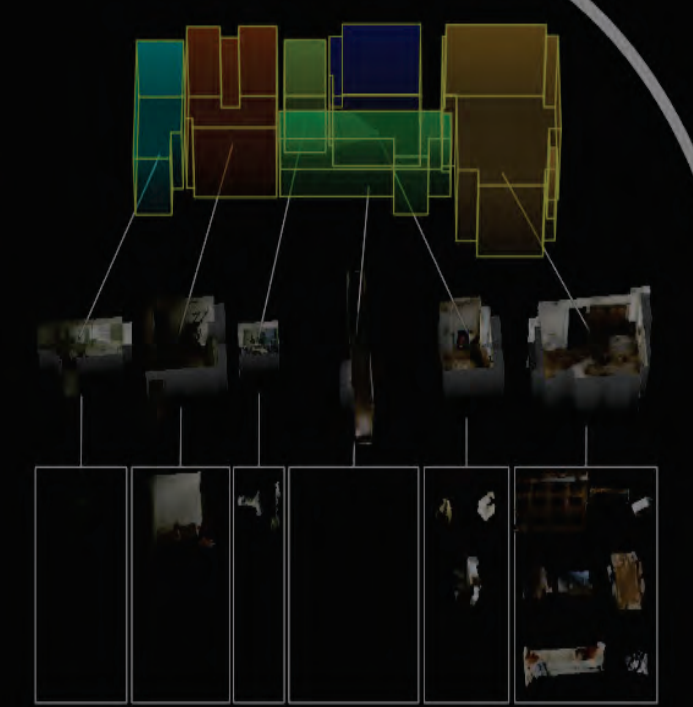


2012

# Perception - ice-age/revolution?



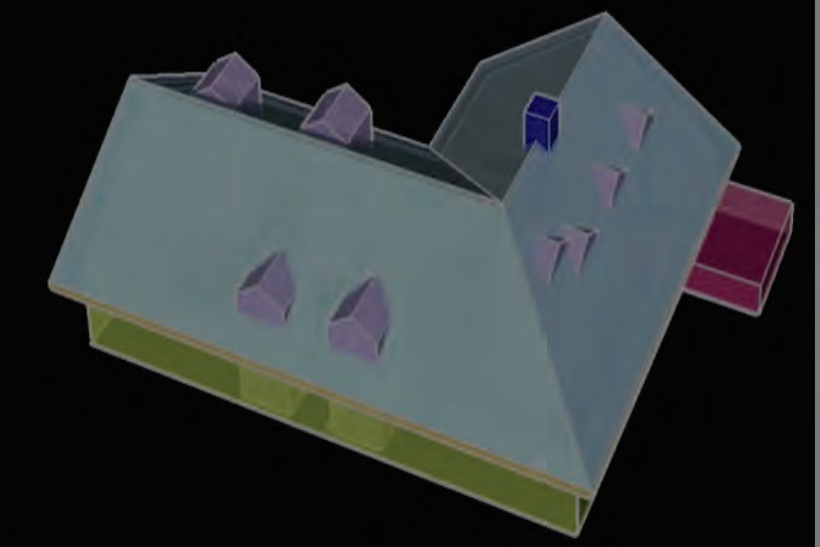
2014



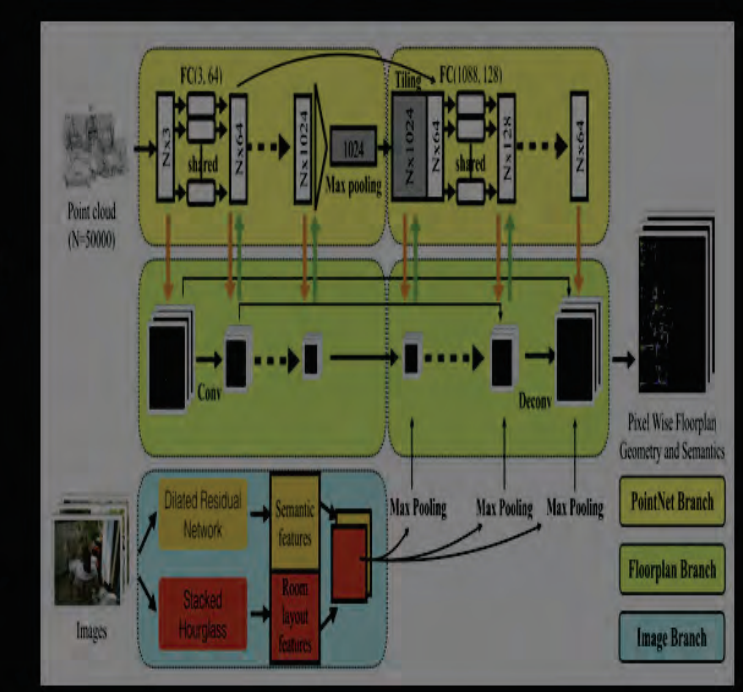
2015



2017



2018



2018



2019

# Skip details and refer to

Furukawa and Hernandez. Multi-View Stereo: A Tutorial.  
Foundations and Trends in Computer Graphics and Vision, 2015.





# First break-through in 2003

## Silhouette and Stereo Fusion for 3D Object Modeling

Carlos Hernández Esteban and Francis Schmitt  
Signal and Image Processing Department, CNRS URA 820  
Ecole Nationale Supérieure des Télécommunications, France  
{carlos.hernandez, francis.schmitt}@enst.fr

### Abstract

In this paper we present a new approach to high quality 3D object reconstruction. Starting from a calibrated sequence of color images, the algorithm is able to reconstruct both the 3D geometry and the texture. The core of the method is based on a deformable model, which defines the framework where texture and silhouette information can be fused. This is achieved by defining two external forces based on the images: a texture driven force and a silhouette driven force. The texture force is computed in two steps: a multi-stereo correlation voting approach and a gradient vector flow diffusion. Due to the high resolution of the voting approach, a multi-grid version of the gradient vector flow has been developed. Concerning the silhouette force, a new formulation of the silhouette constraint is derived. It provides a robust way to integrate the silhouettes in the evolution algorithm. As a consequence, we are able to recover the apparent contours of the model at the end of the iteration process. Finally, a texture map is computed from the original images for the reconstructed 3D model.

### 1. Introduction

As computer graphics and technology become more powerful, attention is being focused on the creation or acquisition of high quality 3D models. As a result, a great effort is being made to exploit the biggest source of 3D models: the real world. Among all the possible techniques of 3D acquisition, there is one which is specially attractive: the image-based modeling. In this kind of approach, the only input data to the algorithm are a set of images, possibly calibrated. Its main advantages are the low cost of the system and the possibility of immediate color. The main disadvantage is the quality of the reconstructions compared to the quality of more active techniques (range scanning or encoded-light techniques). We present in this paper an image-based modeling approach that offers the possibility of high quality reconstructions by mixing two orthogonal image data into a same framework: silhouette information

and texture information. Our two main contributions are a new approach to the silhouette constraint definition and the high quality of the overall system (see Fig.1).

### 2. Related Work

Acquiring 3D models is not an easy task and abundant literature exists on this subject. There are three main approaches to the problem of 3D acquisition: pure image-based rendering techniques, hybrid image-based techniques, and 3D scanning techniques. Pure image-based rendering techniques as [2, 20] try to generate synthetic views from a given set of original images. They do not estimate the real 3D structure behind the images, they only interpolate the given set of images to generate a synthetic view. Hybrid methods as [5, 19] make a rough estimation of the 3D geometry and mix it with a traditional image-based rendering algorithm in order to obtain more accurate results. In both types of methods, the goal is to generate coherent views of the real scene, not to obtain metric measures of it. In opposition to these techniques, the third class of algorithms try to recover the full 3D structure. Among the 3D scanning techniques, we can distinguish two main groups: active methods and passive methods. Active methods use a controlled source of light such as a laser or a coded light in order to recover the 3D information [25, 4, 14]. Passive methods use only the information contained in the images of the scene and are commonly known as *shape from X methods*. They can be classified according to the type of information they use. A first class consists of the shape from silhouette methods [1, 23, 28, 22, 18]. They obtain an initial estimation of the 3D model known as visual hull. They are robust and fast, but because of the type of information used, they are limited to simple shaped objects. We can find commercial products based on this technique. A second class corresponds to the shape from shading methods. They are based on the diffusing properties of Lambertian surfaces. They mainly work for 2.5D surfaces and are very dependent on the light conditions. A third class of methods uses color consistency to carve a voxel volume [26, 17]. But they only provide an output model composed



Figure 5. Hygia model after convergence (159534 vertices). Left: Gouraud shading. Right: texture mapping.



Figure 6. Example of a reconstruction with bad image correlations. Top left: one of the original images. Top right: rendering of the correlation voting volume. Bottom left: snake mesh after convergence. Bottom right: textured mesh.

- [2] S.E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH '93*, pages 279–288, 1993.
- [3] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In A. Leonardis, F. Solina, and R. Bajcsy, editors, *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics*, Ljubljana, Slovenia, pages 25–47, 2000.
- [4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96*, pages 303–312, 1996.
- [5] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach. In *SIGGRAPH '96*, pages 11–20, 1996.
- [6] C. Hernández Esteban and F. Schmitt. Multi-stereo 3d object reconstruction. In *3DPT'02*, pages 159–166, 2002.
- [7] P. Fua. From multiple stereo views to multiple 3d surfaces. *Int. J. of Computer Vision*, 24:19–35, 1997.
- [8] P. Fua and Y.G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *Int. J. of Computer Vision*, 16:35–56, 1995.
- [9] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. J. of Computer Vision*, 1:321–332, 1988.
- [10] R. Keriven and O. Faugeras. Variational principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998.
- [11] L. Kobbelt.  $\sqrt{3}$ -subdivision. In *SIGGRAPH 2000*, pages 103–112, 2000.
- [12] A. Laurentini. The visual hull concept for silhouette based image understanding. *IEEE Trans. on PAMI*, 16(2):150–162, 1994.
- [13] H. Lensch, W. Heidrich, and H. P. Seidel. A silhouette-based algorithm for texture registration and stitching. *Journal of Graphical Models*, pages 245–262, 2001.
- [14] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginszton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH 2000*, pages 131–144, 2000.
- [15] M. Li, H. Schirmacher, M. Magnor, and H.P. Seidel. Combining stereo and visual hull information for on-line reconstruction and rendering of dynamic scenes. In *IEEE Workshop on MMS'02*, pages 9–12, 2002.
- [16] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, volume 21, pages 163–169, 1987.
- [17] Y. Matsumoto, K. Fujimura, and T. Kitamura. Shape-from-silhouette/stereo and its application to 3-d digitizer. In *Proceedings of Discrete Geometry for Computing Imagery*, pages 177–190, 1999.
- [18] Y. Matsumoto, H. Terasaki, K. Sugimoto, and T. Arakawa. A portable three-dimensional digitizer. In *Int. Conf. on Recent Advances in 3D Imaging and Modeling*, pages 197–205, 1997.
- [19] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. *SIGGRAPH 2000*, pages 369–374, 2000.
- [20] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH '95*, pages 39–46, 1995.
- [21] G. Medioni, M.S. Lee, and C.-K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.
- [22] W. Niem and J. Wingbermühle. Automatic reconstruction of 3d objects using a mobile monoscopic camera. In *Int. Conf. on Recent Advances in 3D Imaging and Modeling*, pages 173–181, 1997.
- [23] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *CVGIP*, 40:1–29, 1987.
- [24] A. Sarti and S. Tubaro. Image based multiresolution implicit object modeling. *EURASIP Journal on Applied Signal Processing*, 2002(10):1053–1066, 2002.
- [25] F. Schmitt, B. Barsky, and W. Du. An adaptive subdivision method for surface-fitting from sampled data. In *SIGGRAPH '86*, pages 179–188, 1986.
- [26] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. J. of Computer Vision*, 38(3):197–216, 2000.
- [27] J.A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*. Cambridge University Press, 1996.
- [28] R. Vaillant and O. Faugeras. Using extremal boundaries for 3d object modeling. *IEEE Trans. on PAMI*, 14(2):157–173, 1992.
- [29] C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, pages 359–369, 1998.

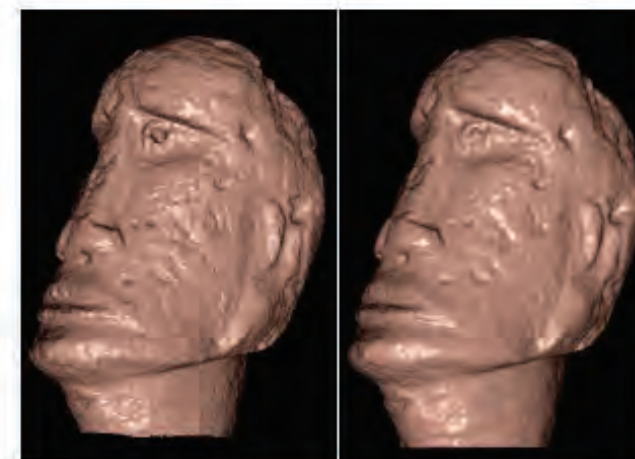


Figure 7. Comparison between the proposed passive method and a laser active method. Left: laser model of 385355 vertices obtained with a Minolta VIVID 910 3D scanner. Right: proposed method after snake convergence (233262 vertices).



Fig. 16. Reconstructions using our proposed approach. Left: one original image used in the reconstruction. Middle: Gouraud shading reconstructed models (45843, 83628 and 114496 vertices respectively). Right: textured models.

# Sensing revolution



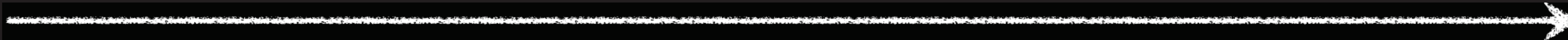
2000

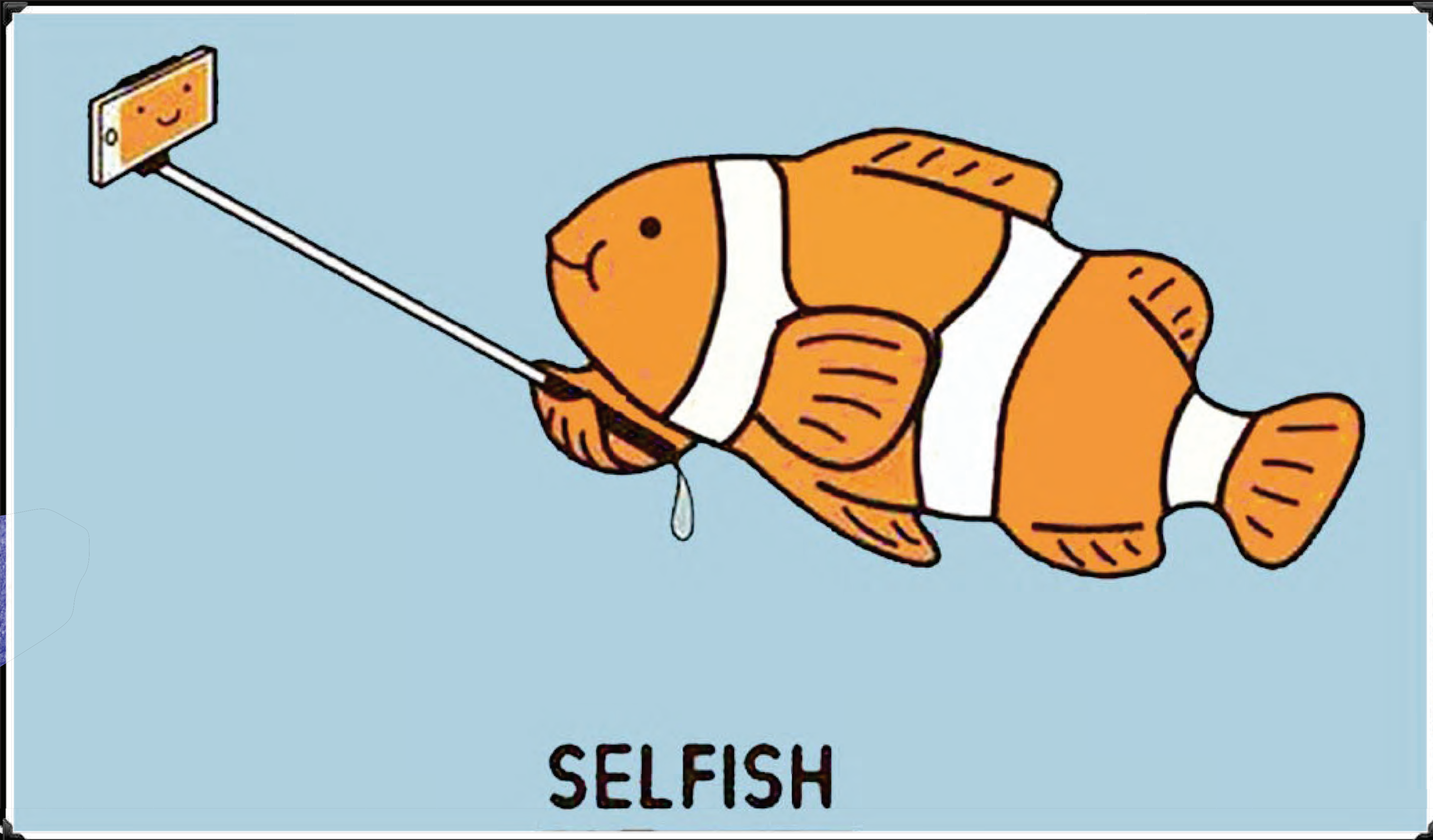
2005

2010

2011

2012





**SELFISH**



2000

2005

2010

2011

2012



# Sensing revolution

Accurate, Dense, and Robust Multi-View Stereopsis  
Yasutaka Furukawa and Jean Ponce  
Computer Vision and Pattern Recognition 2007



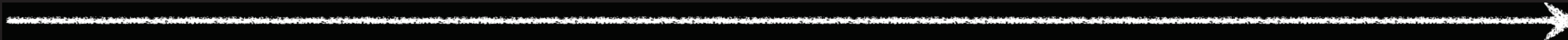
2000

2005

2010

2011

2012



# Sensing revolution

Intel Developer's Conference, 2011

Towards Internet-scale Multi-view Stereo  
Yasutaka Furukawa, Rick Szeliski, Brian Curless, and Steve Seitz  
Computer Vision and Pattern Recognition 2010



2000

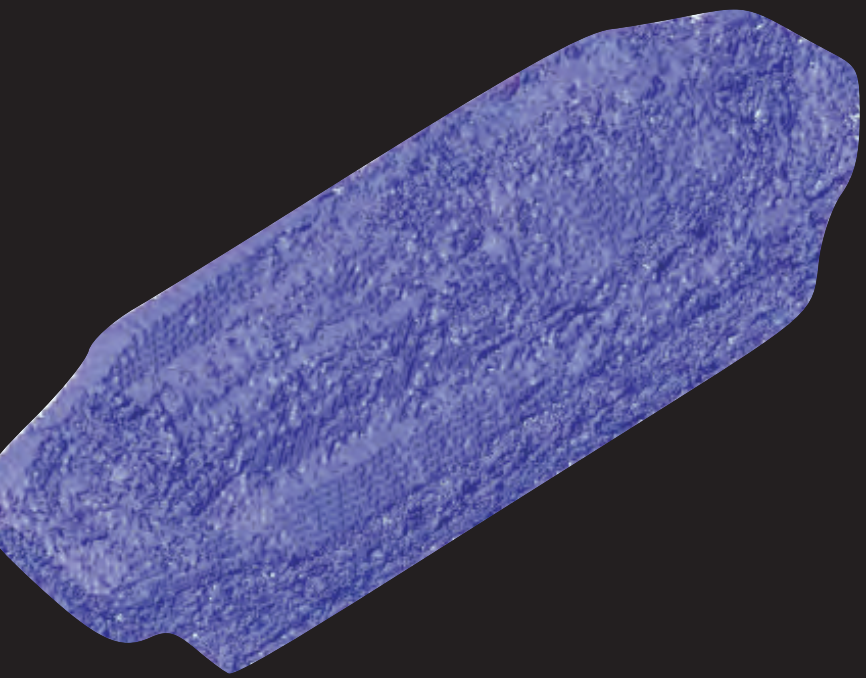
2005

2010

2011

2012

# Sensing revolution



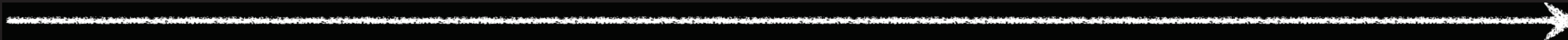
2000

2005

2010

2011

2012



# Sensing revolution



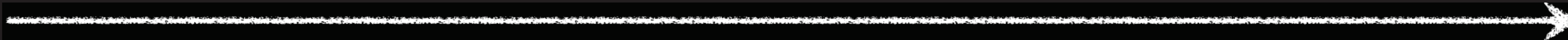
2000

2005

2010

2011

2012



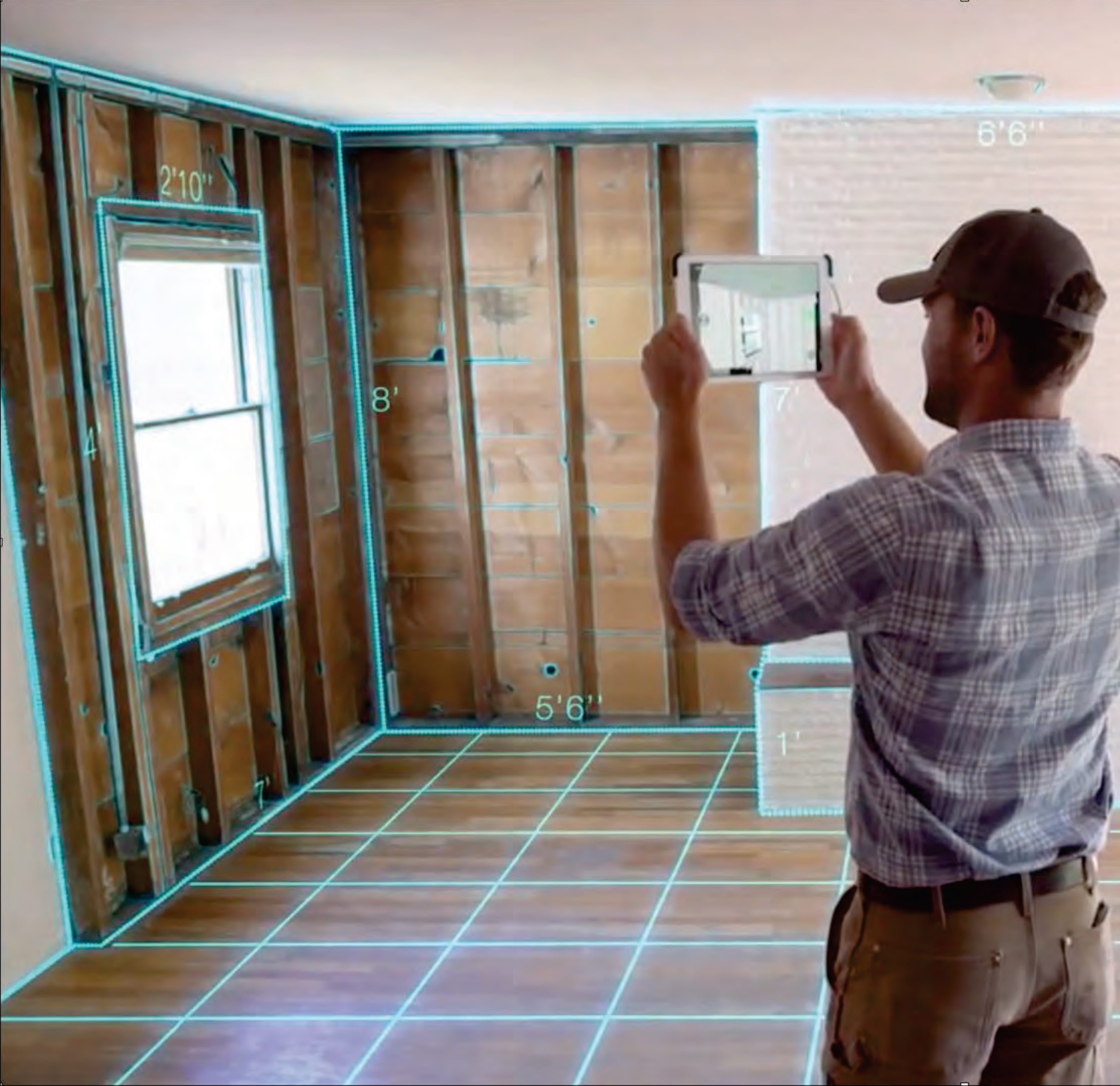


**Photo Tours**  
Avanish Kushal, Yasutaka Furukawa, Rick Szeliski, Brian Curless, and Steve Seitz  
International Conference on 3D Vision 2011







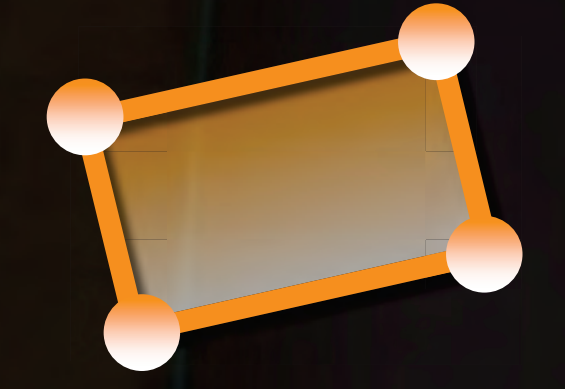


# Geometric Elements

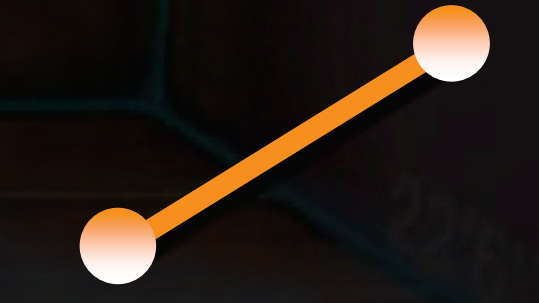
3D primitive



2D primitive



1D primitive



0D primitive

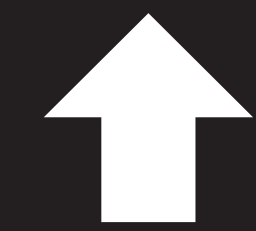
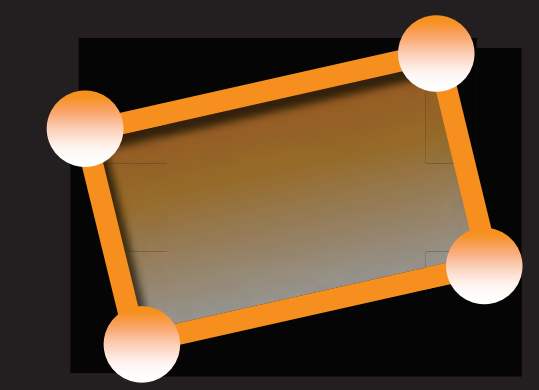




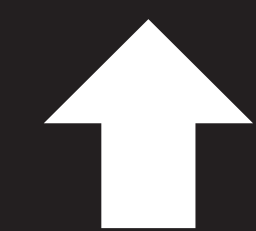
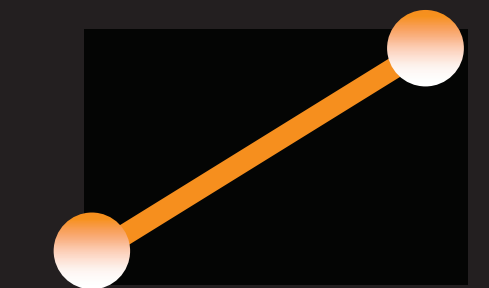
# Bottom-up



2D primitive

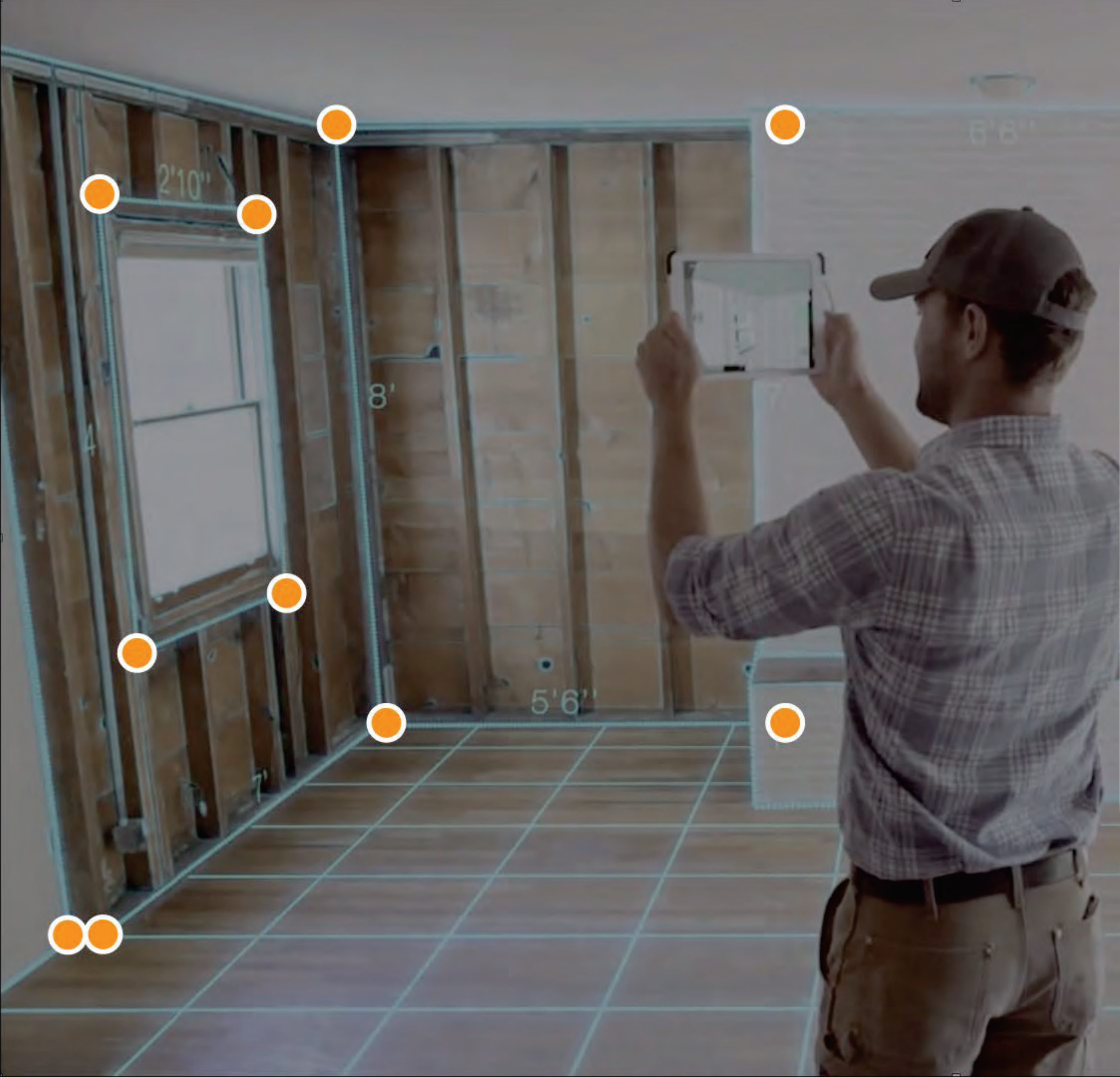


1D primitive



0D primitive

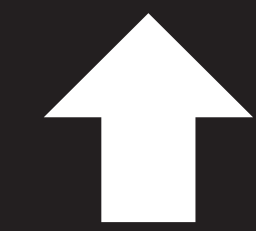
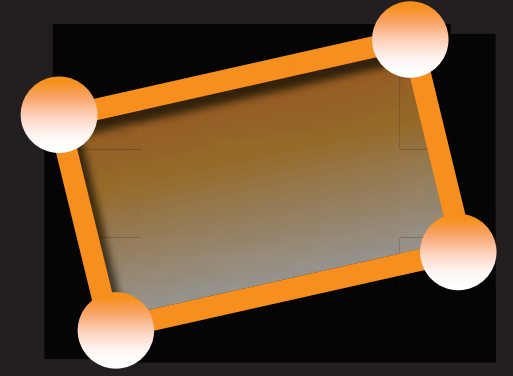




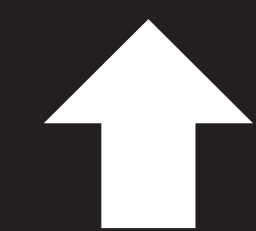
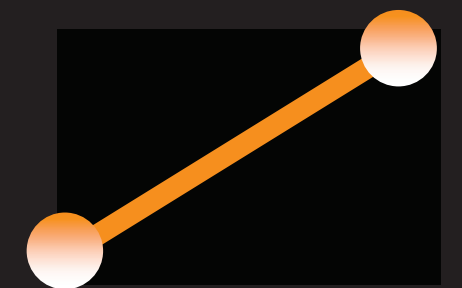
# Bottom-up



2D primitive



1D primitive



0D primitive

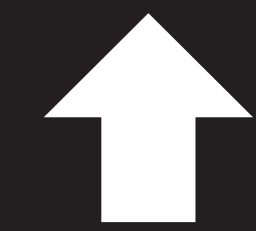
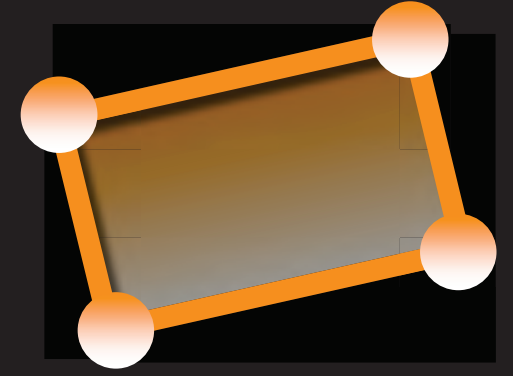




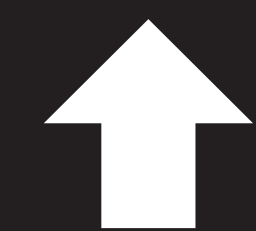
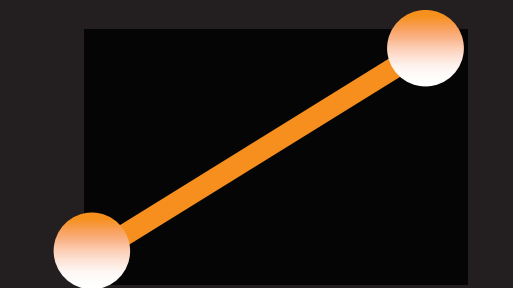
# Bottom-up



2D primitive

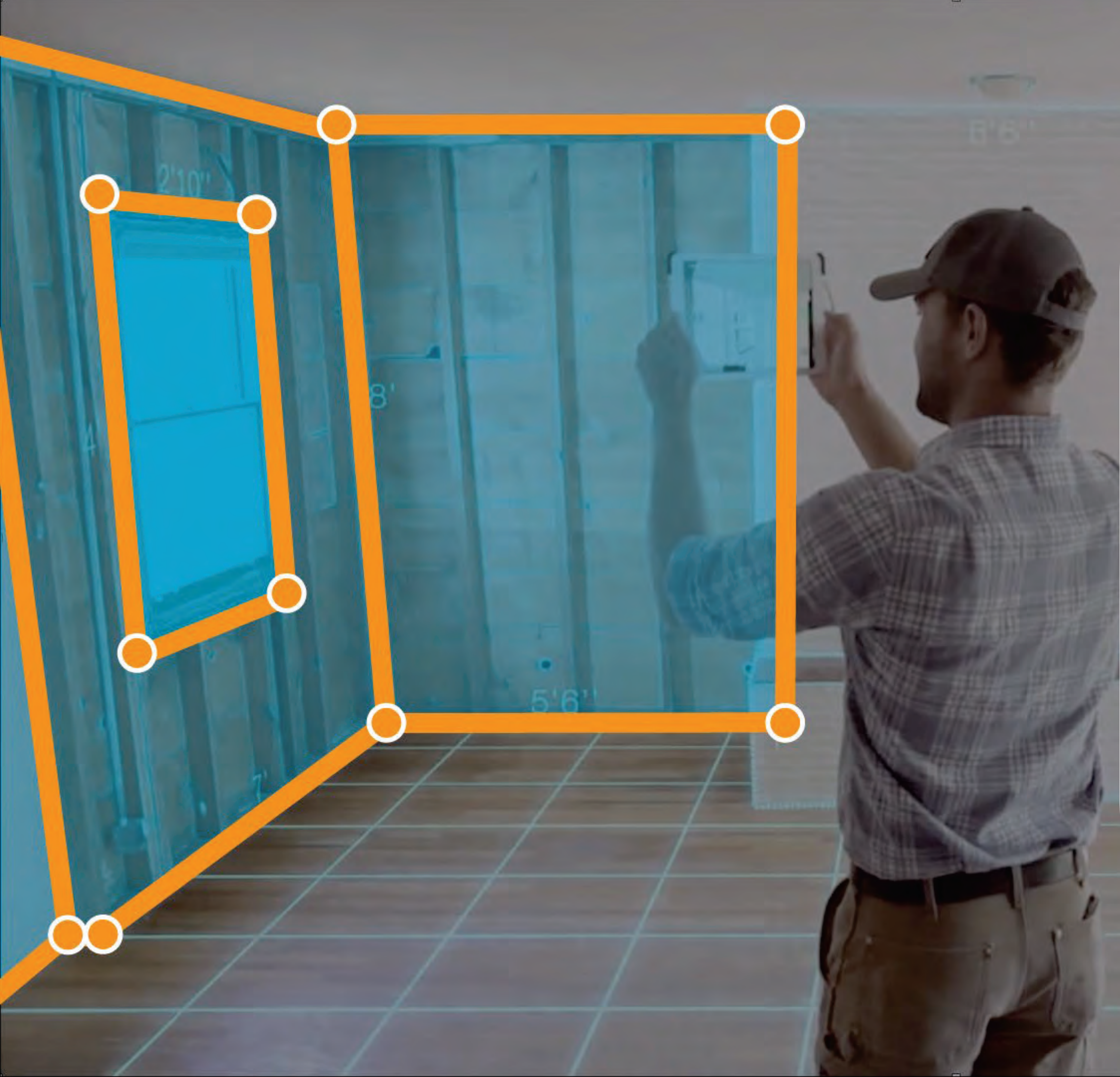


1D primitive



0D primitive

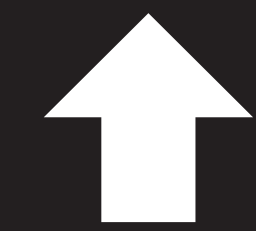
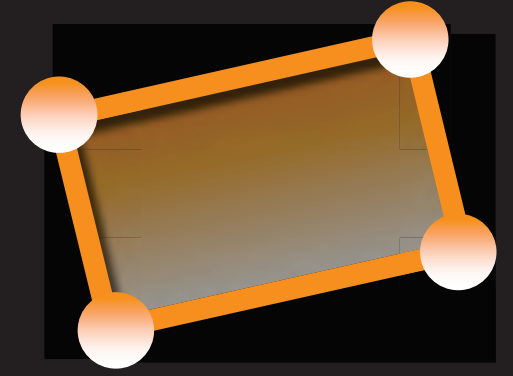




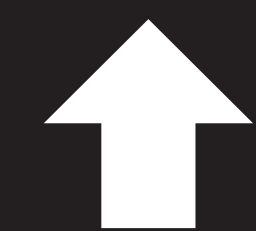
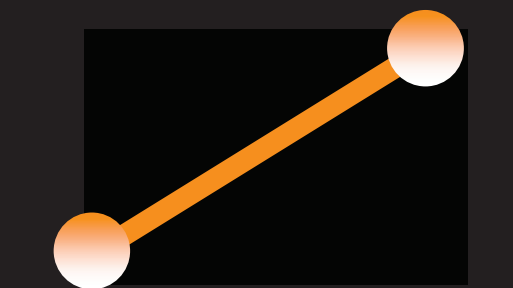
# Bottom-up



2D primitive



1D primitive



0D primitive



### Piecewise Planar and Compact Floorplan Reconstruction from Images

Ricardo Cabral  
Carnegie Mellon University  
rcabral@cmu.edu

Yasutaka Furukawa  
Washington University  
furukawa@wustl.edu

**Abstract**  
This paper presents a system to reconstruct piecewise planar and compact floorplans from images, which are then converted to high quality texture-mapped models for free-viewpoint visualization. There are two main challenges in image-based floorplan reconstruction. The first is the lack of 3D information that can be extracted from images by Structure from Motion and Multi-View Stereo, as indoor scenes abound with non-diffuse and homogeneous surfaces plus clutter. The second challenge is the need of a sophisticated regularization technique that enforces piecewise planarity, to suppress clutter and yield high quality texture-mapped models. Our technical contributions are twofold.




Figure 1. Our system reconstructs high quality texture-mapped mesh models of cluttered indoor scenes from panorama images.

### Structured Indoor Modeling

Satoshi Ikehata      Hang Yan      Yasutaka Furukawa  
Washington University in St. Louis

**Abstract**  
This paper presents a novel 3D modeling framework that reconstructs an indoor scene as a structured model from panorama RGBD images. A scene geometry is represented as a graph, where nodes correspond to structural elements such as rooms, walls, and objects. The approach devises a structure grammar that defines how a scene graph can be manipulated. The grammar then drives a principled new reconstruction algorithm, where the grammar rules are sequentially applied to recover a structured model. The paper also proposes a new room segmentation algorithm and reconstruction approaches exist. However, their output is either a pure polygon soup [30] or a set of planar patches [31]. We establish a computational framework and algorithms for reconstructing structured indoor model from panorama RGBD images. We introduce a novel 3D model representation "structure graph", whose nodes represent structural elements such as rooms, doors, and objects, and the edges represent their geometric relationships. "Structure grammar" then defines a list of possible graph transformations. This grammar drives a principled new reconstruction algorithm, where the rules are sequentially applied to naturally segment, annotate, and reconstruct architectural scenes.

### Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu  
Washington University in St. Louis  
chenliu@wustl.edu

Pushmeet Kohli  
DeepMind  
pushmeet@google.com

Jijun Wu  
Massachusetts Institute of Technology  
jjwu@mit.edu

Yasutaka Furukawa  
Simon Fraser University  
furukawa@sfu.ca




Figure 1. This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a pop-up 3D model.

[cs.CV] 31 Mar 2018

### FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans

Chen Liu\*    Jiaye Wu\*    Yasutaka Furukawa  
Washington University in St. Louis    Simon Fraser University  
{chenliu, jiaye.wu}@wustl.edu    furukawa@sfu.ca

**Abstract.** The ultimate goal of this indoor mapping research is to automatically reconstruct a floorplan simply by walking through a house with

### Neural Procedural Reconstruction for Residential Buildings

Huayi Zeng<sup>1</sup>, Jiaye Wu<sup>1</sup> and Yasutaka Furukawa<sup>2</sup>

<sup>1</sup> Washington University in St. Louis, USA  
{zengh, jiaye.wu}@wustl.edu

<sup>2</sup> Simon Fraser University, Canada  
furukawa@sfu.ca

**Abstract.** This paper proposes a novel 3D reconstruction approach, dubbed Neural Procedural Reconstruction (NPR). NPR infers a sequence of shape grammar rule applications and reconstructs CAD-quality models with procedural structure from 3D points. While most existing methods rely on low-level geometry analysis to extract primitive structures,

### PlaneNet: Piece-wise Planar Reconstruction from a Single RGB Image

Chen Liu<sup>1</sup>    Jimei Yang<sup>2</sup>    Duygu Ceylan<sup>2</sup>    Ersin Yumer<sup>3</sup>    Yasutaka Furukawa<sup>4</sup>

<sup>1</sup>Washington University in St. Louis    <sup>2</sup>Adobe Research    <sup>3</sup>Argo AI    <sup>4</sup>Simon Fraser University  
chenliu@wustl.edu    {jyang, ceylan}@adobe.com    meyumer@gmail.com    furukawa@sfu.ca

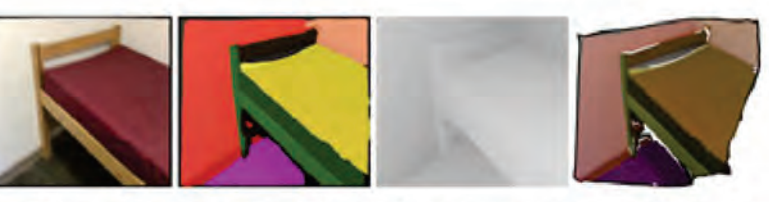


Figure 1. This paper proposes a deep neural architecture for piece-wise planar depthmap reconstruction from a single RGB image. From left to right, an input image, a piece-wise planar segmentation, a reconstructed depthmap, and a texture-mapped 3D model.

### Reconstructing the World's Museums

Jianxiang Xiao - Yasutaka Furukawa

Received date / Accepted date

**Abstract** Virtual exploration tools for large indoor environments (e.g. museums) have so far been limited to either blueprint-style 2D maps that lack photo-realistic views of scenes, or ground-level image-to-image transitions, which are immersive but ill-suited for navigation. On the other hand, photo-realistic aerial maps would be a useful navigational guide for large indoor environments, but it is impossible to directly acquire photographs covering a large indoor environment from aerial viewpoints. This paper presents a 3D reconstruction and visualization system for automatically processing



### Reconstructing Building Interiors from Images

Yasutaka Furukawa, Brian Curless, Steven M. Seitz  
University of Washington, Seattle, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research, Redmond, USA  
rszeliski@microsoft.com

**Abstract**  
This paper proposes a fully automated 3D reconstruction and visualization system for architectural scenes (interiors and exteriors). The reconstruction of indoor environments from photographs is particularly challenging due to texture-poor planar surfaces such as uniformly-painted walls. Our system first uses structure-from-motion, multi-view stereo, and sequential plane-sweep 3D distance



Figure 1: Floor plan and photograph of a house interior.

### Manhattan-world Stereo

Yasutaka Furukawa, Brian Curless, Steven M. Seitz  
Department of Computer Science & Engineering  
University of Washington, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research  
Redmond, USA  
rszeliski@microsoft.com

**Abstract**  
Multi-view stereo (MVS) algorithms now produce reconstructions that rival laser range scanner accuracy. However, stereo algorithms require textured surfaces, and therefore work poorly for many architectural scenes (e.g., building interiors with textureless, painted walls). This paper presents a novel MVS approach to overcome these limitations.




Figure 1. Increasingly oblique images of architectural scenes with texture-poor but highly structured surfaces.

### Floor-SP: Inverse CAD for Floorplans by Sequential Room-wise Shortest Path

Jiacheng Chen<sup>1</sup>    Chen Liu<sup>2</sup>    Jiaye Wu<sup>2</sup>    Yasutaka Furukawa<sup>1</sup>

<sup>1</sup>Simon Fraser University    <sup>2</sup>Washington University in St. Louis  
{jca147, furukawa}@sfu.ca    {chenliu, jiaye.wu}@wustl.edu



Figure 1. The proposed system, dubbed Floor-SP, takes aligned panorama RGBD scans as input, finds room segments, solves an optimization problem to reconstruct a floorplan graph as multiple polygonal loops (one for each room), and merges them into a 2D graph via simple post-processing heuristics. The optimization is the technical contribution of the paper, which employs the room-wise coordinate descent strategy and sequentially solves shortest path problems to optimize the room structure.

### Conv-MPN: Convolutional Message Passing Neural Network for Structure

**Abstract**  
Conv-MPN, a novel message passing neural network, reconstructs outdoor buildings as planar graphs from a single image. The reconstructions after 0, 1, or 3 iterations of message passing are as shown.



Figure 1. Conv-MPN, a novel message passing neural network, reconstructs outdoor buildings as planar graphs from a single image. The reconstructions after 0, 1, or 3 iterations of message passing are as shown.

### Vectorizing World Buildings: Planar Graph Reconstruction by Primitive Detection and Relationship Classification

**Abstract**  
The paper takes a RGB image, detects three geometric primitives (i.e., corners, edges, and regions), classifies their relationships (i.e., corner-to-edge and region-to-region), and fuses the information via Integer Programming to reconstruct a planar graph.

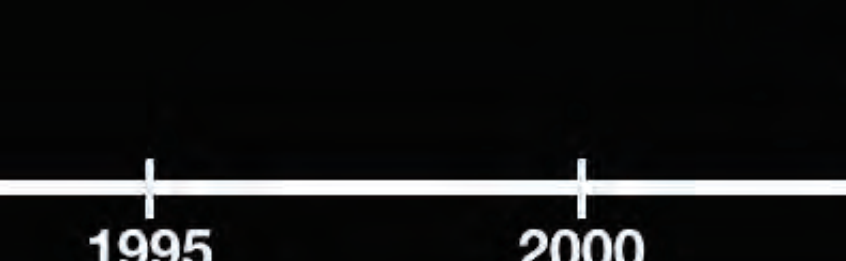
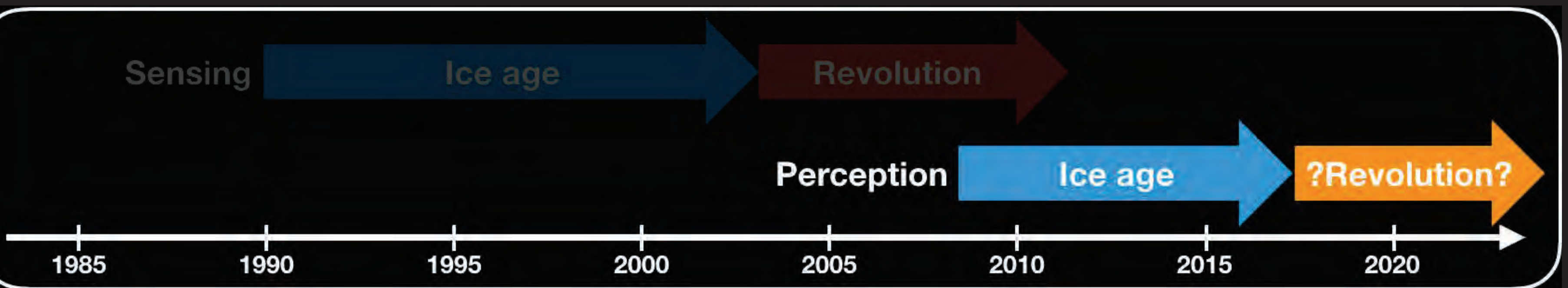


Figure 1. The paper takes a RGB image, detects three geometric primitives (i.e., corners, edges, and regions), classifies their relationships (i.e., corner-to-edge and region-to-region), and fuses the information via Integer Programming to reconstruct a planar graph.

# CVPR-ICCV-ECCV papers for geometry perception...



### PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image

Chen Liu<sup>1,2\*</sup>    Kibwan Kim<sup>1</sup>    Jiwon Gu<sup>1,3\*</sup>    Yasutaka Furukawa<sup>1</sup>    Jan Kautz<sup>4</sup>

<sup>1</sup>NVIDIA    <sup>2</sup>Washington University in St. Louis    <sup>3</sup>SceneTime    <sup>4</sup>Simon Fraser University

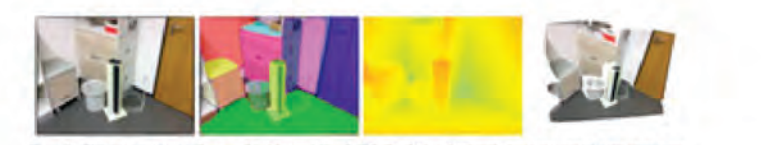


Figure 1. This paper proposes a deep neural architecture, PlaneRCNN, that detects planar regions and reconstructs a piecewise planar depthmap from a single RGB image. From left to right, an input image, segmented plane regions, estimated depthmap, and reconstructed planar surfaces.

### Conv-MPN: Convolutional Message Passing Neural Network for Structure

**Abstract**  
Conv-MPN, a novel message passing neural network, reconstructs outdoor buildings as planar graphs from a single image. The reconstructions after 0, 1, or 3 iterations of message passing are as shown.

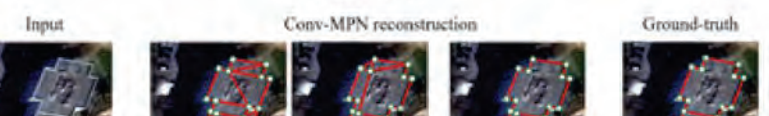


Figure 1. Conv-MPN, a novel message passing neural network, reconstructs outdoor buildings as planar graphs from a single image. The reconstructions after 0, 1, or 3 iterations of message passing are as shown.

### Vectorizing World Buildings: Planar Graph Reconstruction by Primitive Detection and Relationship Classification

**Abstract**  
The paper takes a RGB image, detects three geometric primitives (i.e., corners, edges, and regions), classifies their relationships (i.e., corner-to-edge and region-to-region), and fuses the information via Integer Programming to reconstruct a planar graph.

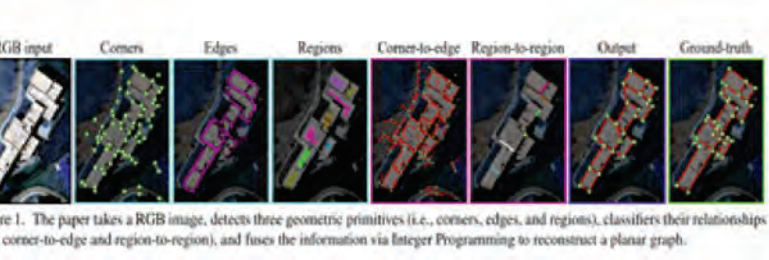


Figure 1. The paper takes a RGB image, detects three geometric primitives (i.e., corners, edges, and regions), classifies their relationships (i.e., corner-to-edge and region-to-region), and fuses the information via Integer Programming to reconstruct a planar graph.

# Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu  
Washington University in St. Louis

chenliu@wustl.edu

Pushmeet Kohli†  
DeepMind

pushmeet@google.com

Jiajun Wu  
Massachusetts Institute of Technology

jiajunwu@mit.edu

Yasutaka Furukawa\*  
Simon Fraser University

furukawa@sfu.ca

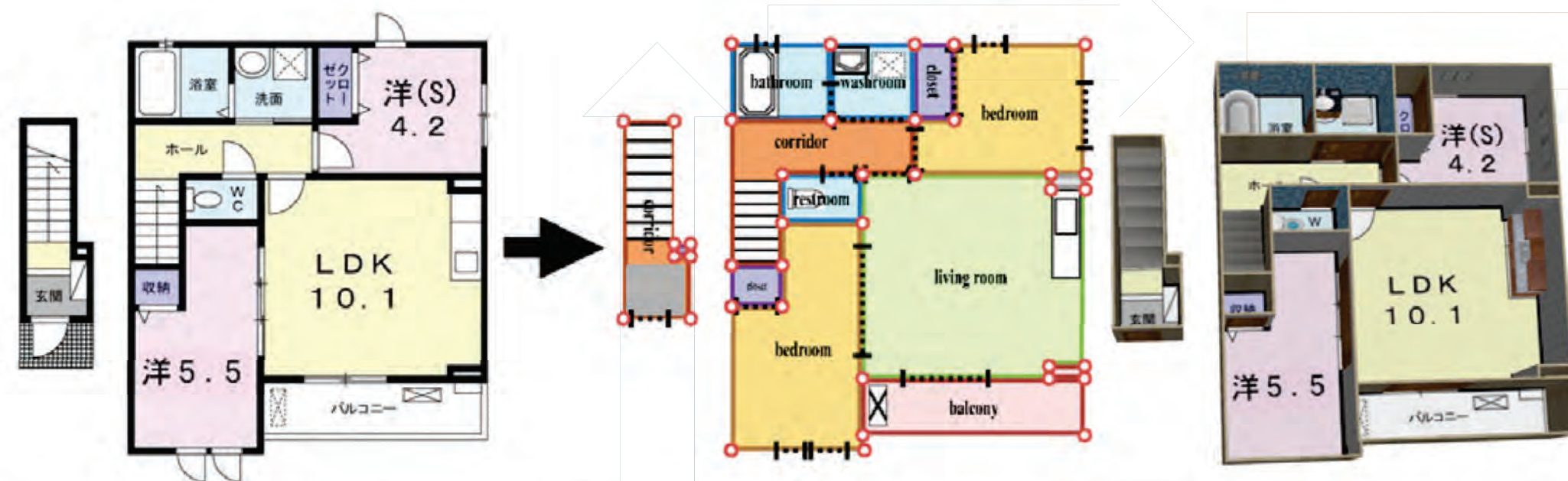


Figure 1: This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a popup 3D model.

民間企業提供データ

- Yahoo!データセット
- 楽天データセット
- ニコニコデータセット
- リクルートデータセット
- クックパッドデータセット
- LIFULL HOME'Sデータセット
- 不満調査データセット
- Sansanデータセット
- インテージデータセット
- オリコンデータセット
- ダイエットロコミデータ

## LIFULL HOME'Sデータセット (旧名称: HOME'Sデータセット)

国立情報学研究所が株式会社LIFULL (旧社名 株式会社ネクスト) から提供を受けて研究者に提供しているデータセットです。

2019/09/12 更新

### データ概要

不動産・住宅情報サイトLIFULL HOME'Sに掲載されたデータです。

- 賃貸物件スナップショットデータ (2015年9月時点, 賃貸物件データ+画像データ)  
全国約533万件についての賃料, 面積, 立地 (市区町村, 郵便番号, 最寄り駅, 徒歩分), 築年数, 間取り, 建物構造, 諸設備などのデータと, 各物件に対する間取り図や室内写真など約8,300万枚の画像データです。IDはユニーク番号に変換済みで, 特定の物件に紐づく属性は含んでおりません。賃貸物件データはTSV形式のファイルで約1.6GBです。画像データは最大横120ピクセル×縦120ピクセルのJPEG形式で, 圧縮ファイルで約210GBとなります。画像のメタデータには「玄関」「キッチン」といった画像の種類や, 一部にはフリーテキストによる説明が付与されています。
- 高精細度間取り図画像データ (賃貸物件スナップショットデータに対応)  
賃貸物件スナップショットの画像データのうち, 間取り図に関しての高精細度版の画像データ約515万枚です。JPEG形式で, 圧縮ファイルで約140GBとなります。本データに関しては別途お申し込みが必要となります (詳細は「お問い合わせ」欄をご覧ください)。

Satoshi Ikehata



1985 1990 1995 2000 2005 2010 2015 2020



Yoji Kiyota



Tomoko Osuga

• 「HOME'Sデータセット」の配布を開始しました。(2015/11/24)



## Reconstructing Building Interiors from Images

Yasutaka Furukawa, Brian Curless, Steven M. Seitz  
University of Washington, Seattle, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research, Redmond, USA  
szeliski@microsoft.com

### Abstract

This paper proposes a fully automated 3D reconstruction and visualization system for architectural scenes (interiors and exteriors). The reconstruction of indoor environments from photographs is particularly challenging due to texture-poor planar surfaces such as uniformly-painted walls. Our system first uses structure-from-motion, multi-view stereo, and a stereo algorithm specifically designed for



Figure 1: Floor plan and photograph of a house interior.

## Manhattan-world Stereo

Yasutaka Furukawa Brian Curless Steven M. Seitz  
Department of Computer Science & Engineering  
University of Washington, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research  
Redmond, USA  
szeliski@microsoft.com

### Abstract

Multi-view stereo (MVS) algorithms now produce reconstructions that rival laser range scanner accuracy. However, stereo algorithms require textured surfaces, and therefore work poorly for many architectural scenes (e.g., building interiors with textureless, painted walls). This paper presents a novel MVS approach to overcome these limi-



Figure 1. Increasingly ubiquitous on the Internet are images of architectural scenes with texture-poor but highly structured surfaces.

## Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu  
Washington University in St. Louis  
chenliu@wustl.edu  
Pushmeet Kohli†  
DeepMind  
pushmeet@google.com

Jiajun Wu  
Massachusetts Institute of Technology  
jiajunwu@mit.edu  
Yasutaka Furukawa\*  
Simon Fraser University  
furukawa@sfu.ca

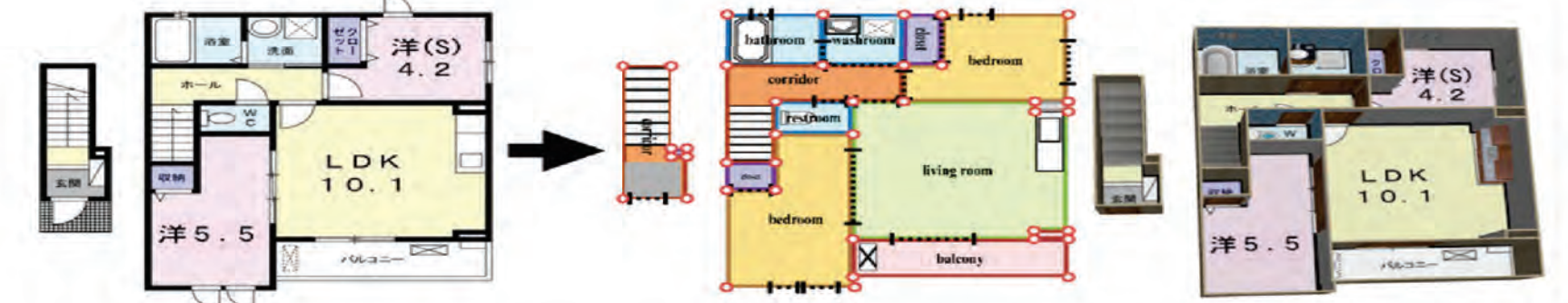


Figure 1: This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a popup 3D model.

## FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans

Chen Liu\* Jiaye Wu\*  
Washington University in St. Louis  
{chenliu, jiaye.wu}@wustl.edu

Yasutaka Furukawa  
Simon Fraser University  
furukawa@sfu.ca

**Abstract.** The ultimate goal of this indoor mapping research is to automatically reconstruct a floorplan simply by walking through a house with

Sensing

Ice age

Revolution

Perception

Ice age

?Revolution?

1985

1990

1995

2000

2005

2010

2015

2020

## Manhattan-world Stereo

Yasutaka Furukawa   Brian Curless   Steven M. Seitz  
Department of Computer Science & Engineering  
University of Washington, USA

{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research  
Redmond, USA

szeliski@microsoft.com

### Abstract

*Multi-view stereo (MVS) algorithms now produce reconstructions that rival laser range scanner accuracy. However, stereo algorithms require textured surfaces, and therefore work poorly for many architectural scenes (e.g., building interiors with textureless, painted walls). This paper presents a novel MVS approach to overcome these limi-*

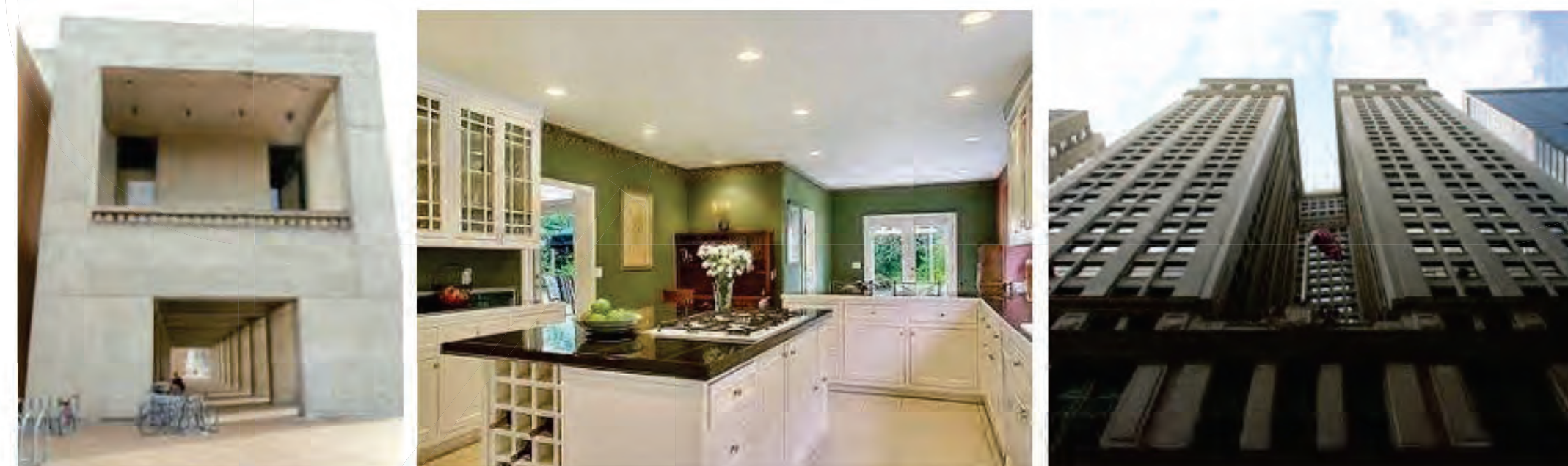
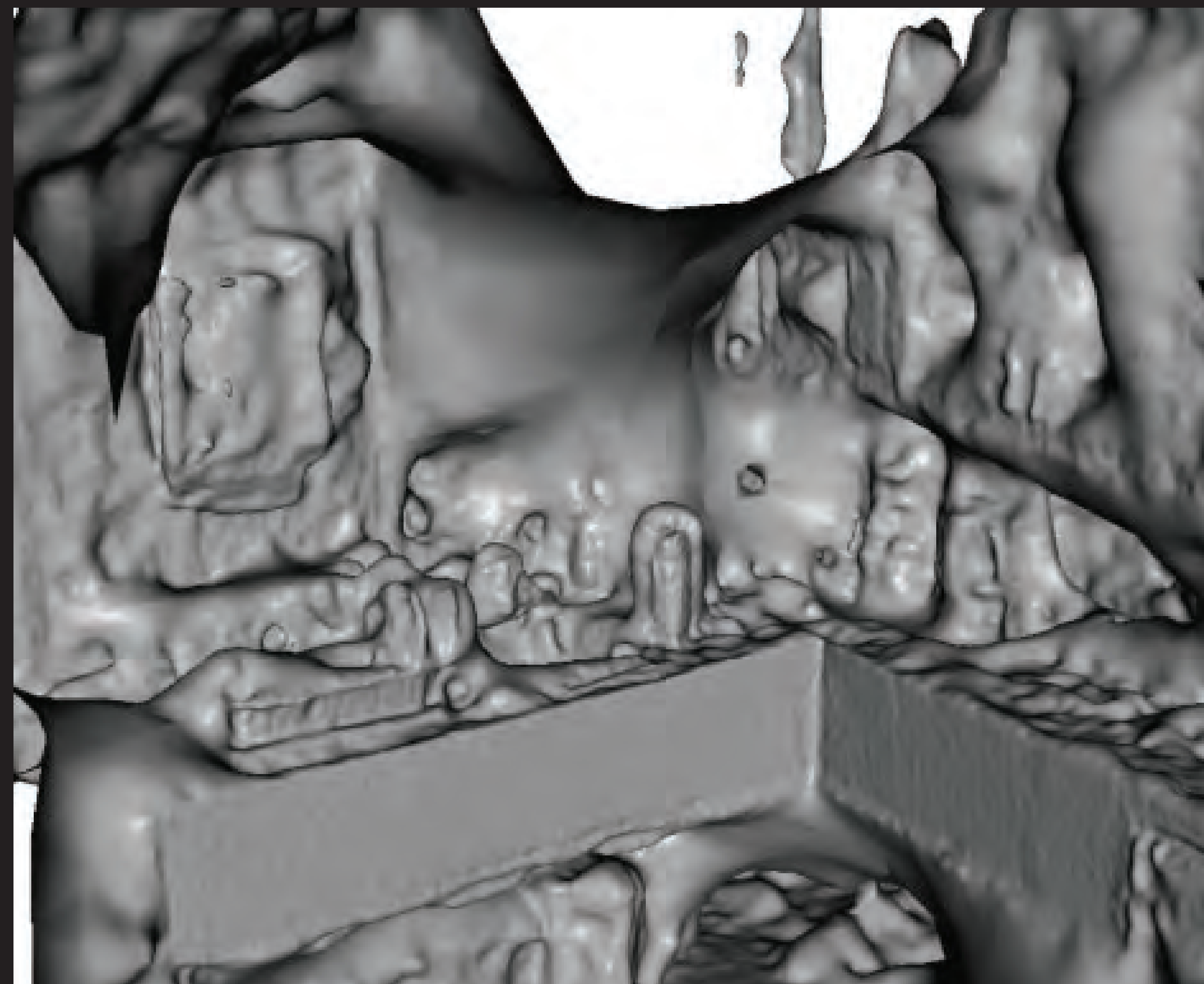


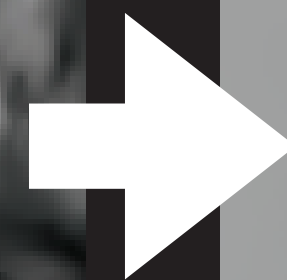
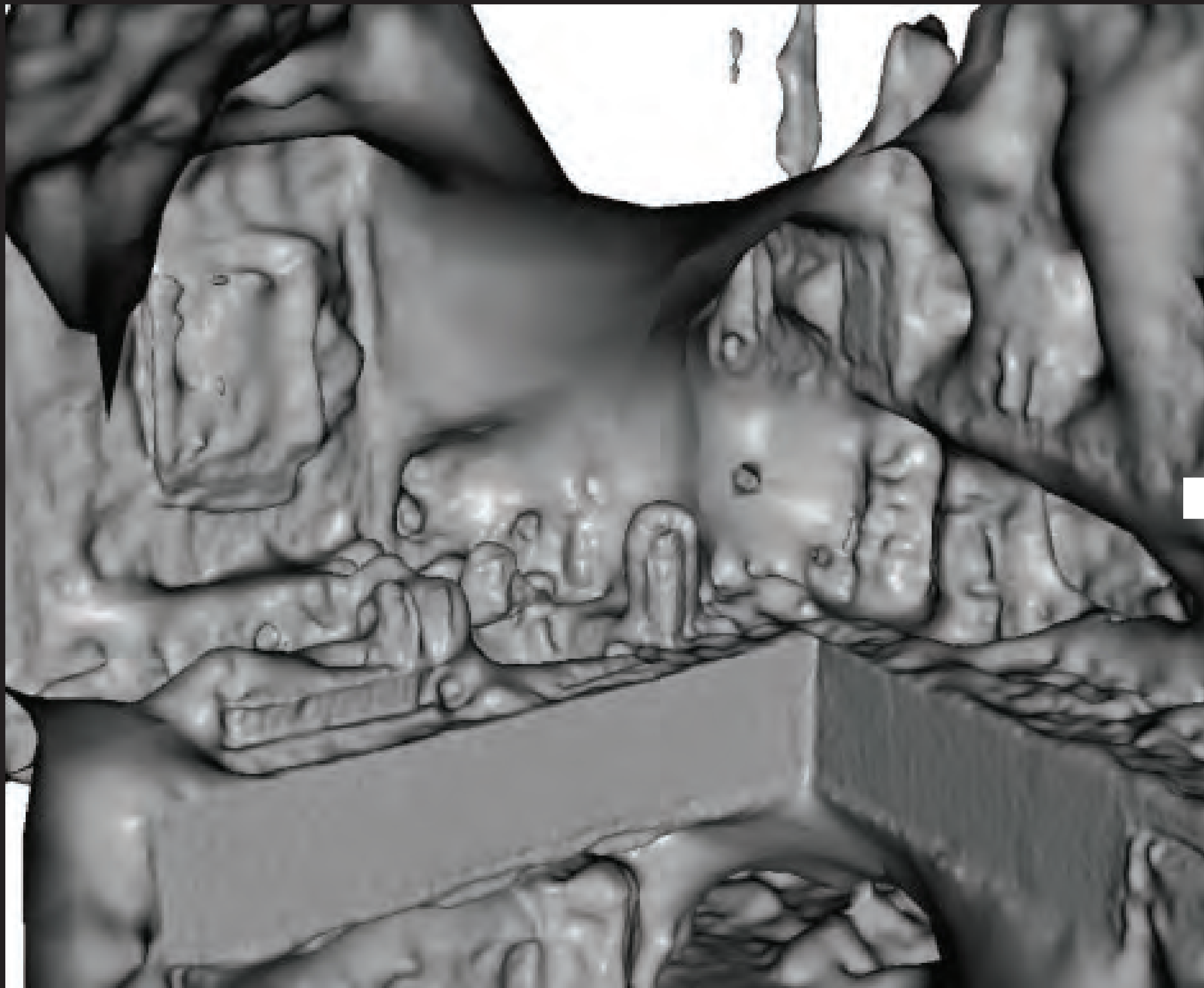
Figure 1. Increasingly ubiquitous on the Internet are images of architectural scenes with texture-poor but highly structured surfaces.

# Geometry sensing



[Furukawa and Ponce, 2007]

# Geometry perception

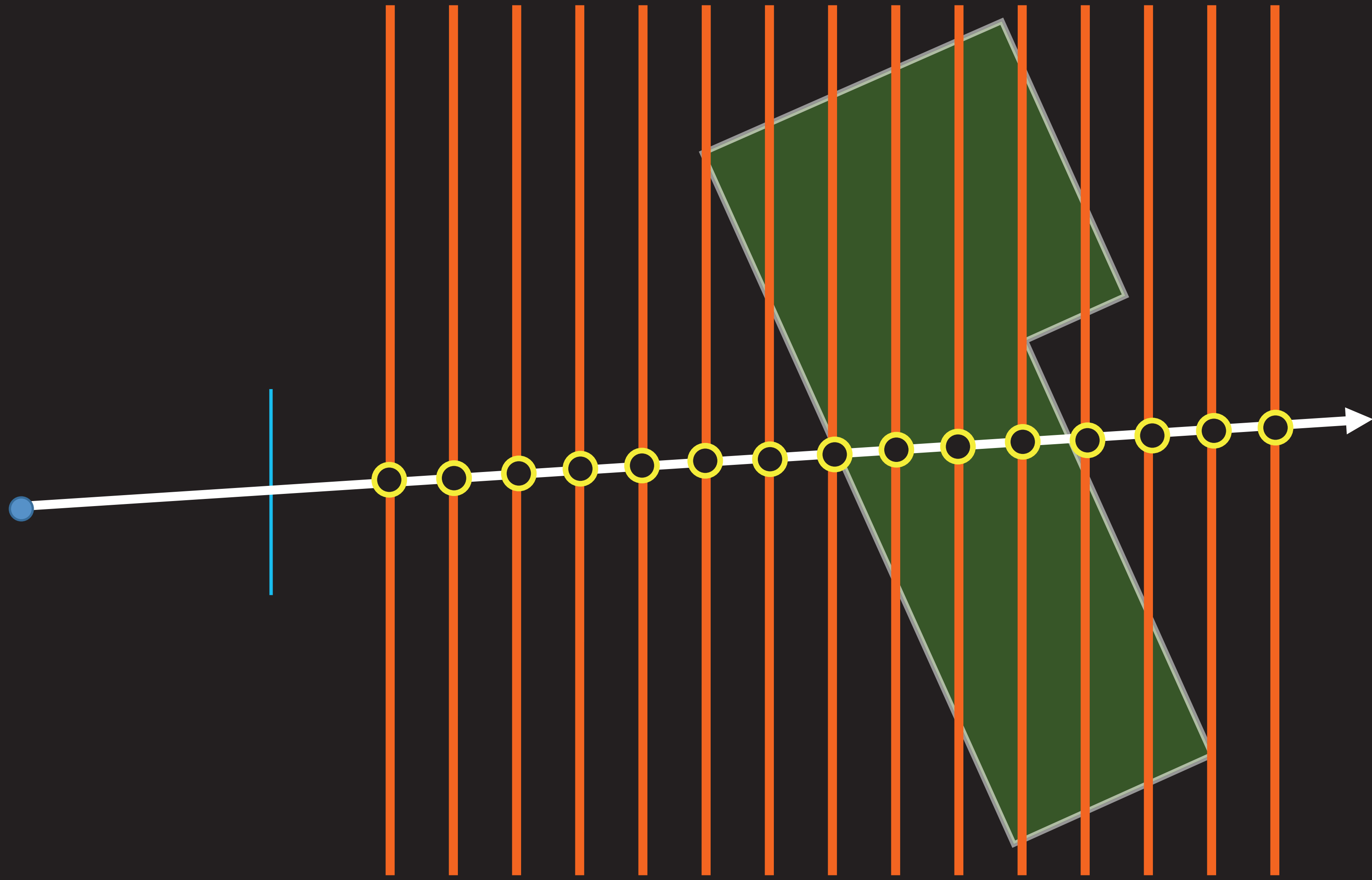


[Furukawa and Ponce, 2007]

Planarity/orthogonality enforcement

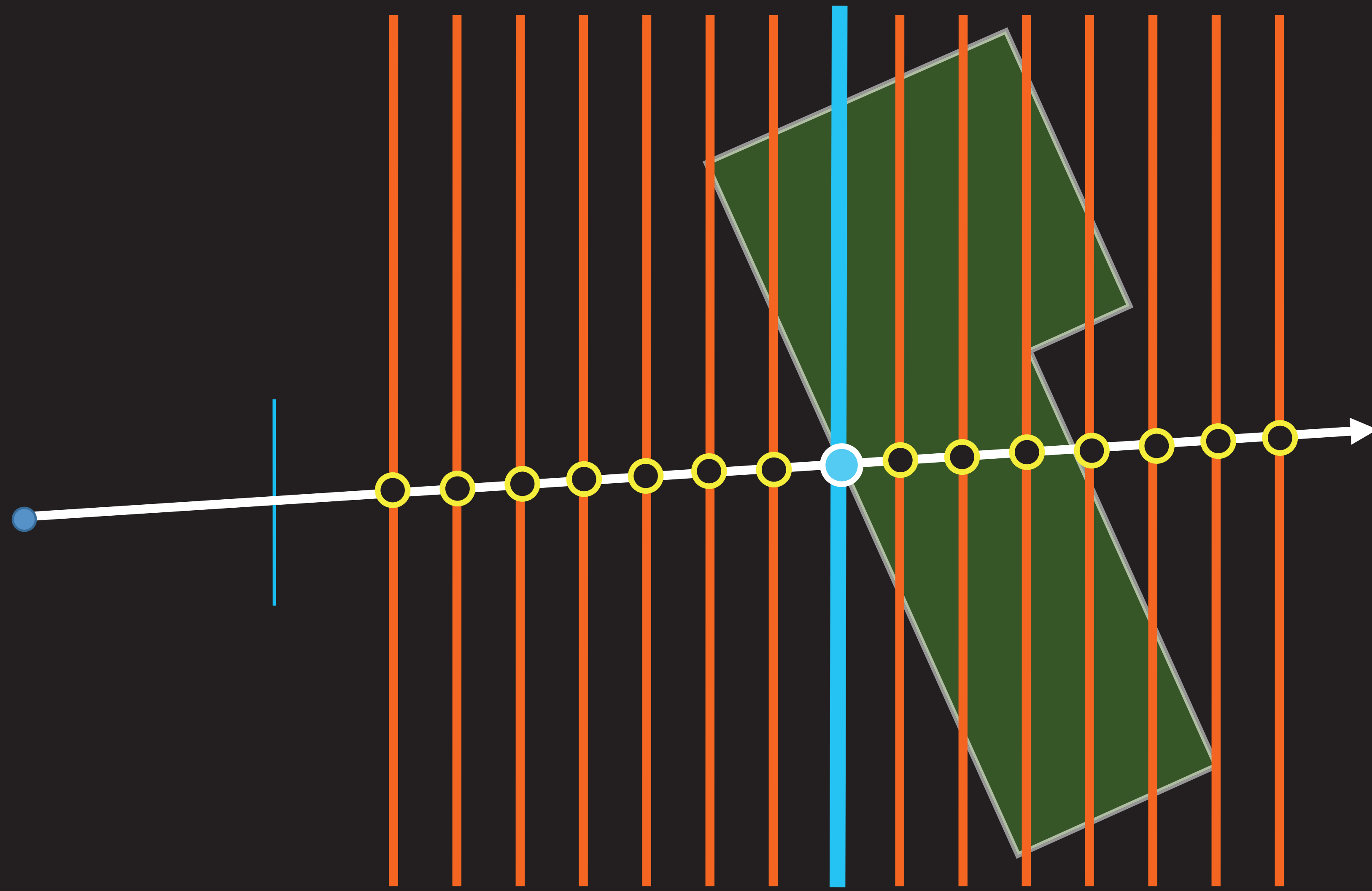
# (Sensing) Depthmap estimation

Possible depth values



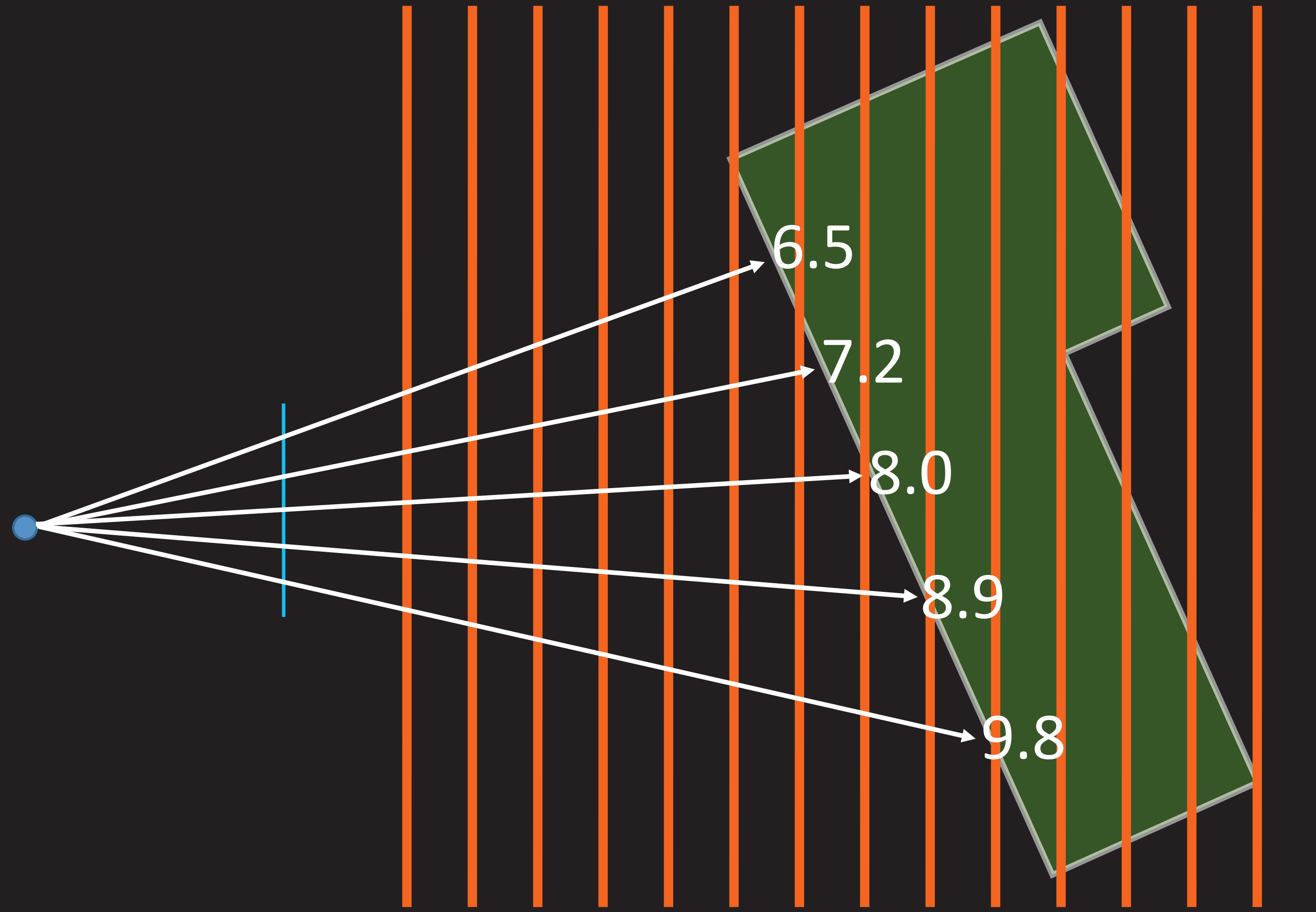
# (Sensing) Depthmap estimation

Possible depth values



# (Sensing) Depthmap estimation

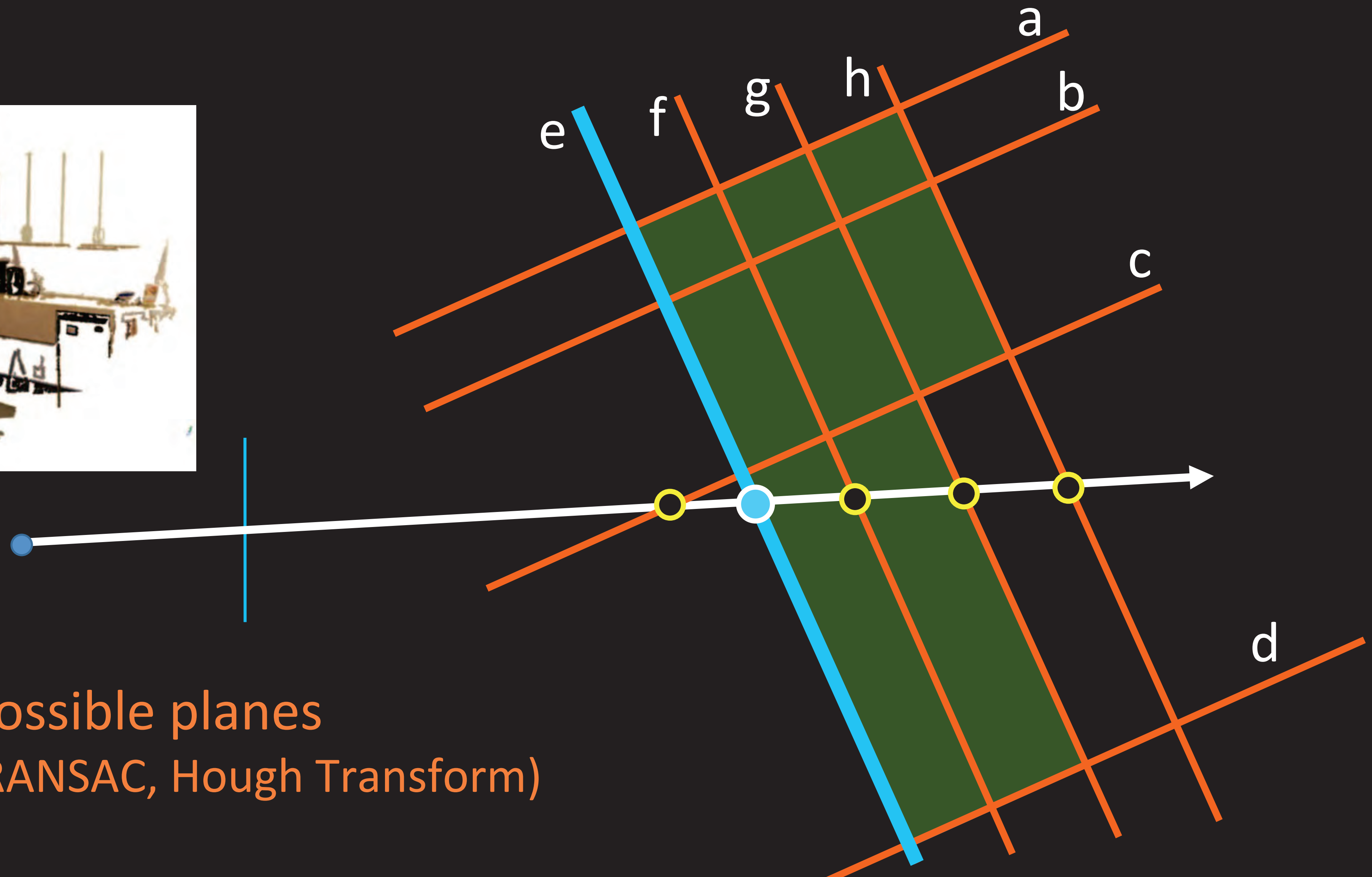
Possible depth values



~~(Sensing)~~ Depthmap estimation  
(Perception)

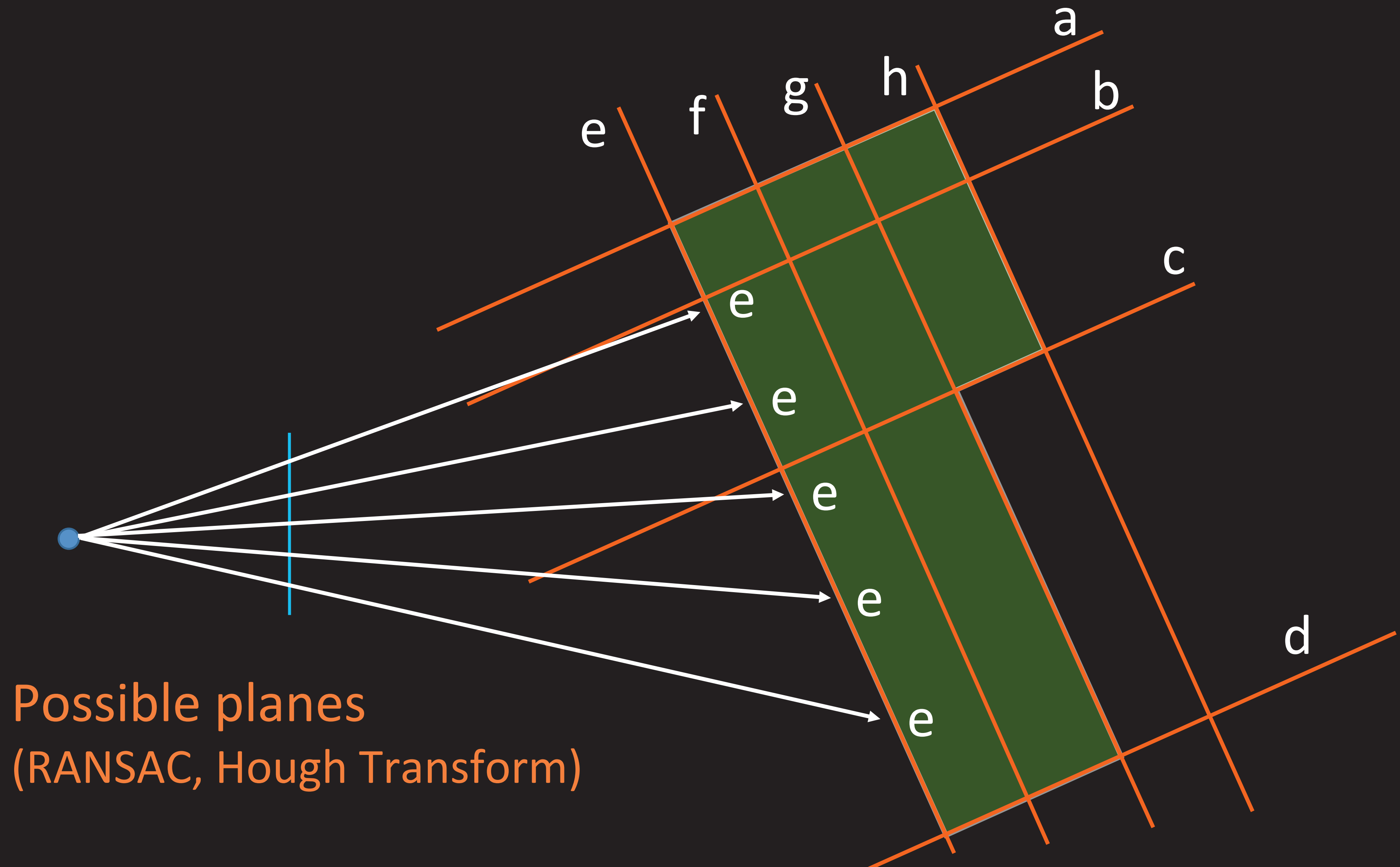


# (Perception) Depthmap estimation



Possible planes  
(RANSAC, Hough Transform)

# (Perception) Planemap estimation



# Graphical Model Inference

## What Energy Functions Can Be Minimized via Graph Cuts?

Vladimir Kolmogorov, Member, IEEE, and Ramin Zabih, Member, IEEE

**Abstract**—In the last few years, several new algorithms based on graph cuts have been developed to solve energy minimization problems in computer vision. Each of these techniques constructs a graph such that the minimum cut on the graph also minimizes the energy. Yet, because these graph constructions are complex and highly specific to a particular energy function, graph cuts have seen limited application to date. In this paper, we give a characterization of the energy functions that can be minimized by graph cuts. Our results are restricted to functions of binary variables. However, our work generalizes many previous constructions and is easily applicable to vision problems that involve large numbers of labels, such as stereo, motion, image restoration, and scene reconstruction. We give a precise characterization of what energy functions can be minimized using graph cuts, among the energy functions that can be written as a sum of terms containing three or fewer binary variables. We also provide a general-purpose construction to minimize such an energy function. Finally, we give a necessary condition for any energy function of binary variables to be minimized by graph cuts. Researchers who are considering the use of graph cuts to optimize a particular energy function can use our results to determine if this is possible and then follow our construction to create the appropriate graph. A software implementation is freely available.

**Index Terms**—Energy minimization, optimization, graph algorithms, minimum cut, maximum flow, Markov Random Fields.

### 1 INTRODUCTION AND OVERVIEW

Many of the problems that arise in early vision can be naturally expressed in terms of energy minimization. The computational task of minimizing the energy is usually quite difficult as it generally requires minimizing a nonconvex function in a space with thousands of dimensions. If the functions have a very restricted form, they can be solved efficiently using dynamic programming [2]. However, researchers typically have needed to rely on general purpose optimization techniques such as simulated annealing [3], [16], which requires exponential time in theory and is extremely slow in practice.

In the last few years, however, a new approach has been developed based on graph cuts. The basic technique is to construct a specialized graph for the energy function to be minimized such that the minimum cut on the graph also minimizes the energy (either globally or locally). The minimum cut, in turn, can be computed very efficiently by max flow algorithms. These methods have been successfully used for a wide variety of vision problems, including image restoration [9], [10], [18], [21], stereo and motion [4], [9], [10], [20], [24], [27], [32], [35], [36], image synthesis [29], image segmentation [8], voxel occupancy [39], multicamera scene reconstruction [28], and medical imaging [5], [6], [25], [26]. The output of these algorithms is generally a solution with some interesting theoretical quality guarantee. In some cases [9], [18], [20], [21], [35], it is the global minimum, in other cases, a local minimum in a strong sense [10] that is within a known factor of the global

minimum. The experimental results produced by these algorithms are also quite good. For example, two recent evaluations of stereo algorithms using real imagery with dense ground truth [37], [41] found that the best overall performance was due to algorithms based on graph cuts.

Minimizing an energy function via graph cuts, however, remains a technically difficult problem. Each paper constructs its own graph specifically for its individual energy function and, in some of these cases (especially [10], [27], [28]), the construction is fairly complex. One consequence is that researchers sometimes use heuristic methods for optimization, even in situations where the exact global minimum can be computed via graph cuts. The goal of this paper is to precisely characterize the class of energy functions that can be minimized via graph cuts and to give a general-purpose graph construction that minimizes any energy function in this class. Our results play a key role in [28], provide a significant generalization of the energy minimization methods used in [4], [5], [6], [10], [18], [26], [32], [39], and show how to minimize an interesting new class of energy functions.

In this paper, we only consider energy functions involving binary-valued variables. At first glance, this restriction seems severe since most work with graph cuts considers energy functions with variables that have many possible values. For example, the algorithms presented in [10] use graph cuts to address the standard pixel labeling problem that arises in early vision, including stereo, motion, and image restoration. In the pixel-labeling problem, the possible variables represent individual pixels and the possible values for an individual variable represent, e.g., its possible displacements or intensities. However, as we discuss in Section 2, many of the graph cut methods that repeatedly minimize a multiple possible values work by repeatedly minimizing an energy function involving only binary variables. As we will see, our results generalize many graph cut algorithms [4], [5], [6], [10], [18], [26], [39] and can be easily applied to

The authors are with the Computer Science Department, Cornell University, Ithaca, NY 14853. E-mail: {vovk, rz}@cs.cornell.edu.  
Manuscript received 2 May 2002; revised 17 Mar. 2003; accepted 16 June 2003.  
Recommended for acceptance by M.A.T. Figueiredo, E.R. Hancock, M. Phillis, and J. Zerubia.  
For information on obtaining reprints of this article, please send email to: [transac@computer.org](mailto:transac@computer.org), and reference IEEECS Log Number 118731.

## Fast Approximate Energy Minimization via Graph Cuts

Yuri Boykov, Olga Veksler and Ramin Zabih\*

### Abstract

Many tasks in computer vision involve assigning a label (such as disparity) to every pixel. A common constraint is that the labels should vary smoothly almost everywhere while preserving sharp discontinuities that may exist, e.g., at object boundaries. These tasks are naturally stated in terms of energy minimization. In this paper we consider a wide class of energies with various smoothness constraints. Global minimization of these energy functions is NP-hard even in the simplest discontinuity-preserving case. Our focus is therefore on efficient approximation algorithms. We present two algorithms based on graph cuts that efficiently find a local minimum with respect to two types of large moves, namely *expansion* moves and *swap* moves. These moves can simultaneously change the labels of arbitrarily large sets of pixels. In contrast, many standard algorithms (including simulated annealing) use small moves where only one pixel changes its label at a time. Our expansion algorithm finds a labeling within a known factor of the global minimum, while our swap algorithm handles more general energy functions. Both algorithms allow important cases of discontinuity preserving energies. We experimentally demonstrate the effectiveness of our approach for image restoration, stereo and motion. On real data with ground truth we achieve 98% accuracy.

**Index Terms** — Energy minimization, graph, minimum cut, maximum flow, stereo, motion, image restoration, Markov Random Fields, Potts model, multiway cut.

### 1 Energy minimization in early vision

Many early vision problems require estimating some spatially varying quantity (such as intensity or disparity) from noisy measurements. Such quantities tend to be piecewise smooth;

\*Yuri Boykov ([yuri@csd.uwo.ca](mailto:yuri@csd.uwo.ca)) and Olga Veksler ([olga@csd.uwo.ca](mailto:olga@csd.uwo.ca)) are with the Computer Science Department at the University of Western Ontario, Canada. Ramin Zabih ([rdz@cs.cornell.edu](mailto:rdz@cs.cornell.edu)) is with the Computer Science Department, Cornell University, Ithaca, NY. Note that most of this work was done while Yuri Boykov and Olga Veksler were at Cornell.

## Minimizing Sparse Higher Order Energy Functions of Discrete Variables

Carsten Rother, Pushmeet Kohli  
Microsoft Research Cambridge  
{rother,prohli}@microsoft.com

Wei Feng, Jiaya Jia  
The Chinese University of Hongkong  
{wfeng,lejia}@cse.cuhk.edu.hk

### Abstract

Higher order energy functions have the ability to encode high level structural dependencies between pixels, which have been shown to be extremely powerful for image labeling problems. Their use, however, is severely hampered in practice by the intractable complexity of representing and minimizing such functions. We observed that higher order functions encountered in computer vision are very often "sparse", i.e. many labelings of a higher order clique are equally unlikely and hence have the same high cost. In this paper, we address the problem of minimizing such sparse higher order energy functions. Our method works by transforming the problem into an equivalent quadratic function minimization problem. The resulting quadratic function can be minimized using popular message passing or graph cut based algorithms for MAP inference. Although this is primarily a theoretical paper, it also shows how higher order functions can be used to obtain impressive results for the binary texture restoration problem.

### 1. Introduction

Many computer vision problems such as object segmentation, disparity estimation, and 3D reconstruction can be formulated as pixel or voxel labeling problems. The conventional methods for solving these problems use pairwise Conditional and Markov Random Field (CRF/MRF) formulations [20], which allow for the exact or approximate inference of Maximum a Posteriori (MAP) solutions using extremely efficient algorithms such as Belief Propagation (BP) [4, 15, 22], graph cuts [2] and Tree-Resewighted (TRW) [9, 21] message passing. Although pairwise random field models permit efficient inference, they have restricted expressive power as they can only model interactions between pairs of structural variables. They are unable to enforce the high level structural dependencies between pixels which have been shown to be extremely powerful for image labeling problems.

The last few years have seen the successful application of higher order CRFs and MRFs to some low level vision problems such as image restoration, disparity estimation and object segmentation [7, 14, 16, 23, 24]. In spite of these encouraging results, the use of such models have not spread to other labeling problems. We believe that this is primarily due to the lack of efficient algorithms for performing in-

ference in such models. This paper proposes a method for minimizing general higher order functions that can be used to perform MAP inference in higher order random fields. We follow the classical approach for minimizing higher order functions which can be broken down into two essential steps [1]: (a) Transformation of the higher order energy into a quadratic function, and (b) Minimization of the resulting function using efficient inference algorithms. The first step in this approach is also the most crucial one. Transformation of a general  $m$ -order function to an equivalent quadratic function involves the addition of exponential number of auxiliary variables [1, 5]. Alternatively, the addition of a single random variable with an exponential label space is needed. Both these approaches make the resulting quadratic function minimization problem intractable. Recent work on solving higher order functions in vision have side-stepped the problem of minimizing general higher order potential functions (such as the  $P^m$  Potts model [7]) which can be transformed to quadratic ones by the addition of a few auxiliary variables.

In this paper, we address the problem of minimizing general higher order functions. This is intrinsically a computationally expensive problem since even the parametrization of a general  $m$  order function of  $k$ -state variables requires  $k^m$  parameters. However, the higher order functions used in computer vision have certain properties like *sparseness* which makes them easy to handle. A typical example would be the patch based potentials used for image restoration. It is well-known that the set of  $5 \times 5$  patches of natural images is a small subset of the set of all possible  $5 \times 5$  patches. Higher order potentials used for image restoration enforce that patches in the restored image come from the set of natural image patches. In other words, these functions assign a low cost (or energy) to only a few label assignments (natural patches). The rest of the labelings (artificial patches) are given a high (almost constant) cost (see section 5 for more details). We show how such sparse higher order functions can be transformed to quadratic ones with the addition of only a small number of auxiliary variables and edges. It should be noted that our method allows for the exact transformation of general higher order functions to quadratic functions albeit with an addition of exponential number of auxiliary variables in the worst case.

**Outline of the Paper** We provide our notation and review

$m$   
 $p_1$

$E$   
 $(j)$

If

$\geq$

$v_2, v_3$

$v_3$

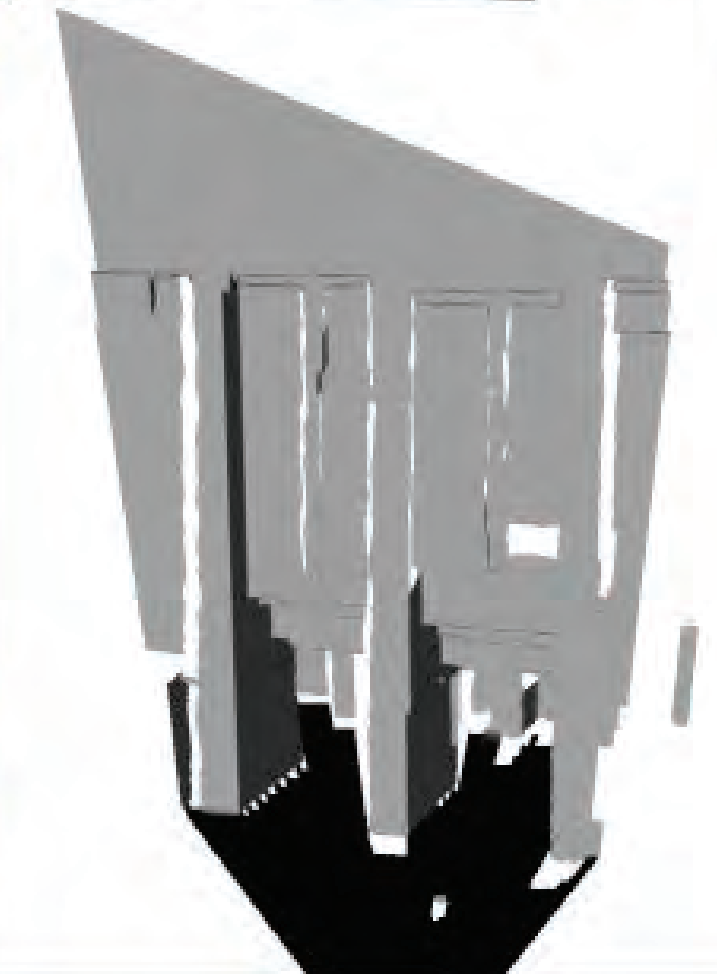
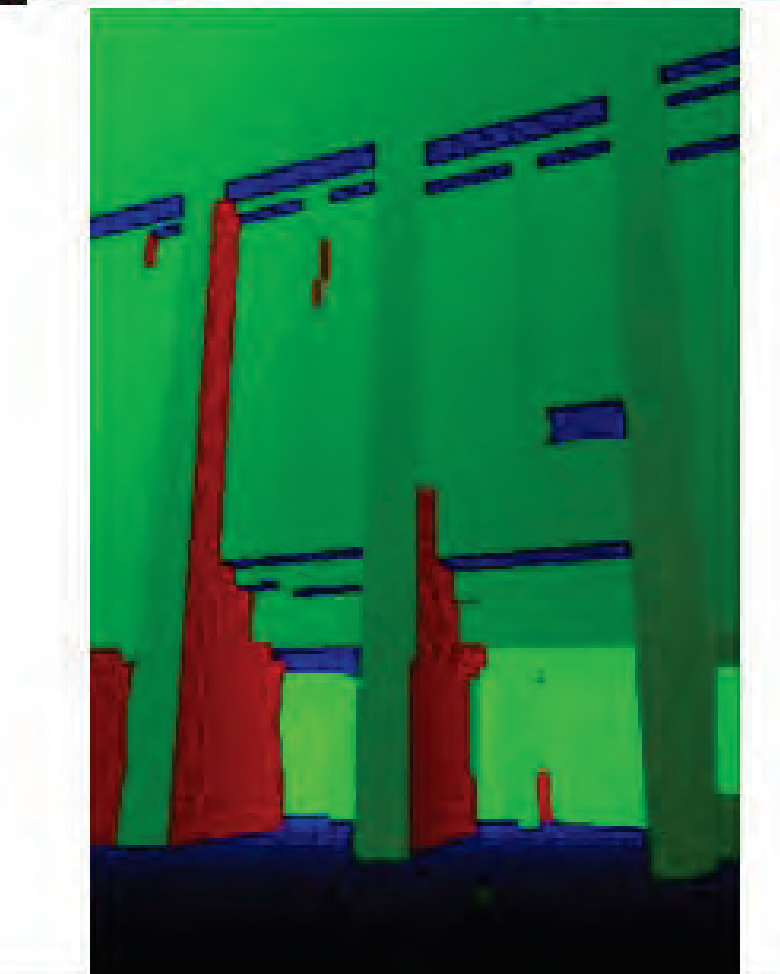
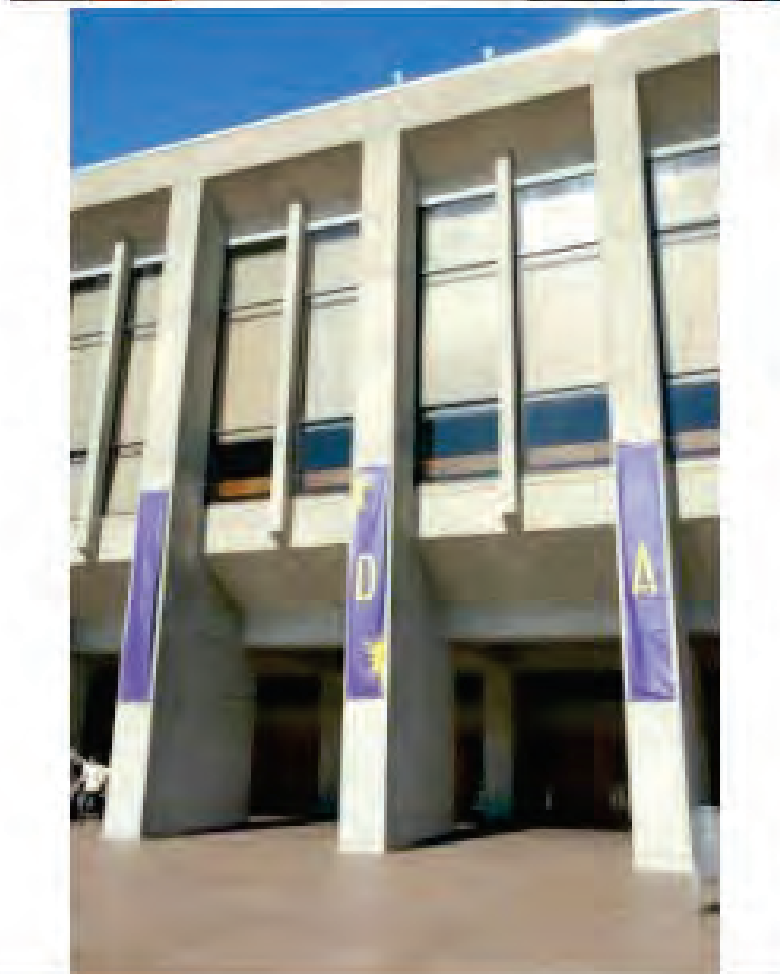
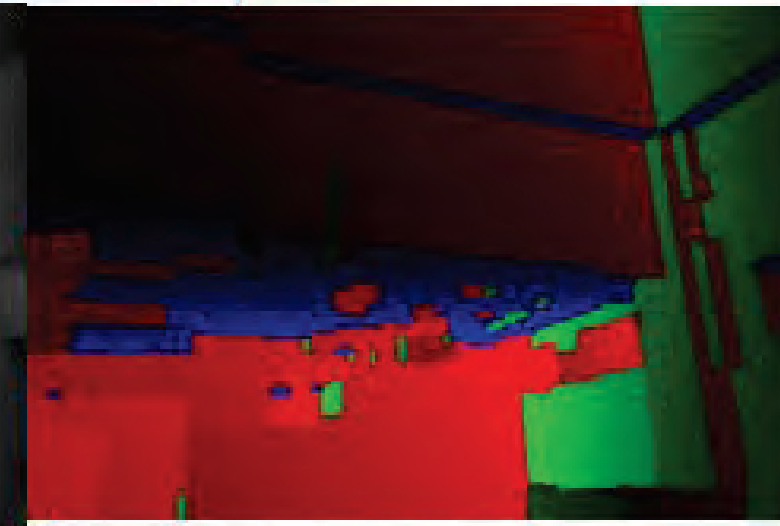
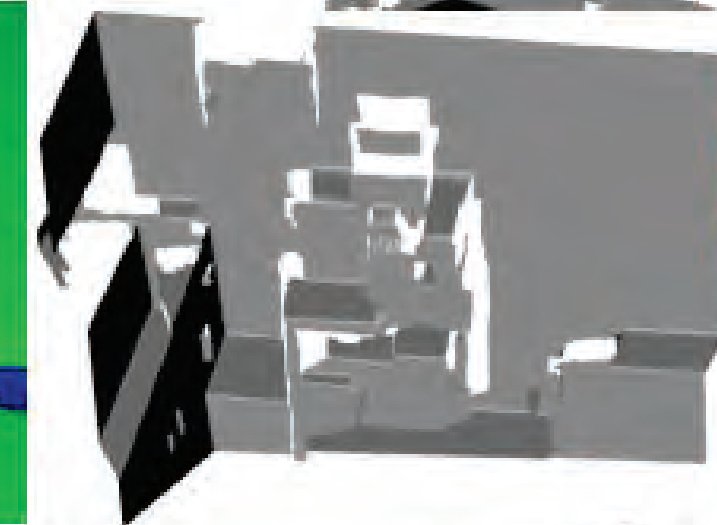
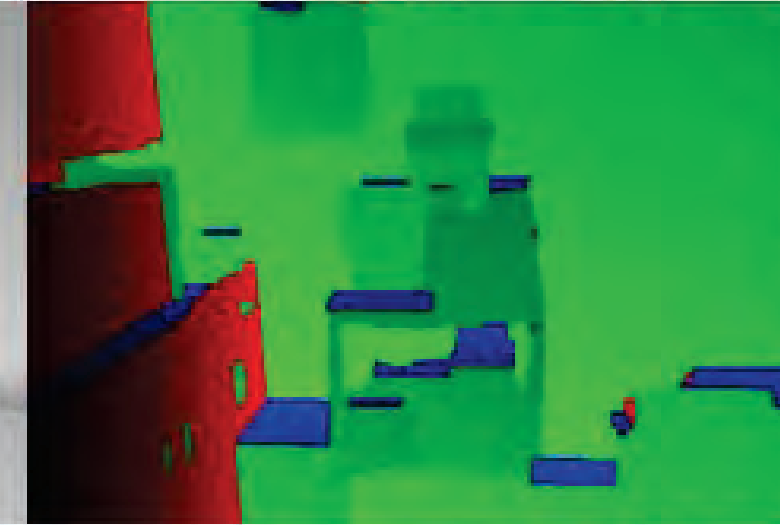
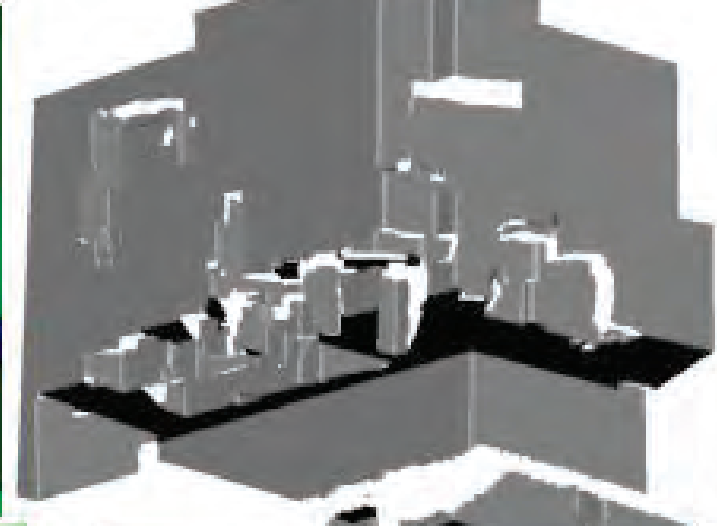
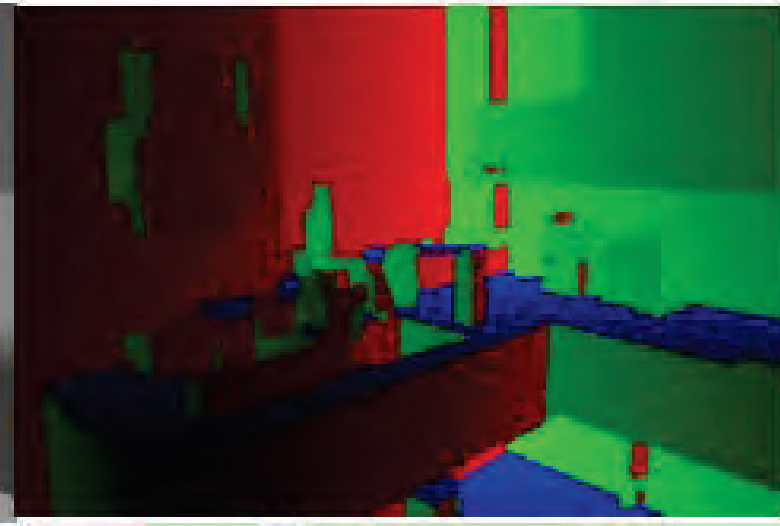
Target image

Depth map

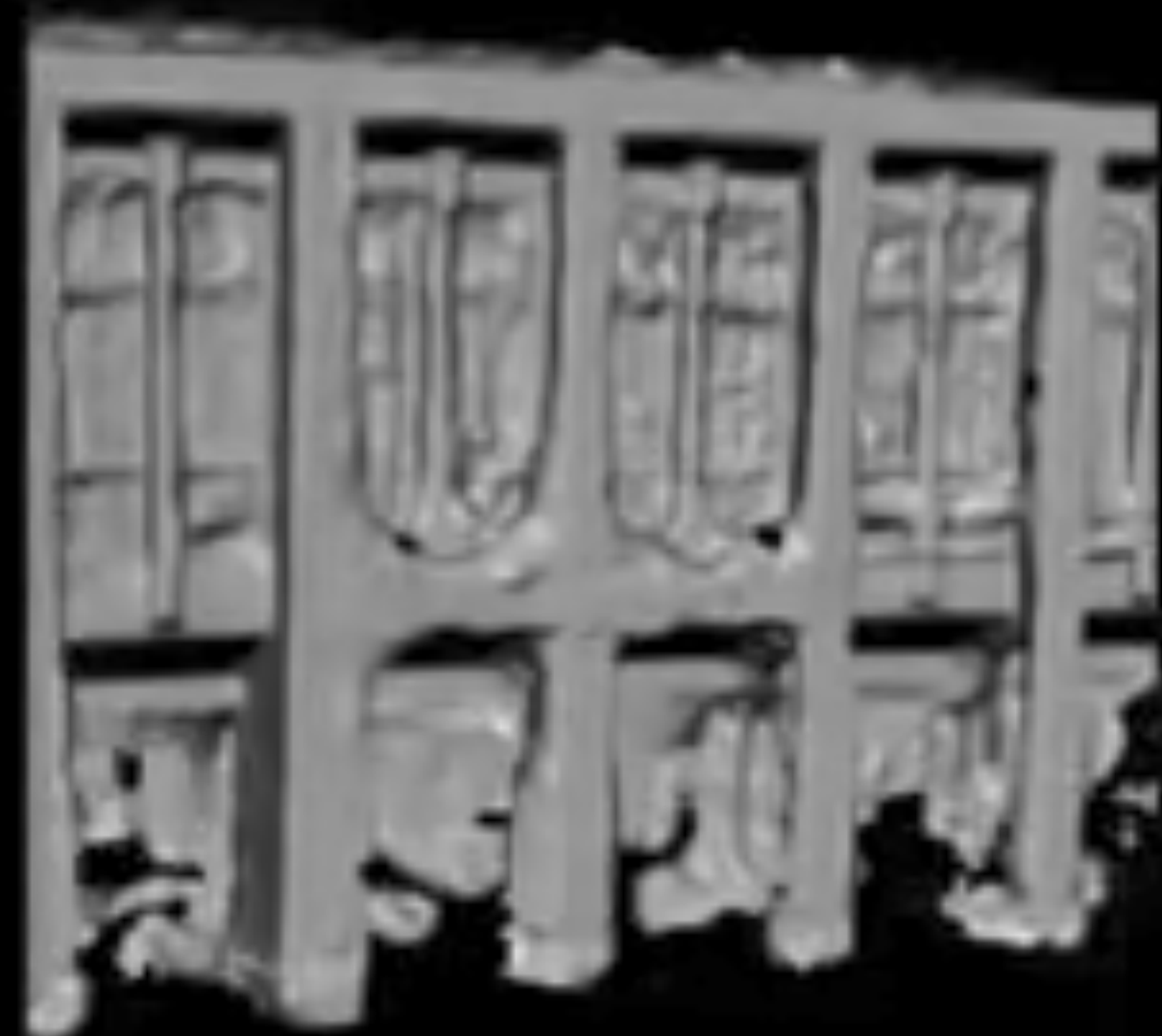
Depth normal map

Mesh model

Texture mapped  
mesh model



PMVS+Poisson



Our Result

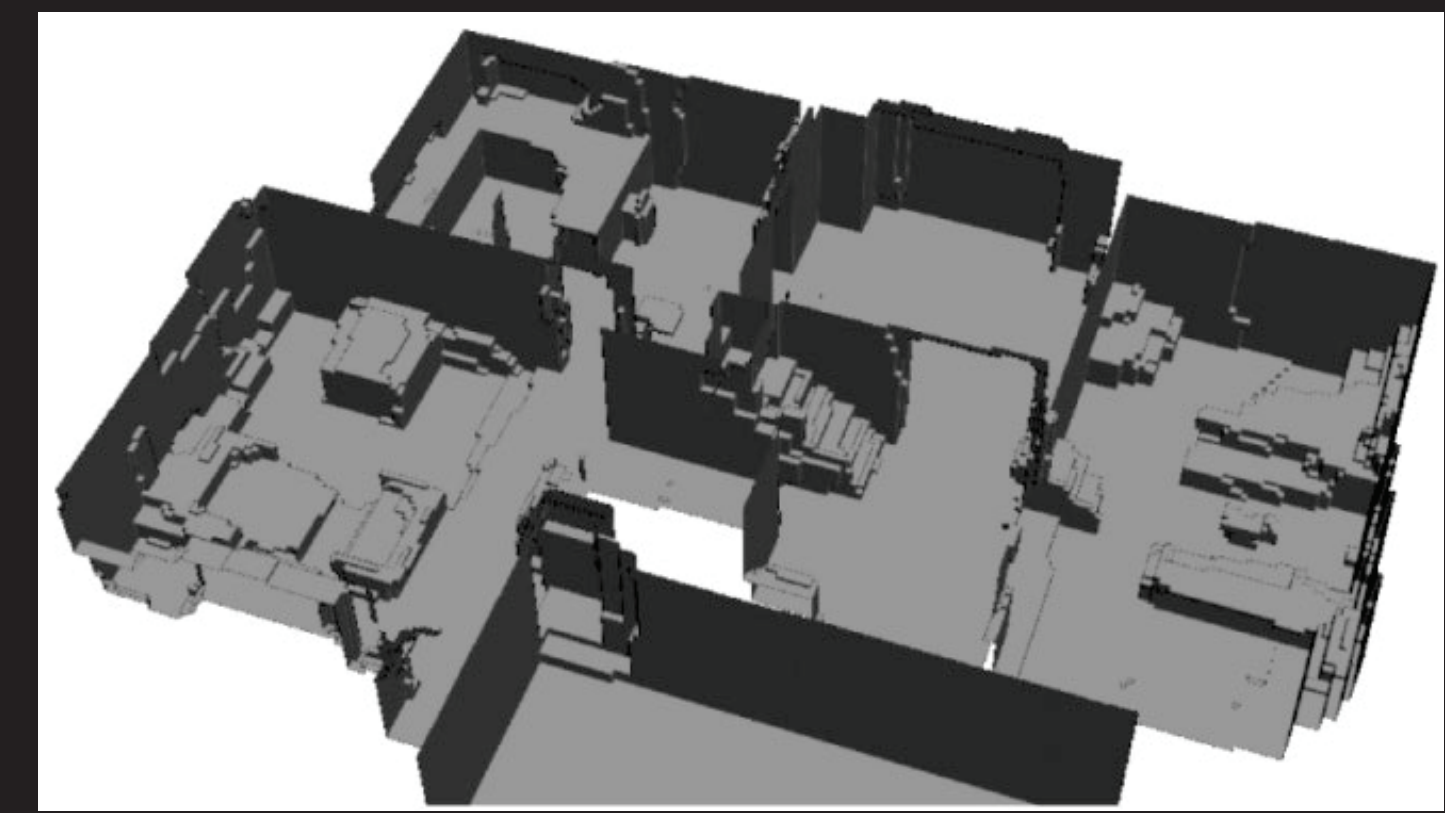


Reconstructing building interiors from images  
Yasutaka Furukawa, Rick Szeliski, Brian Curless, and Steve Seitz  
International Conference on Computer Vision 2009

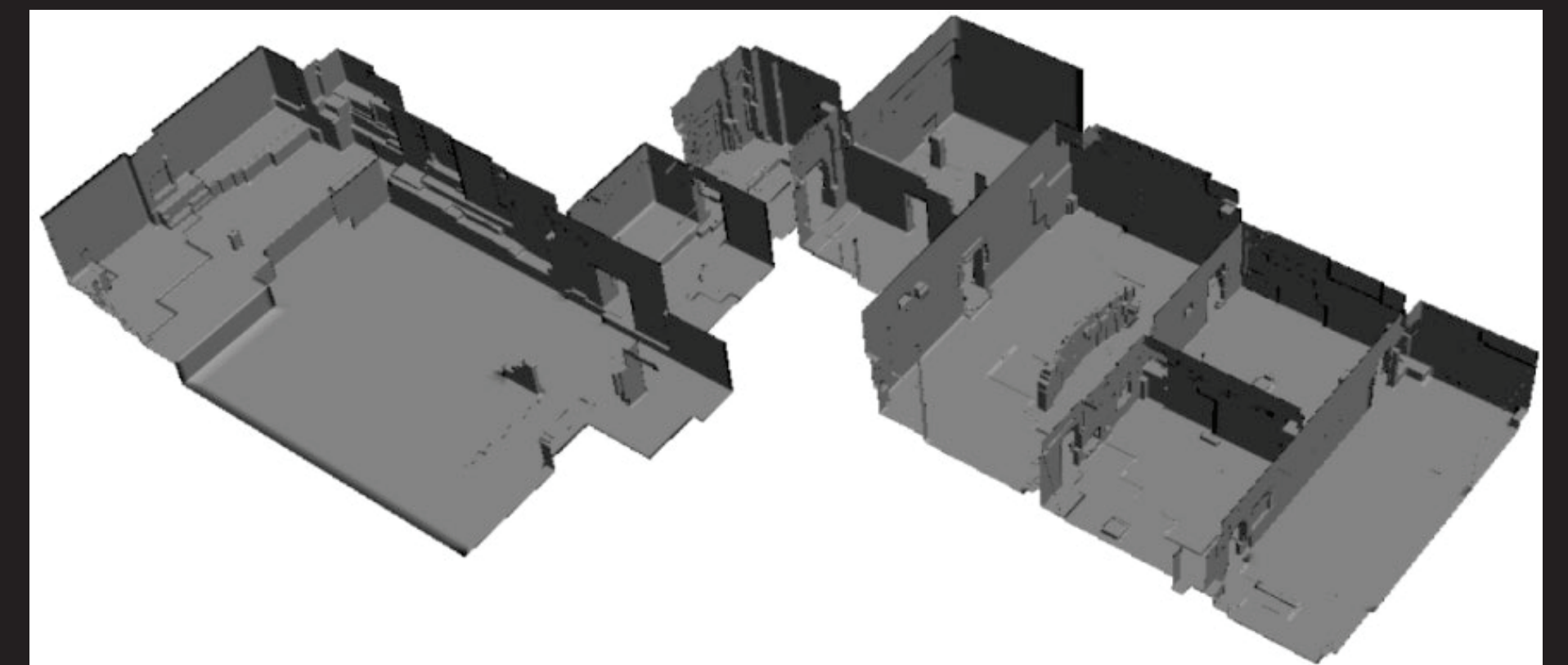
*Kitchen* - 22 images



*house* - 148 images



*gallery* - 492 images



## Reconstructing Building Interiors from Images

Yasutaka Furukawa, Brian Curless, Steven M. Seitz  
University of Washington, Seattle, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research, Redmond, USA  
szeliski@microsoft.com

### Abstract

This paper proposes a fully automated 3D reconstruction and visualization system for architectural scenes (interiors and exteriors). The reconstruction of indoor environments from photographs is particularly challenging due to texture-poor planar surfaces such as uniformly-painted walls. Our system first uses structure-from-motion, multi-view stereo, and a stereo algorithm specifically designed for



Figure 1: Floor plan and photograph of a house interior.

## Manhattan-world Stereo

Yasutaka Furukawa Brian Curless Steven M. Seitz  
Department of Computer Science & Engineering  
University of Washington, USA  
{furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research  
Redmond, USA  
szeliski@microsoft.com

### Abstract

Multi-view stereo (MVS) algorithms now produce reconstructions that rival laser range scanner accuracy. However, stereo algorithms require textured surfaces, and therefore work poorly for many architectural scenes (e.g., building interiors with textureless, painted walls). This paper presents a novel MVS approach to overcome these limi-



Figure 1. Increasingly ubiquitous on the Internet are images of architectural scenes with texture-poor but highly structured surfaces.

## Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu  
Washington University in St. Louis  
chenliu@wustl.edu  
Pushmeet Kohli†  
DeepMind  
pushmeet@google.com

Jiajun Wu  
Massachusetts Institute of Technology  
jiajunwu@mit.edu  
Yasutaka Furukawa\*  
Simon Fraser University  
furukawa@sfu.ca

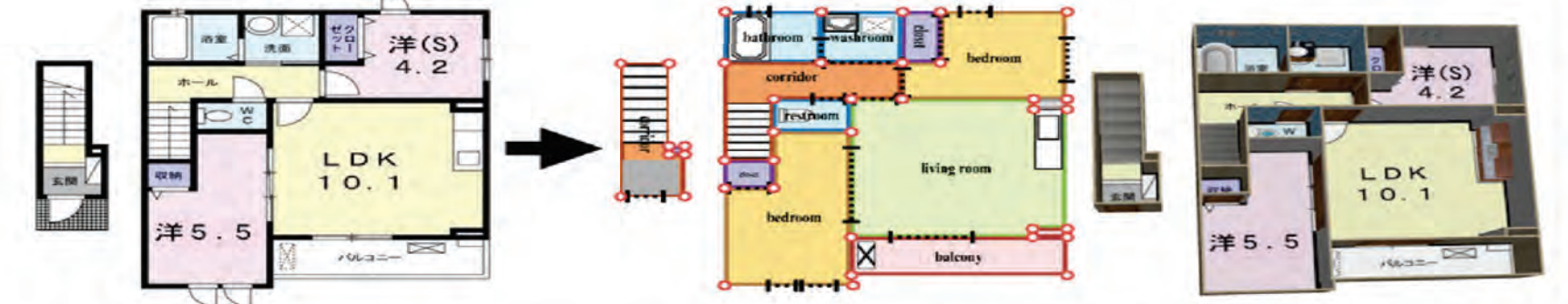


Figure 1: This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a popup 3D model.

## FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans

Chen Liu\* Jiaye Wu\*  
Washington University in St. Louis  
{chenliu, jiaye.wu}@wustl.edu

Yasutaka Furukawa  
Simon Fraser University  
furukawa@sfu.ca

**Abstract.** The ultimate goal of this indoor mapping research is to automatically reconstruct a floorplan simply by walking through a house with

Sensing

Ice age

Revolution

Perception

Ice age

?Revolution?

1985

1990

1995

2000

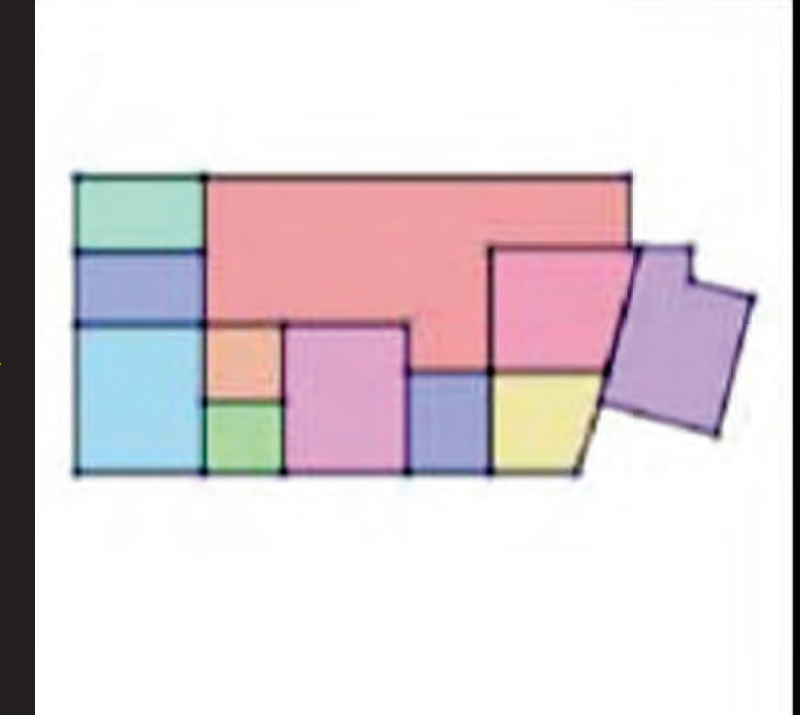
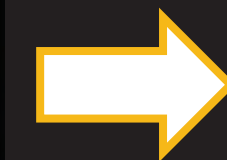
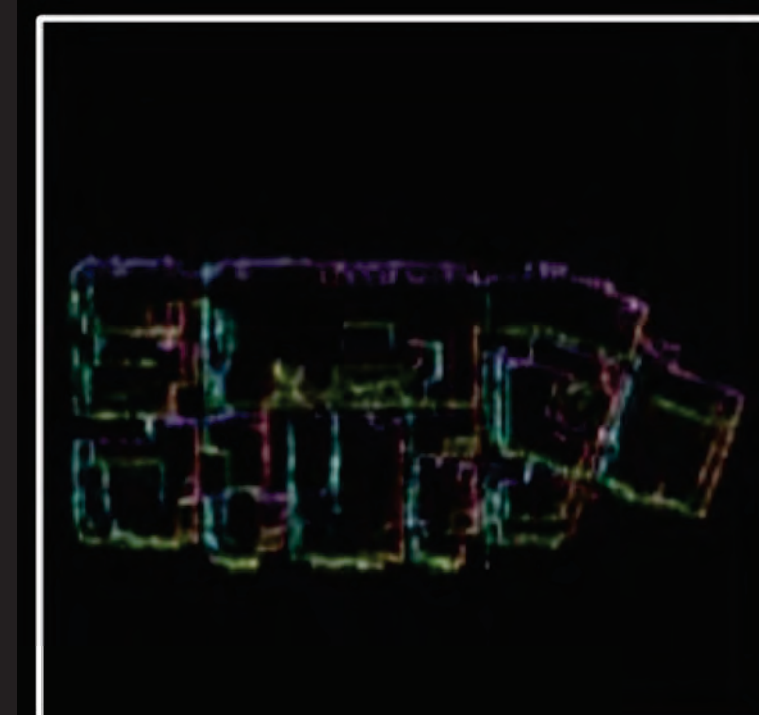
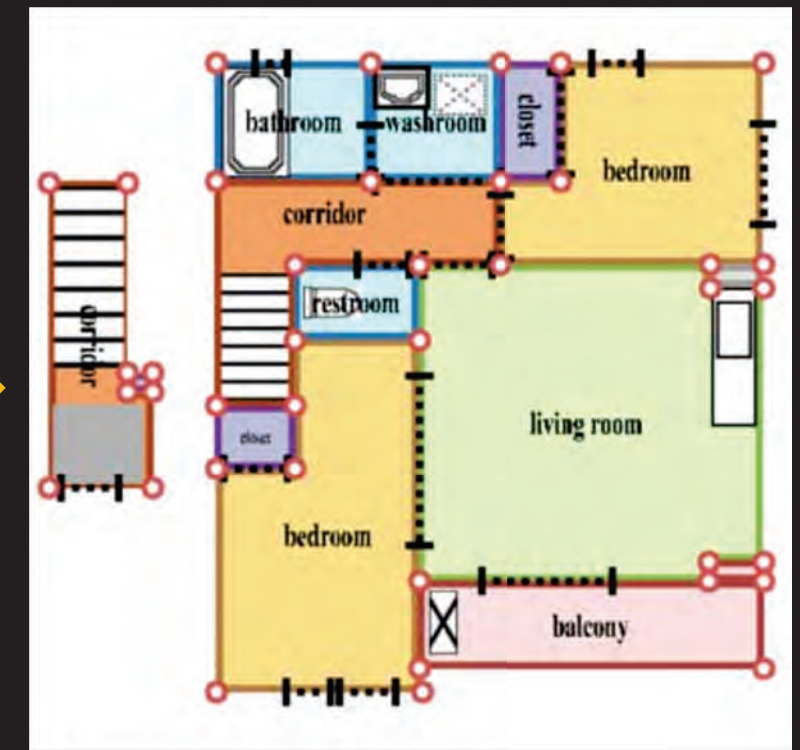
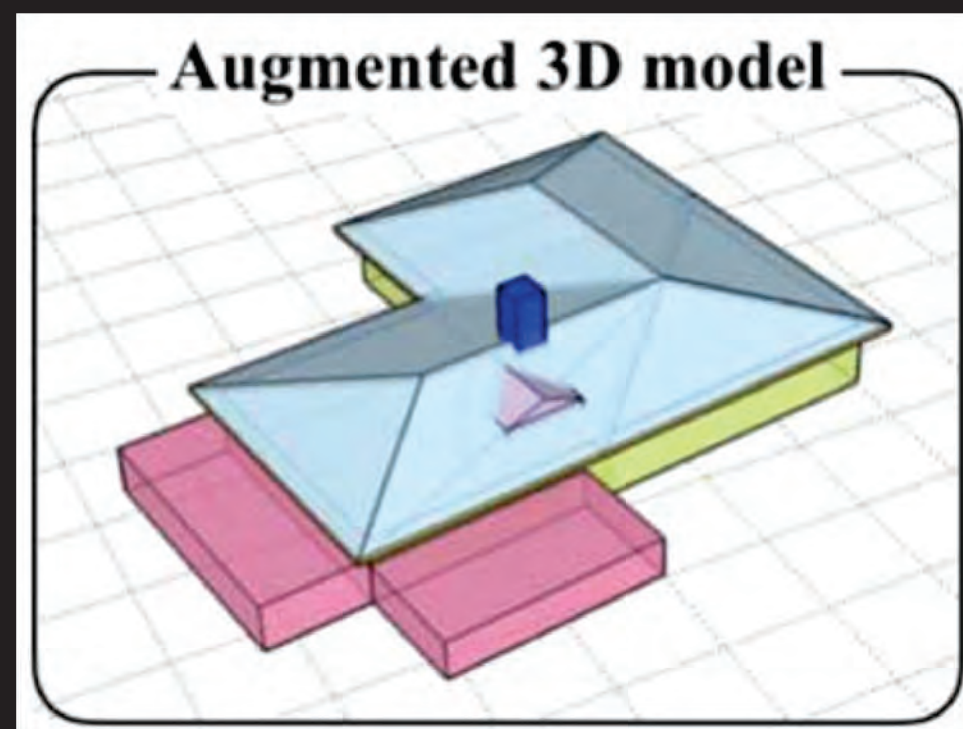
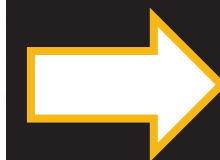
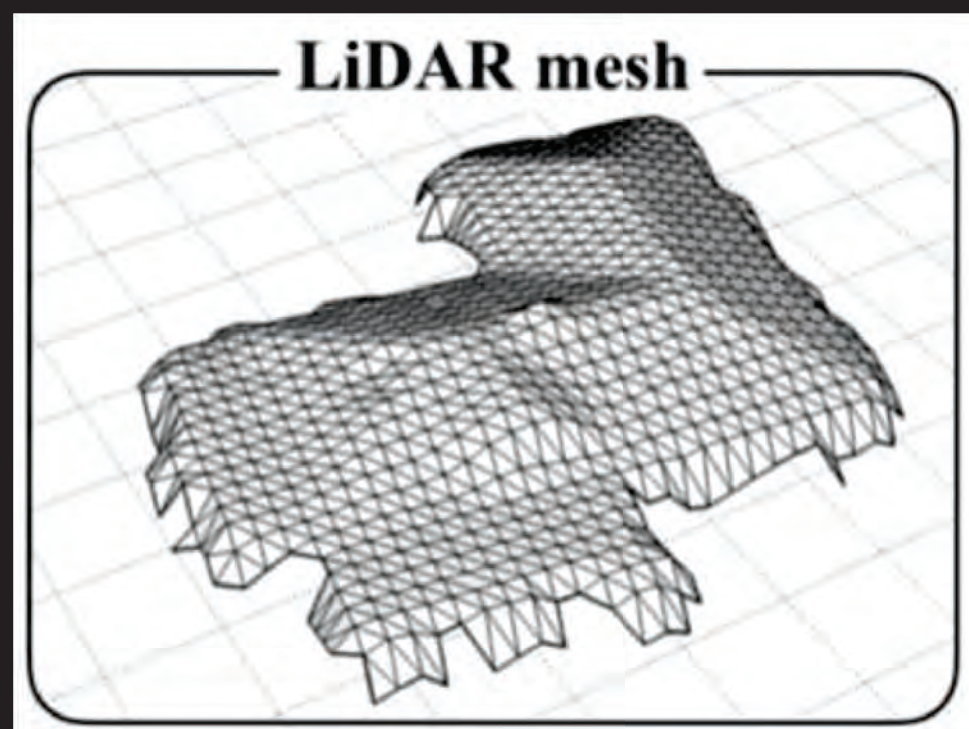
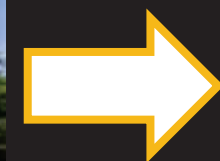
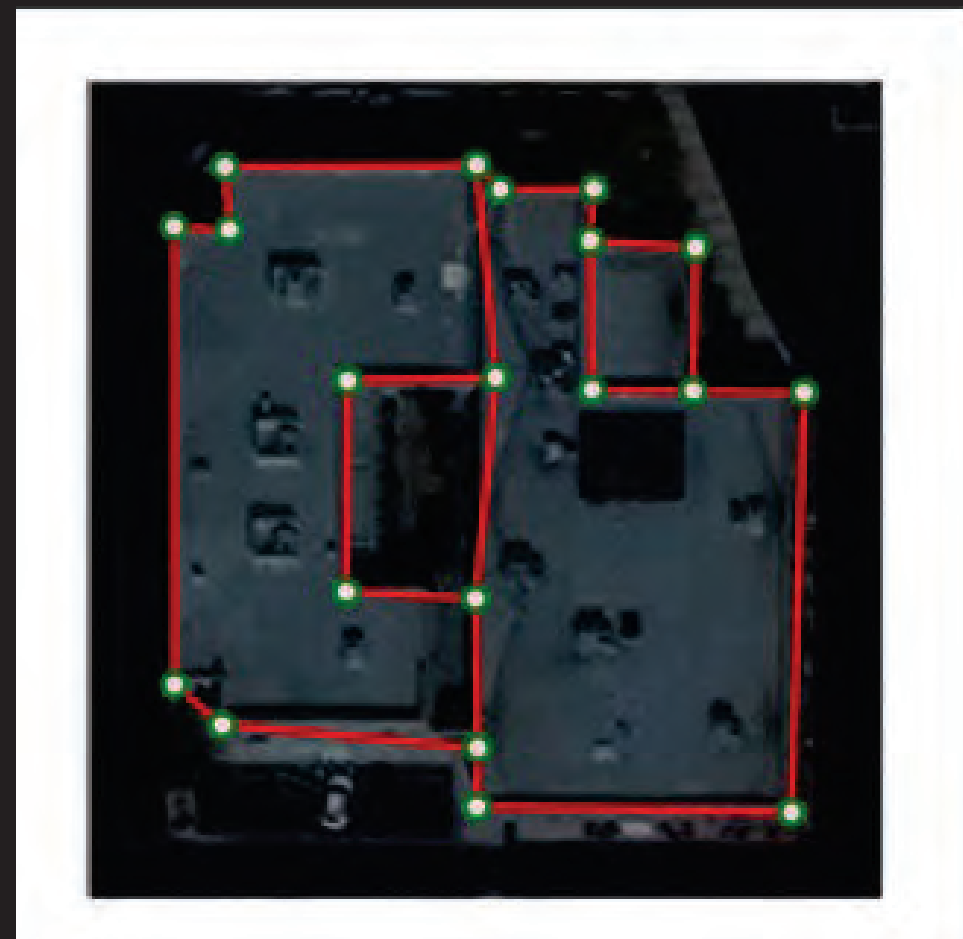
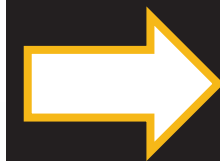
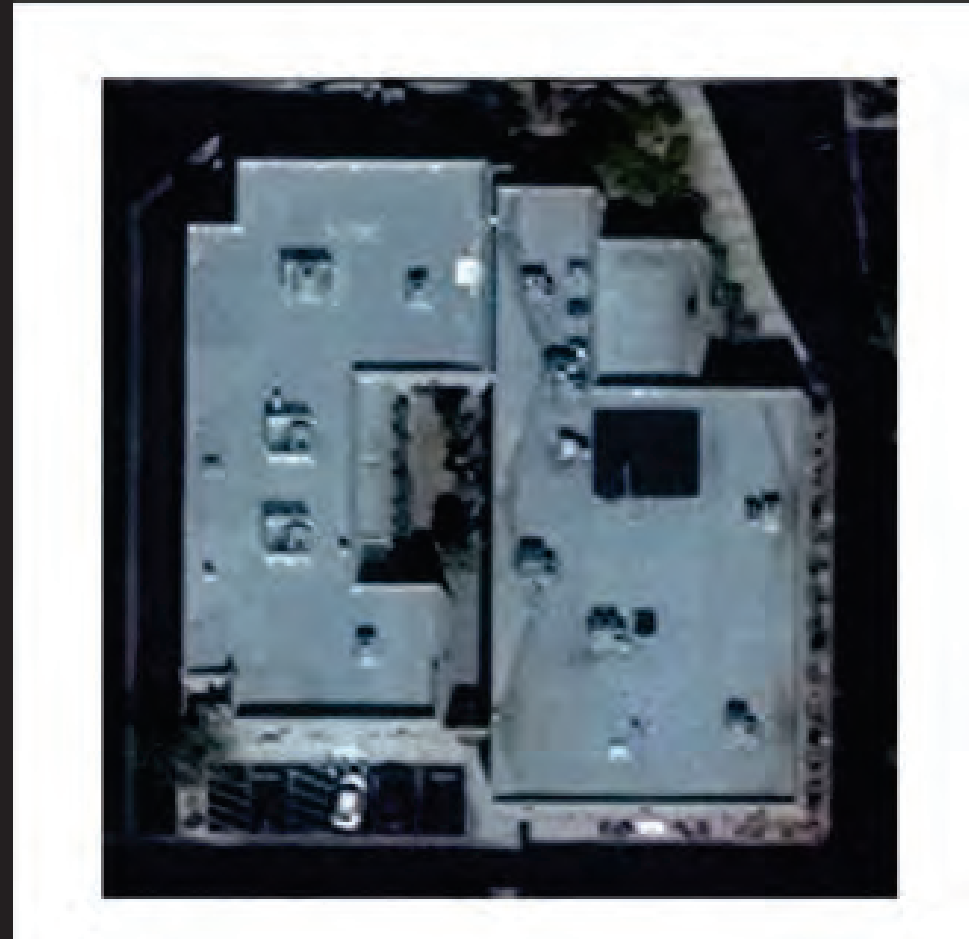
2005

2010

2015

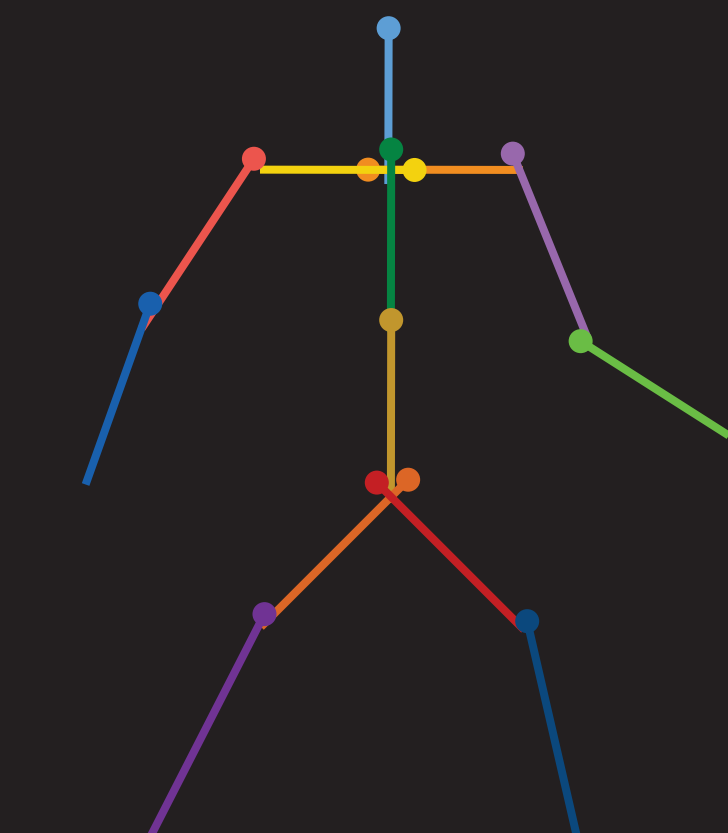
2020

# Geometry perception

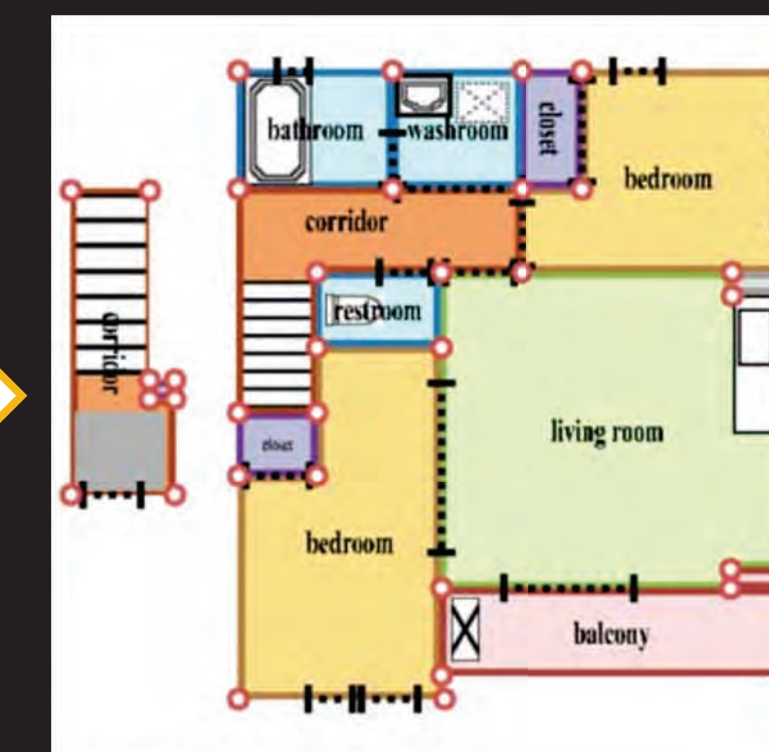




# Why is this hard?

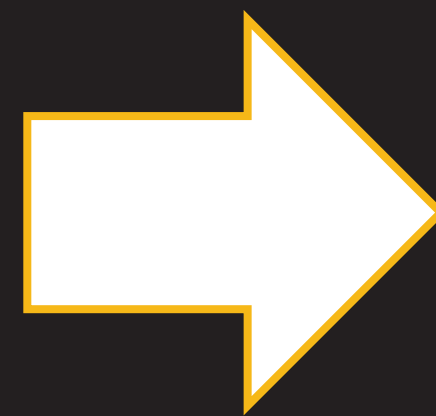


**Fixed topology**

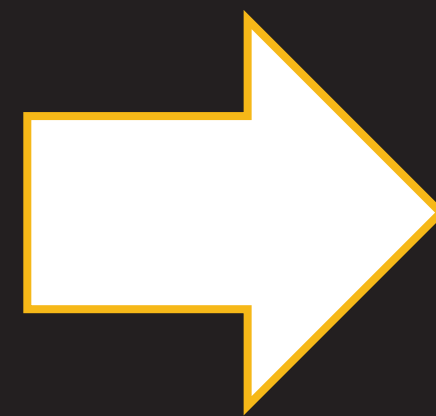


**Arbitrary topology**

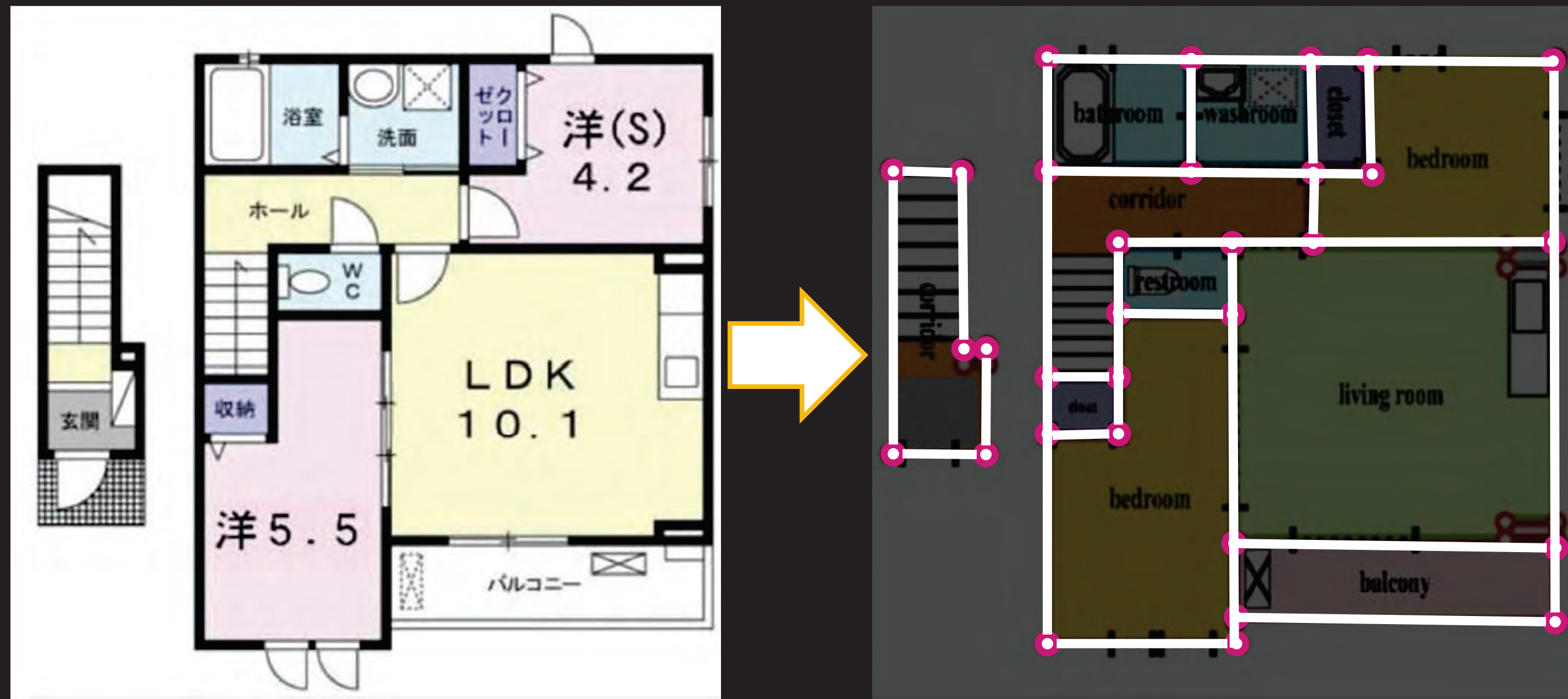
# Planar graph inference



# Planar graph inference

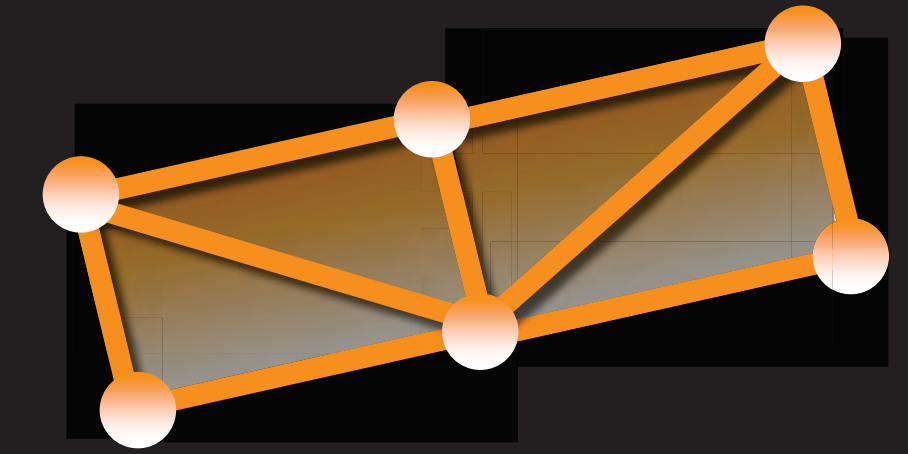


# Planar graph inference

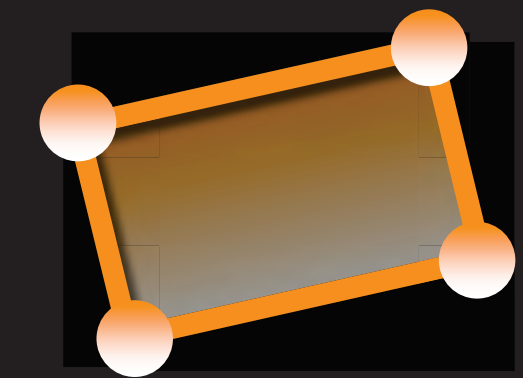


## Geometric Elements

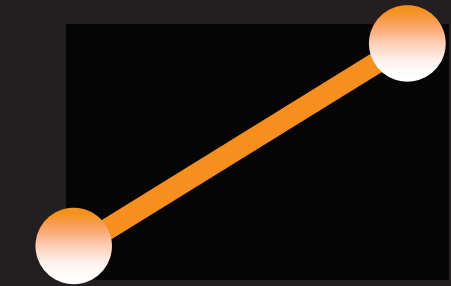
Graph



2D primitive

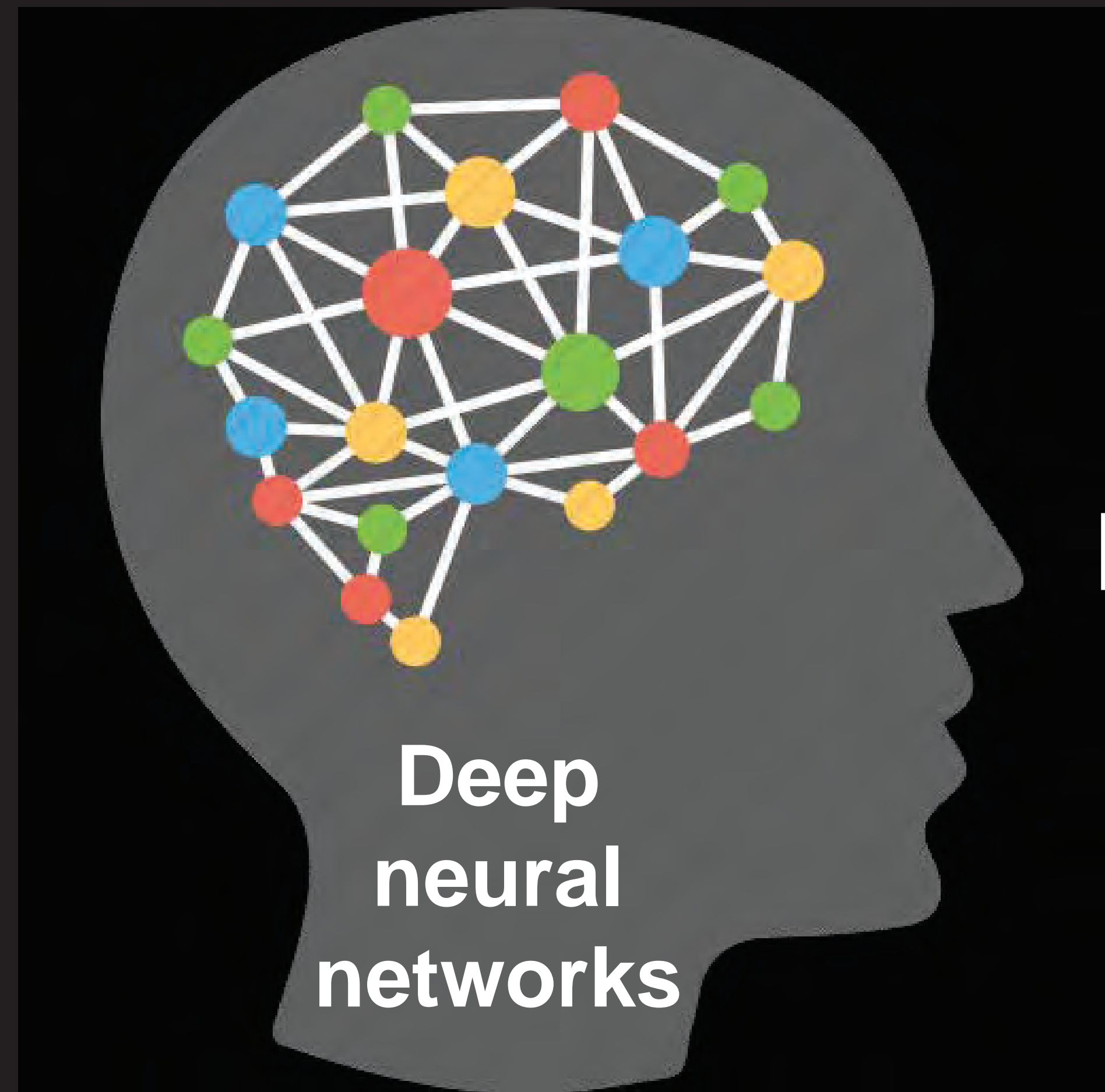
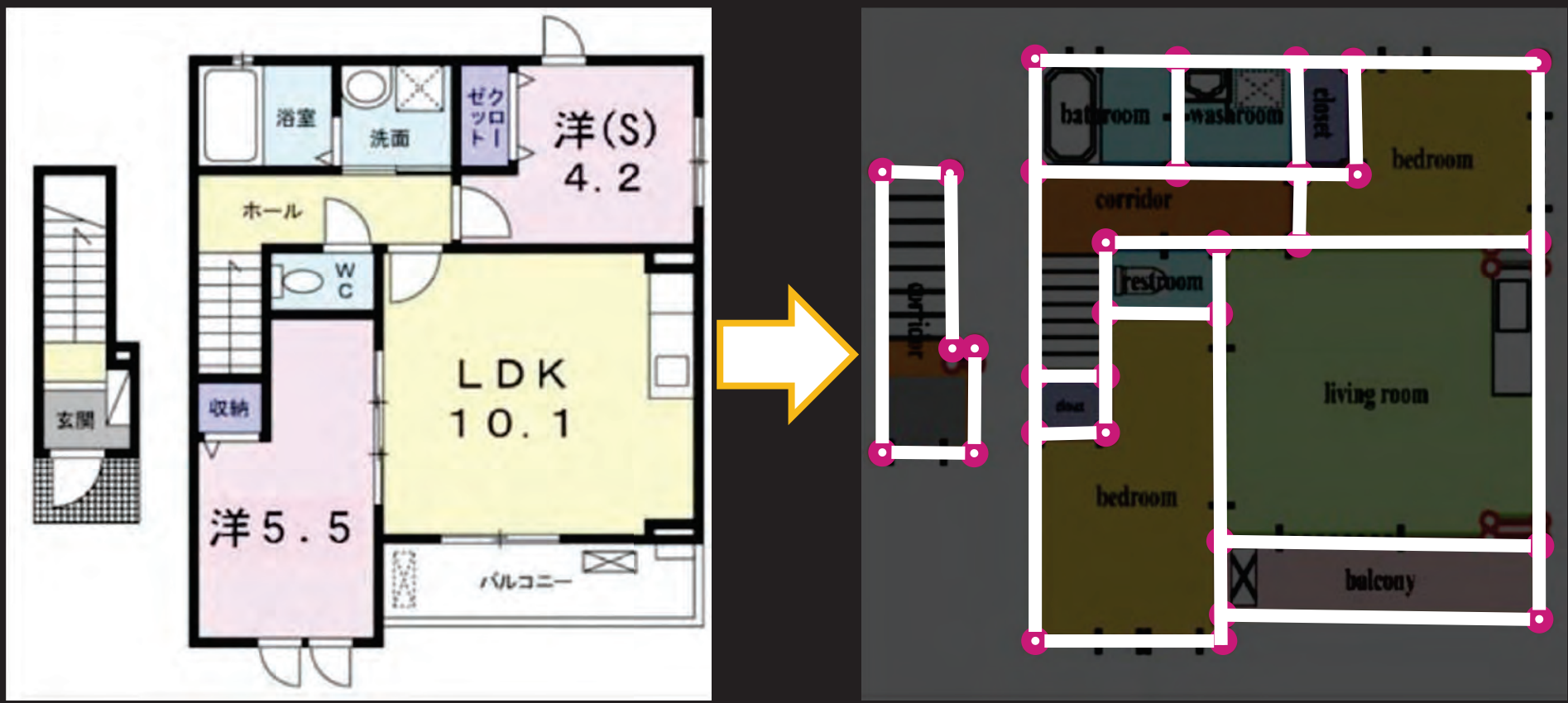


1D primitive

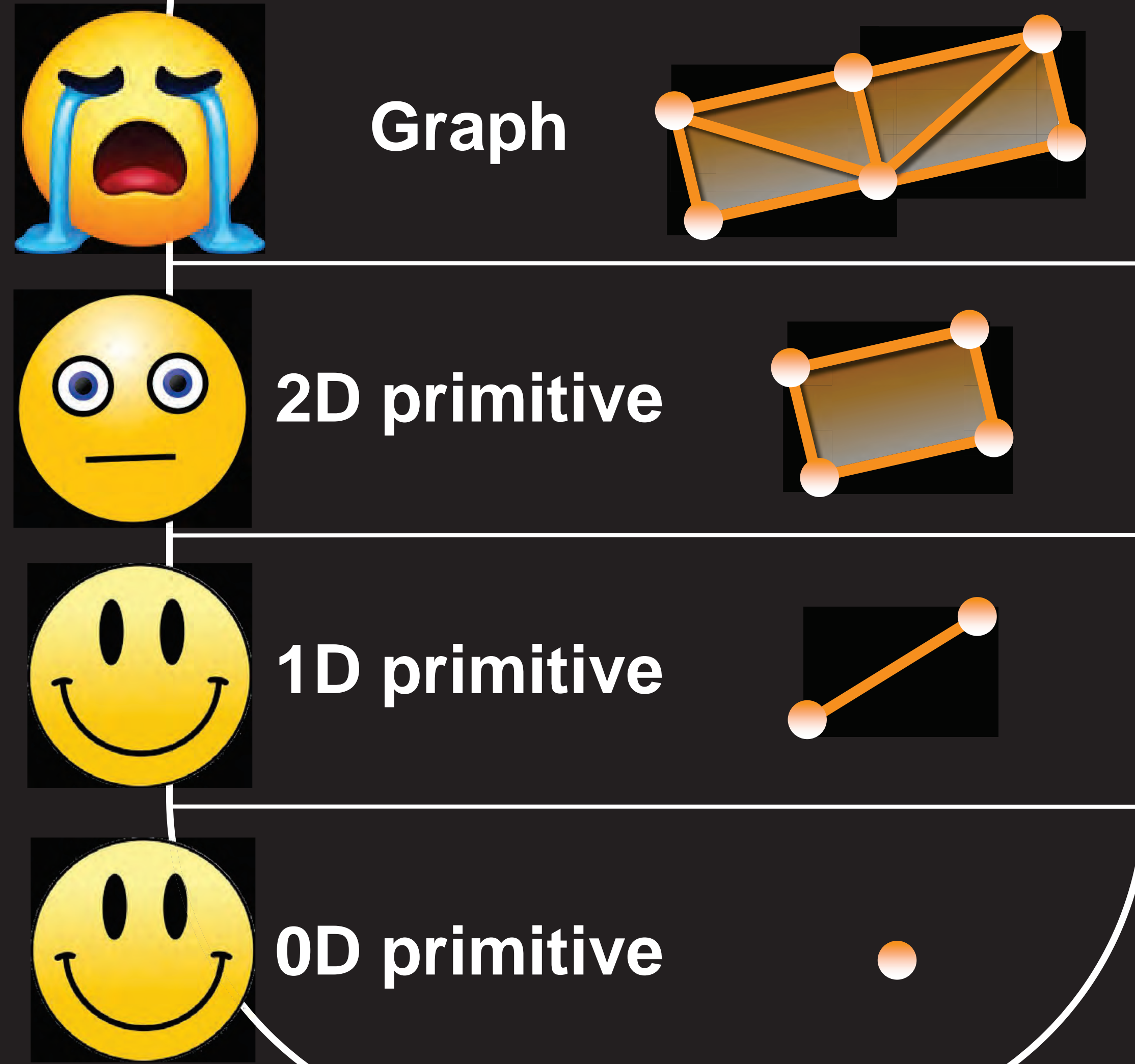


0D primitive

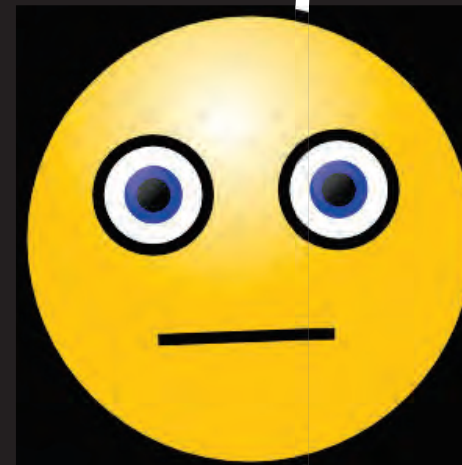
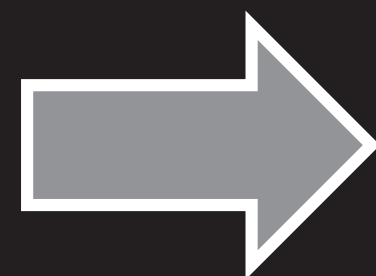




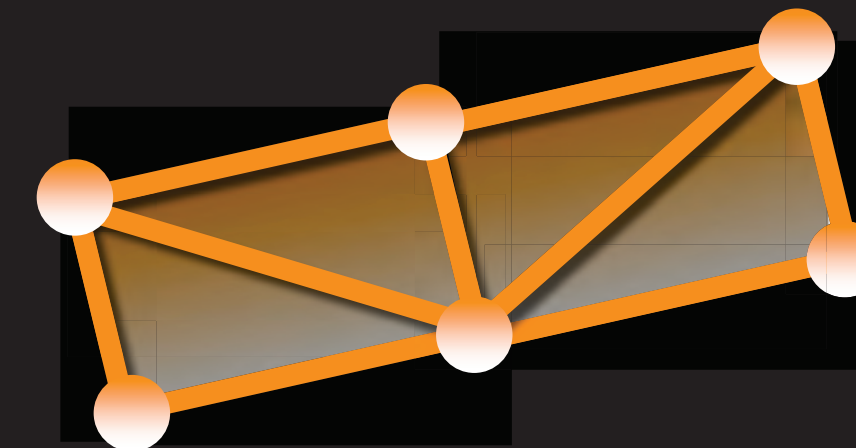
# Geometric Elements



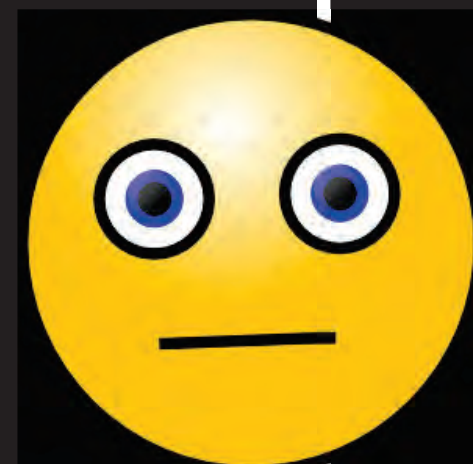
**Optimization  
(Integer Programming)**



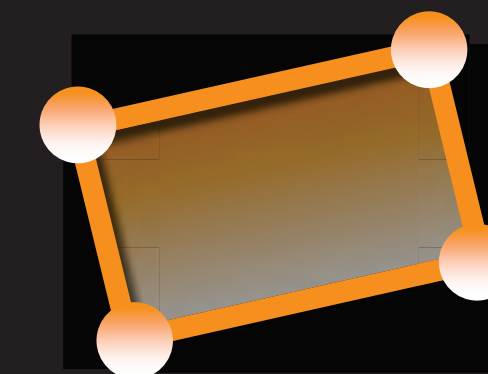
**Graph**



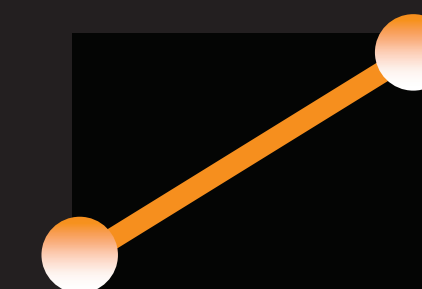
**Geometric Elements**



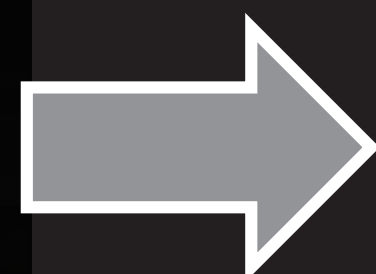
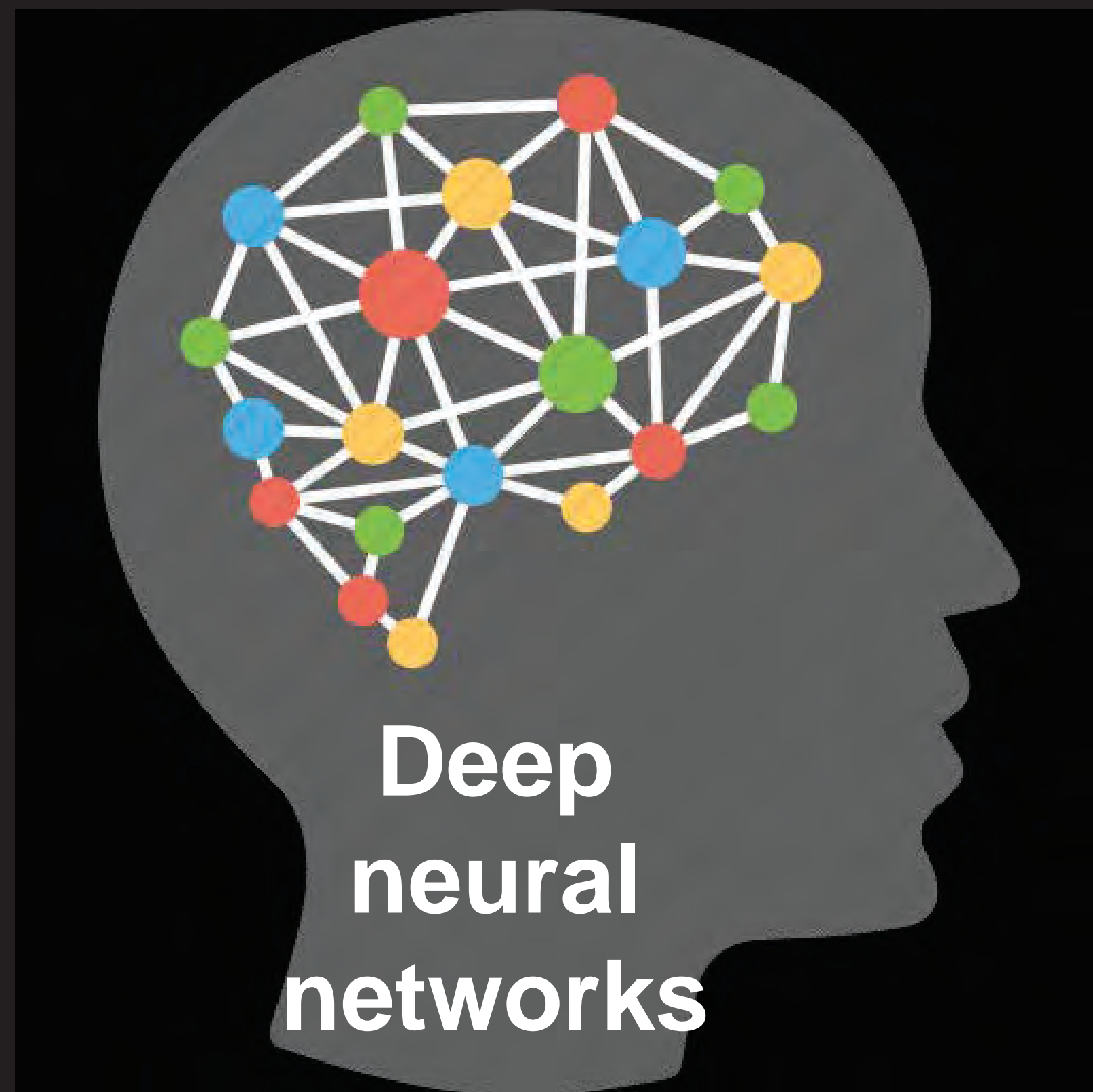
**2D primitive**



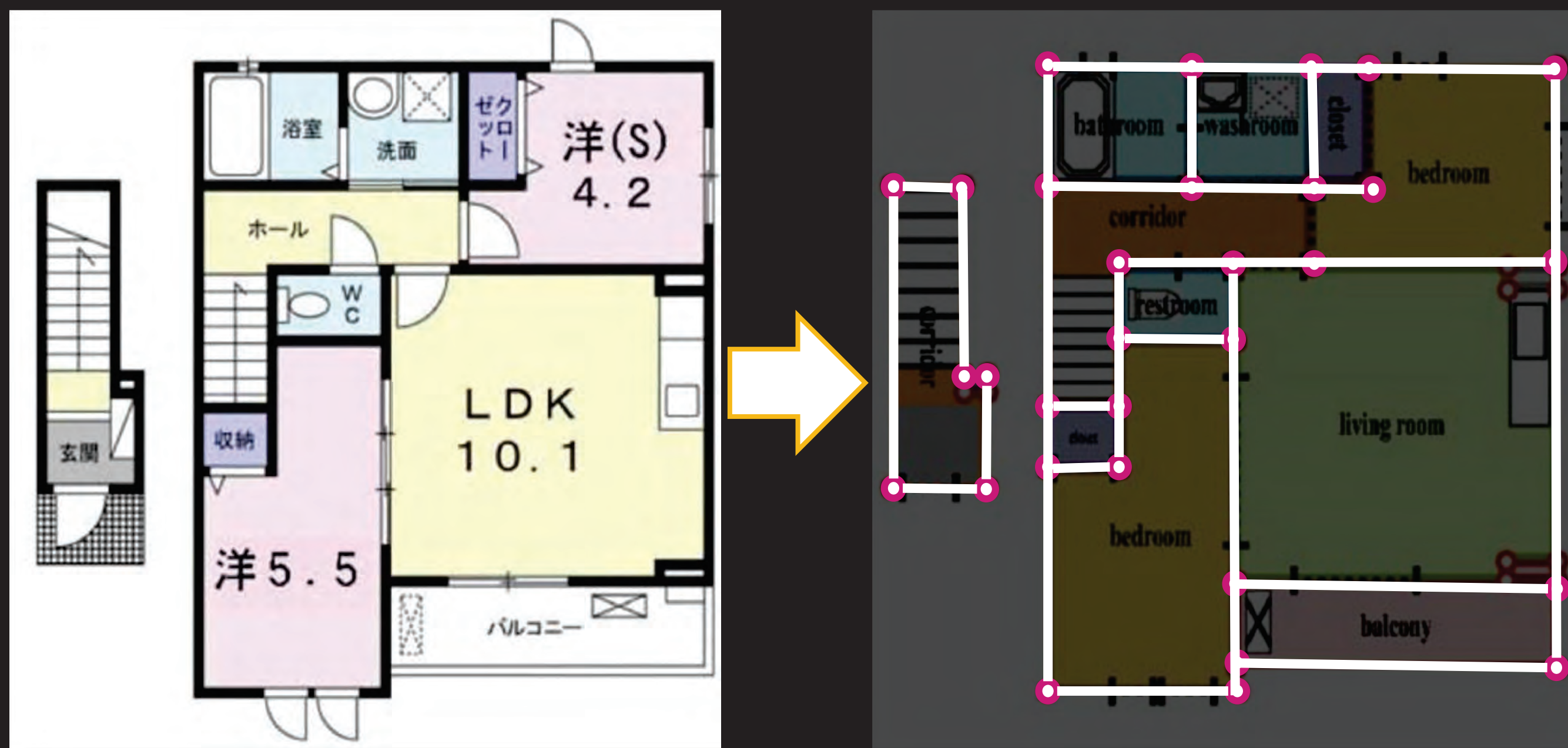
**1D primitive**



**0D primitive**



# Planar graph inference



## Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu  
Washington University in St. Louis  
chenliu@wustl.edu

Pushmeet Kohli<sup>†</sup>  
DeepMind  
pushmeet@google.com

Jiajun Wu  
Massachusetts Institute of Technology  
jiajunwu@mit.edu

Yasutaka Furukawa\*  
Simon Fraser University  
furukawa@sfu.ca

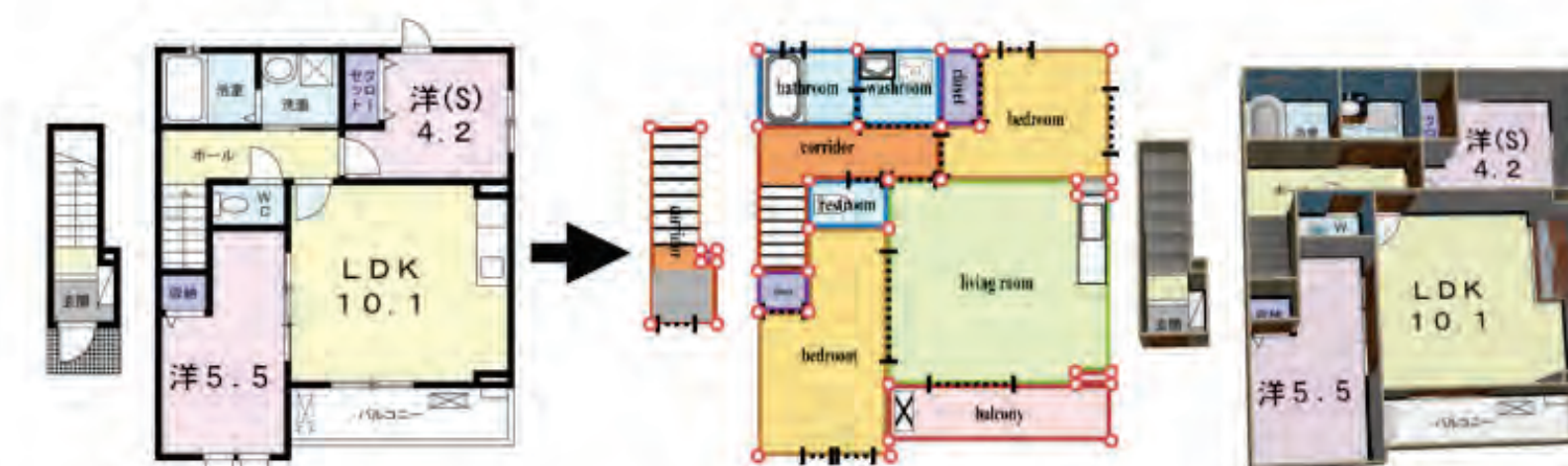


Figure 1: This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a popup 3D model.

### Abstract

This paper addresses the problem of converting a rasterized floorplan image into a vector-graphics representation. Unlike existing approaches that rely on a sequence of low-level image processing heuristics, we adopt a learning-based approach. A neural architecture first transforms a rasterized image to a set of junctions that represent low-level geometric and semantic information (e.g., wall corners or door end-points). Integer programming is then formulated to aggregate junctions into a set of simple primitives (e.g., wall lines, door lines, or icon boxes) to produce a vectorized floorplan, while ensuring a topologically and geometrically consistent result. Our algorithm significantly outperforms existing methods and achieves around 90% precision and recall, getting to the range of production-ready performance. The vector representation allows 3D model popup for better indoor scene visualization, direct model manipulation for architectural remodeling, and further computational applications such as data analysis. Our system is efficient: we have

converted hundred thousand production-level floorplan images into the vector representation and generated 3D popup models.

### 1. Introduction

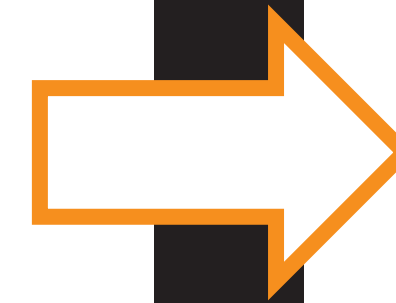
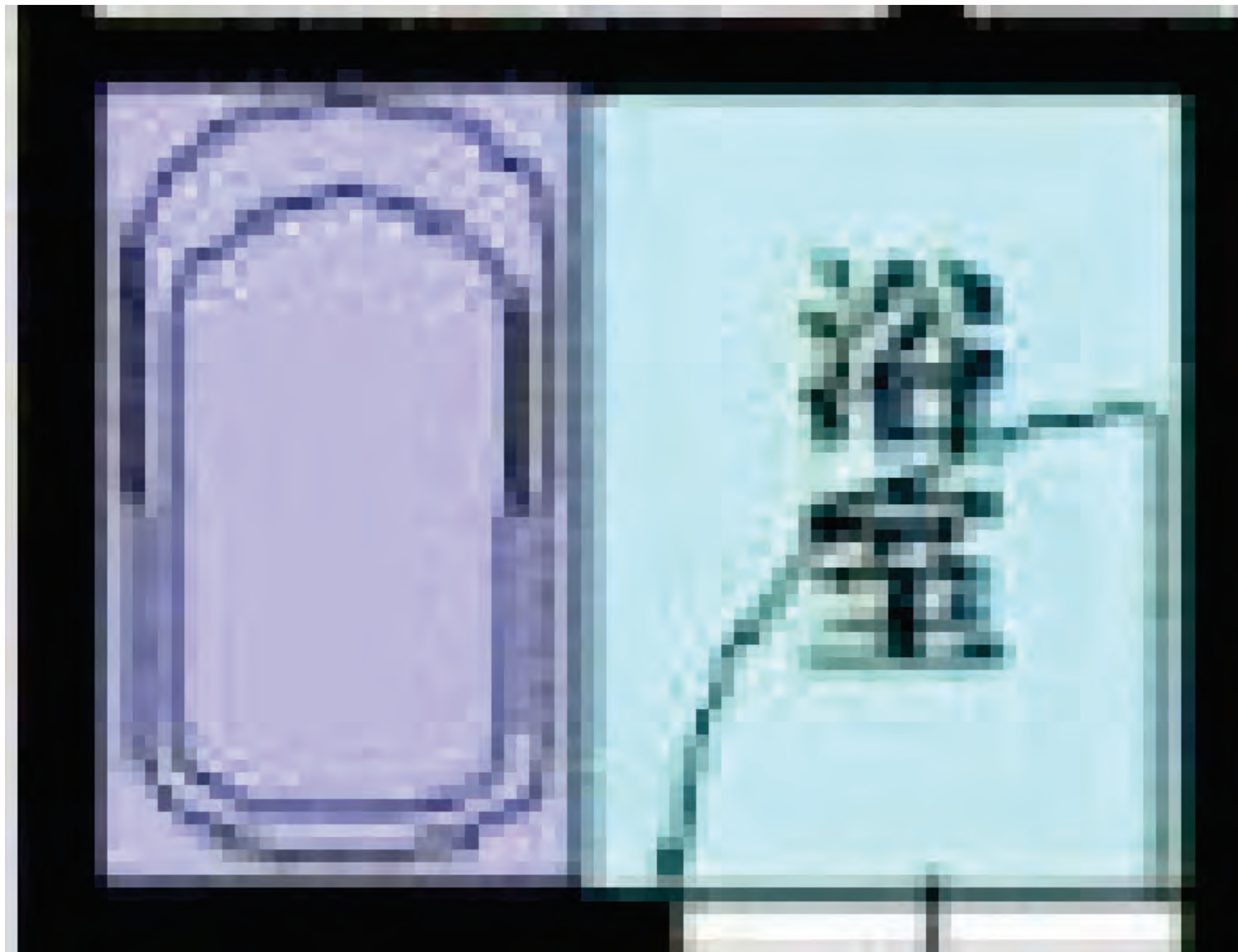
Architectural floorplans play a crucial role in designing, understanding, or remodeling indoor spaces. Their drawings are very effective in conveying geometric and semantic information of a scene. For instance, we can quickly identify room extents, the locations of doors, or object arrangements (geometry). We can also recognize the types of rooms, doors, or objects easily through texts or icon styles (semantics).

While professional architects or designers draw floorplans in a vector-graphics representation using software such as AutoCAD [1], HomeStyler [3], or Sketchup [5], the final use of an artwork is often just visualization for clients (e.g., home buyers or renters). As a result, floorplans are rasterized to print or digital media for publication. This process discards all the structured geometric and semantic information, limiting human post-processing or further computing capabilities such as model analysis, synthesis, or modification.

\*At Washington University in St. Louis at the time of the project.

<sup>†</sup>At Microsoft Research Redmond at the time of the project.

# Toy example





# Corner detection

## Convolutional Pose Machines

Shih-En Wei    Varun Ramakrishna    Takeo Kanade    Yaser Sheikh  
 shihenw@cmu.edu    vramakri@cs.cmu.edu    Takeo.Kanade@cs.cmu.edu    yaser@cs.cmu.edu

The Robotics Institute  
 Carnegie Mellon University

### Abstract

*Pose Machines provide a sequential prediction framework for learning rich implicit spatial models. In this work we show a systematic design for how convolutional networks can be incorporated into the pose machine framework for learning image features and image-dependent spatial models for the task of pose estimation. The contribution of this paper is to implicitly model long-range dependencies between variables in structured prediction tasks such as articulated pose estimation. We achieve this by designing a sequential architecture composed of convolutional networks that directly operate on belief maps from previous stages, producing increasingly refined estimates for part locations, without the need for explicit graphical model-style inference. Our approach addresses the characteristic difficulty of vanishing gradients during training by providing a natural learning objective function that enforces intermediate supervision, thereby replenishing back-propagated gradients and conditioning the learning procedure. We demonstrate state-of-the-art performance and outperform competing methods on standard benchmarks including the MPII, LSP, and FLIC datasets.*



**Figure 1:** A Convolutional Pose Machine consists of a sequence of predictors trained to make dense predictions at each image location. Here we show the increasingly refined estimates for the location of the right elbow in each stage of the sequence. (a) Predicting from local evidence often causes confusion. (b) Multi-part context helps resolve ambiguity. (c) Additional iterations help converge to a certain solution.

of each part. At each stage in a CPM, image features and the belief maps produced by the previous stage are used as input. The belief maps provide the subsequent stage an expressive non-parametric encoding of the spatial uncertainty of location for each part, allowing the CPM to learn rich image-dependent spatial models of the relationships between parts. Instead of explicitly parsing such belief maps either using graphical models [28, 38, 39] or specialized post-processing steps [38, 40], we learn convolutional networks that directly operate on intermediate belief maps and learn implicit image-dependent spatial models of the relationships between parts. The overall proposed multi-stage architecture is fully differentiable and therefore can be trained in an end-to-end fashion using backpropagation.

At a particular stage in the CPM, the spatial context of part beliefs provide strong disambiguating cues to a subsequent stage. As a result, each stage of a CPM produces belief maps with increasingly refined estimates for the locations of each part (see Figure 1). In order to capture long-range interactions between parts, the design of the network in each stage of our sequential prediction framework is motivated by the goal of achieving a large receptive field on both the image and the belief maps. We find, through experiments, that large receptive fields on the belief maps are crucial for learning long range spatial relationships and re-

maps described are closely related to beliefs produced in message passing inference in graphical models. The overall architecture can be viewed as an unrolled mean-field message passing inference algorithm [31] that is learned end-to-end using backpropagation.

### 1. Introduction

We introduce *Convolutional Pose Machines (CPMs)* for the task of articulated pose estimation. CPMs inherit the benefits of the *pose machine* [29] architecture—the implicit learning of long-range dependencies between image and multi-part cues, tight integration between learning and inference, a modular sequential design—and combine them with the advantages afforded by convolutional architectures: the ability to learn feature representations for both image and spatial context directly from data; a differentiable architecture that allows for globally joint training with backpropagation; and the ability to efficiently handle large training datasets.

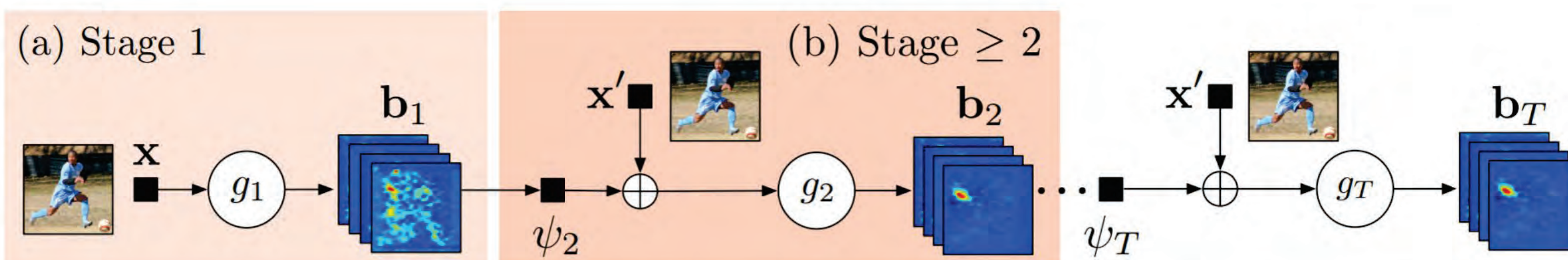
CPMs consist of a sequence of convolutional networks that repeatedly produce 2D belief maps<sup>1</sup> for the location

<sup>1</sup>We use the term *belief* in a slightly loose sense, however the belief

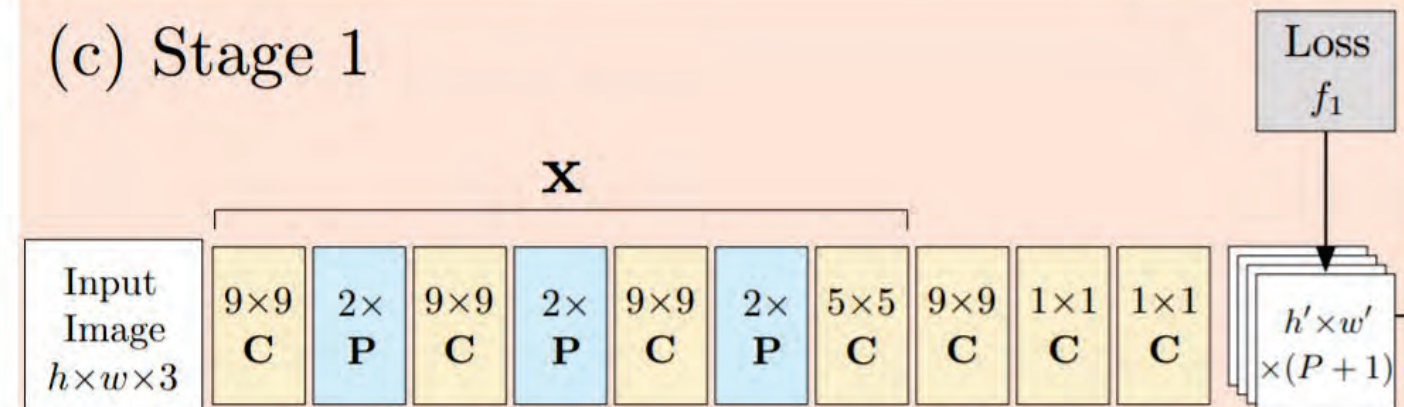
## Convolutional Pose Machines [ Wei, Ramakrishna, Kanade, Sheikh, 2016 ]

### Convolutional Pose Machines (T-stage)

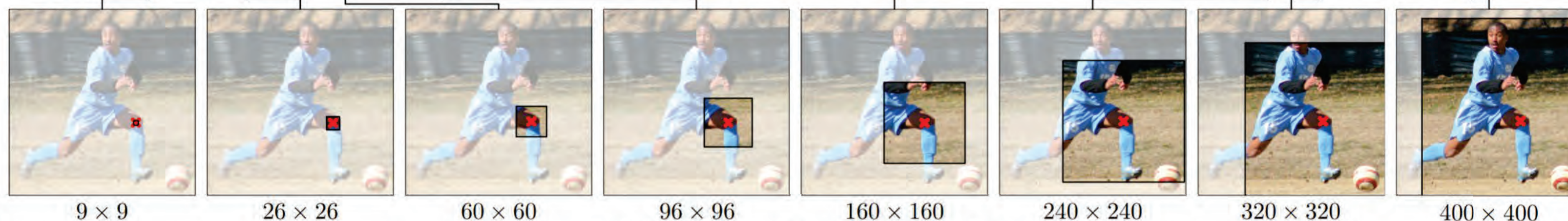
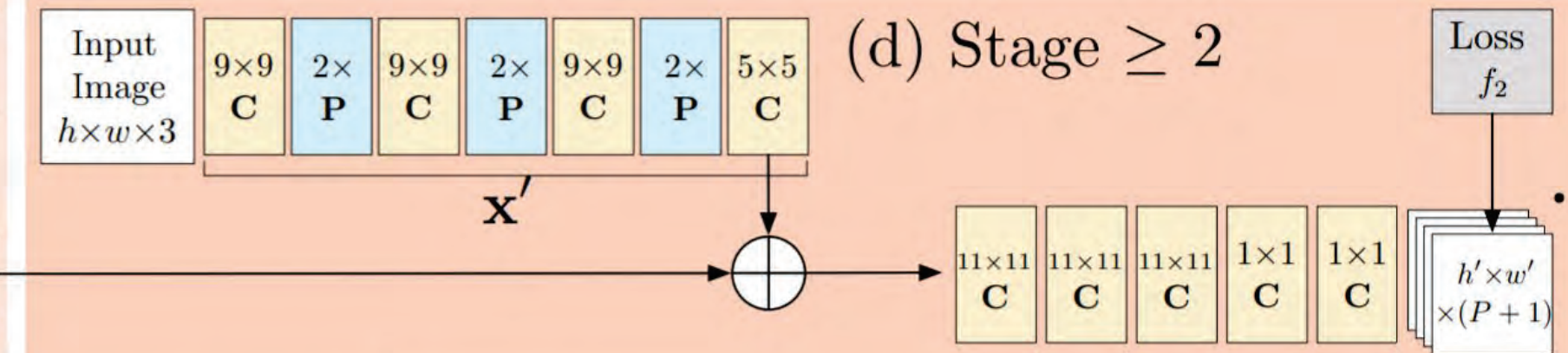
- P** Pooling
- C** Convolution



### (c) Stage 1

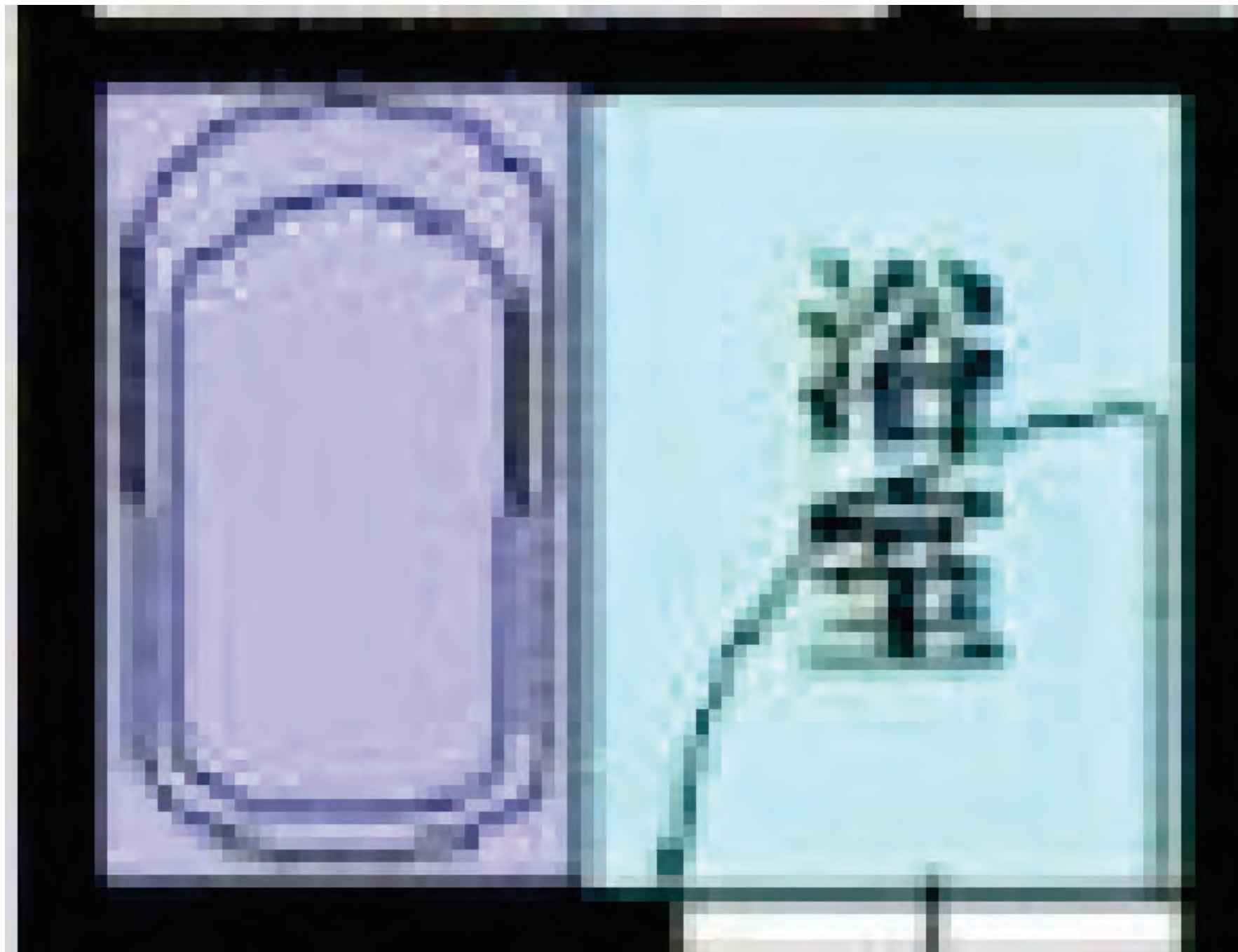


### (d) Stage >= 2

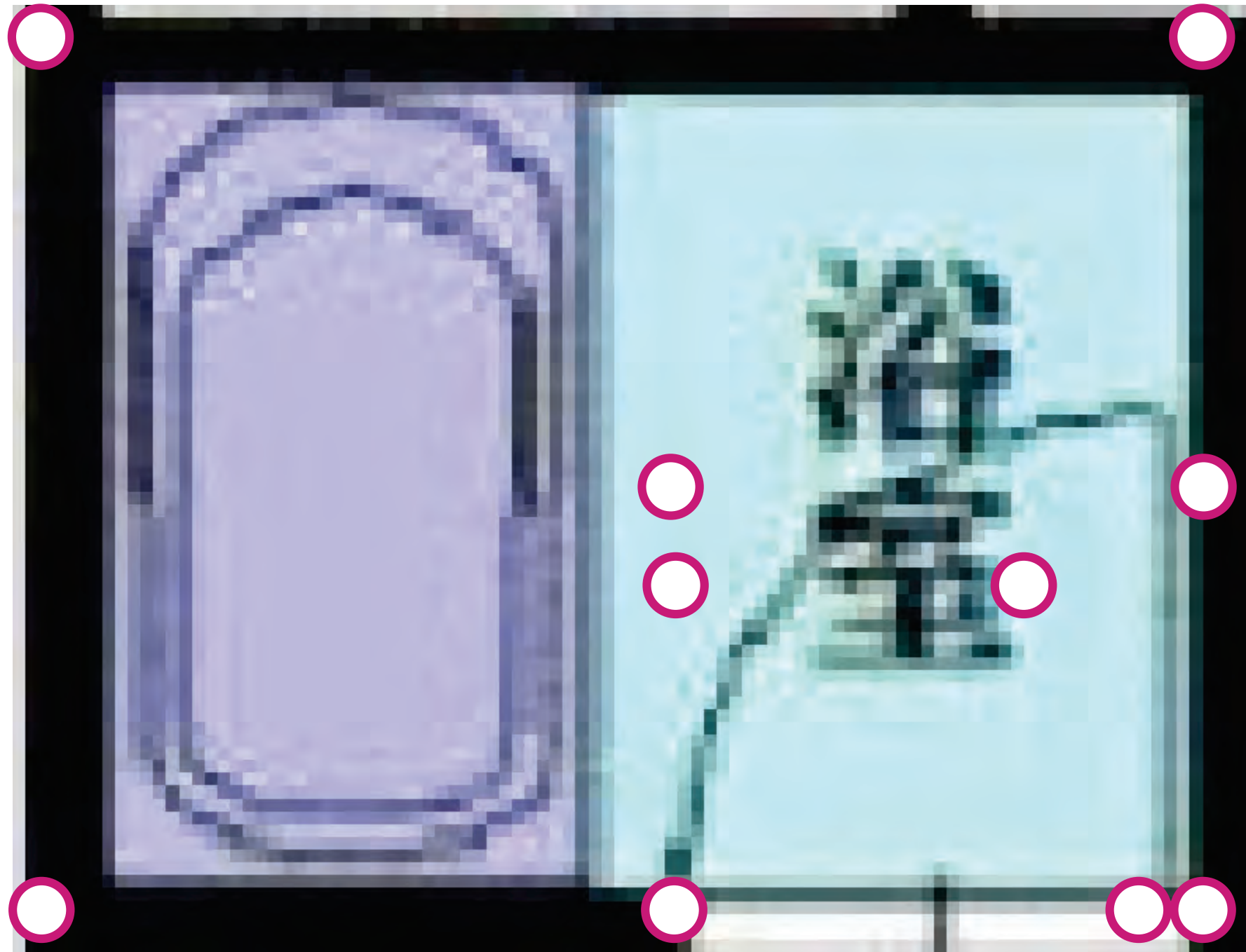


(e) Effective Receptive Field

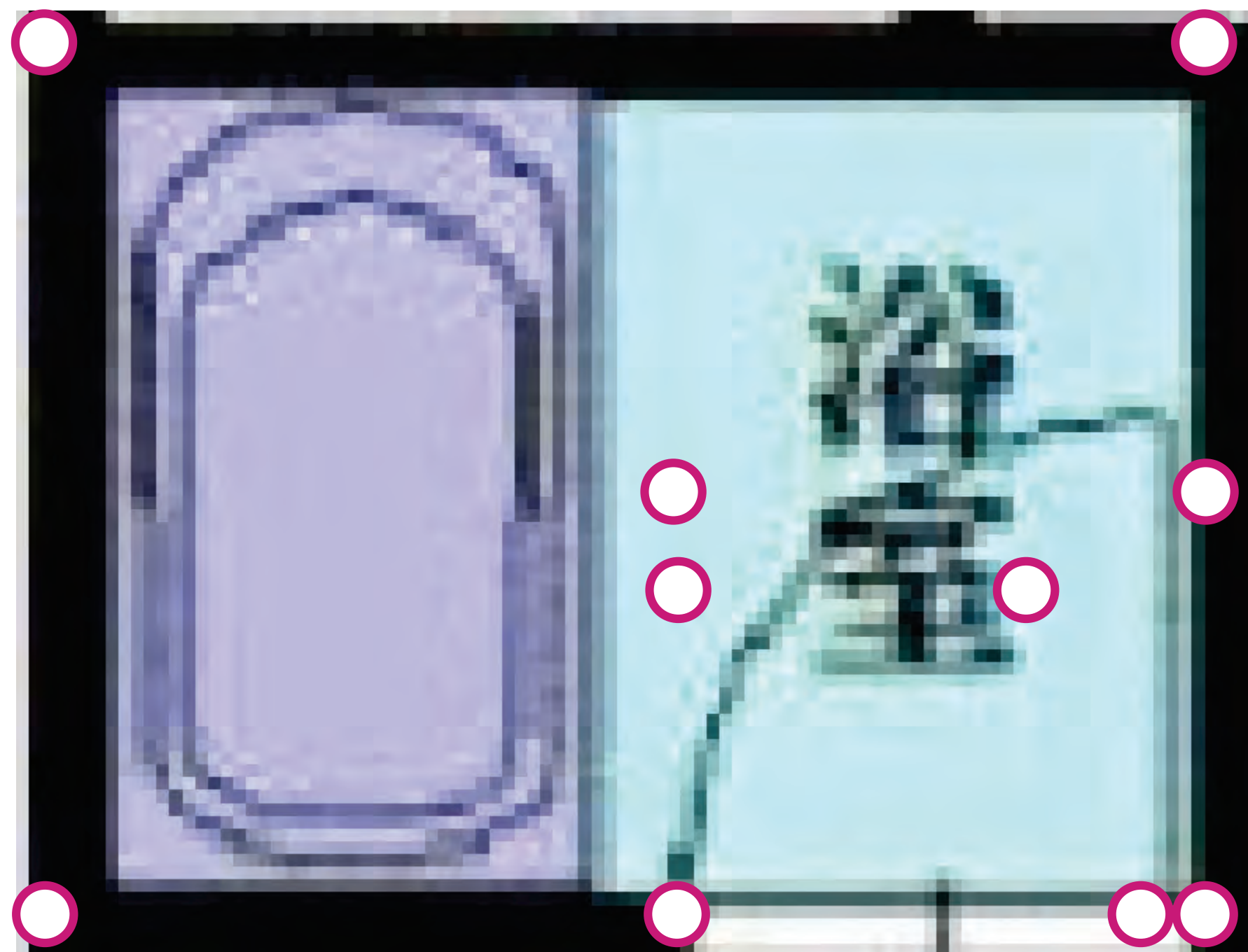
# Corner candidates



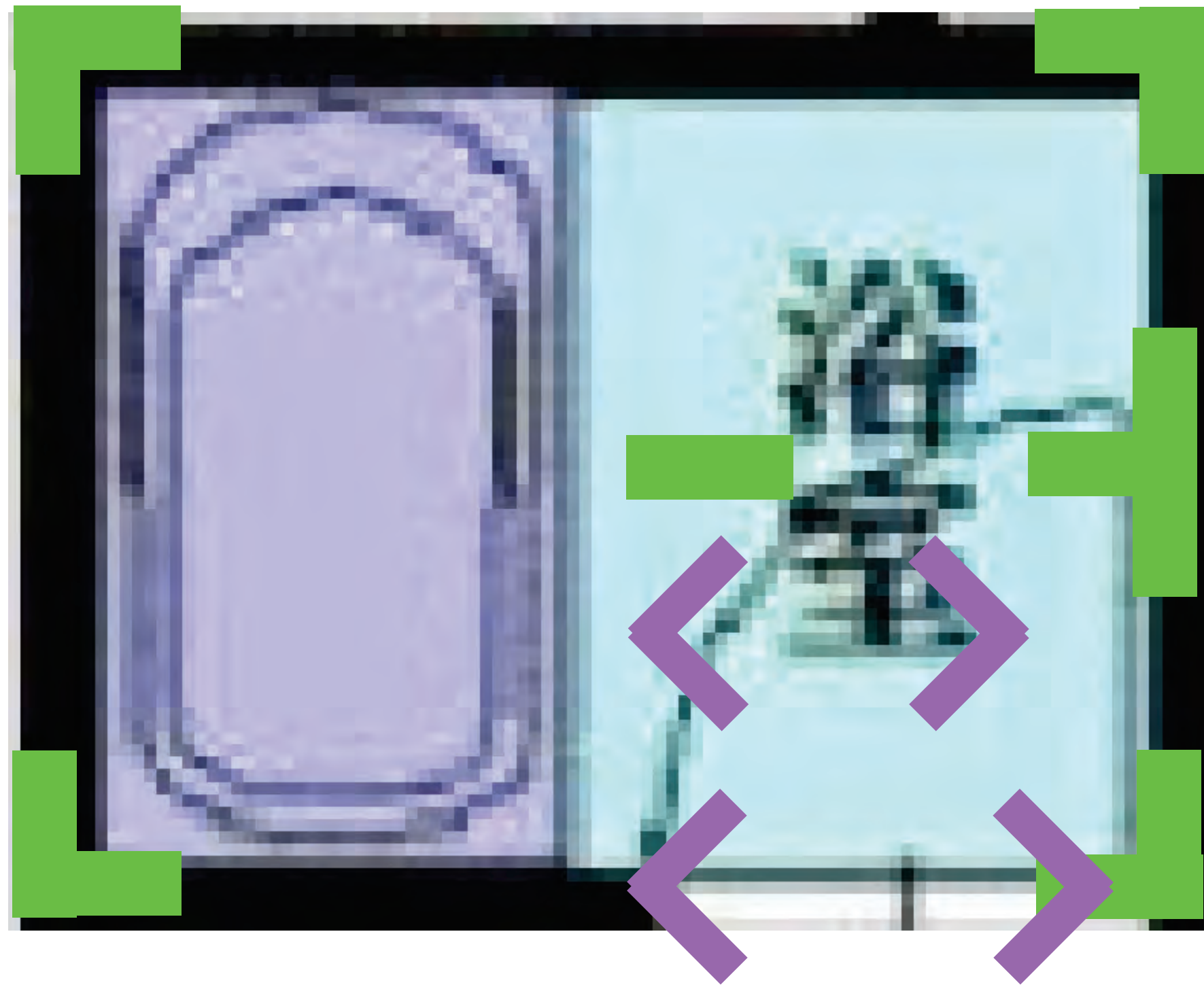
# Corner candidates



# Corner can



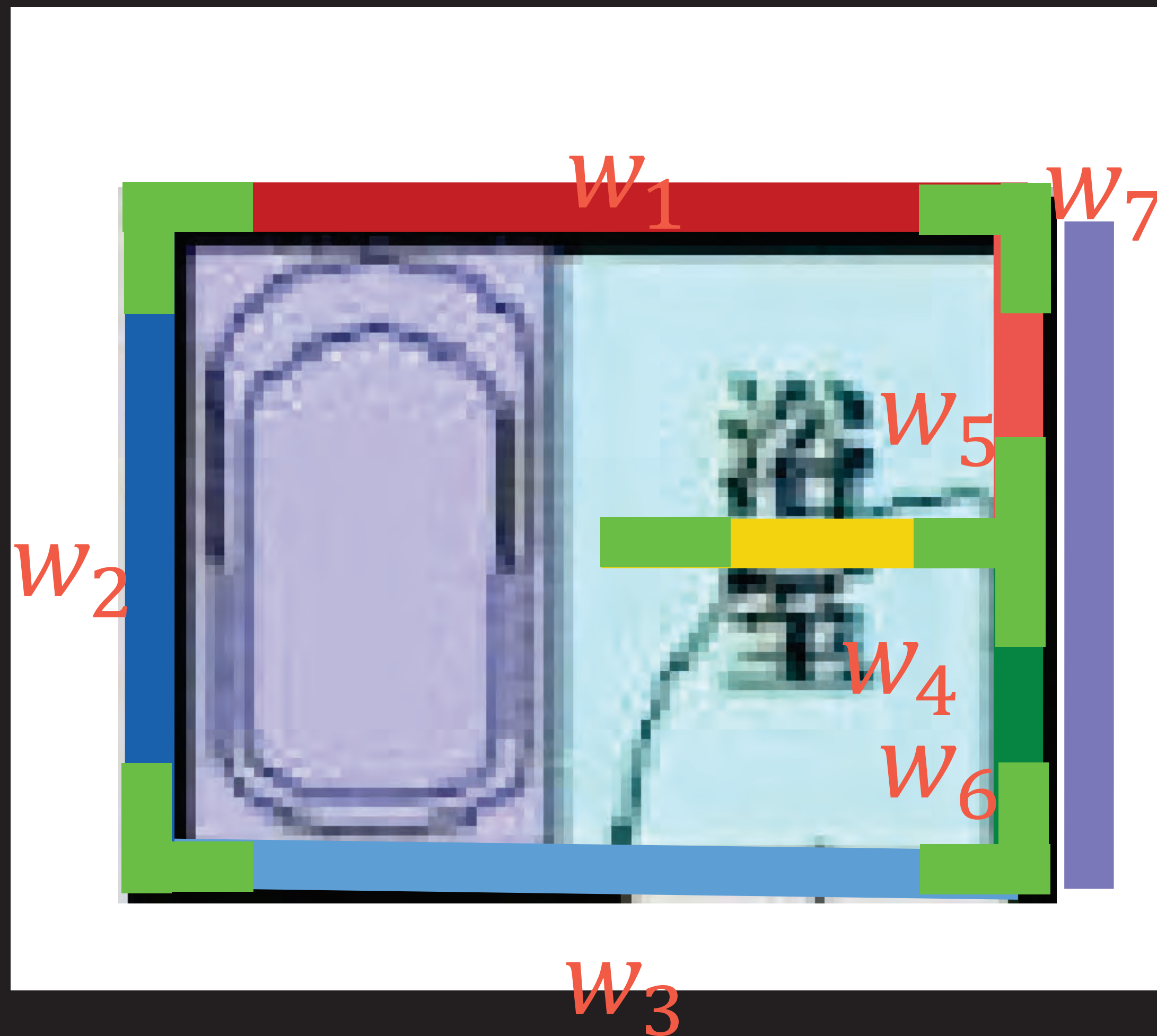
# Corner candidates



# Wall candidates



# Wall candidates

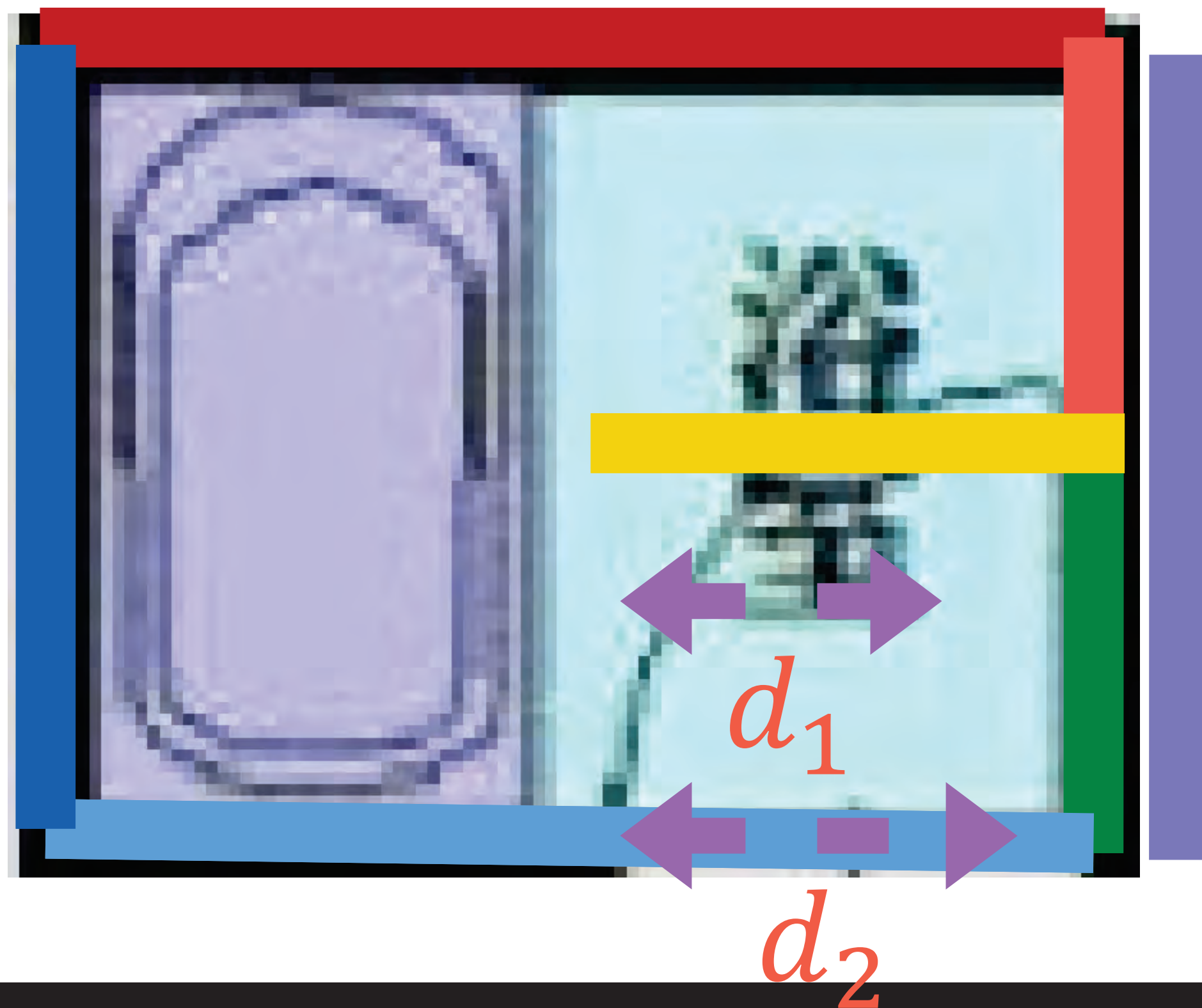


# Door candidates

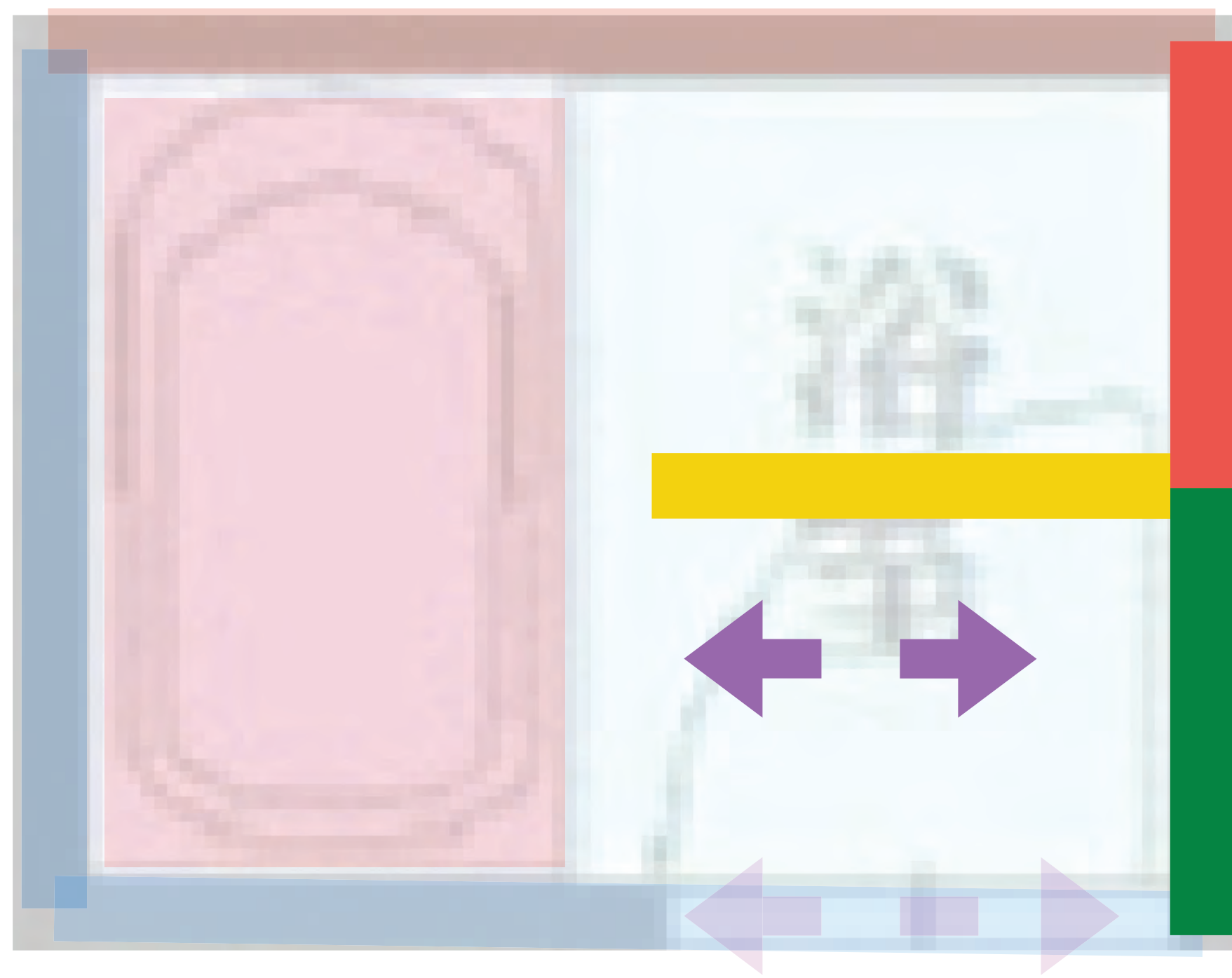




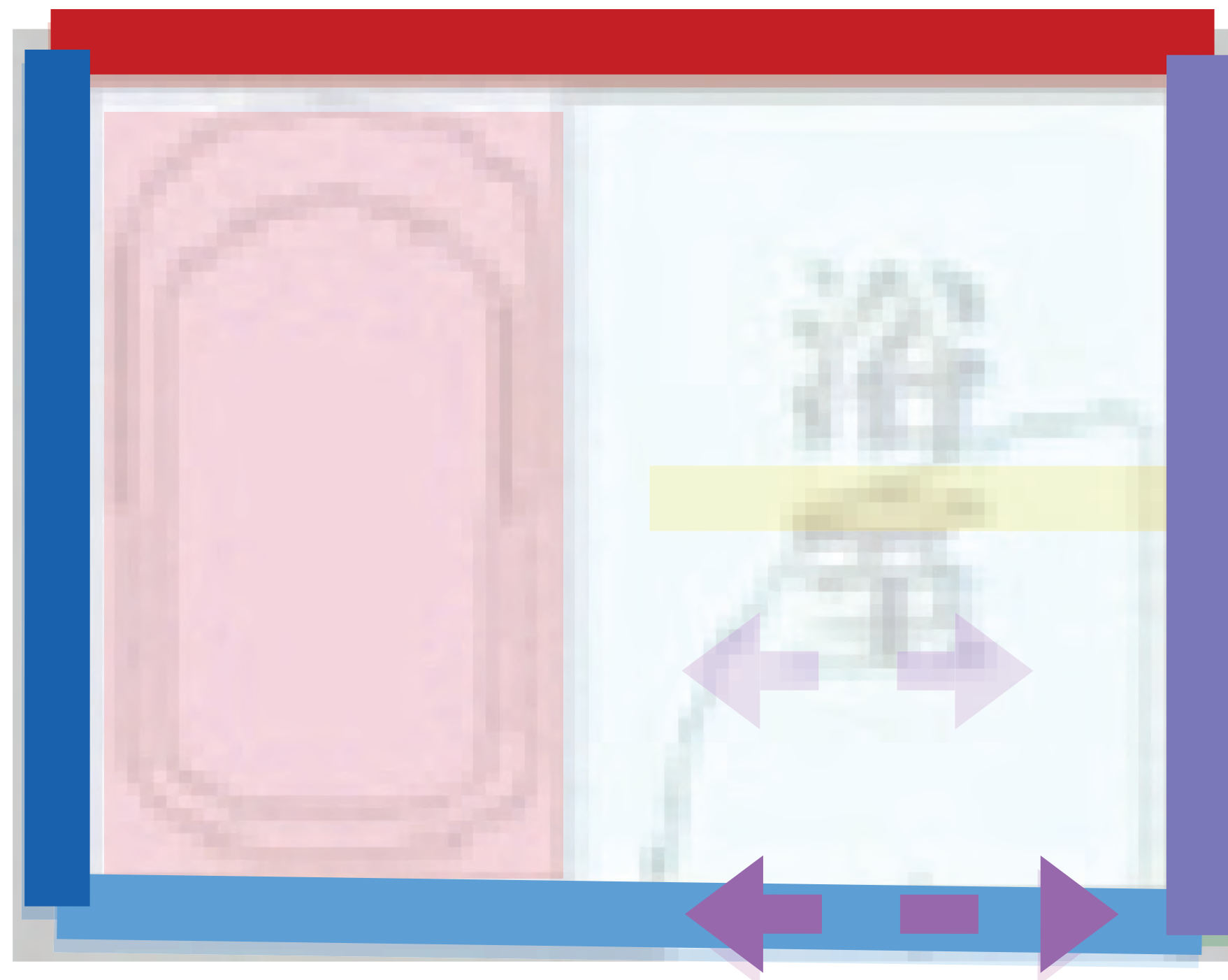
# Door candidates



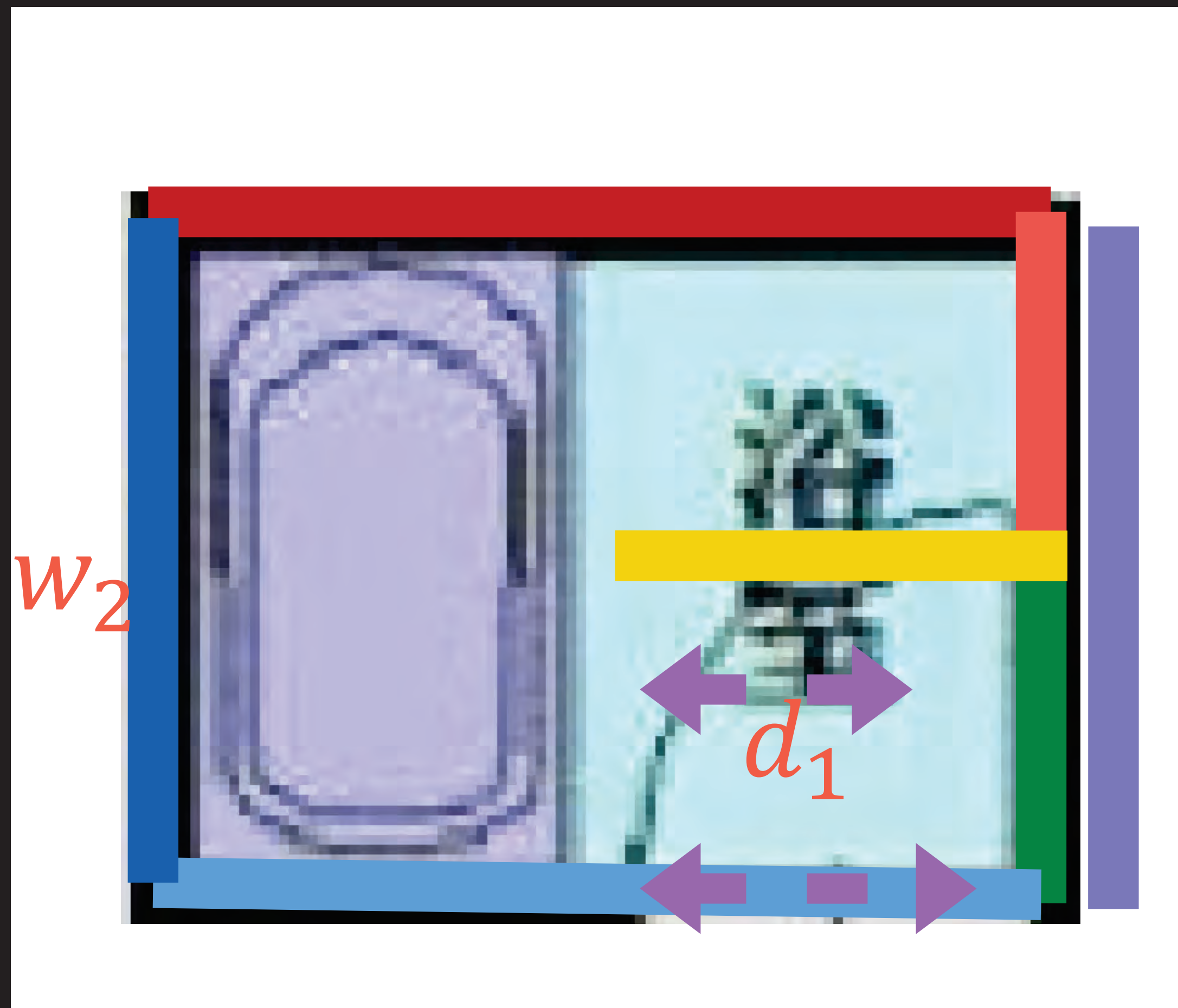
# Incorrect primitive candidates



# Correct primitive candidates



# Optimization formulation

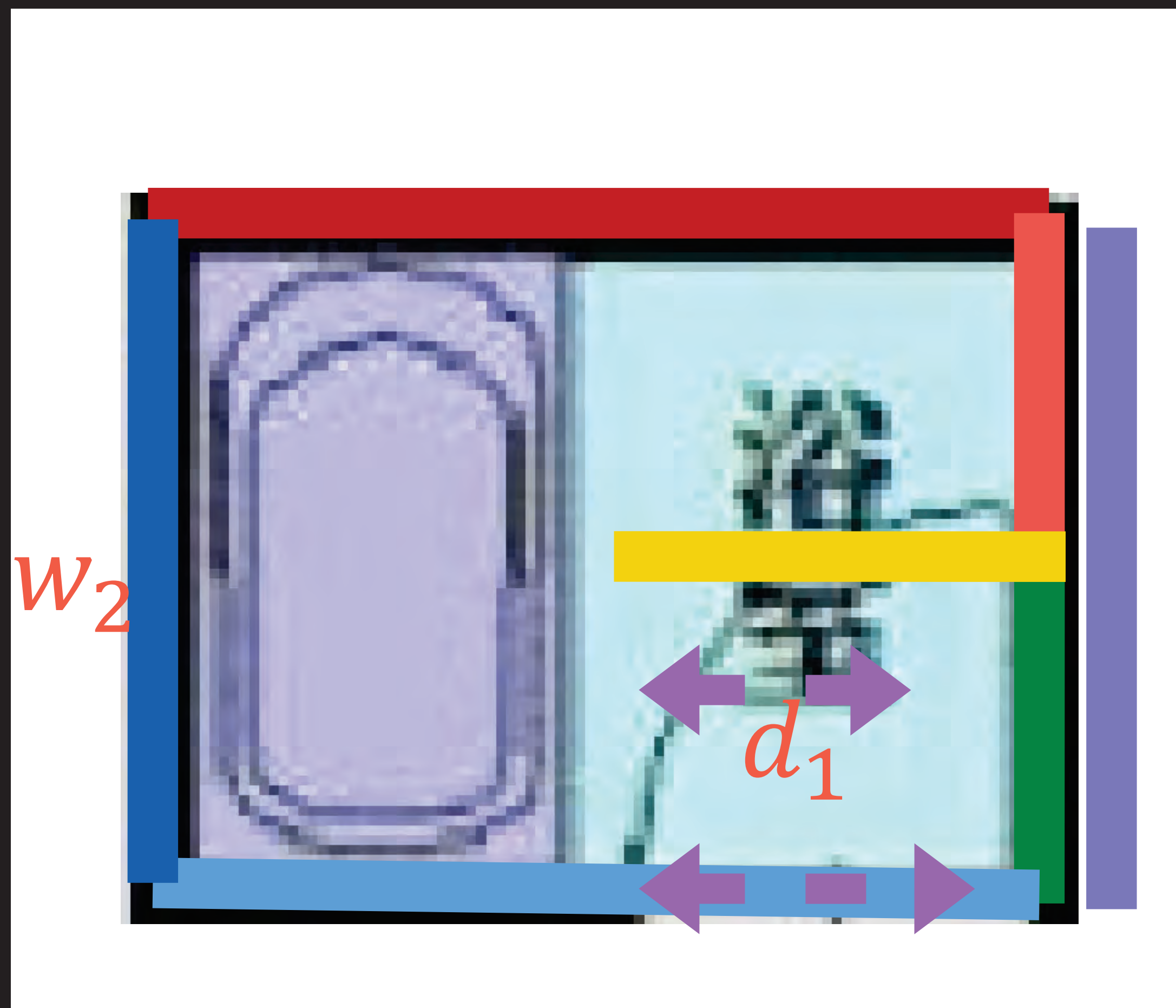


$v \leftarrow \begin{cases} 1 & \text{if correct} \\ 0 & \text{if incorrect} \end{cases}$

$d_1 \leftarrow 0$

$w_2 \leftarrow 1$

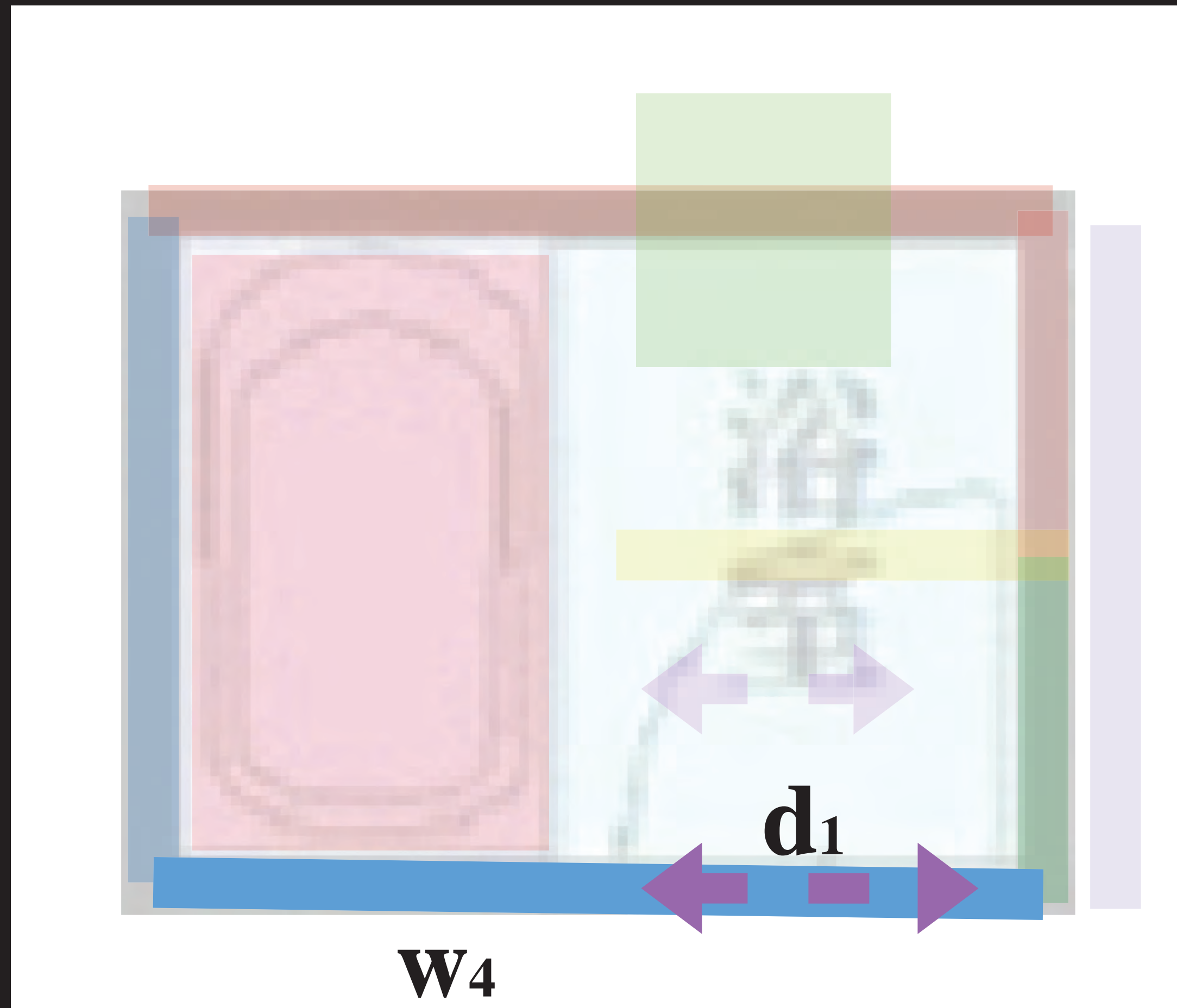
# Optimization formulation



$v \leftarrow \begin{cases} 1 & \text{if correct} \\ 0 & \text{if incorrect} \end{cases}$

$$\max \sum_i w_i + \sum_j d_j$$

# Door constraints



$$\max \sum_i w_i + \sum_j d_j$$

Subject to

$$d_1 \leq w_4 (+w_? + w_? \dots)$$

# Loop constraints

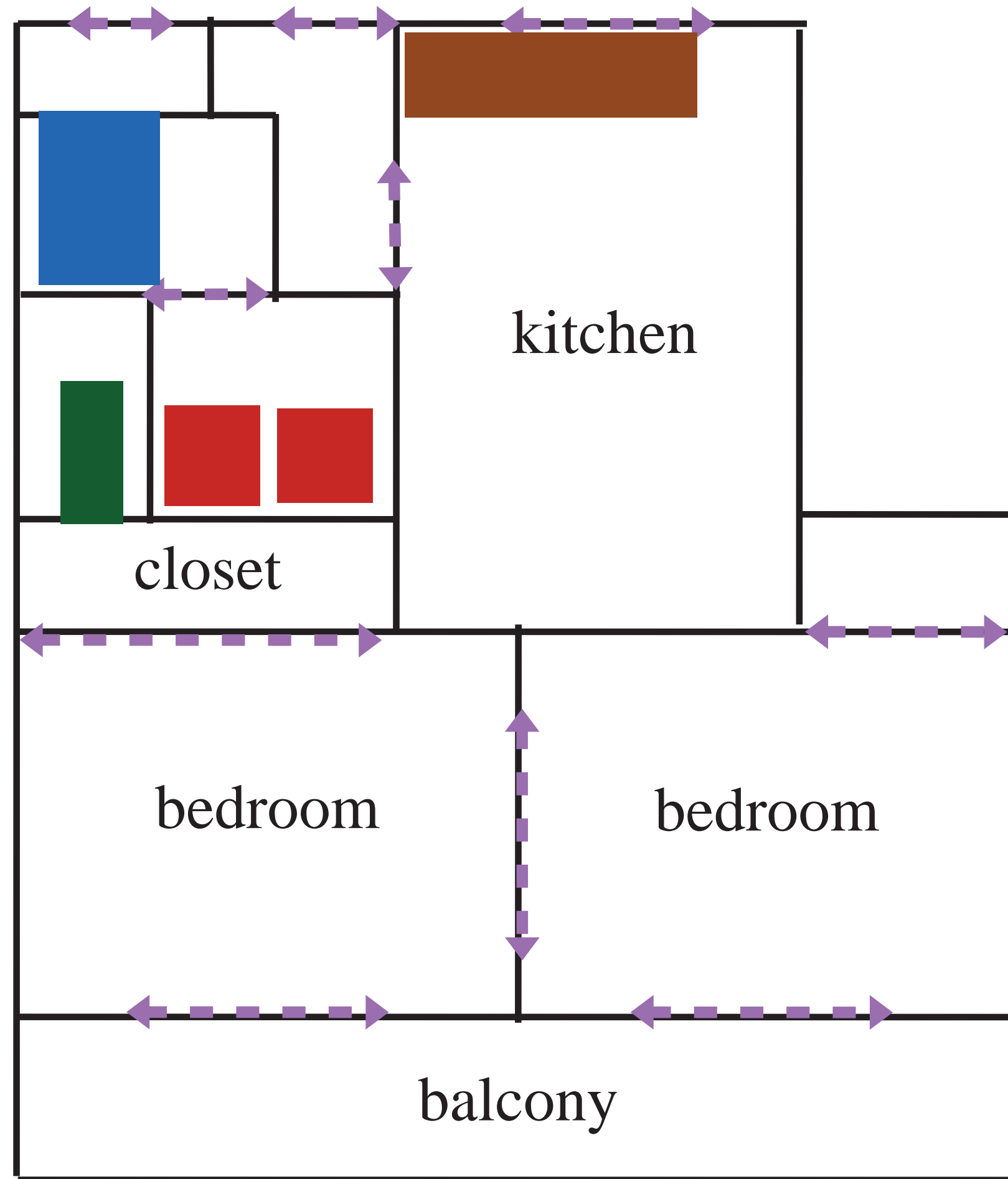


# Loop constraints



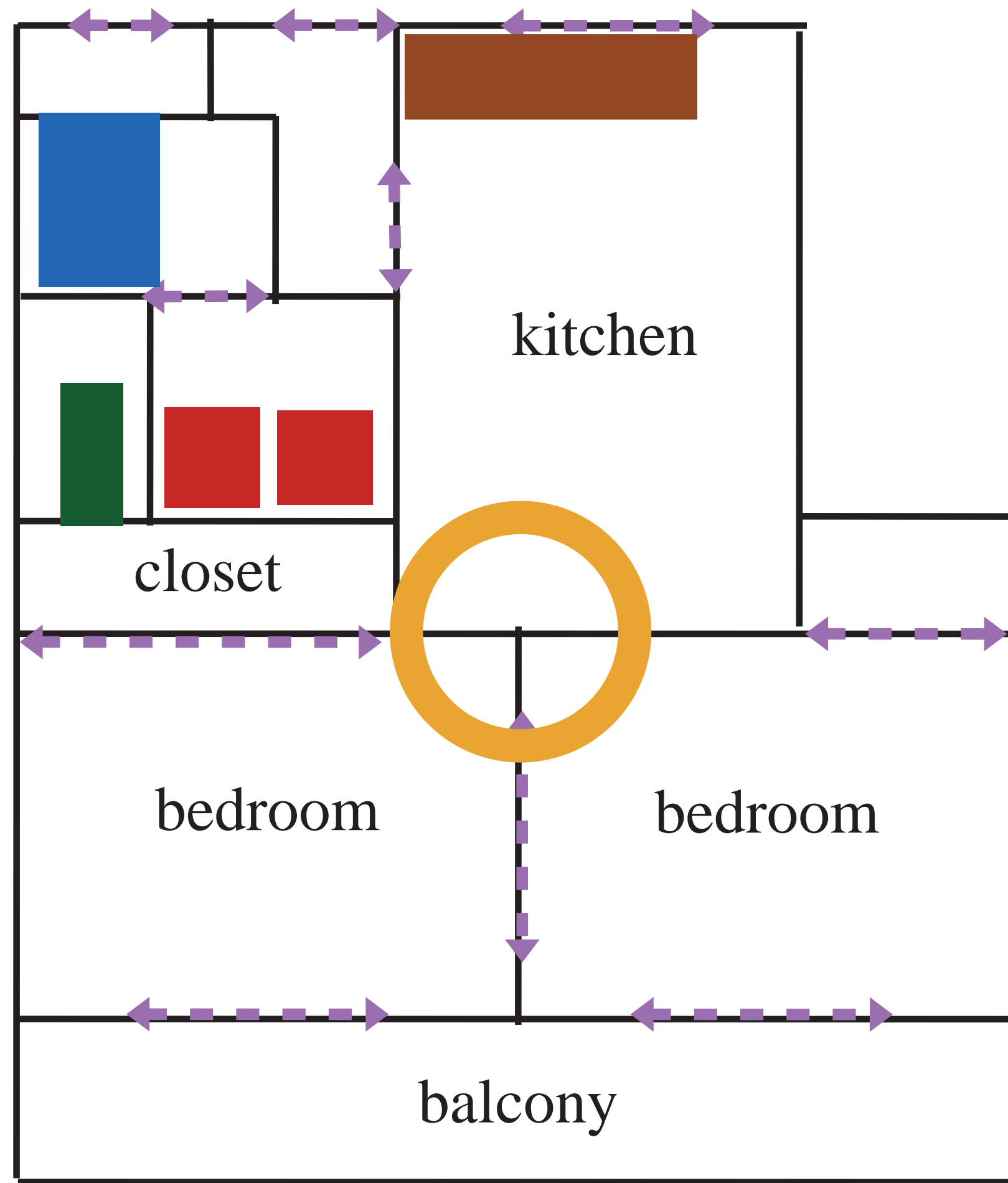


# Loop constraints

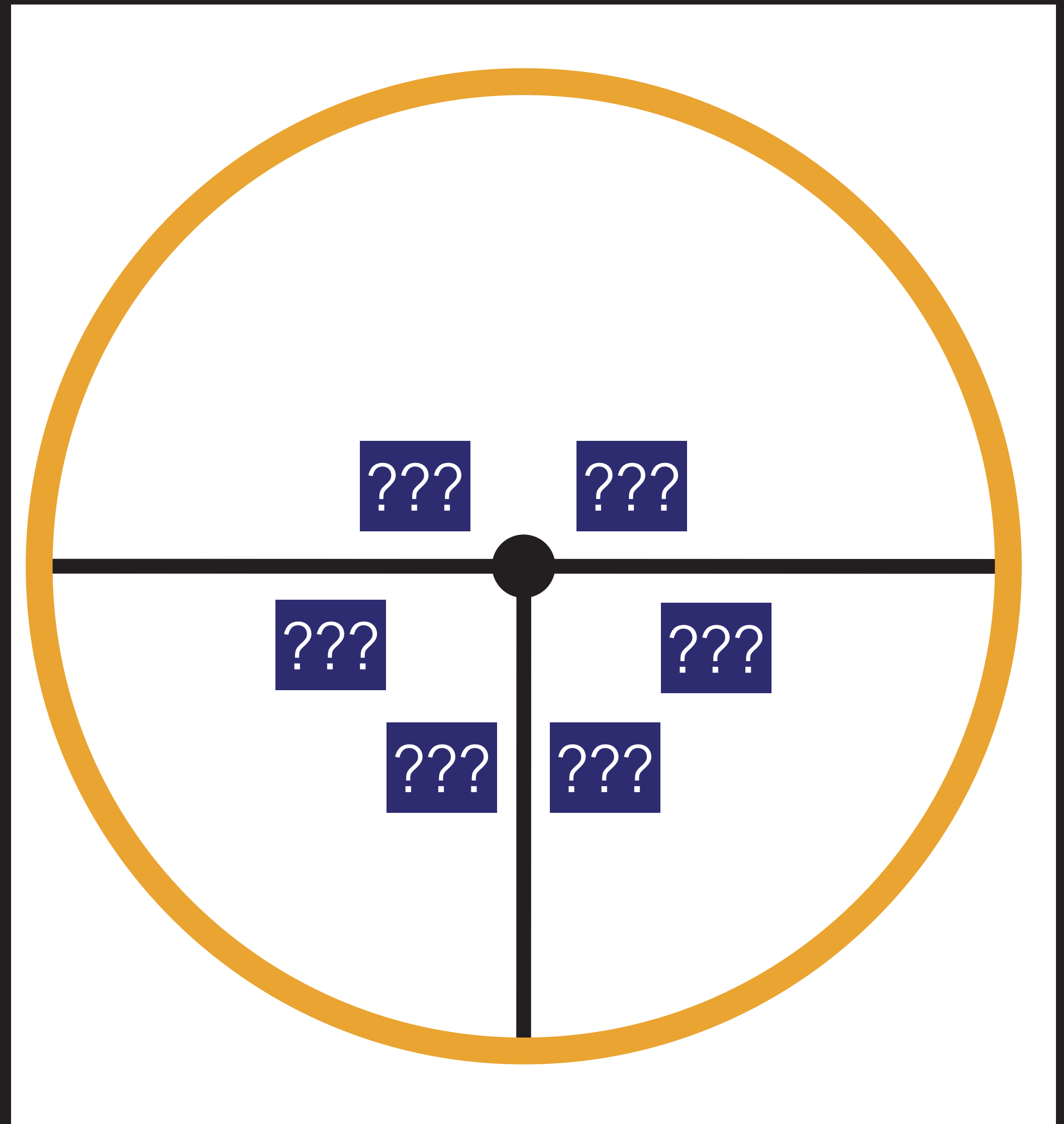


Vectorized floorplan

# Loop constraints



Vectorized floorplan





# Integer Programming

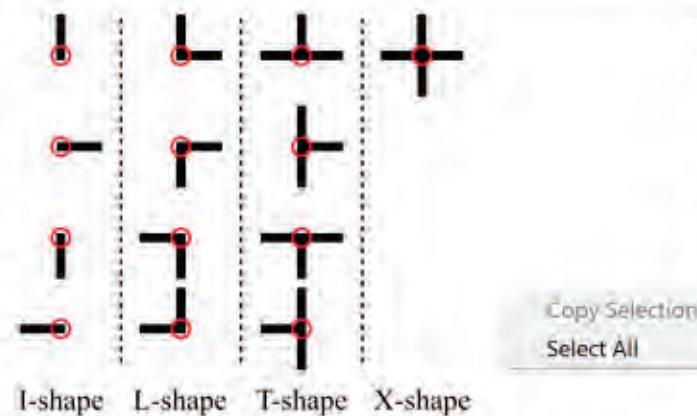


Figure 3: There are four wall junction types: I-, L-, T-, and X-shaped, depending on the degrees of incident wall segments. Considering orientations, we have in total 13 ( $= 4 + 4 + 4 + 1$ ) types.

can be connected given their orientations. Each primitive is also associated with some semantics information.

- A wall primitive is associated with room types at its both sides. The possible room types are the same as in the junction layer.
- An opening primitive, which is either a door or a window, does not have any semantics associated at this layer, as doors and windows are indistinguishable on our floorplan images.
- An icon primitive is associated with one of the icon types, which are the same as in the junction layer.

Primitives must satisfy a variety of constraints so that a simple post-processing can extract a floorplan vector representation with high-level structure. For example, a bedroom must be represented by a set of wall primitives that form a closed-loop and have consistent room type annotation on one side. The constraints are enforced in solving the Integer Programming as explained in Section 4.2.

## 4. Raster to vector conversion

Our system consists of three steps. First, we employ a CNN to convert a raster floorplan image into the first junction layer (*i.e.*, junction maps and per-pixel room-classification scores). Second, after generating primitive candidates from junctions based on simple heuristics, we use Integer Programming to select the right subset while enforcing high-level geometric and semantic constraints. Third, a simple post-processing is used to convert the floorplan data into the final vector-graphics representation. We now explain each step.

### 4.1. Junction layer conversion via a CNN

CNNs have been proven to be very powerful in extracting low-level information from images. Our junction and per-pixel classification representation allows straight-forward application of CNNs. We borrow the residual part of the detection network from [7], which modified ResNet-152 [15] to predict heatmaps at pixel-level. We drop their last deconvolution layer, and append three deconvolution layers in

parallel, one for junction heatmap regression and two for per-pixel classifications. For junctions, there are at total 21 ( $= 13 + 4 + 4$ ) different types, and one heatmap is regressed for each type, where pixelwise sigmoid cross entropy loss is applied. For classification tasks, we use pixelwise softmax cross entropy loss. We train three branches jointly, and the final loss is a weighted summation with larger weight, only for junction heatmap regression. Both the input output have resolution 256x256. Besides common data augmentation techniques like random cropping and color jittering, we also rotate the image with an angle randomly picked from  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ , and  $270^\circ$ . During inference, we threshold junction heatmaps with 0.4 (slightly lower than 0.5 to bring in more junction candidates for IP to choose), and apply non-maximum suppression to extract junctions.

While the network makes very good predictions of junction locations, it sometimes mis-classifies junction types (*e.g.*, mis-classifies a L-shaped junction as T-shaped). To make the detection robust, we allow one mistake in the estimation of the degree. For example, for each detected T-shaped junction, we hallucinate two L-shaped junctions and one X-shaped junction at the same location. The integer programming will enforce later that at most one of the junctions can be used. We found that mis-classification between I and L is rare, and perform the hallucination for all the other cases.

### 4.2. Primitive layer conversion via IP

Deep network makes very good predictions and simple heuristics suffice to extract primitive candidates (*i.e.*, walls, openings, and icons). Integer programming then finds the correct subset while enforcing various geometric and semantic constraints. With the small problem size, it takes around 2s to find the optimal solution to IP using Gurobi [2].

#### 4.2.1 Primitive candidate generation

A wall primitive can be formed by two wall junctions if 1) they are axis-aligned with a tolerance of 10 pixels, and 2) their aligned orientation is compatible with the junction orientations. Similarly, two door junctions can form a door primitive if qualified. For icons, four axis-aligned junctions (top-left, top-right, bottom-right, and bottom-left) together form an icon primitive.

#### 4.2.2 Integer programming

Integer Programming enforces geometric and semantic constraints among primitives to filter out spurious primitive candidates and guarantee properties of floorplan data, which must hold true. For instance, a bedroom must be surrounded by a set of walls forming a 1D loop, with a bedroom type associated with the correct side of each wall primitive.

**Variable definition:** We define indicator variables for junctions, primitives, and semantic types:

- $J_{wall}(j)$ ,  $J_{open}(j)$ ,  $J_{icon}(j)$  for junctions,
- $P_{wall}(p)$ ,  $P_{open}(p)$ ,  $P_{icon}(p)$  for primitives,
- $S_{wall}^L(p, s)$ ,  $S_{wall}^R(p, s)$ ,  $S_{icon}(p, s)$  for semantics.

$j$ ,  $p$  and  $s$  denote indexes for junctions, primitives and possible semantic types, respectively. For instance,  $P_{open}(p)$  is an indicator variable encoding if the  $p$ th opening primitive exists or not. Indicator variables for semantics employ one-hot encoding and have two indexes, a primitive index and a semantic type index. Lastly, a wall primitive is associated with two room types as semantics on its both sides. For instance,  $S_{wall}^L(p, s)$  indicates if the  $p$ th wall primitive has the  $s$ th room type on the left hand side.

**Objective function:** The objective function for maximization is a linear combination of the junction and semantic indicator variables, where weights are defined as follows:

- The weight is 10 for all the junctions except for the hallucinated wall junctions, whose weight is set to  $-5$  (See Section 4.1). In other words, we encourage the use of the primitives as much as possible, but discourage the use of the hallucinated ones.
- The weights for the semantic indicator variables are calculated based on the per-pixel classification scores in the junction layer. For an icon type indicator variable, the weight is simply the average icon type classification score inside the box. For a room type indicator variable associated with a wall primitive on one side, we use the average room type classification score in its neighborhood. A neighborhood is obtained by sweeping pixels on the wall primitives along its perpendicular direction (on the side of the room type variable). Each pixel is swept until it hit another wall primitive candidate.

We have not used primitive indicator variables in the objective function, as similar information has been already captured by the junction primitives.

**Constraints:** We enforce a variety of constraints either as linear equalities or linear inequalities.

- One-hot encoding constraints: When a wall primitive does not exist (*i.e.*,  $P_{wall}(p) = 0$ ), its wall semantic variables must be all zero. When it exists, one and only one semantic variable must be chosen. The same is true for icon primitives, yielding the following constraints.

$$P_{wall}(p) = \sum_s S_{wall}^L(p, s) = \sum_s S_{wall}^R(p, s),$$

$$P_{icon}(p) = \sum_s S_{icon}(p, s).$$

- Connectivity constraint: The degree (*i.e.*, the number of connections) of a junction must match the number of incident primitives that are chosen. This applies to



Figure 4: Loop constraints can be enforced locally at each junction. The room types must be the same for each pair of walls marked with the same color.

walls, openings and icons, and here we only show the constraint for the walls, where the summation is over all the wall primitives connected with the wall junction:

$$\{\# \text{ degree}\} J_{wall}(j) = \sum P_{wall}(p).$$

- Mutual exclusion constraints: Two primitives cannot be chosen when they are spatially close, in particular, within 10 pixels. We find every pair of such primitives and enforce that the sum of their indicator variables is at most 1. The same constraint is also applied to wall junctions to ensure that two wall junctions are not close and hallucinated junctions are not selected simultaneously with the original one.
- Loop constraints: Bedroom, bathroom, restroom, balcony, closet, pipe-space, and the exterior boundary must form a closed loop (allowing some walls that stick out). It turns out that this high-level rule can be enforced by local constraints at every wall junction. We use a T-shaped wall junction to explain our idea in Fig. 4. Room types must be the same for a pair of walls with arrows of the same color in the figure.
- Opening constraints: An opening (a door or a window) must be on a wall. For each opening primitive, we find a list of wall primitives that contain the opening (parallel to the opening with distance smaller than 10 pixels) and enforce the following. Note that the right summation is over all the collected wall primitives.

$$P_{open}(p) \leq \sum P_{wall}(p).$$

### 4.3. Final data conversion

The IP output is close to the final representation with a few issues remaining. First, junctions are not well-aligned, because we allow some coordinate error when finding connections. The alignment issue can be simply fixed by averaging the junction coordinates along a straight line. The second issue is that doors are not sitting exactly on walls. To fix this issue, we move each door to align with its closest wall. The last issue is the missing of high-level room information, as room labels are currently associated with walls locally. To derive room information, we first find all the closed polygons formed by walls. If all the walls of a polygon share the same

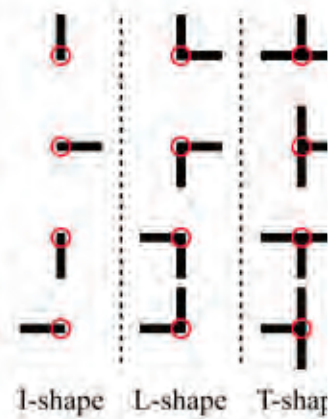


Figure 3: There are four wall junction shapes, depending on the degrees of incidence. Considering orientations, we have in total 13

can be connected given their orientations, and also associated with some semantics:

- A wall primitive is associated with two sides. The possible room types are the layer.
- An opening primitive, which is either a door or a window, does not have any semantics associated with it, and windows are indistinguishable from doors.
- An icon primitive is associated with a room type, which are the same as in the junction layer.

Primitives must satisfy a variety of constraints. A simple post-processing can extract a set of primitives with high-level structure. It must be represented by a set of wall primitives that form a closed-loop and have consistent orientations on one side. The constraints are enforced in the programming as explained in Section 4.

#### 4. Raster to vector conversion

Our system consists of three steps: first, a CNN to convert a raster floorplan image into a vector representation (i.e., junction maps and per-pixel scores). Second, after generating primitive junctions based on simple heuristics, a post-processing step to select the right subset which satisfies geometric and semantic constraints. Finally, a post-processing step is used to convert the floorplan into a vector-graphics representation. We describe each step in detail below.

##### 4.1. Junction layer conversion

CNNs have been proven to be very effective at extracting low-level information from images. In this work, we use a pixel classification representation as the output of an application of CNNs. We borrow the architecture from [7], which mostly consists of a convolution layer, and append three



ng



constraints can be enforced locally at each junction. The indicator variables must be the same for each pair of walls marked with the same color.

For each wall junction, we only show the indicator variables for the walls, where the summation is over all primitives connected with the wall junction:

$$J_{wall}(j) = \sum P_{wall}(p)$$

Consistency constraints: Two primitives cannot be connected if they are spatially close, in particular, if they share a wall. We find every pair of such primitives that share a wall and the sum of their indicator variables must be zero.

The same constraint is also applied to ensure that two wall junctions are not selected if they share a wall. Hallucinated junctions are not selected if they share a wall with the original one.

Room types: Bedroom, bathroom, restroom, balcony, pipe-space, and the exterior boundary must form a closed loop (allowing some walls that stick out), so that this high-level rule can be enforced locally at every wall junction. We use a wall junction to explain our idea in Fig. 4. The indicator variables must be the same for a pair of walls with the same color in the figure.

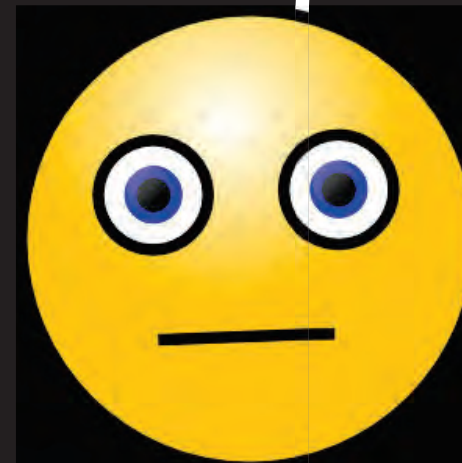
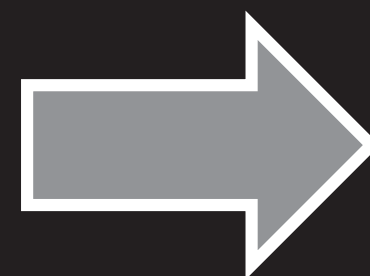
Opening constraints: An opening (a door or a window) is a hole in a wall. For each opening primitive, we find all wall primitives that contain the opening with a distance smaller than 10 pixels. We enforce the following. Note that the right side of the equation is over all the collected wall primitives.

$$J_{open}(p) \leq \sum P_{wall}(p)$$

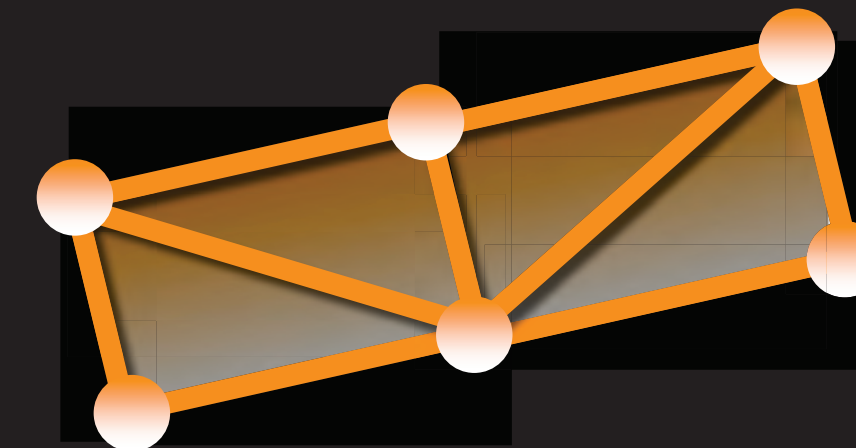
##### Post-processing

This step is close to the final representation with a few adjustments. First, junctions are not well-aligned, so there is some coordinate error when finding common walls. This issue can be simply fixed by averaging the coordinates along a straight line. The second issue is that the junctions are not sitting exactly on walls. To fix this, we move each door to align with its closest wall. The final step is the missing of high-level room information, as the current representation is only associated with walls locally. To recover this information, we first find all the closed polygons and then assign a room type to all the walls of a polygon that share the same

**Optimization  
(Integer Programming)**



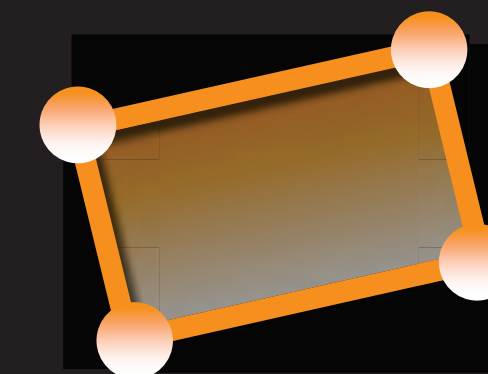
**Graph**



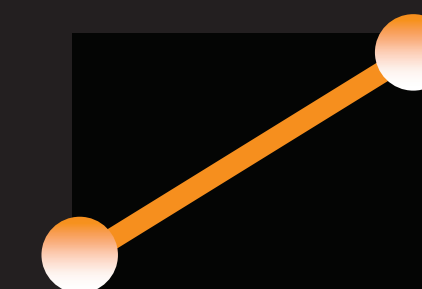
**Geometric Elements**



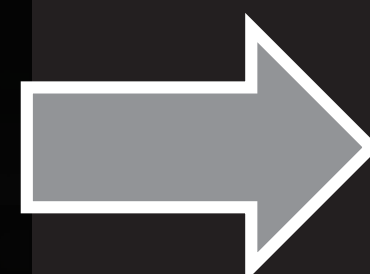
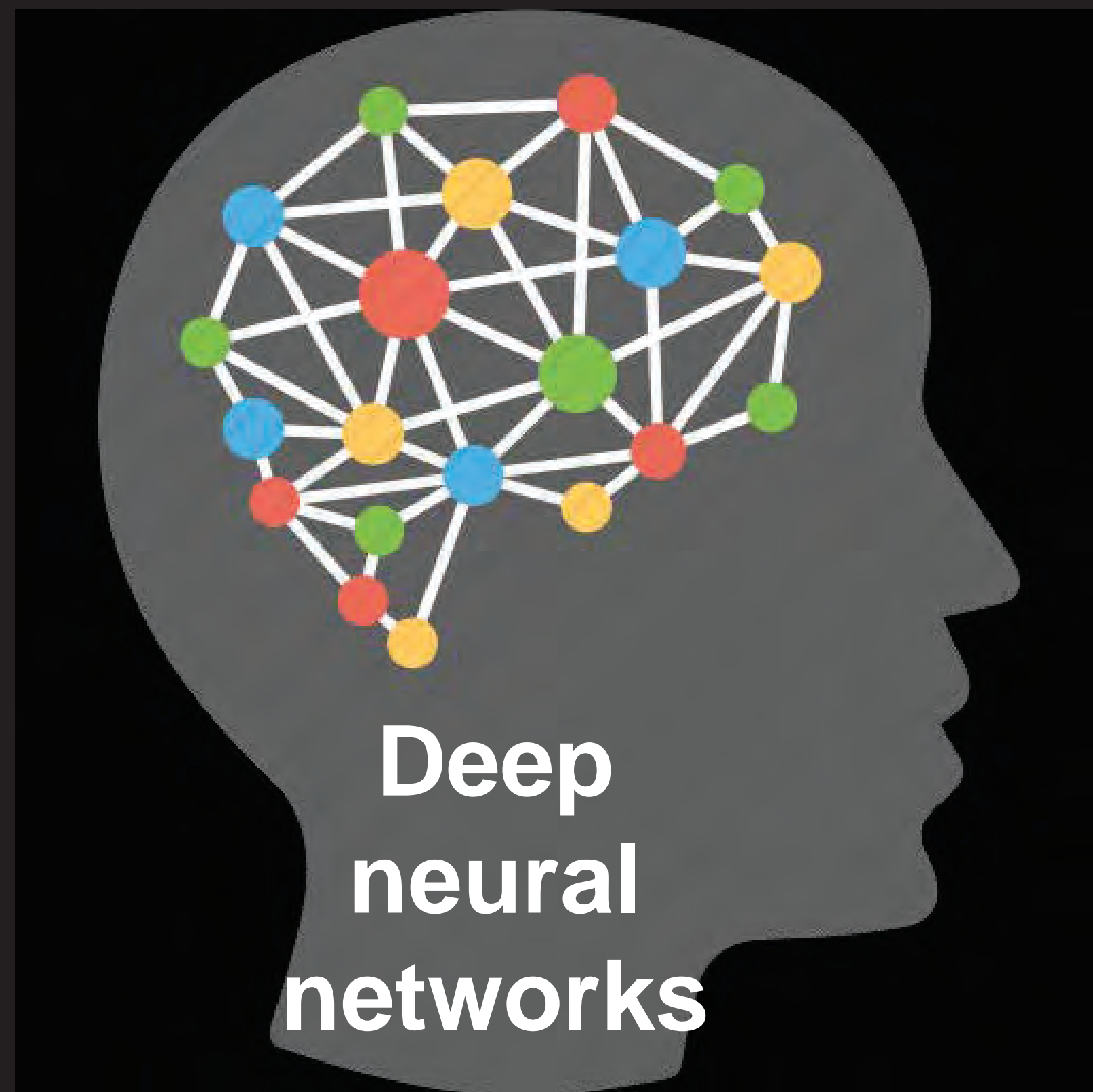
**2D primitive**

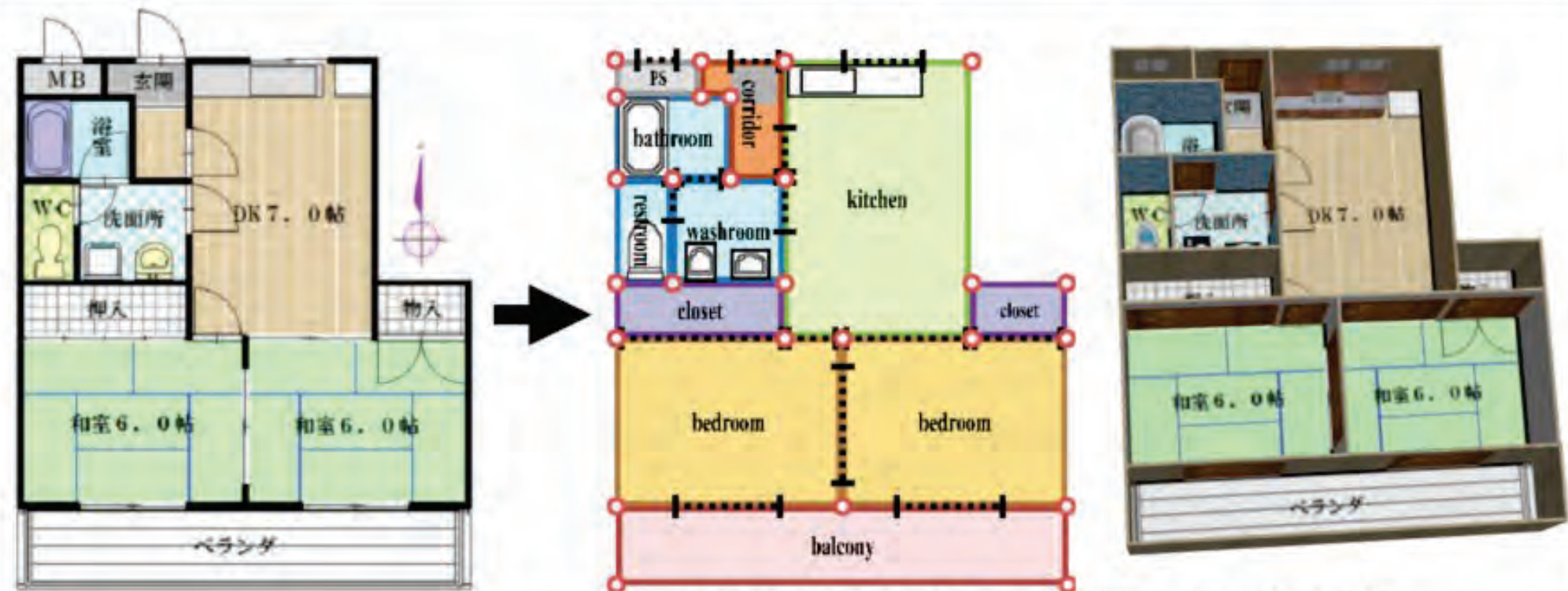


**1D primitive**



**0D primitive**





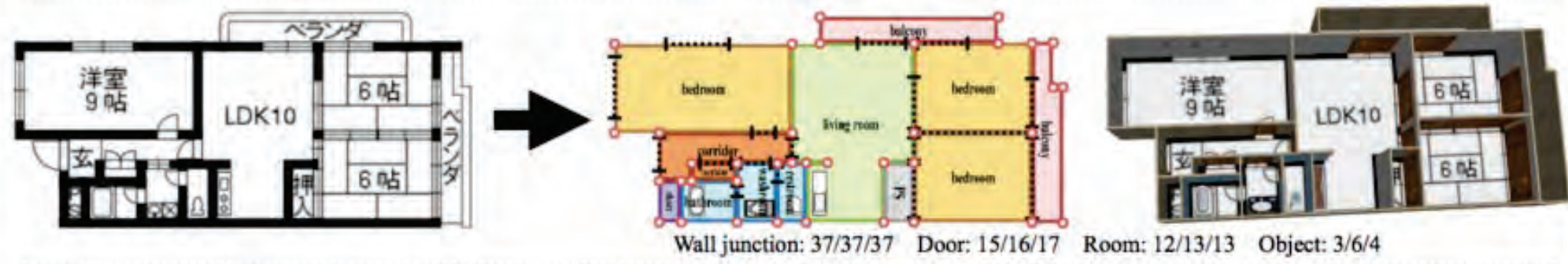
Wall junction: 26/26/26 Door: 14/14/14 Room: 11/11/11 Object: 6/6/6



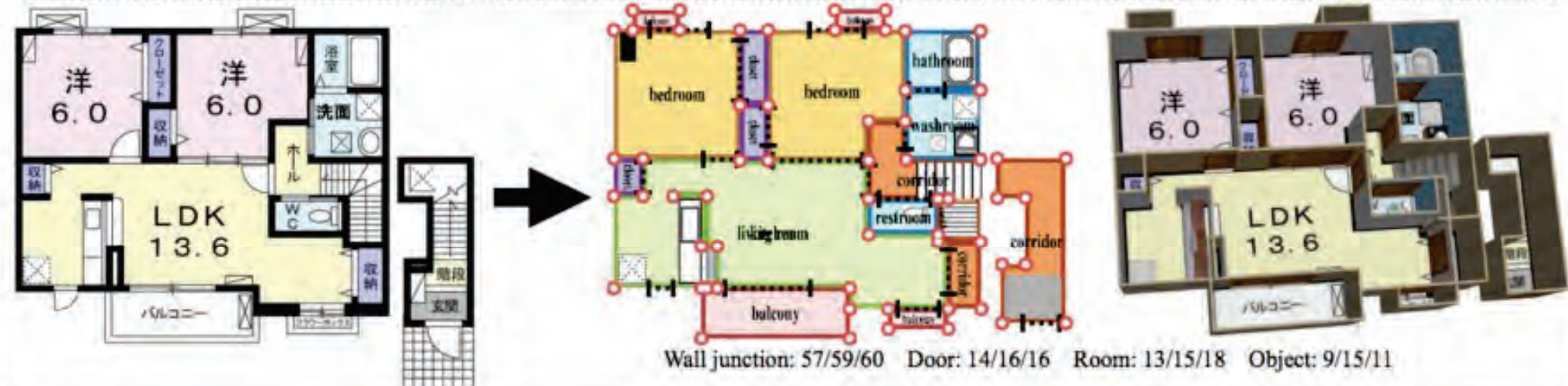
Wall junction: 26/28/26 Door: 16/17/16 Room: 10/10/10 Object: 7/7/7



Wall junction: 35/35/35 Door: 16/16/16 Room: 13/13/14 Object: 5/6/6



Wall junction: 37/37/37 Door: 15/16/17 Room: 12/13/13 Object: 3/6/4



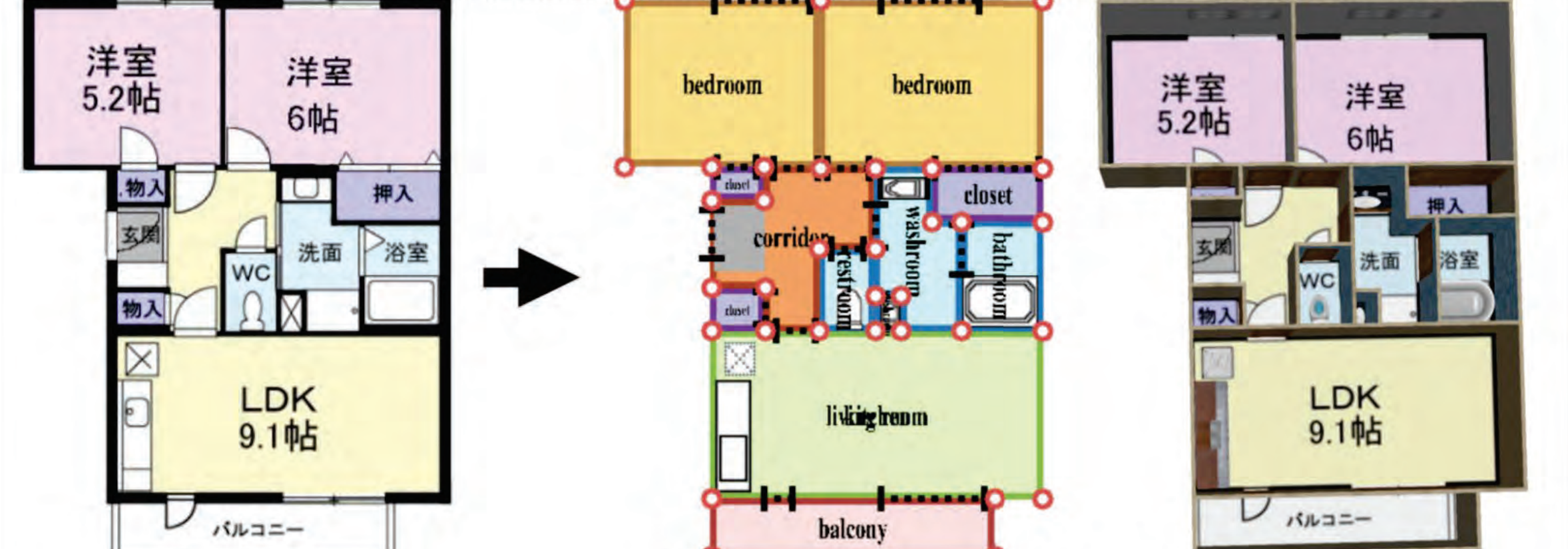
Wall junction: 57/59/60 Door: 14/16/16 Room: 13/15/18 Object: 9/15/11



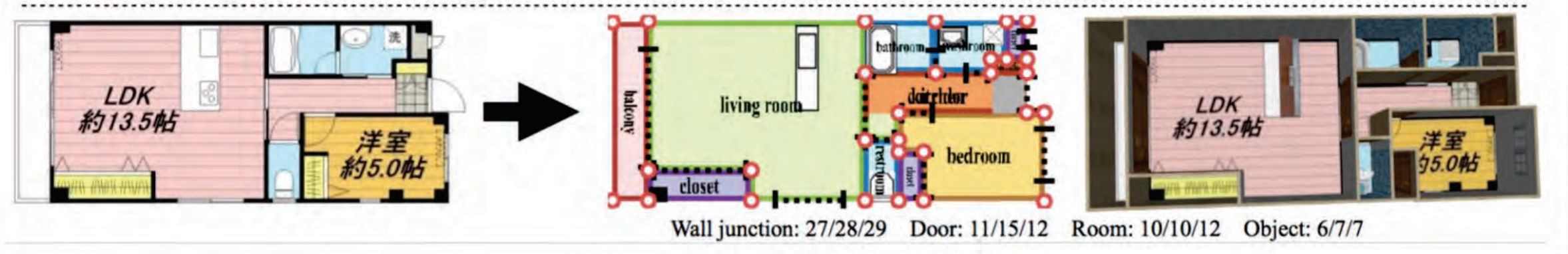
Wall junction: 31/31/31 Door: 13/17/15 Room: 13/13/13 Object: 5/5/6



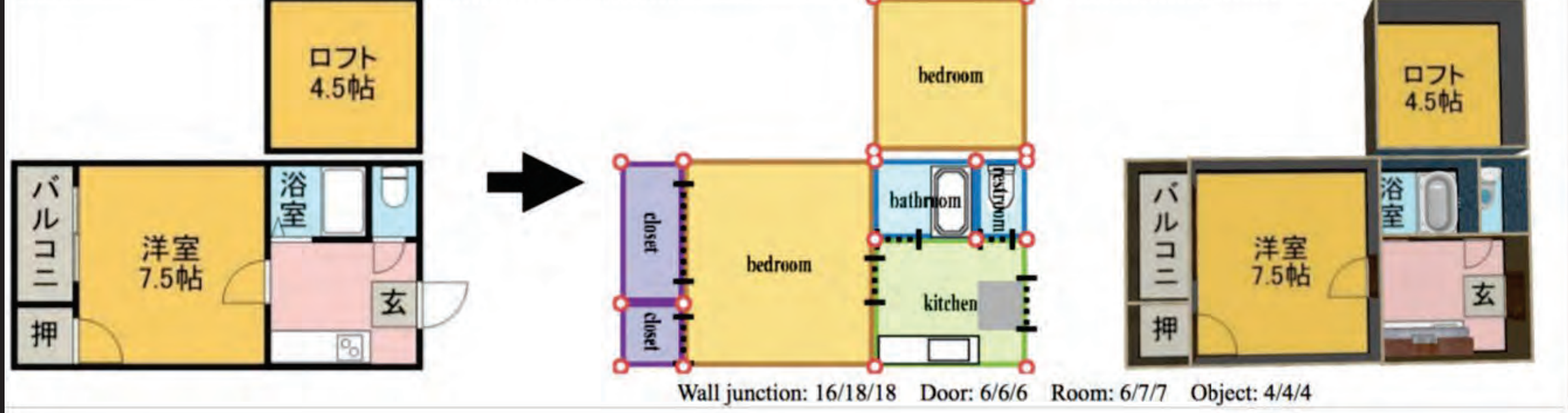
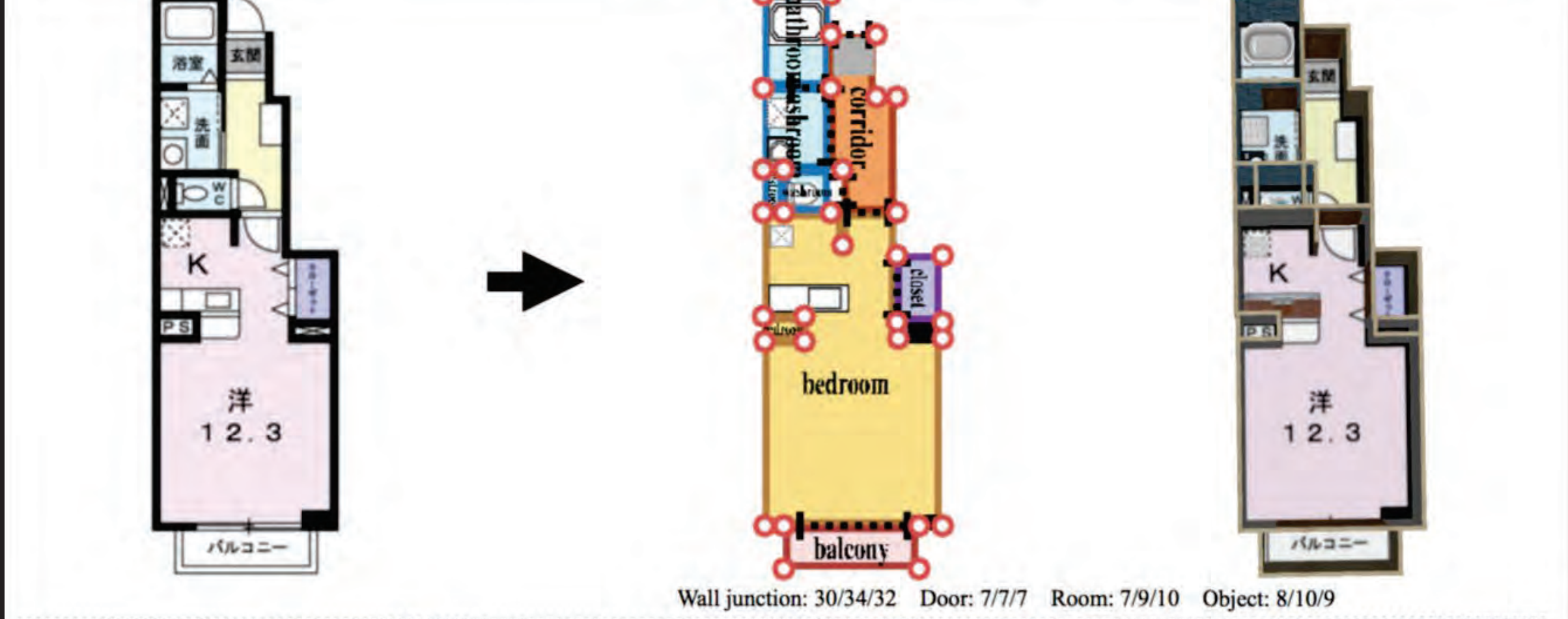
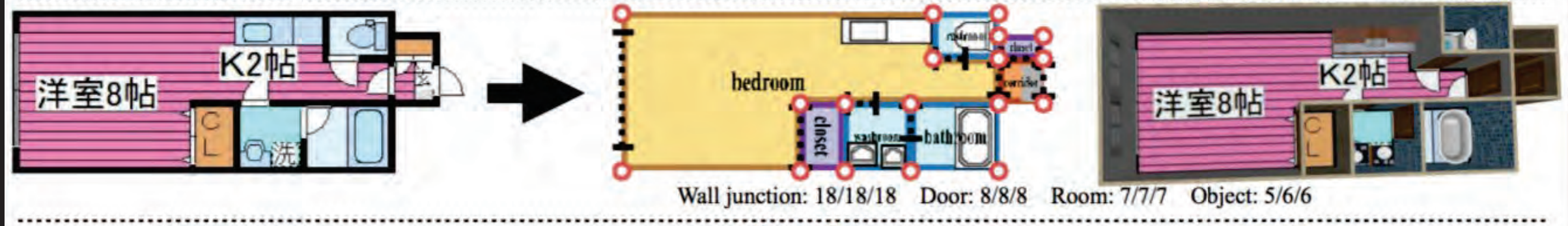
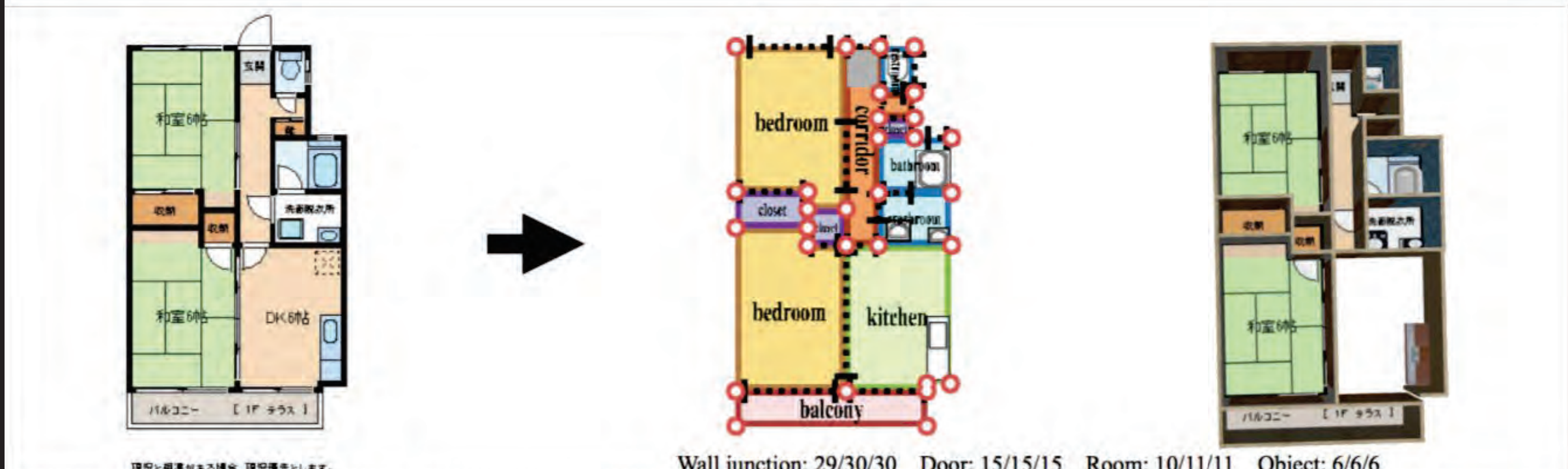
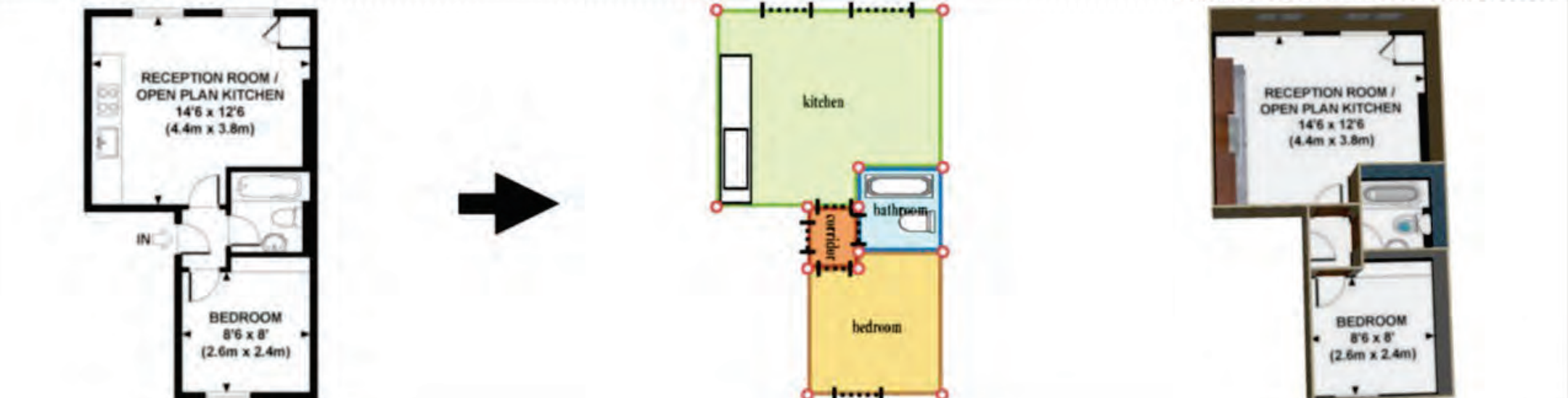
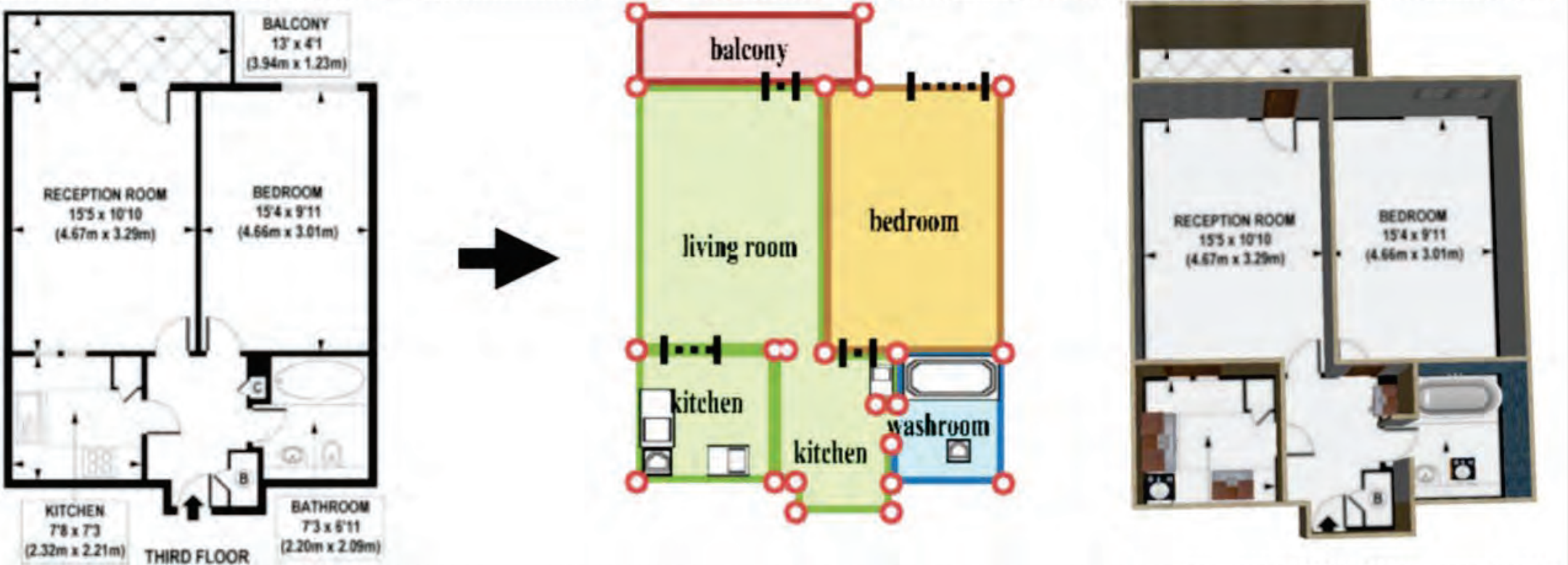
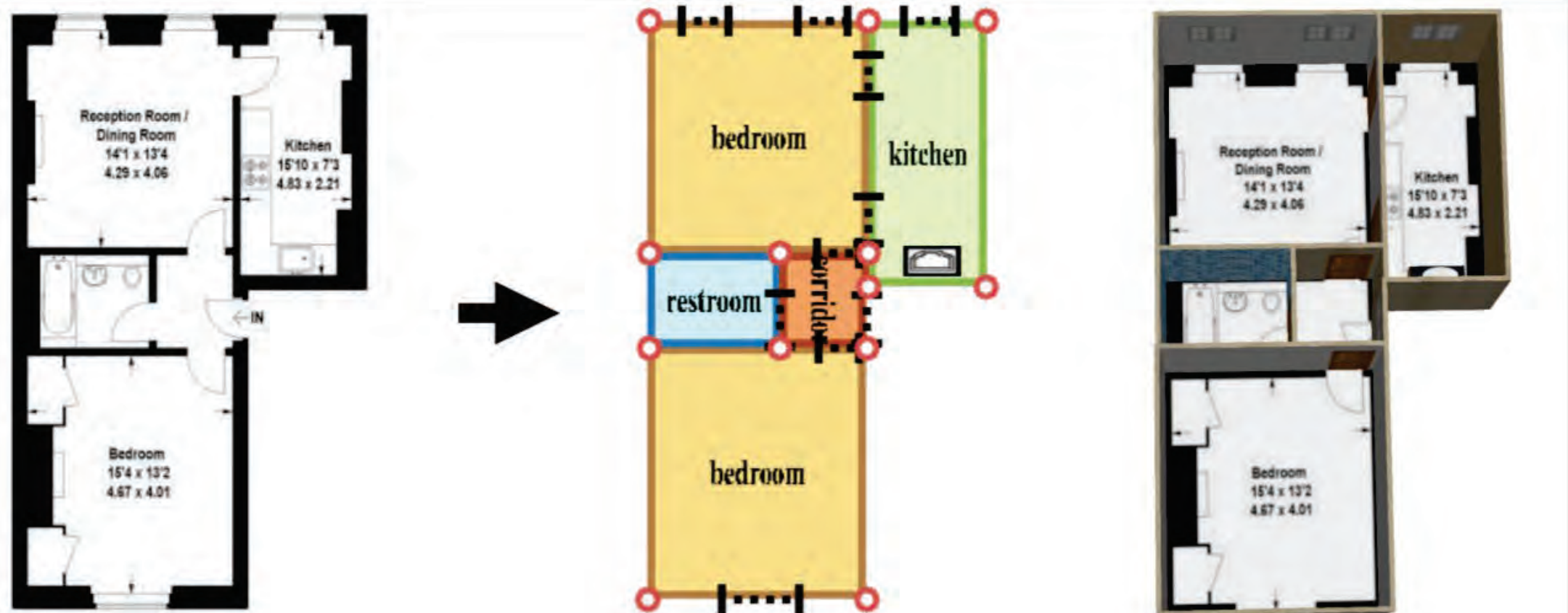
Wall junction: 19/19/19 Door: 7/7/7 Room: 7/7/7 Object: 4/4/4



Wall junction: 33/33/33 Door: 13/14/14 Room: 11/11/13 Object: 6/8/7



Wall junction: 27/28/29 Door: 11/15/12 Room: 10/10/12 Object: 6/7/7





# Quantitative evaluations

Method	Wall Junction		Opening		Icon		Room	
	Acc.	Recall	Acc.	Recall	Acc.	Recall	Acc.	Recall
Ahmed <i>et al.</i> [6]	74.9	57.5	61.3	48.7	N/A	N/A	N/A	N/A
Ours (without IP)	70.7	95.1	67.9	91.4	22.3	77.4	80.9	78.5
Ours (without mutual exclusion constraints)	92.8	91.7	68.5	91.1	22.0	76.2	82.8	87.5
Ours (without loop constraints)	94.2	91.5	91.9	90.2	84.3	75.0	82.5	88.2
Ours (without opening constraints)	94.6	91.7	91.7	90.1	84.0	74.8	84.3	88.3
Ours (with full IP)	94.7	91.7	91.9	90.2	84.0	74.6	84.5	88.4

Table 1: Quantitative evaluations based on our benchmark.

### Piecewise Planar and Compact Floorplan Reconstruction from Images

Ricardo Cabral  
Carnegie Mellon University  
rcabral@cmu.edu

Yasutaka Furukawa  
Washington University  
furukawa@wustl.edu

**Abstract**  
This paper presents a system to reconstruct piecewise planar and compact floorplans from images, which are then converted to high quality texture-mapped models for free-viewpoint visualization. There are two main challenges in image-based floorplan reconstruction. The first is the lack of 3D information that can be extracted from images by Structure from Motion and Multi-View Stereo, as indoor scenes abound with non-diffuse and homogeneous surfaces plus clutter. The second challenge is the need of a sophisticated regularization technique that enforces piecewise planarity, to suppress clutter and yield high quality texture-mapped models. Our technical contributions are twofold.

Figure 1. Our system reconstructs high quality texture-mapped mesh models of cluttered indoor scenes from panorama images.

### Structured Indoor Modeling

Satoshi Ikebata Hang Yan Yasutaka Furukawa  
Washington University in St. Louis

**Abstract**  
This paper presents a novel 3D modeling framework that reconstructs an indoor scene as a structured model from panorama RGBD images. A scene geometry is represented as a graph, where nodes correspond to structural elements such as rooms, walls, and objects. The approach devises a structure grammar that defines how a scene graph can be manipulated. The grammar then drives a principled new reconstruction algorithm, where the grammar rules are sequentially applied to recover a structured model. The paper also proposes a new room segmentation algorithm and reconstruction approaches exist. However, their output is either a pure polygon soup [30] or a set of planar patches [31]. We establish a computational framework and algorithms for reconstructing structured indoor model from panorama RGBD images. We introduce a novel 3D model representation "structure graph", whose nodes represent structural elements such as rooms, doors, and objects, and the edges represent their geometric relationships. "Structure grammar" then defines a list of possible graph transformations. This grammar drives a principled new reconstruction algorithm, where the rules are sequentially applied to naturally segment, annotate, and reconstruct architectural scenes.

### Raster-to-Vector: Revisiting Floorplan Transformation

Chen Liu Washington University in St. Louis chenliu@wustl.edu  
Pushmeet Kohli DeepMind pushmeet@google.com

Jijun Wu Massachusetts Institute of Technology jijunwu@mit.edu  
Yasutaka Furukawa Simon Fraser University furukawa@sfu.ca

Figure 1. This paper makes a breakthrough in the problem of converting raster floorplan images to vector-graphics representations. From left to right, an input floorplan image, reconstructed vector-graphics representation visualized by our custom renderer, and a pop-up 3D model.

[cs.CV] 31 Mar 2018

### FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans

Chen Liu\* Jiaye Wu\* Yasutaka Furukawa  
Washington University in St. Louis Simon Fraser University  
{chenliu, jiaye.wu}@wustl.edu furukawa@sfu.ca

**Abstract.** The ultimate goal of this indoor mapping research is to automatically reconstruct a floorplan simply by walking through a house with

### Neural Procedural Reconstruction for Residential Buildings

Huayi Zeng<sup>1</sup>, Jiaye Wu<sup>1</sup> and Yasutaka Furukawa<sup>2</sup>

<sup>1</sup> Washington University in St. Louis, USA {zengh, jiaye.wu}@wustl.edu  
<sup>2</sup> Simon Fraser University, Canada furukawa@sfu.ca

**Abstract.** This paper proposes a novel 3D reconstruction approach, dubbed Neural Procedural Reconstruction (NPR). NPR infers a sequence of shape grammar rule applications and reconstructs CAD-quality models with procedural structure from 3D points. While most existing methods rely on low-level geometry analysis to extract primitive structures,

### PlaneNet: Piece-wise Planar Reconstruction from a Single RGB Image

Chen Liu<sup>1</sup> Jimei Yang<sup>2</sup> Duygu Ceylan<sup>2</sup> Ersin Yumer<sup>3</sup> Yasutaka Furukawa<sup>4</sup>

<sup>1</sup>Washington University in St. Louis chenliu@wustl.edu  
<sup>2</sup>Adobe Research [jisyang, ceylan]@adobe.com  
<sup>3</sup>Argo AI meyumer@gmail.com  
<sup>4</sup>Simon Fraser University furukawa@sfu.ca

Figure 1. This paper proposes a deep neural architecture for piece-wise planar depthmap reconstruction from a single RGB image. From left to right, an input image, a piece-wise planar segmentation, a reconstructed depthmap, and a texture-mapped 3D model.

### Reconstructing the World's Museums

Jianxiang Xiao - Yasutaka Furukawa

Received date / Accepted date

**Abstract** Virtual exploration tools for large indoor environments (e.g. museums) have so far been limited to either blueprint-style 2D maps that lack photo-realistic views of scenes, or ground-level image-to-image transitions, which are immersive but ill-suited for navigation. On the other hand, photo-realistic aerial maps would be a useful navigational guide for large indoor environments, but it is impossible to directly acquire photographs covering a large indoor environment from aerial viewpoints. This paper presents a 3D reconstruction and visualization system for automatically pro-

### Reconstructing Building Interiors from Images

Yasutaka Furukawa, Brian Curless, Steven M. Seitz  
University of Washington, Seattle, USA {furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research, Redmond, USA rszeliski@microsoft.com

**Abstract**  
This paper proposes a fully automated 3D reconstruction and visualization system for architectural scenes (interiors and exteriors). The reconstruction of indoor environments from photographs is particularly challenging due to texture-poor planar surfaces such as uniformly-painted walls. Our system first uses structure-from-motion, multi-view stereo, and sequential plane fitting to reconstruct the

Figure 1: Floor plan and photograph of a house interior.

### Manhattan-world Stereo

Yasutaka Furukawa Brian Curless Steven M. Seitz  
Department of Computer Science & Engineering University of Washington, USA {furukawa, curless, seitz}@cs.washington.edu

Richard Szeliski  
Microsoft Research Redmond, USA rszeliski@microsoft.com

**Abstract**  
Multi-view stereo (MVS) algorithms now produce reconstructions that rival laser range scanner accuracy. However, stereo algorithms require textured surfaces, and therefore work poorly for many architectural scenes (e.g., building interiors with textureless, painted walls). This paper presents a novel MVS approach to overcome these limi-

Figure 1. Increasingly ubiquitous on the Internet are images of architectural scenes with texture-poor but highly structured surfaces.

### PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image

Chen Liu<sup>1,2\*</sup> Kibwan Kim<sup>1</sup> Jiwon Gu<sup>1,3\*</sup> Yasutaka Furukawa<sup>1</sup> Jan Kazda<sup>2</sup>

<sup>1</sup>NVIDIA  
<sup>2</sup>SceneTime  
<sup>3</sup>Washington University in St. Louis  
<sup>4</sup>Simon Fraser University

Figure 1. This paper proposes a deep neural architecture, PlaneRCNN, that detects planar regions and reconstructs a piecewise planar depthmap from a single RGB image. From left to right, an input image, segmented plane regions, estimated depthmap, and reconstructed planar surfaces.

**Abstract** plane reconstruction from point clouds [1], and Manhattan

### Floor-SP: Inverse CAD for Floorplans by Sequential Room-wise Shortest Path

Jiacheng Chen<sup>1</sup> Chen Liu<sup>2</sup> Jiaye Wu<sup>2</sup> Yasutaka Furukawa<sup>1</sup>

<sup>1</sup>Simon Fraser University {jca447, furukawa}@sfu.ca  
<sup>2</sup>Washington University in St. Louis {chenliu, jiaye.wu}@wustl.edu

Figure 1. The proposed system, dubbed Floor-SP, takes aligned panorama RGBD scans as input, finds room segments, solves an optimization problem to reconstruct a floorplan graph as multiple polygonal loops (one for each room), and merges them into a 2D graph via simple post-processing heuristics. The optimization is the technical contribution of the paper, which employs the room-wise coordinate descent strategy and sequentially solves shortest path problems to optimize the room structure.

### Conv-MPN: Convolutional Message Passing Neural Network for Structure

**Input** Conv-MPN reconstruction Ground-truth

Iteration 0 Iteration 1 Iteration 3

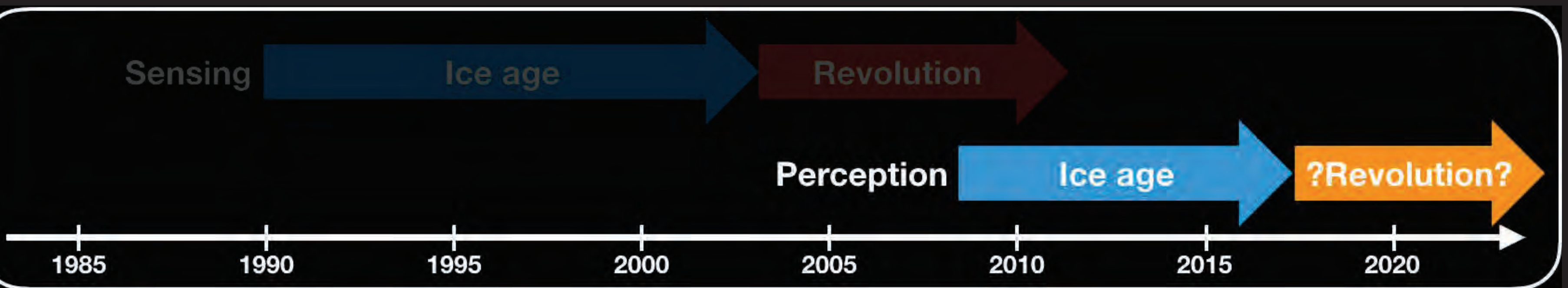
Figure 1. Conv-MPN, a novel message passing neural network, reconstructs outdoor buildings as planar graphs from a single image. The reconstructions after 0, 1, or 3 iterations of message passing are as shown.

### Vectorizing World Buildings: Planar Graph Reconstruction by Primitive Detection and Relationship Classification

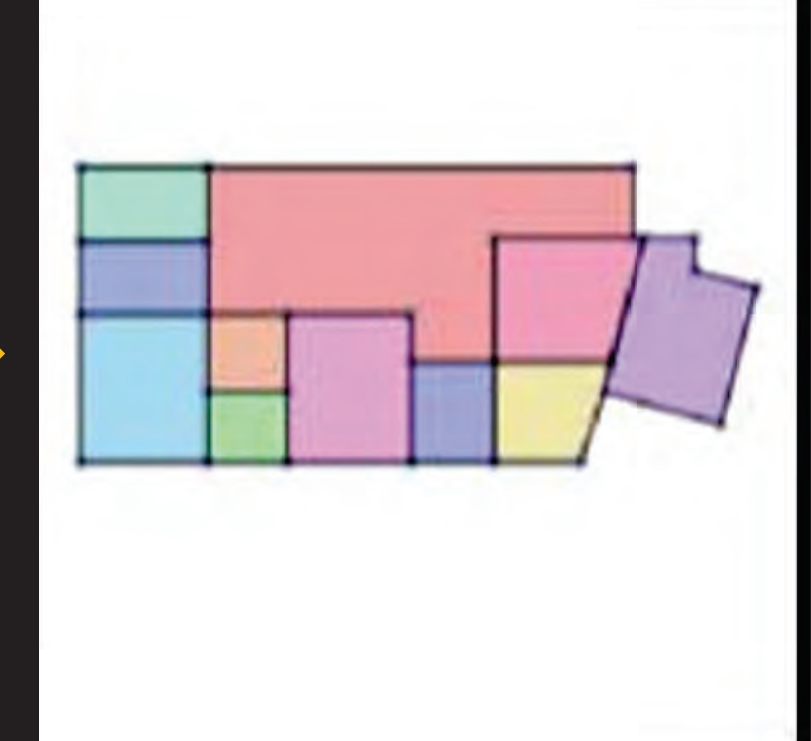
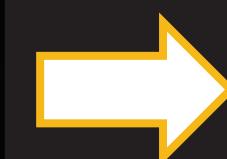
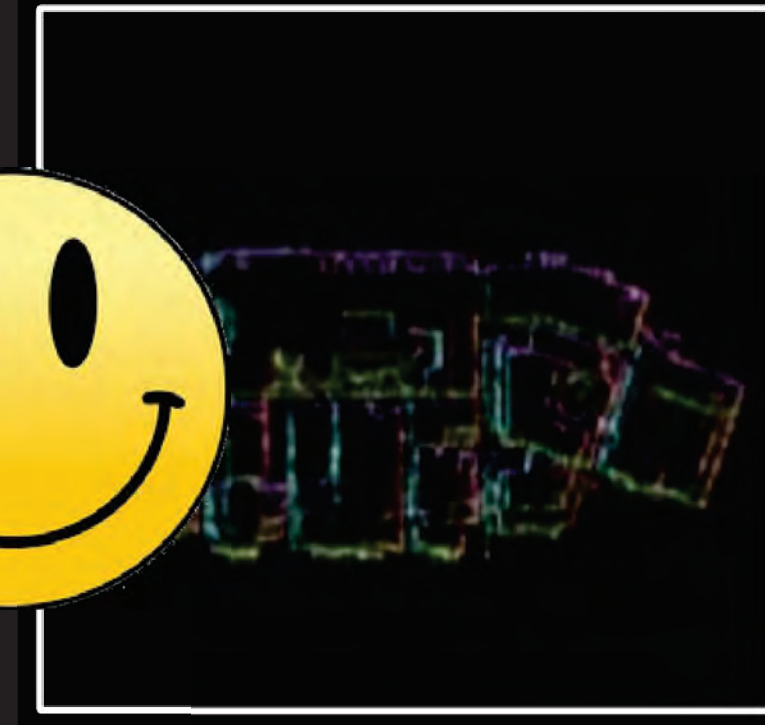
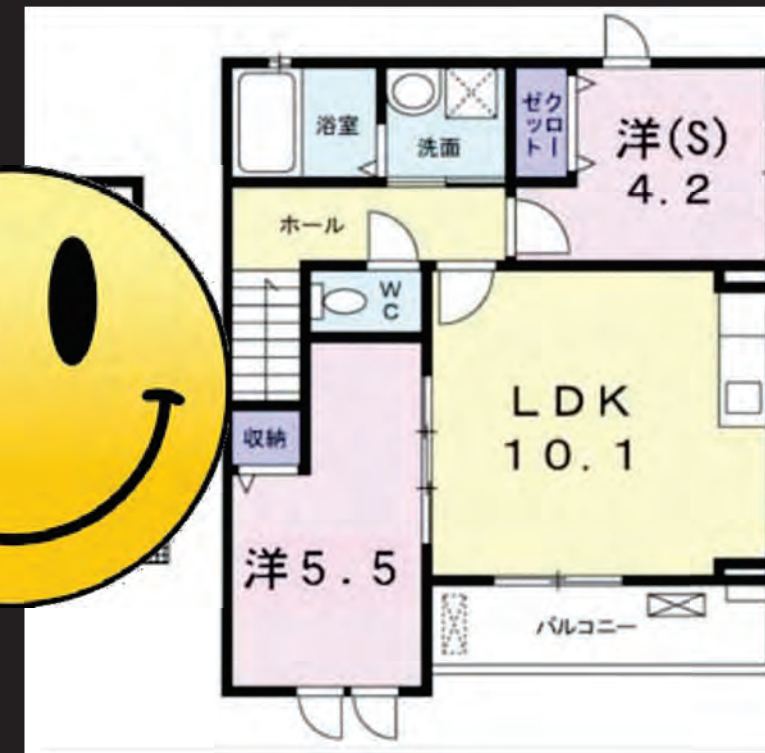
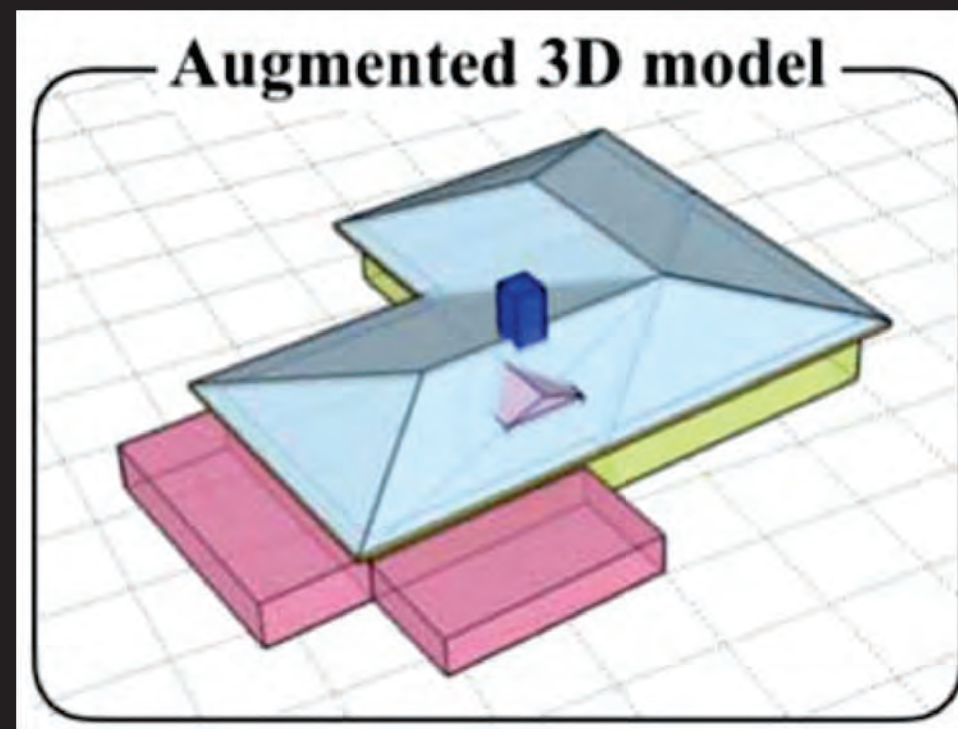
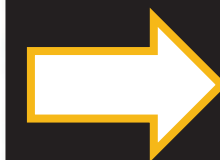
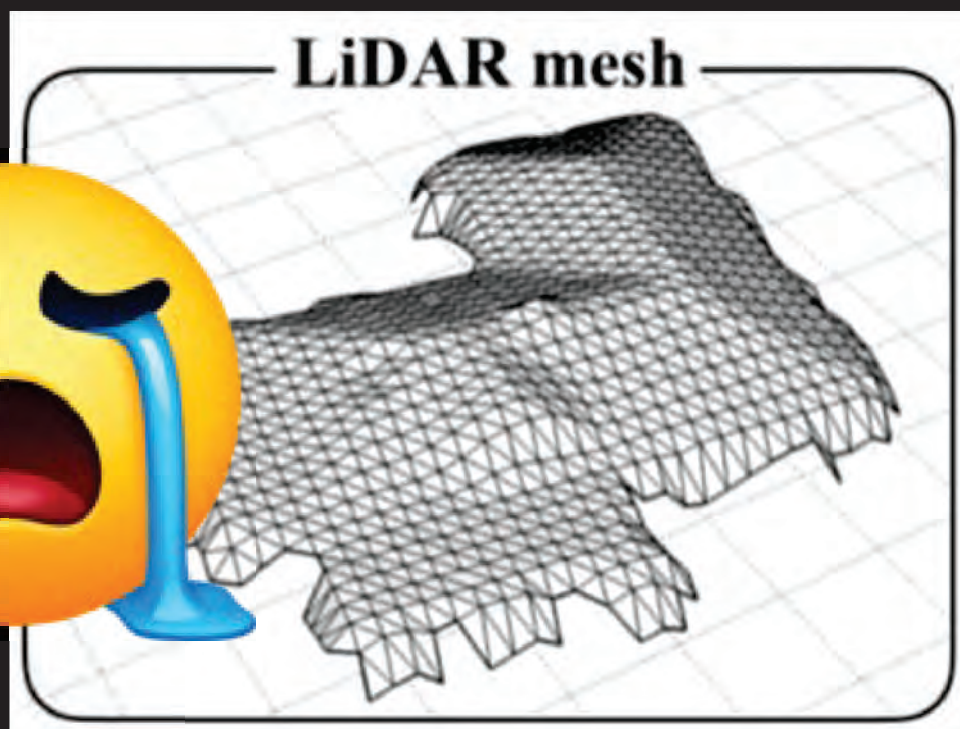
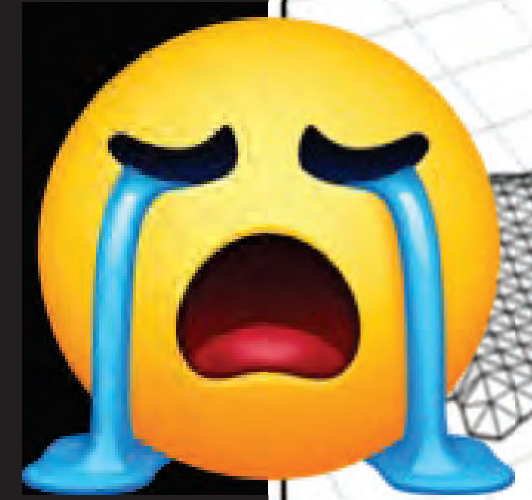
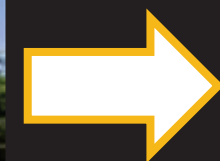
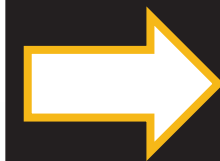
RGB input Corners Edges Regions Corner-to-edge Region-to-region Output Ground-truth

Figure 1. The paper takes a RGB image, detects three geometric primitives (i.e., corners, edges, and regions), classifies their relationships (i.e., corner-to-edge and region-to-region), and fuses the information via Integer Programming to reconstruct a planar graph.

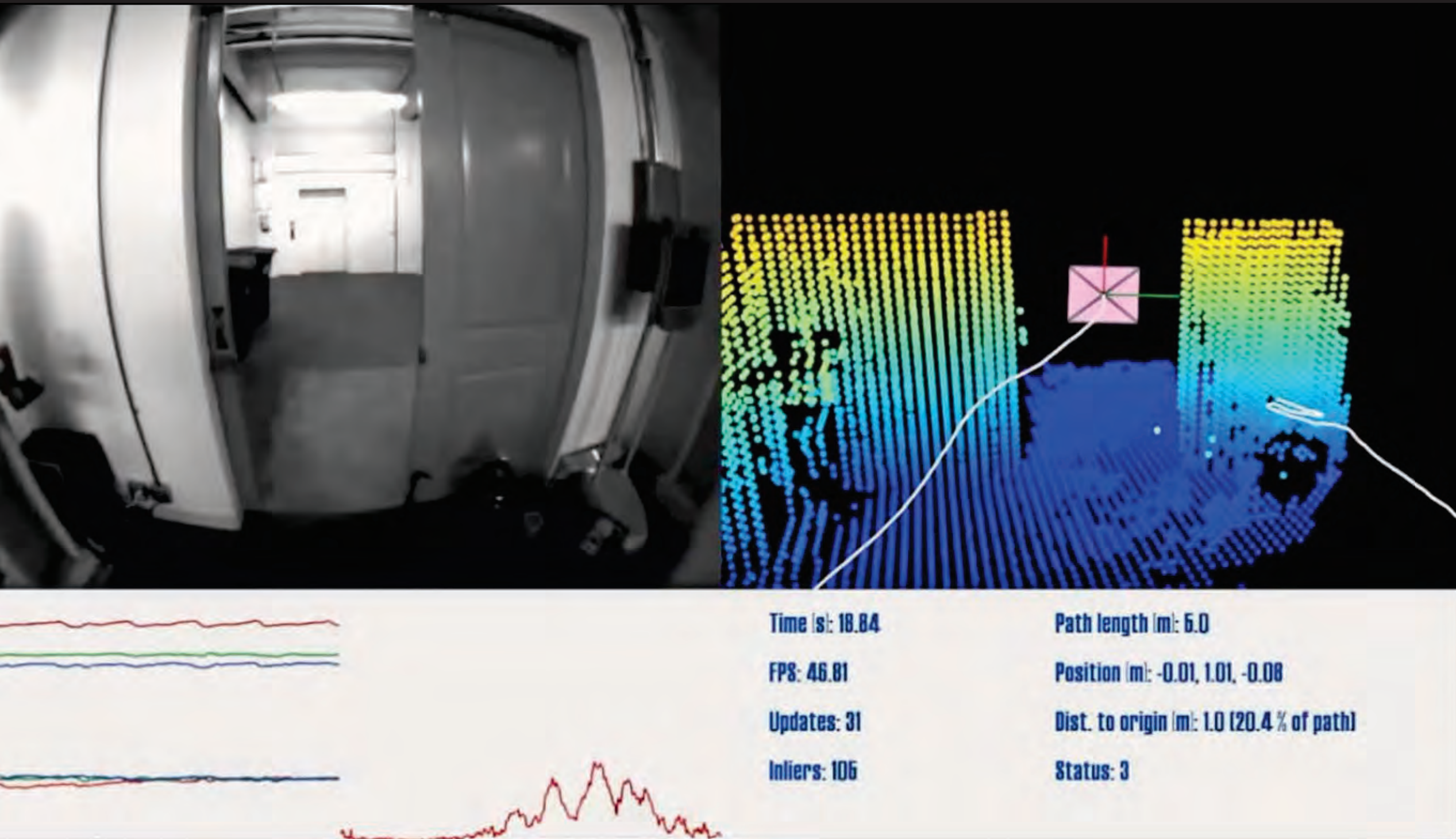
# CVPR-ICCV-ECCV papers for geometry perception...



# Ice-age or revolution?



# Sensing

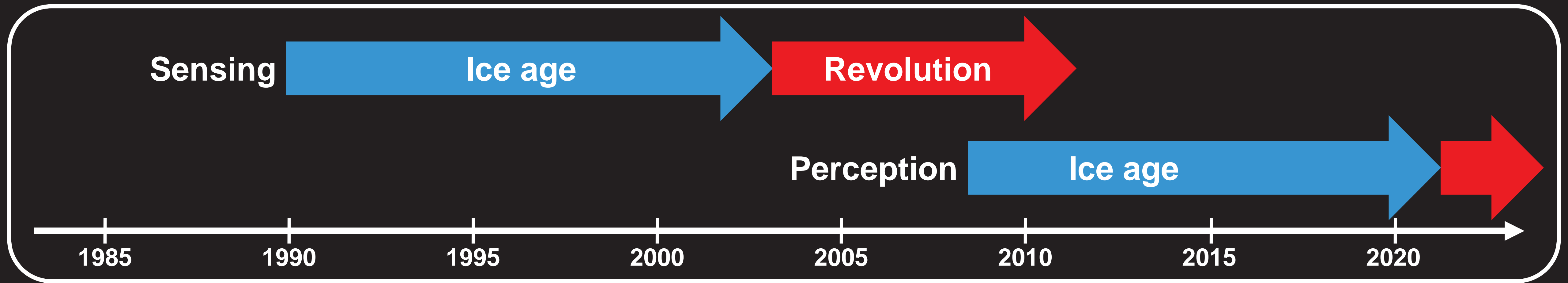


[ Google Project Tango ]

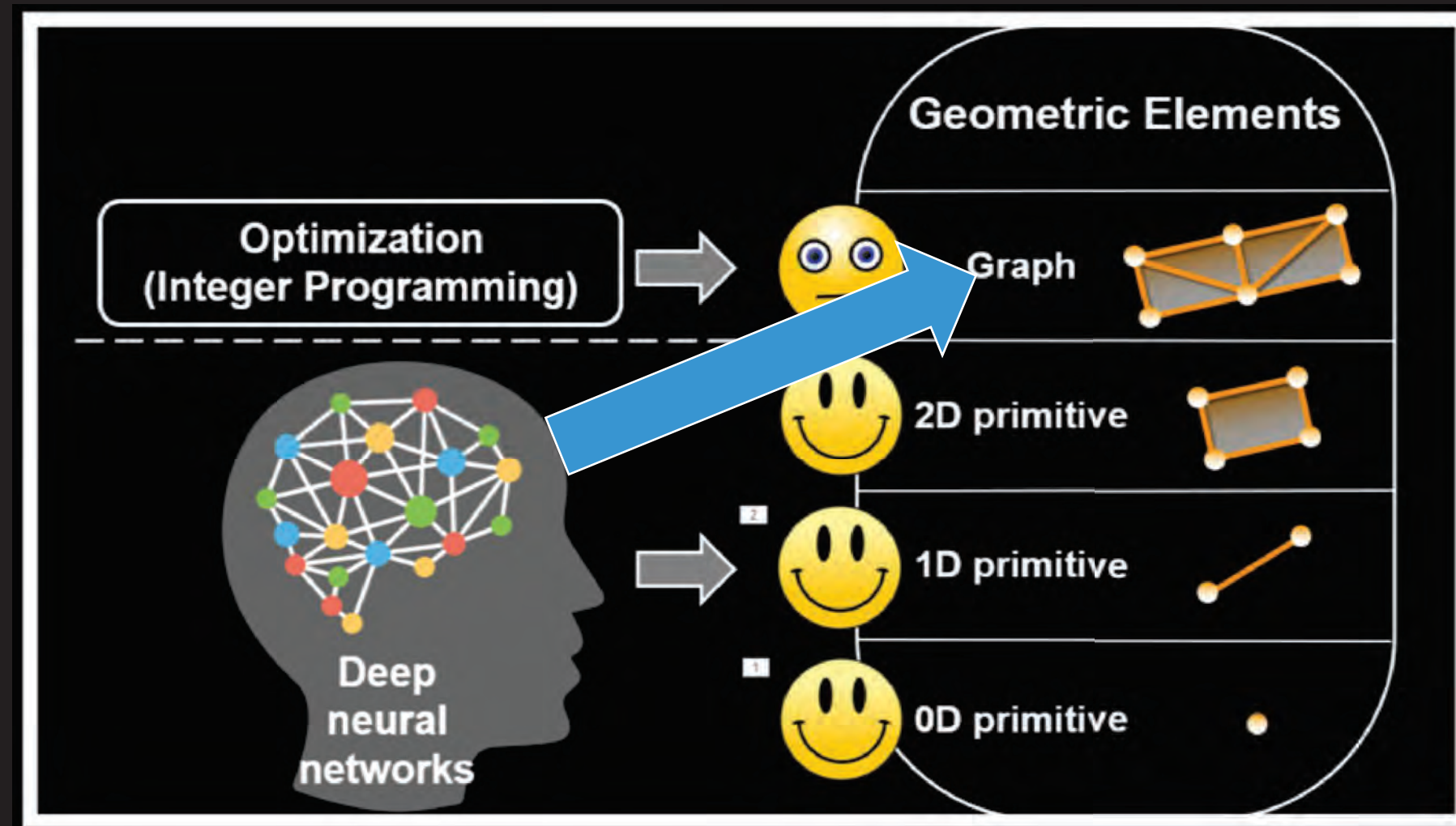
# Perception



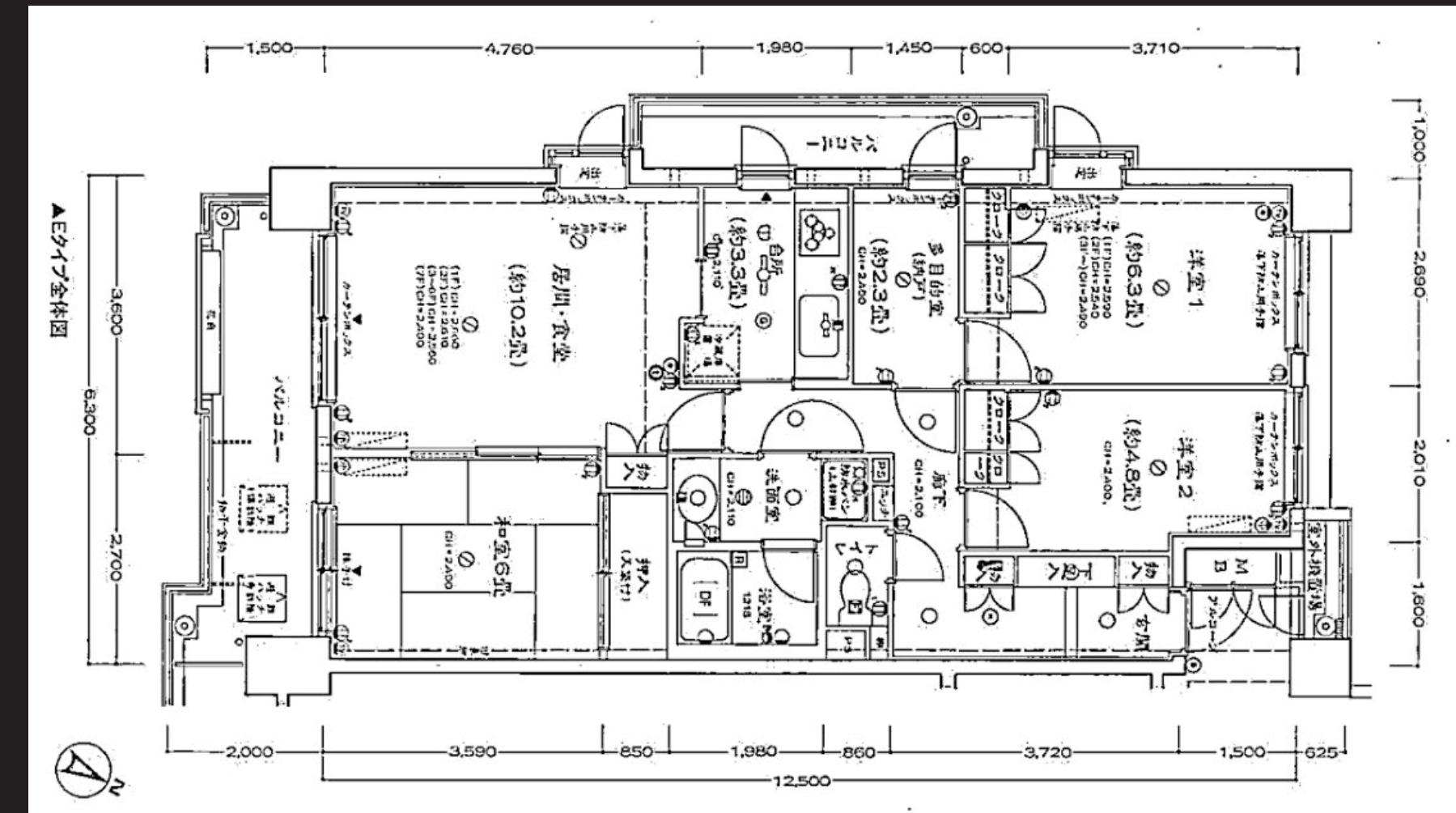
[ CANVAS by Occipital (<https://canvas.io>) ]



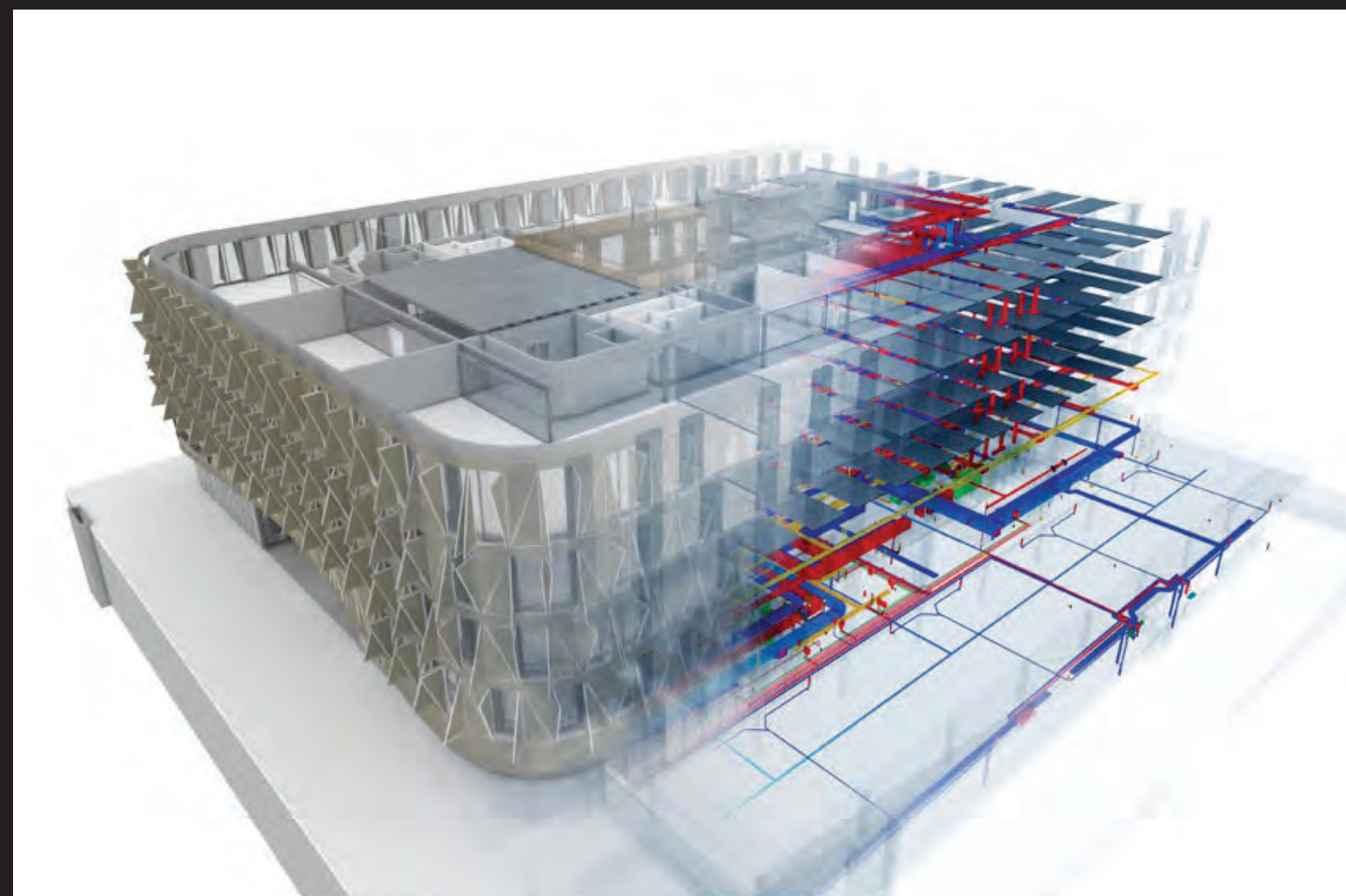
# Future challenges



DNN for high-level perception



Construction-level perception



3D perception



Image (smartphone) only

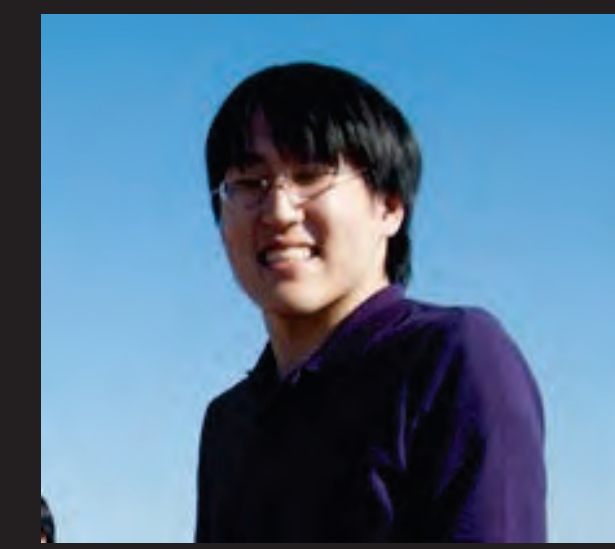
# Acknowledgements



Jean Ponce   Steve Seitz   Brian Curless   Rick Szeliski



Yoji Kiyota



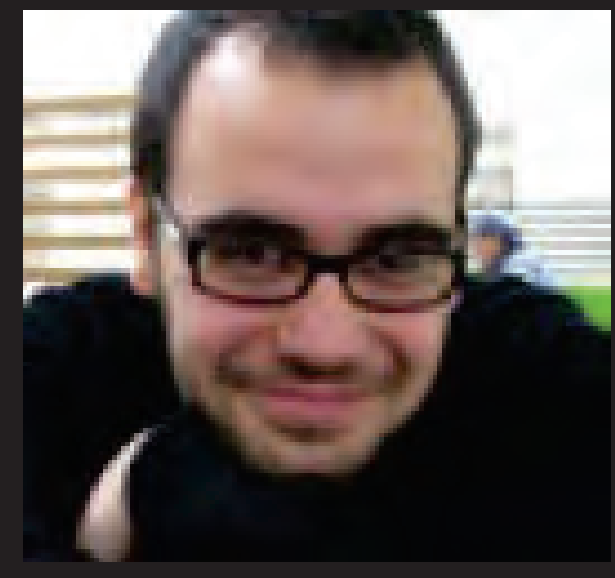
Satoshi Ikehata



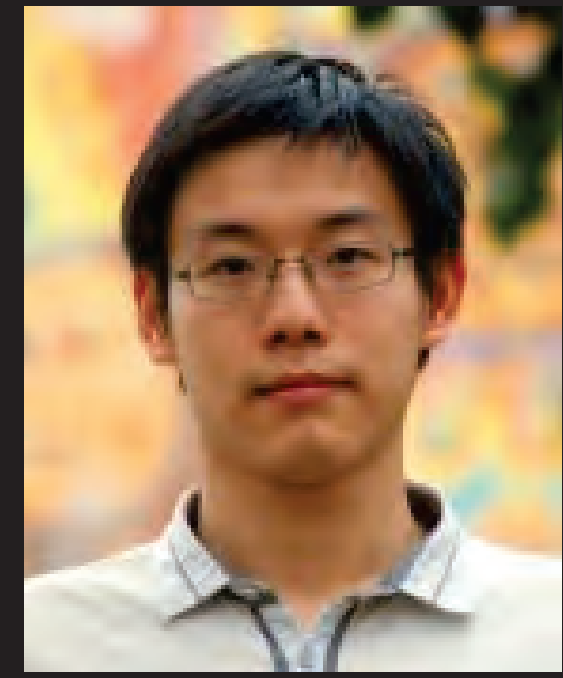
Tomoko Ohsuga



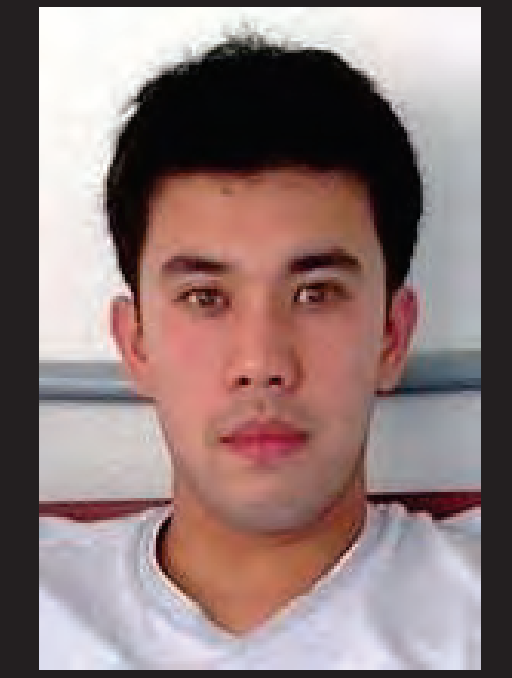
Pushmeet Kohli



Ricardo Cabral



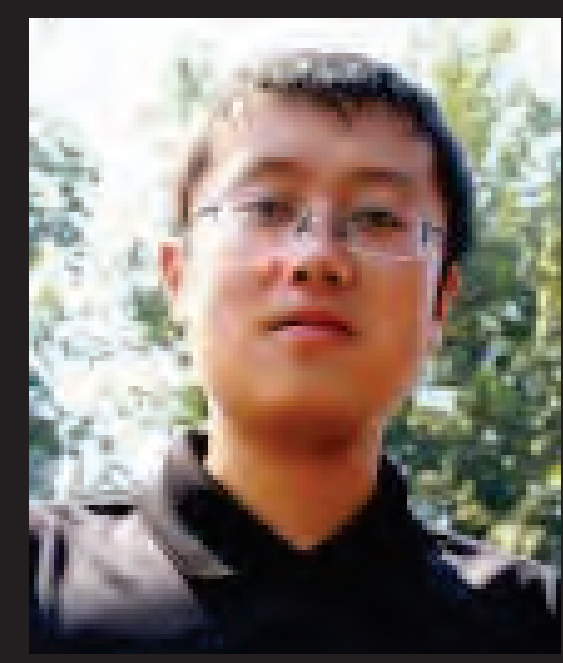
Jiajun Wu



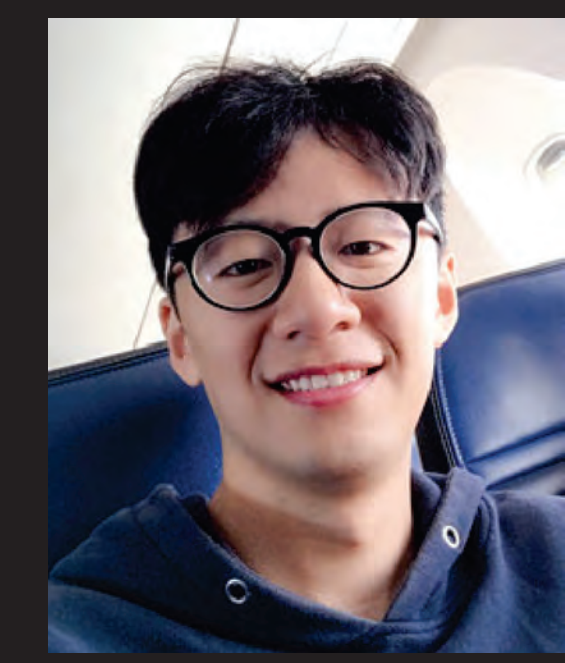
Nelson Nauata



Chen Liu



Hang Yan



Huayi Zeng



Jiachen Chen