Structured 3D Reconstruction:
From geometric sensing to perception

Yasutaka Furukawa

SFU AI

SIMON FRASER UNIVERSITY

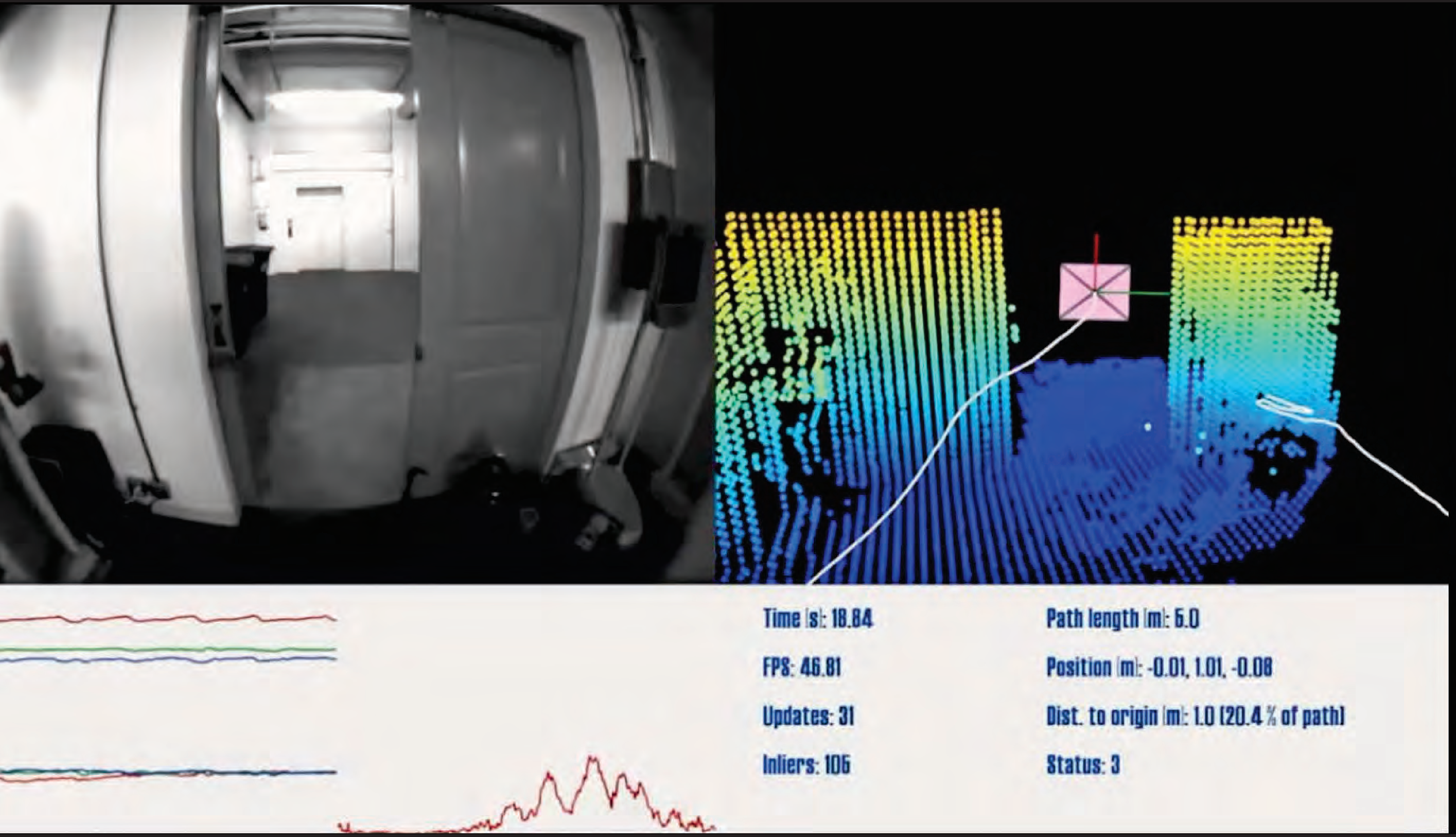
Evolution of 3D Reconstruction Techniques

Sensing

Perception

Sensing

Perception



[Google Project Tango]

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

Ice-age, 20th Century Fox

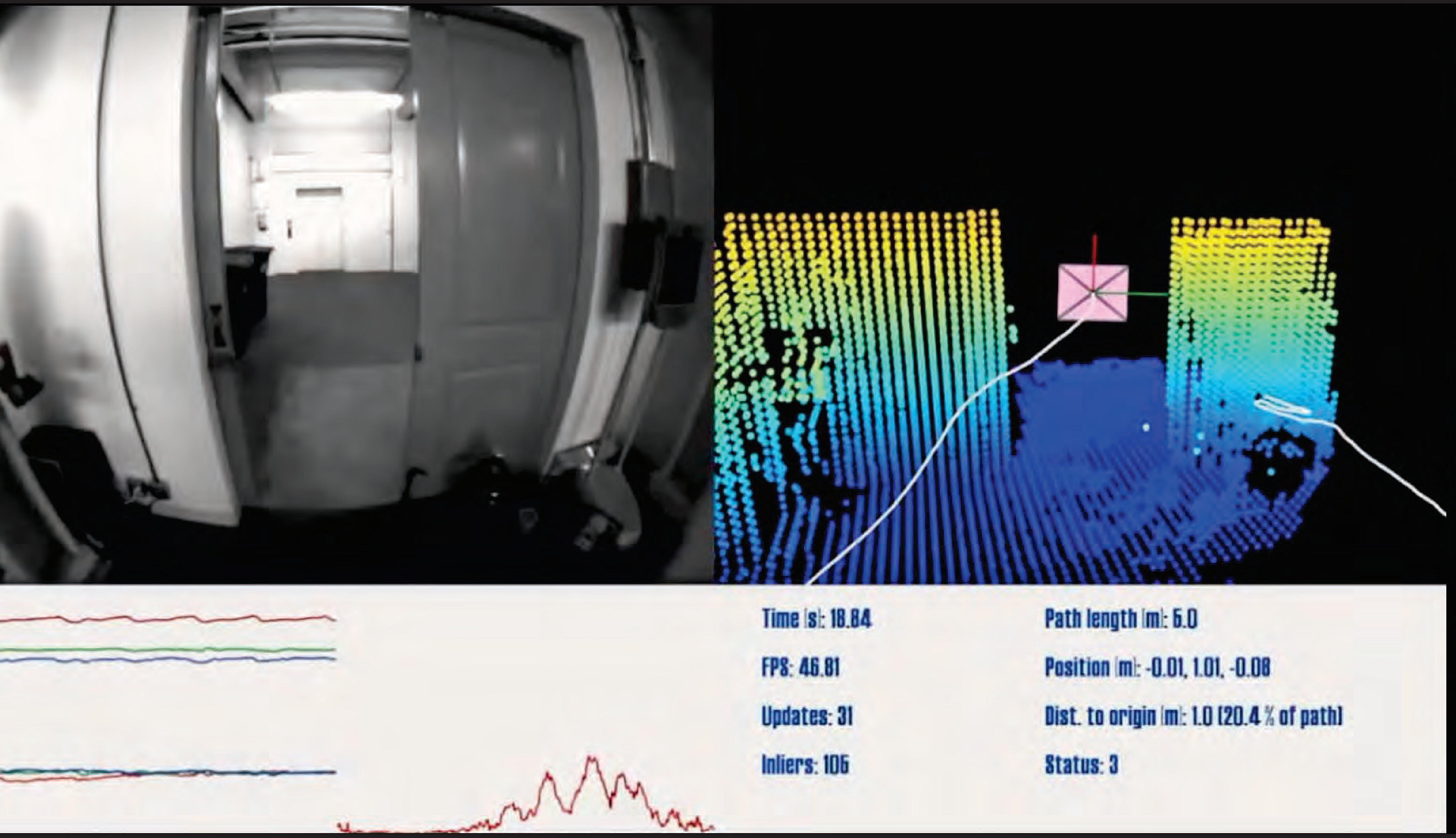
Ice-age: A long period of reduction in the temperature of the Earth’s surface and atmosphere [wikipedia]

Ice-age, 20th Century Fox

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

Sensing

Perception

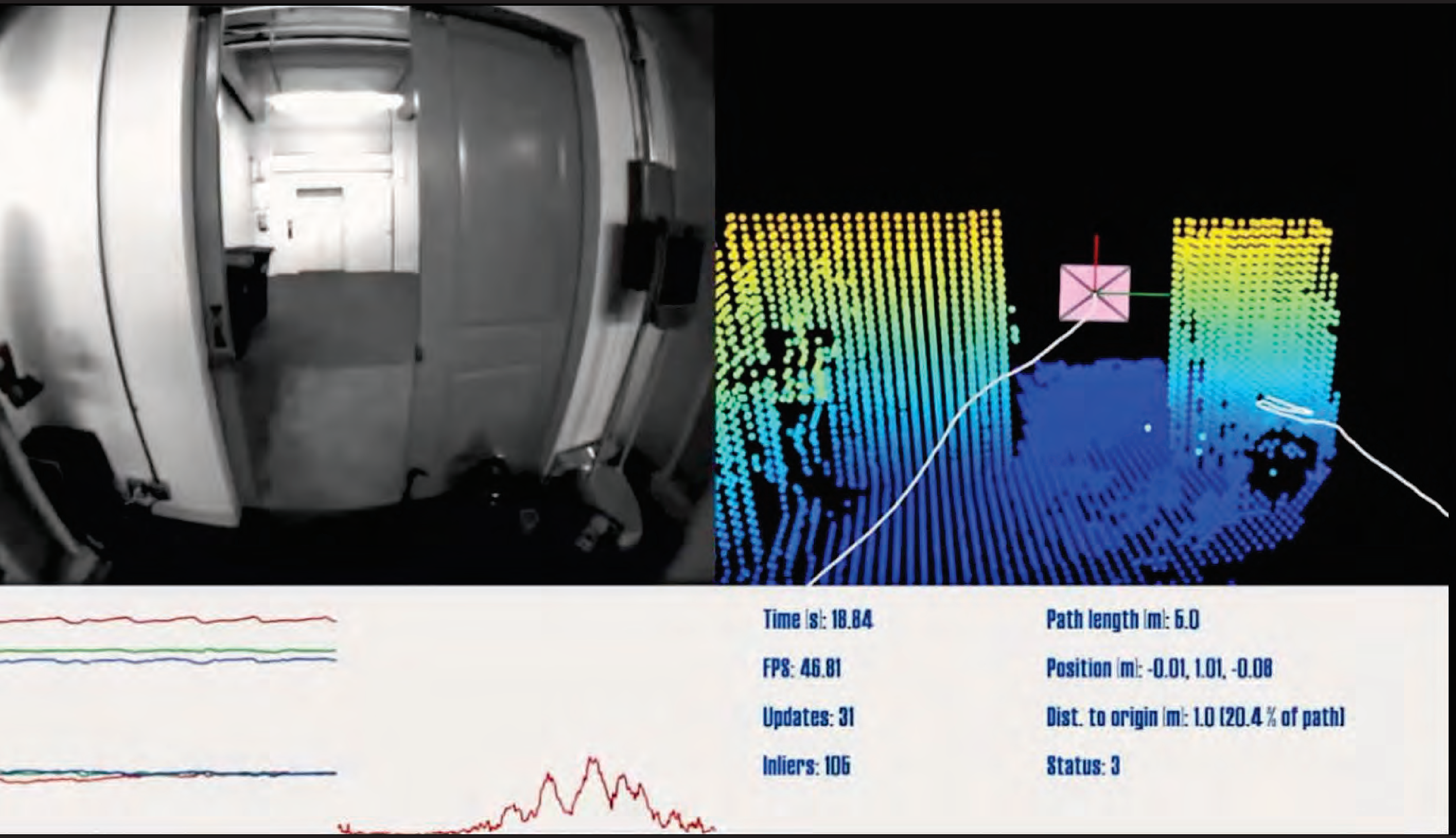


[Google Project Tango]

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

Sensing

Perception

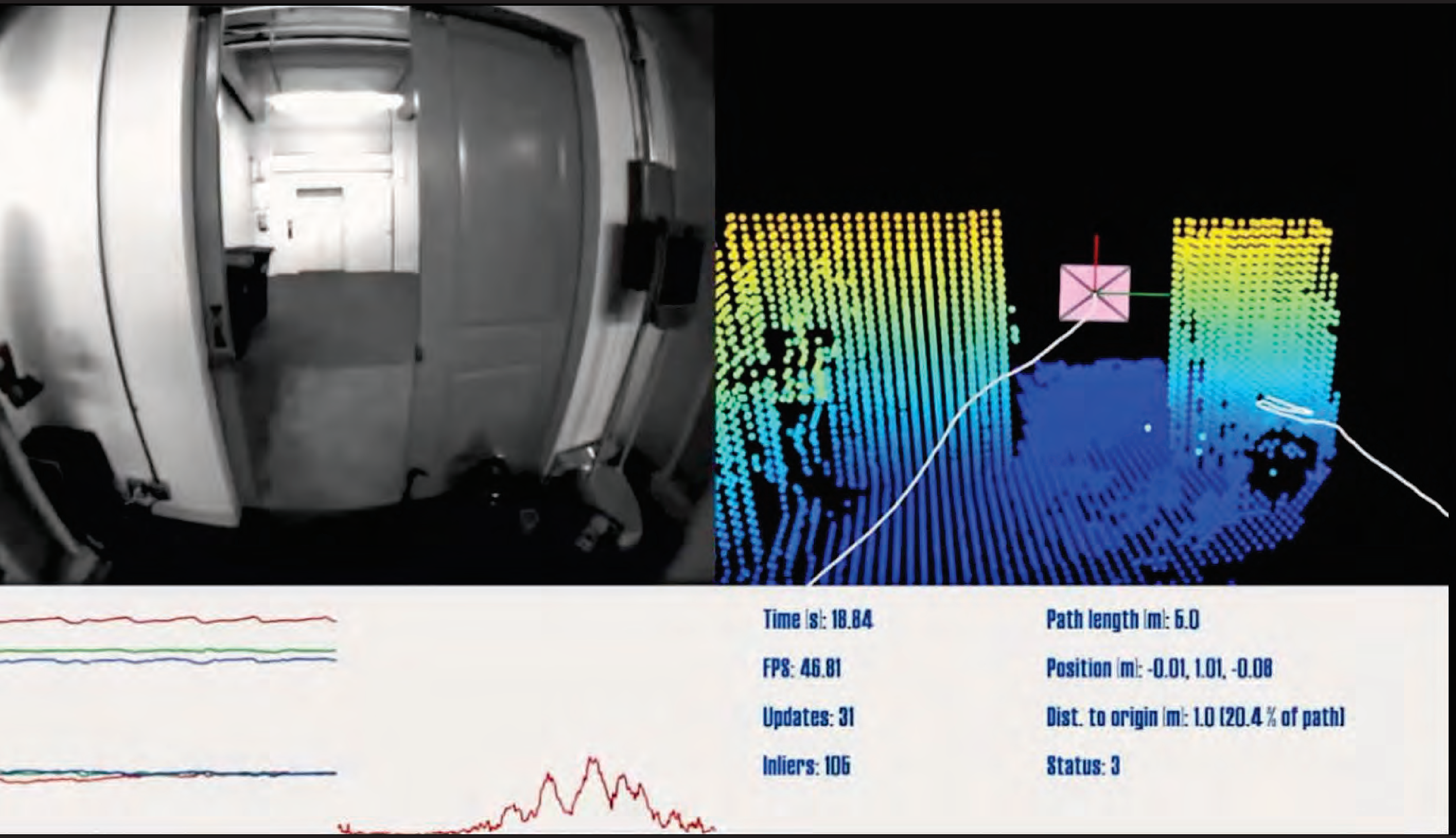


[Google Project Tango]

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

Sensing

Perception



[Google Project Tango]

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


Sensing revolution
2007
2009
2010
2011
2012
The Metropolitan Museum of Art
2012
Perception - ice-age/revolution?
2014
2015
bedroom
bedroom
bathroom
washroom
corridor
restroom
balcony
2017
2018
Point cloud (N=50000)
Images
PointNet Branch
Floorplan Branch
Image Branch
2018
2019

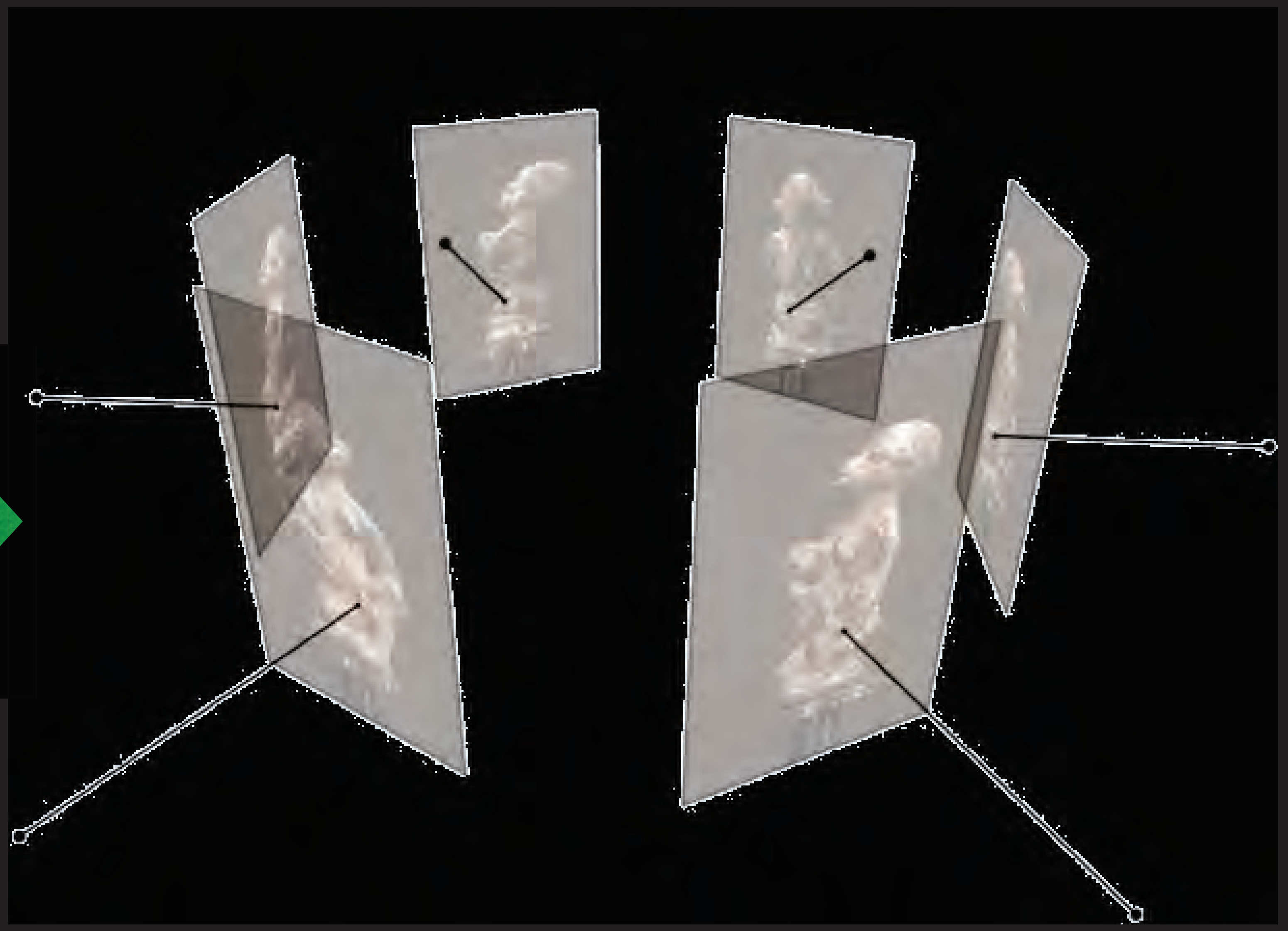

Sensing ice age
ICE AGE
THE MELTDOWN
2000
Sensing revolution
2007
2009
2010
2011
2012
The Metropolitan Museum of Art
2012
Perception - ice-age/revolution?
2014
2015
bedroom
bedroom
bathroom
washroom
corridor
restroom
balcony
2017
2018
2018
2019

Problem setting

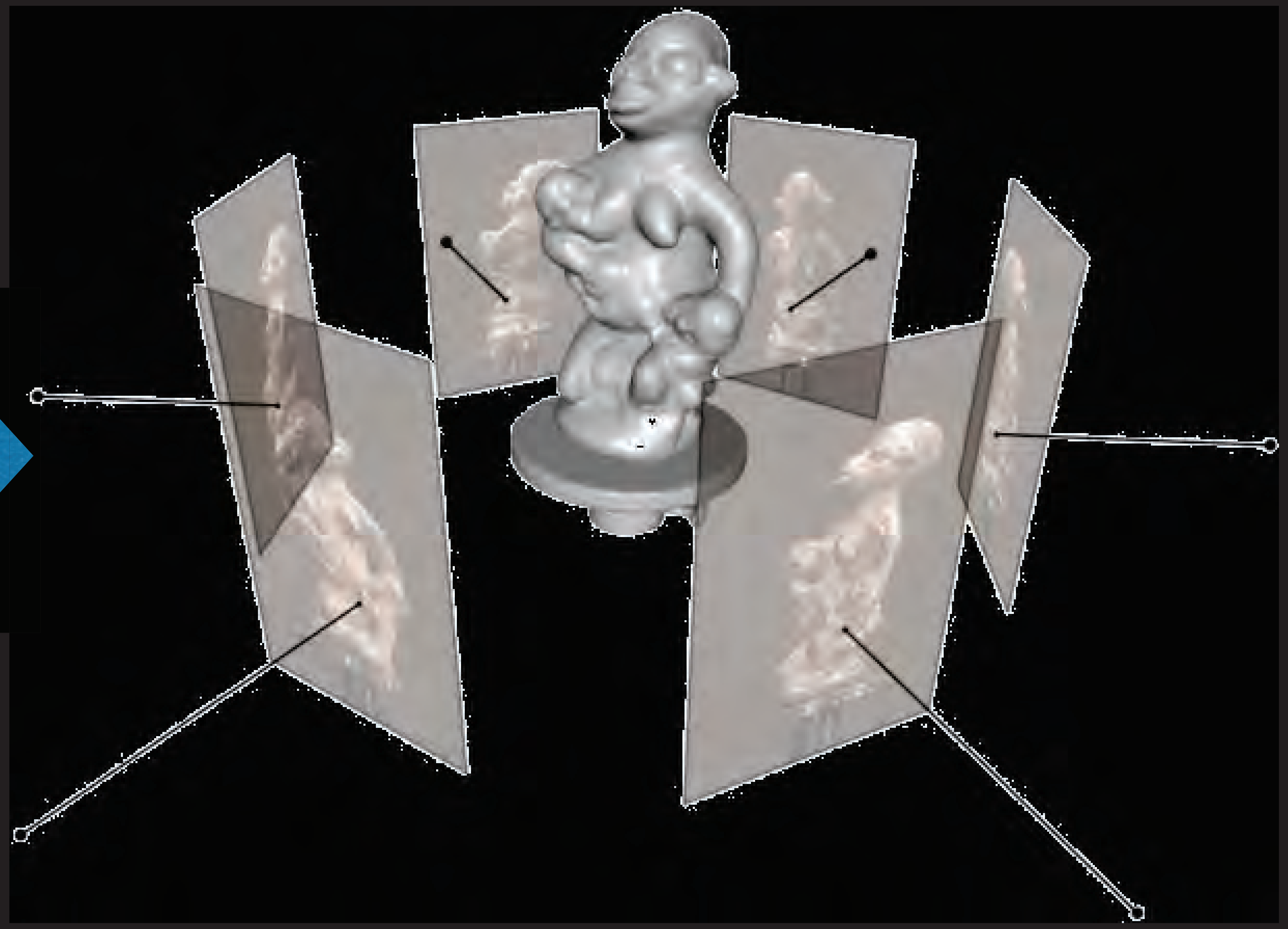
- Input
  - Visual sensor data (e.g., image, depth-images, point-clouds, …)
  - Calibration and camera poses (e.g., intrinsic/extrinsic camera params)
- Output
  - Geometric model (e.g., point-clouds, meshes, …)

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

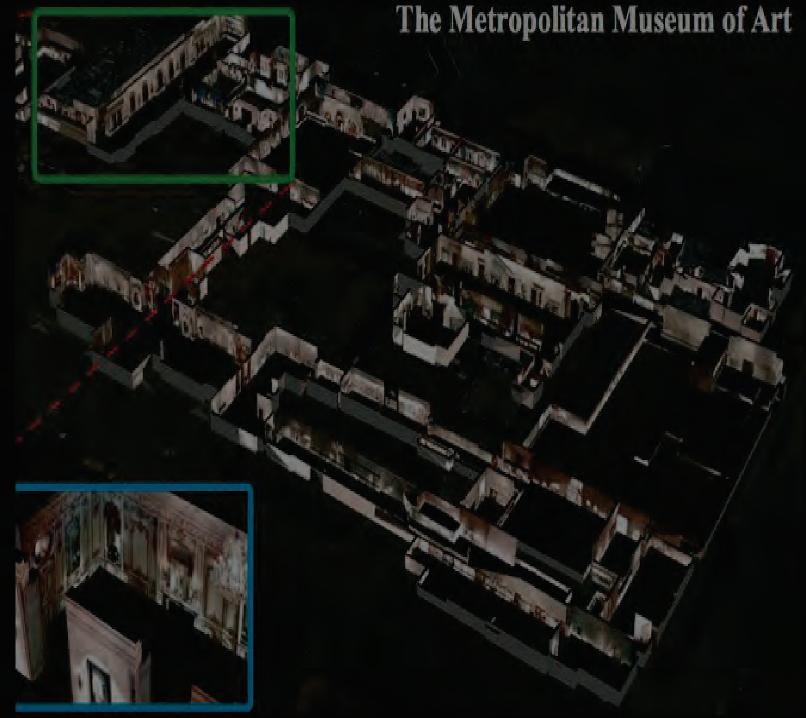
2007

2009

2010

2011

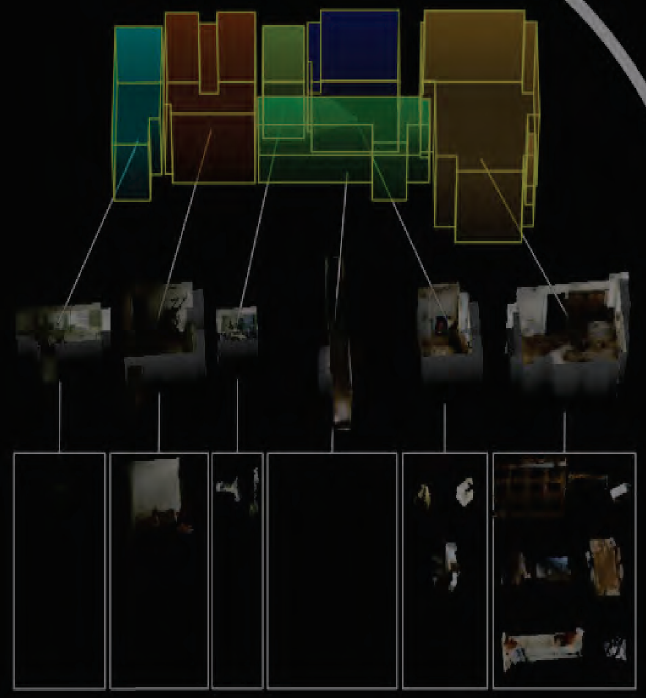
2012



2012

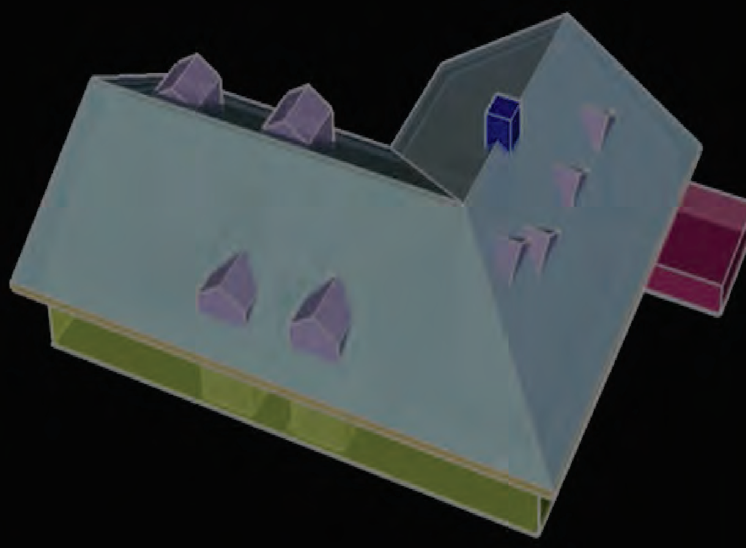
Perception - ice-age/revolution?

2014

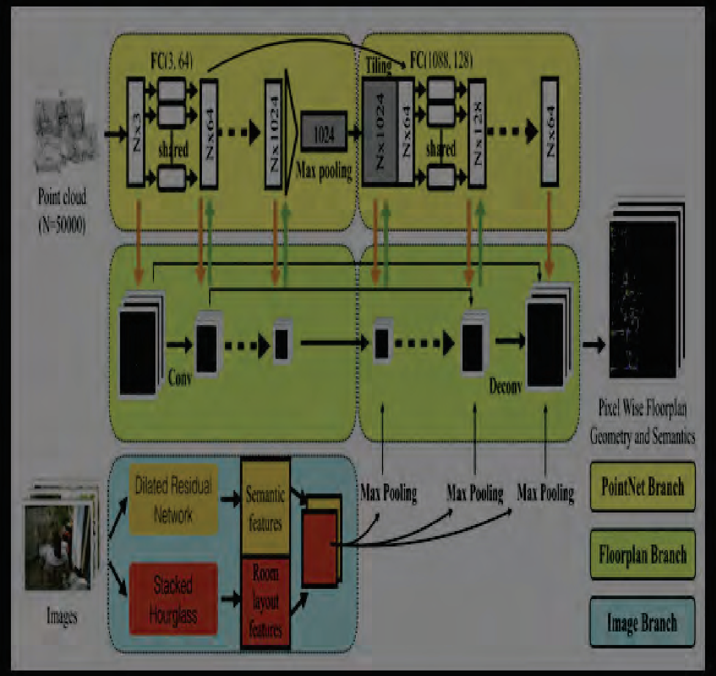


2015

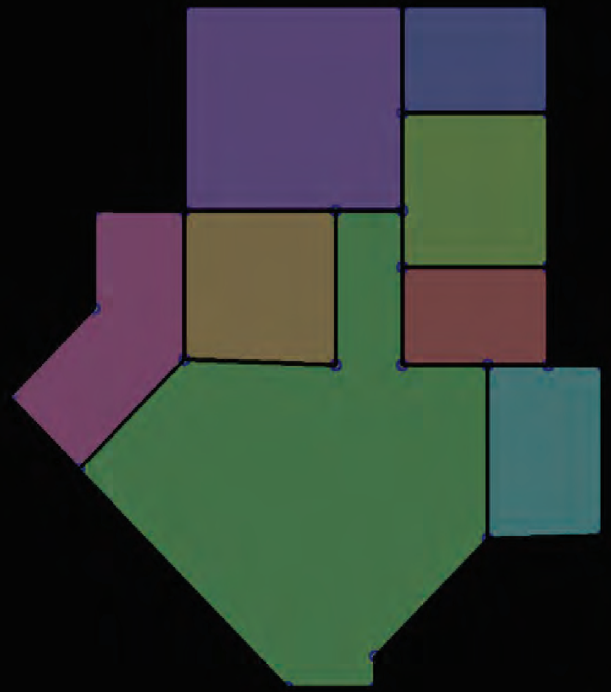
2017



2018



2018



2019

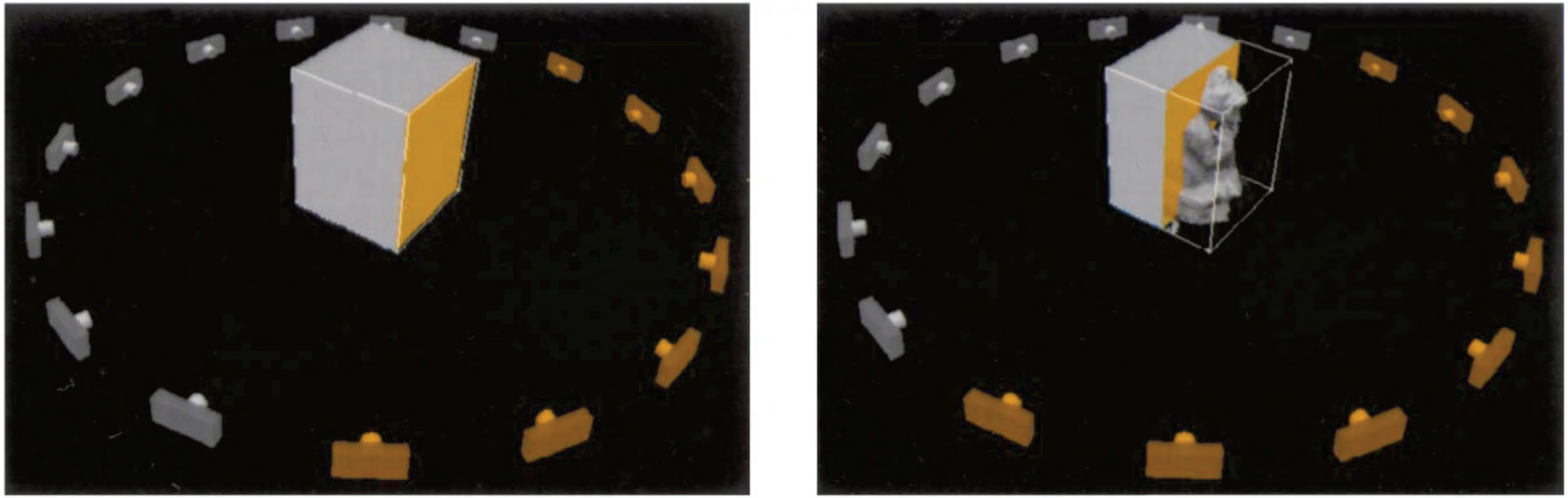
3 mistakes in “sensing ice-age”

- 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.

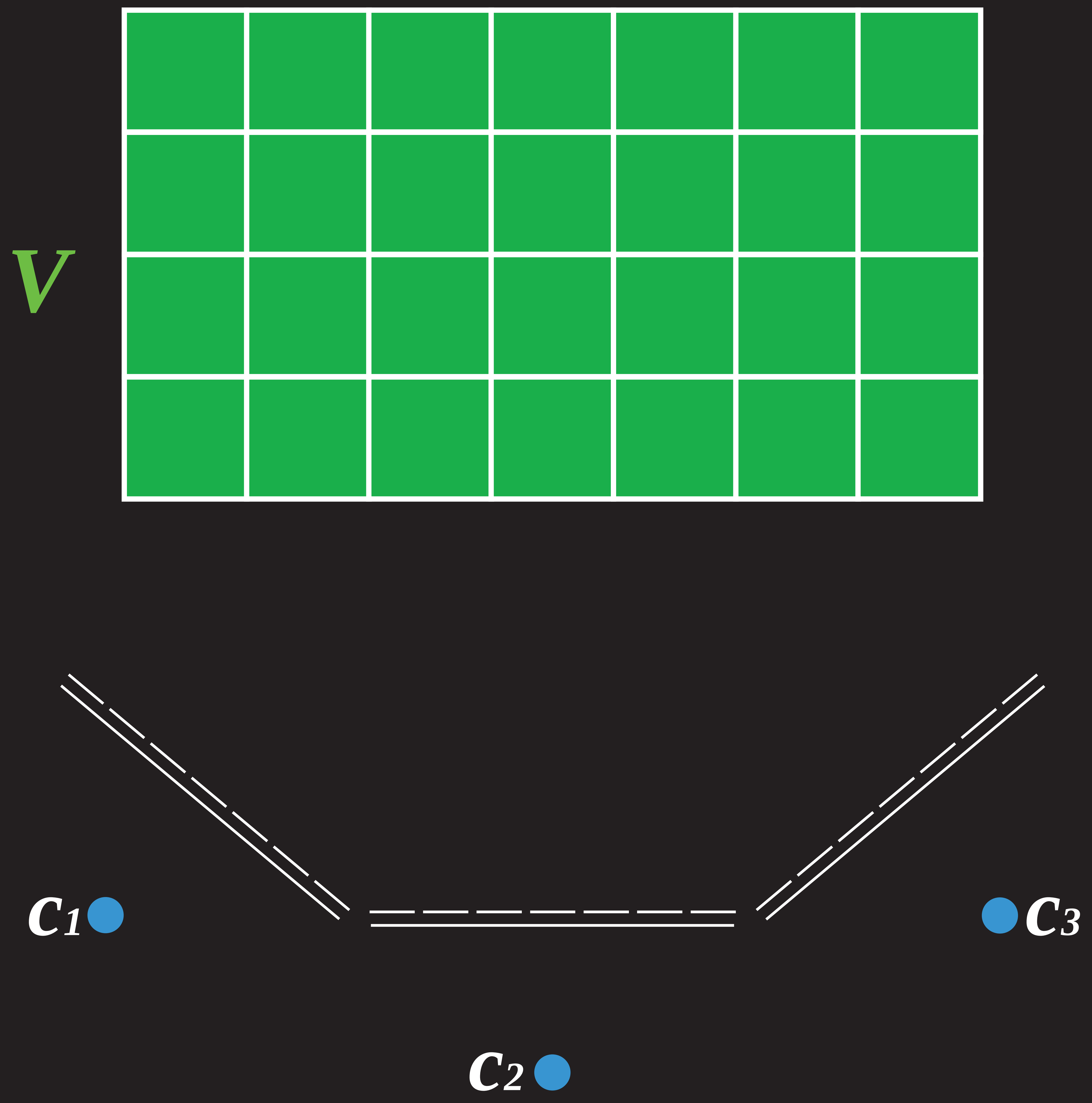
Space Carving: An algorithm

Plane Sweep Algorithm

Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).
Step 2: Intersect Π with the current shape $\mathcal{V}$.
Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

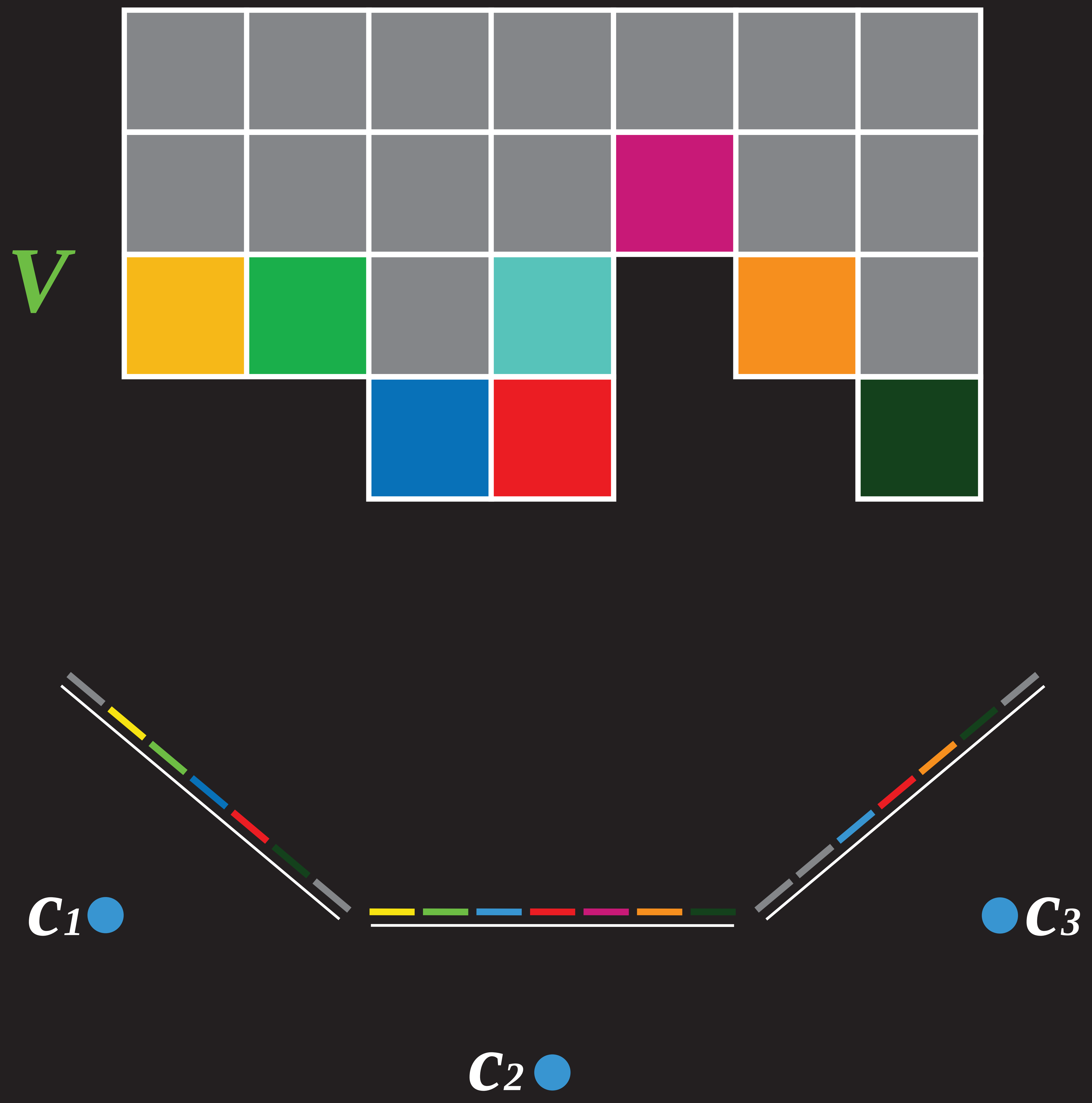
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

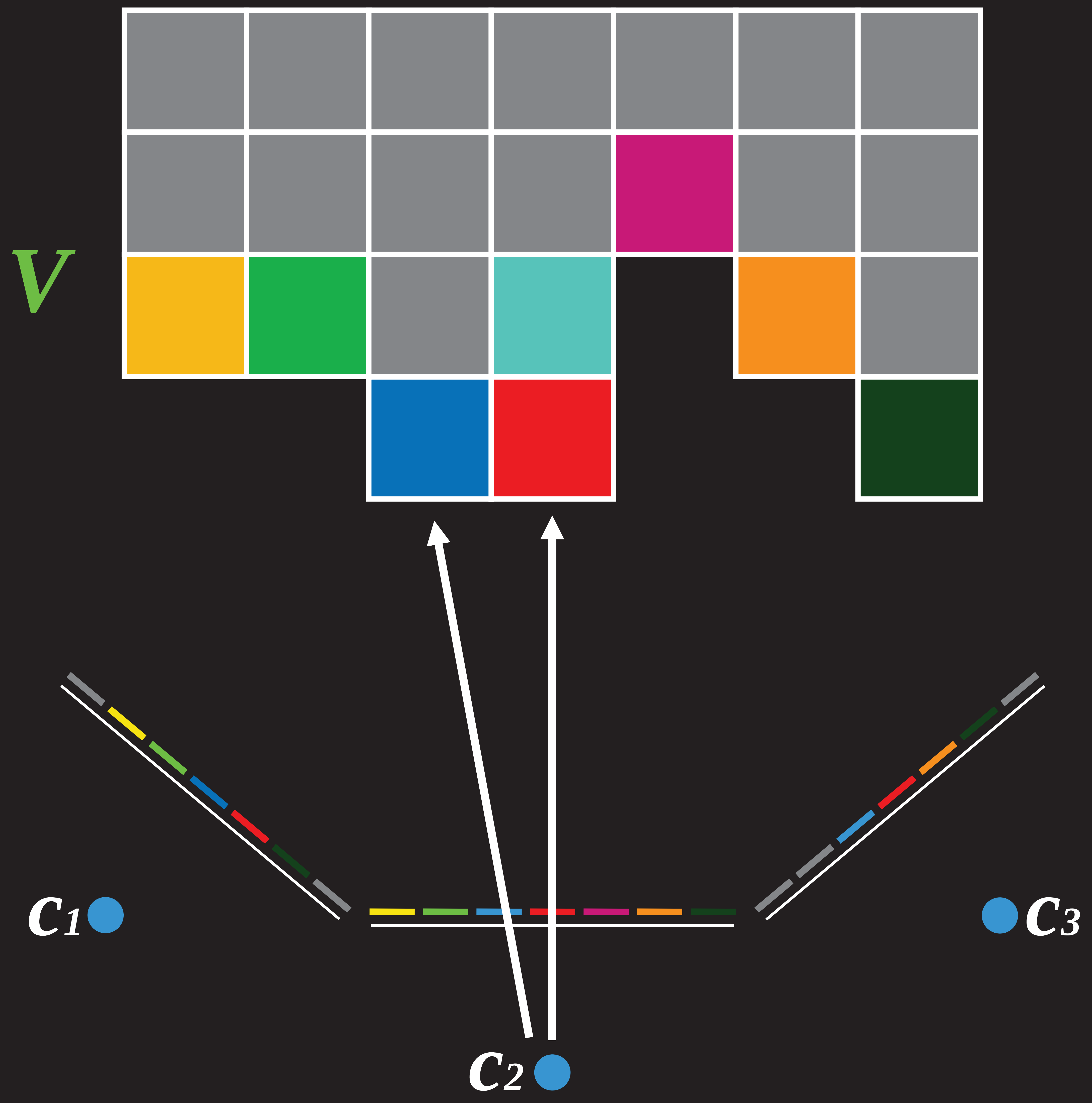
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

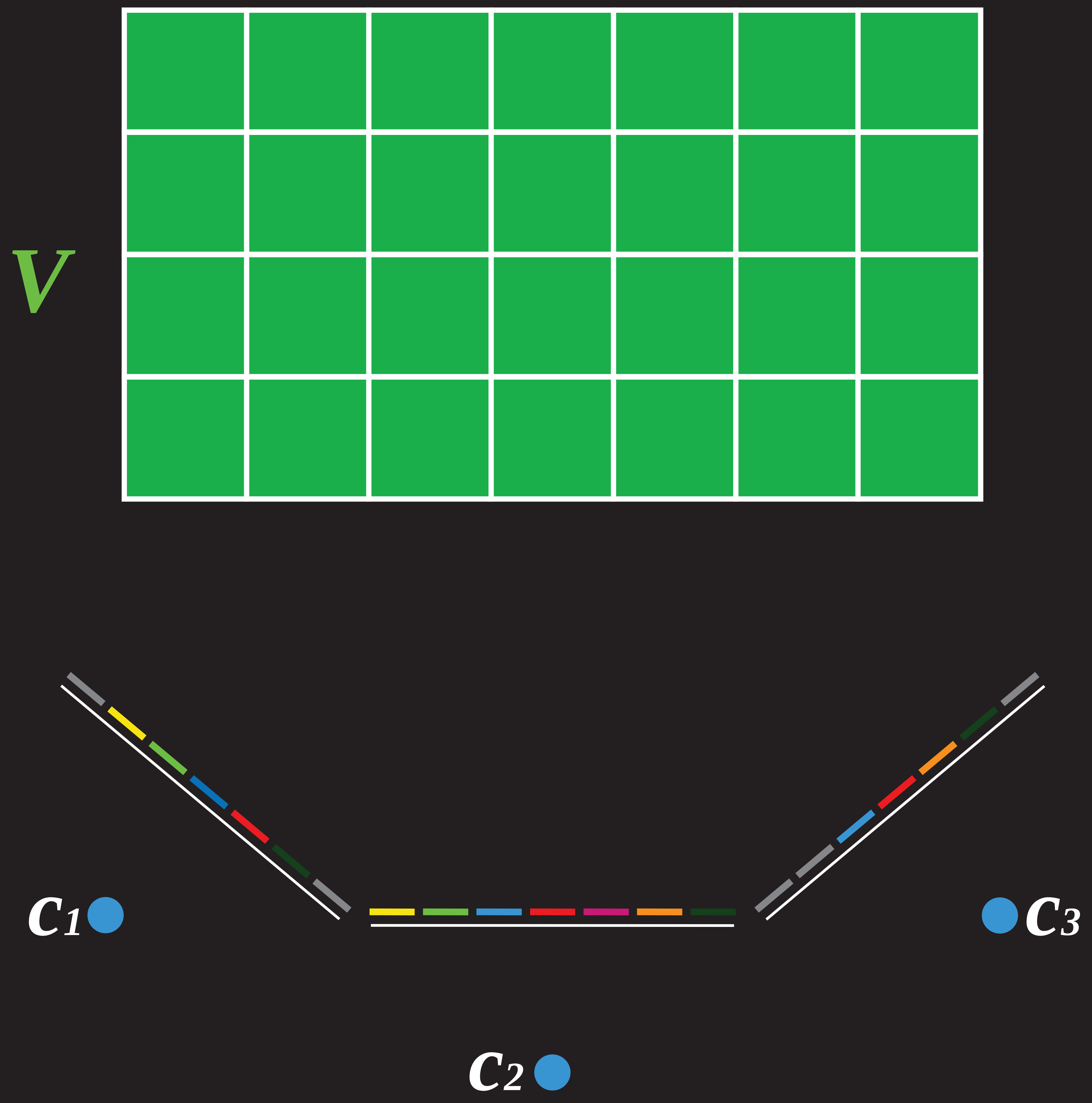
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

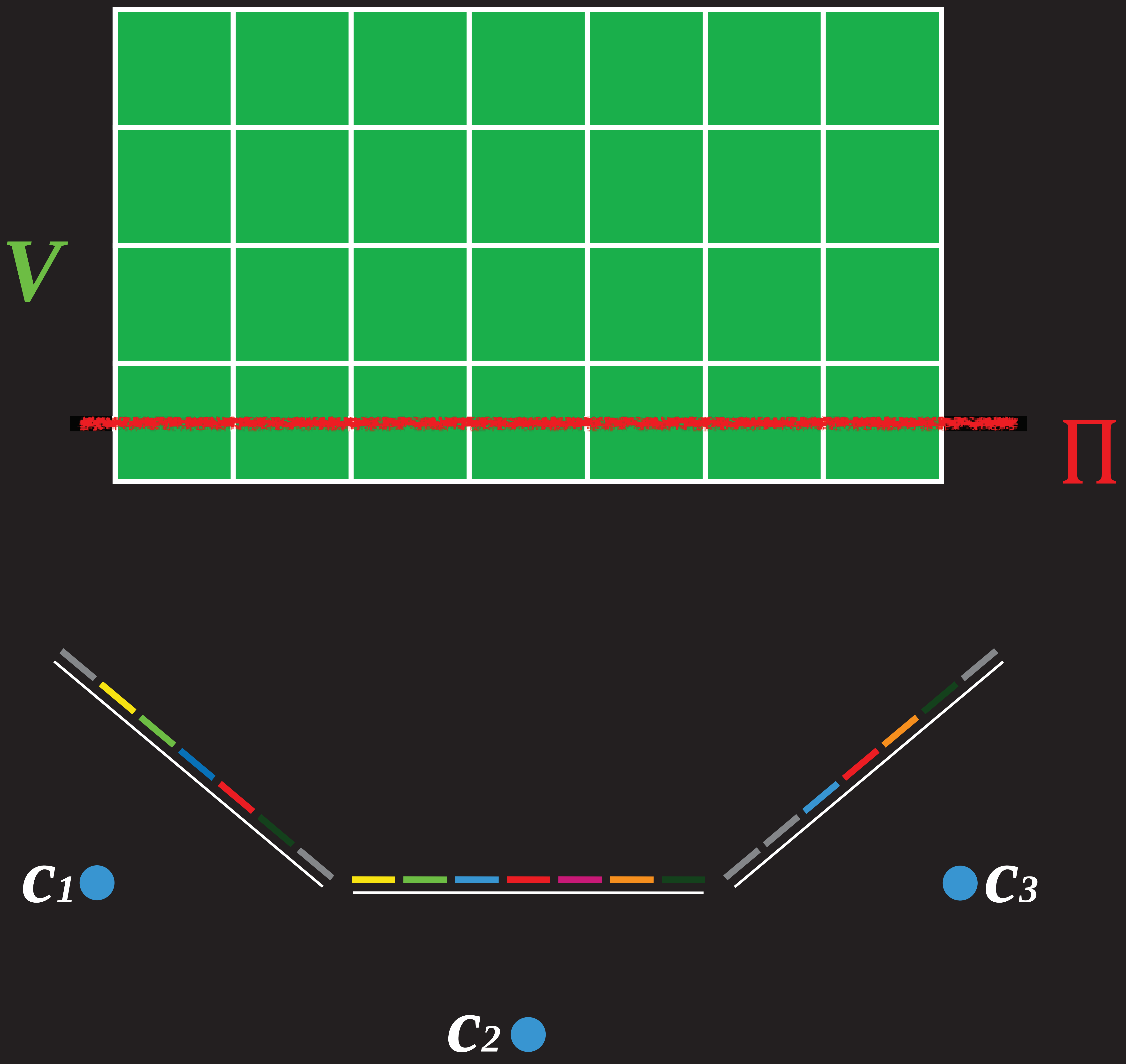
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

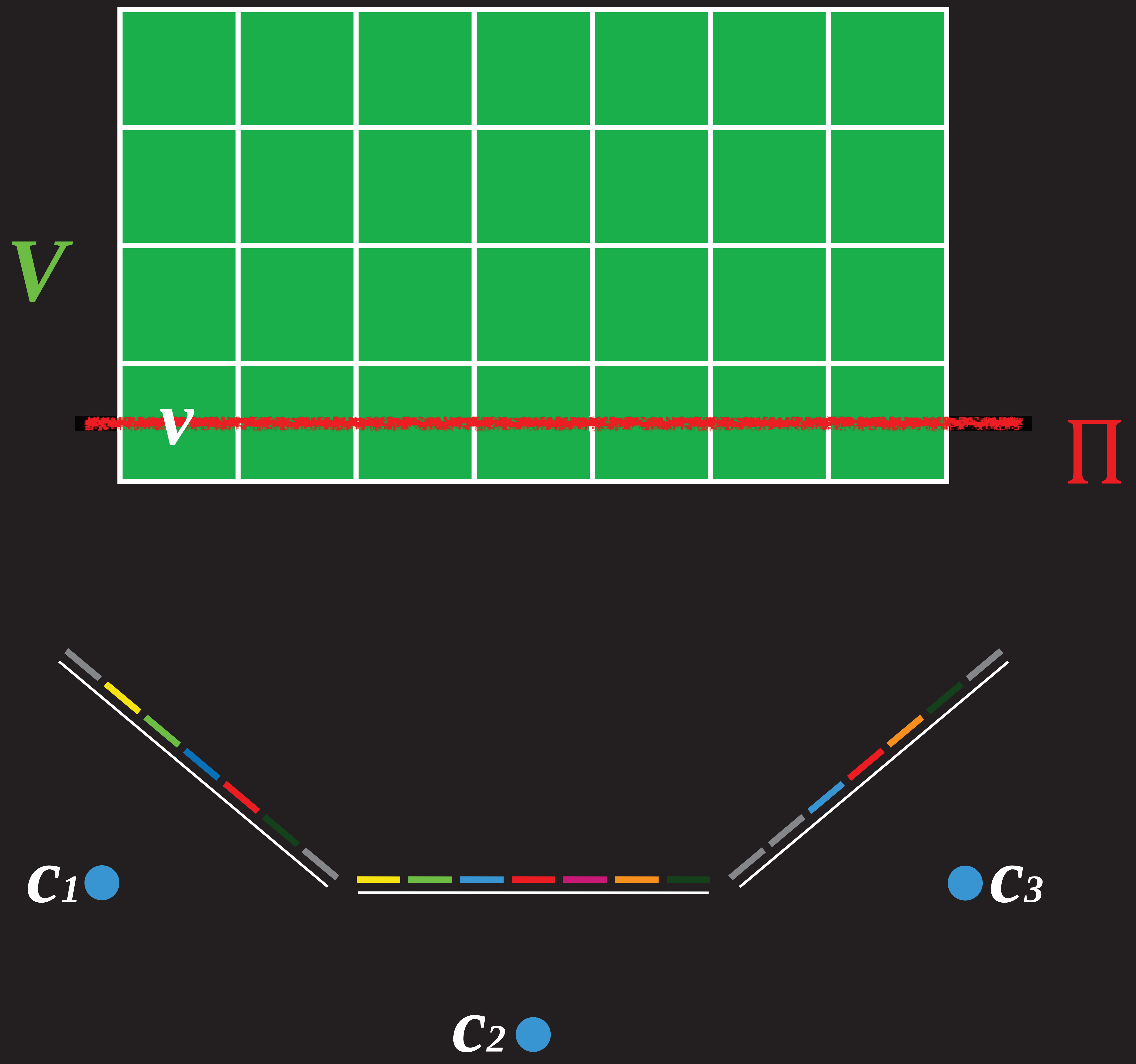
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

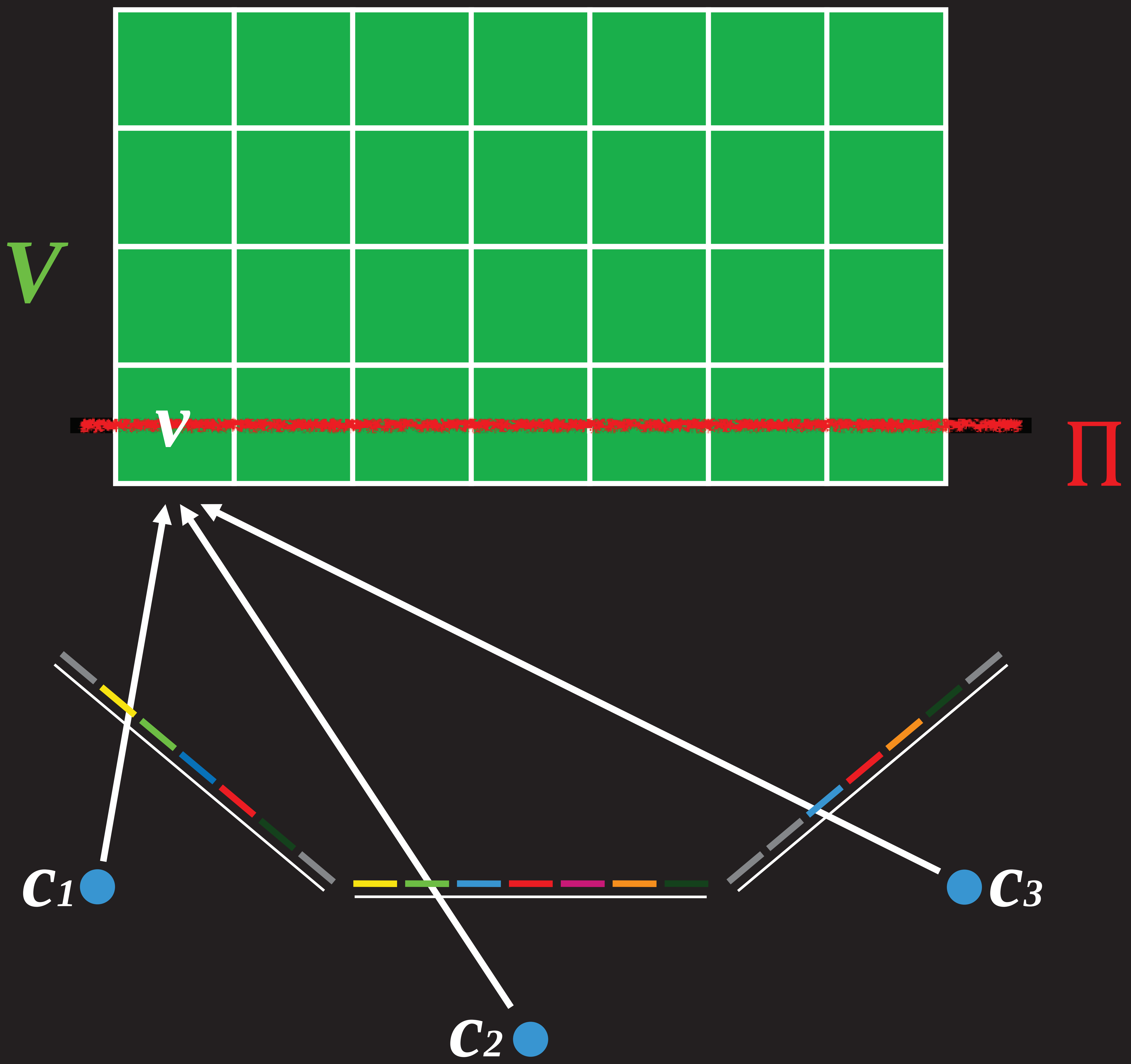
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

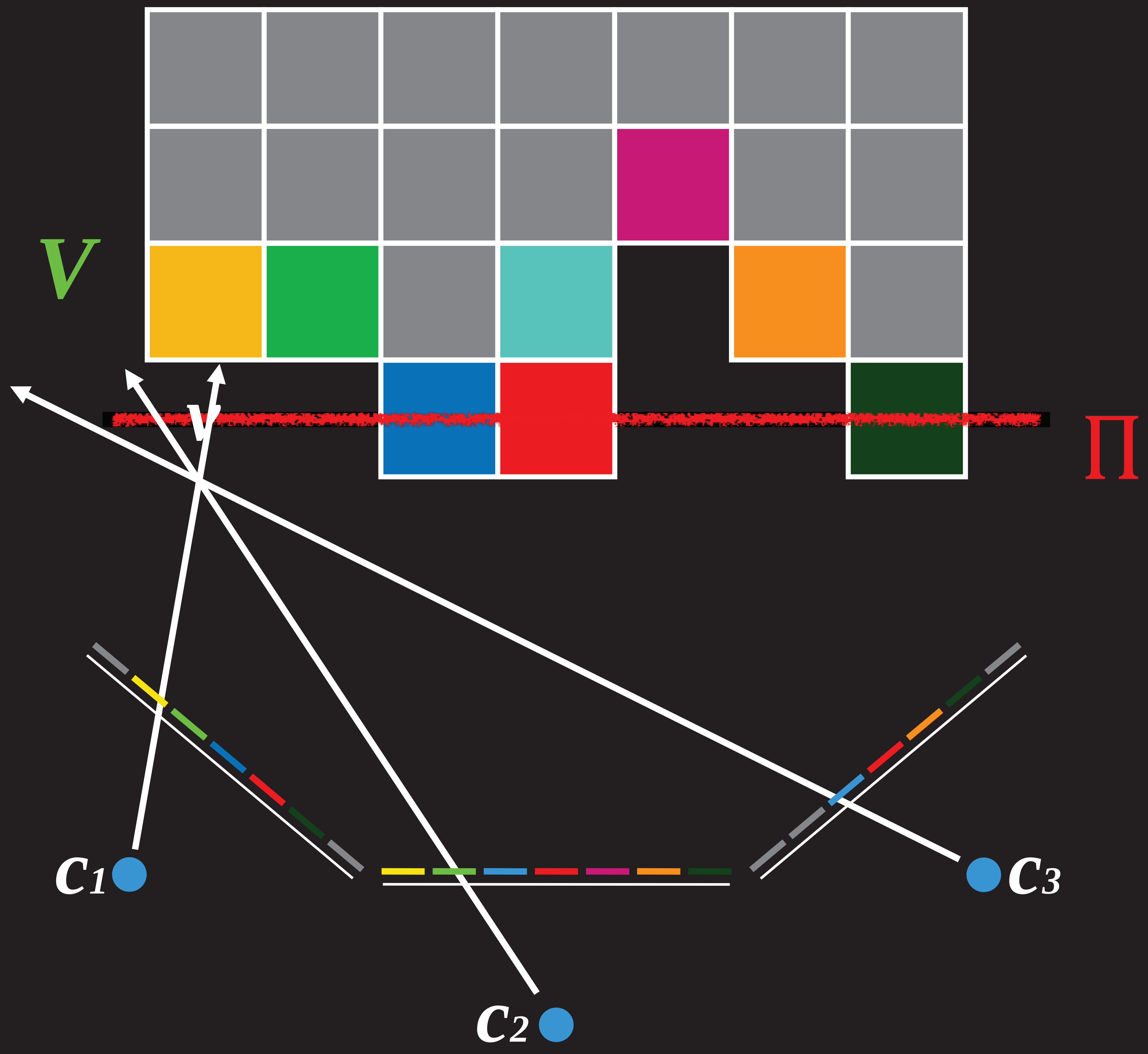
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

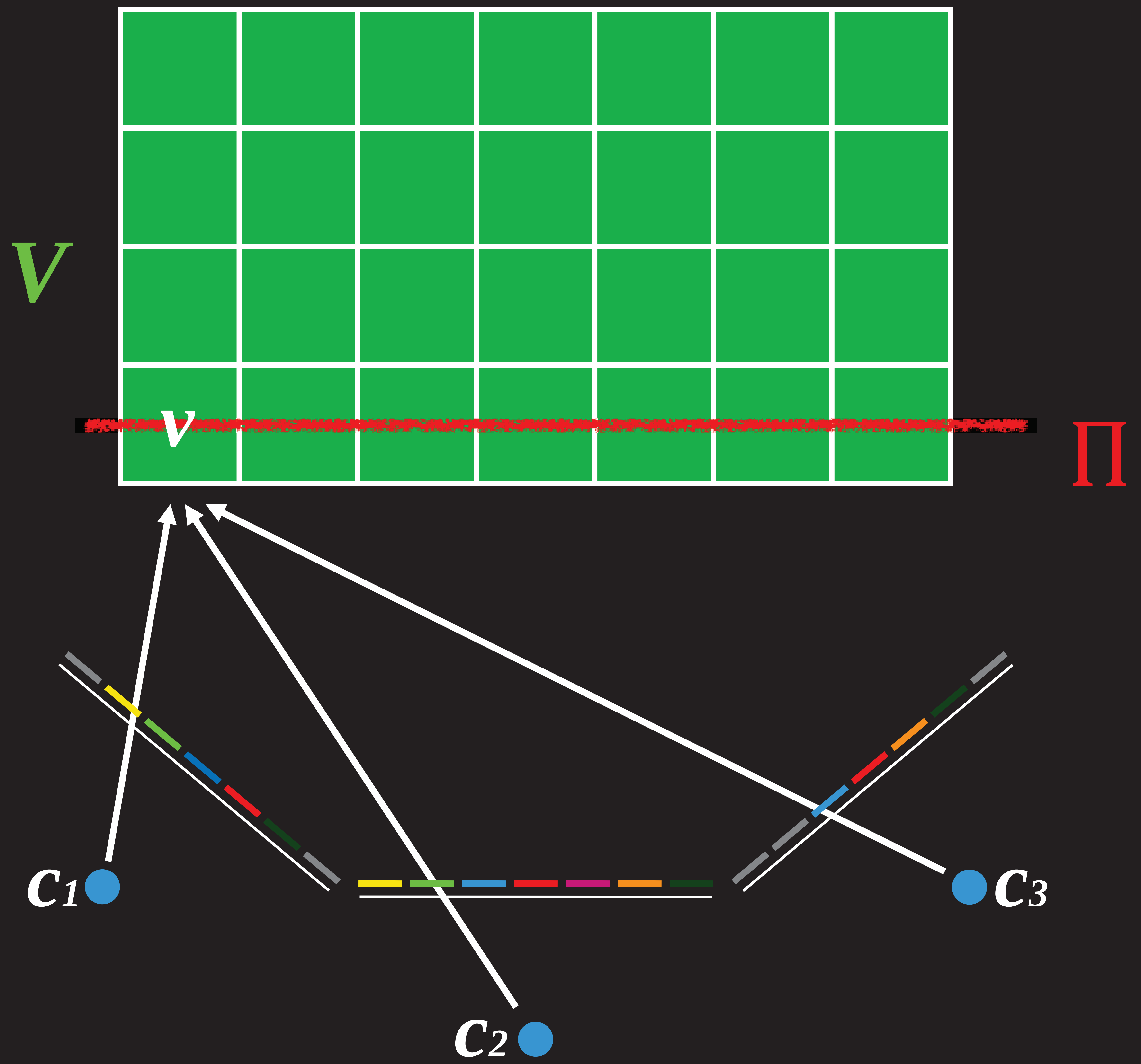
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\mathtt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

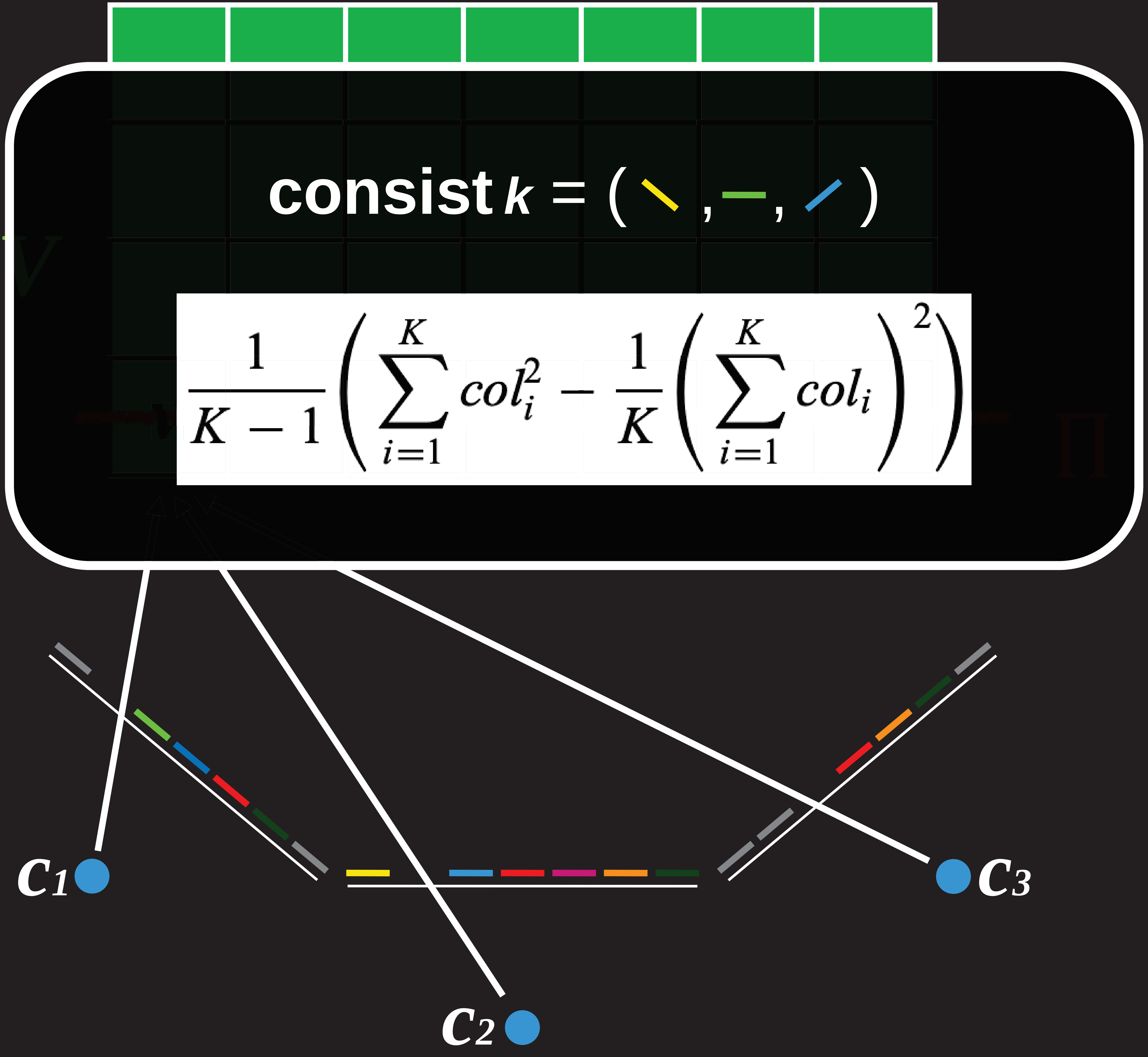
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

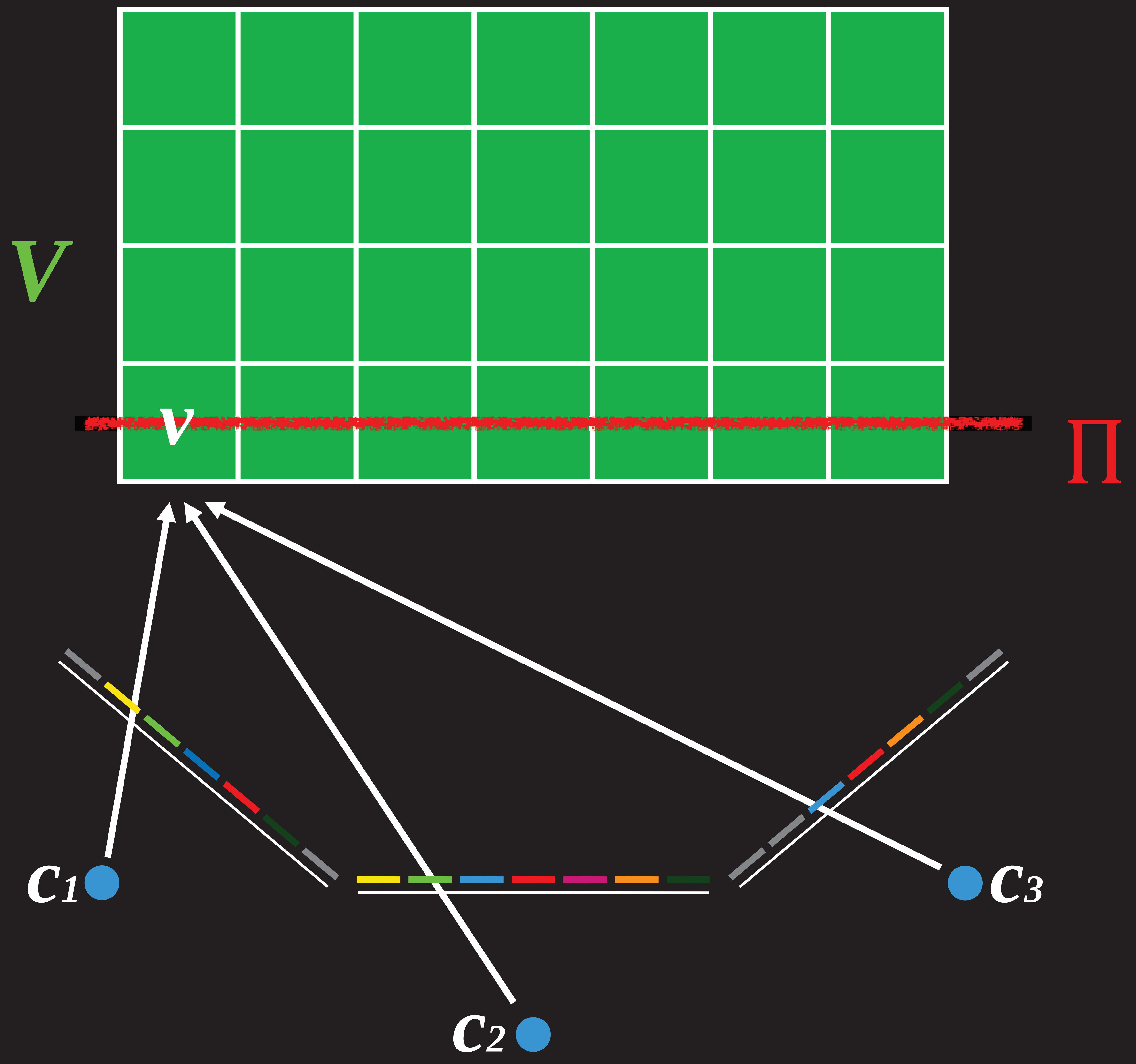
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

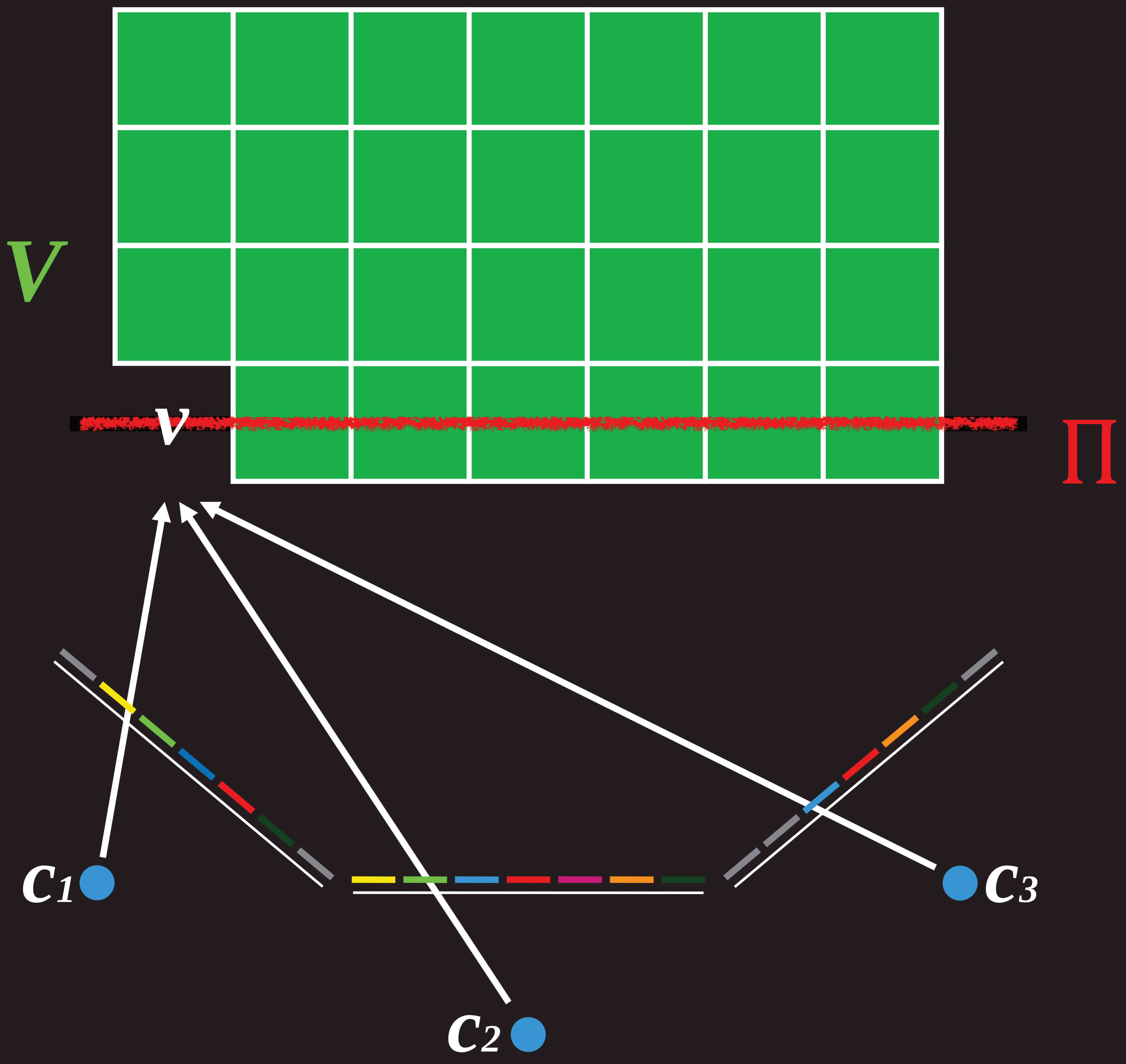
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

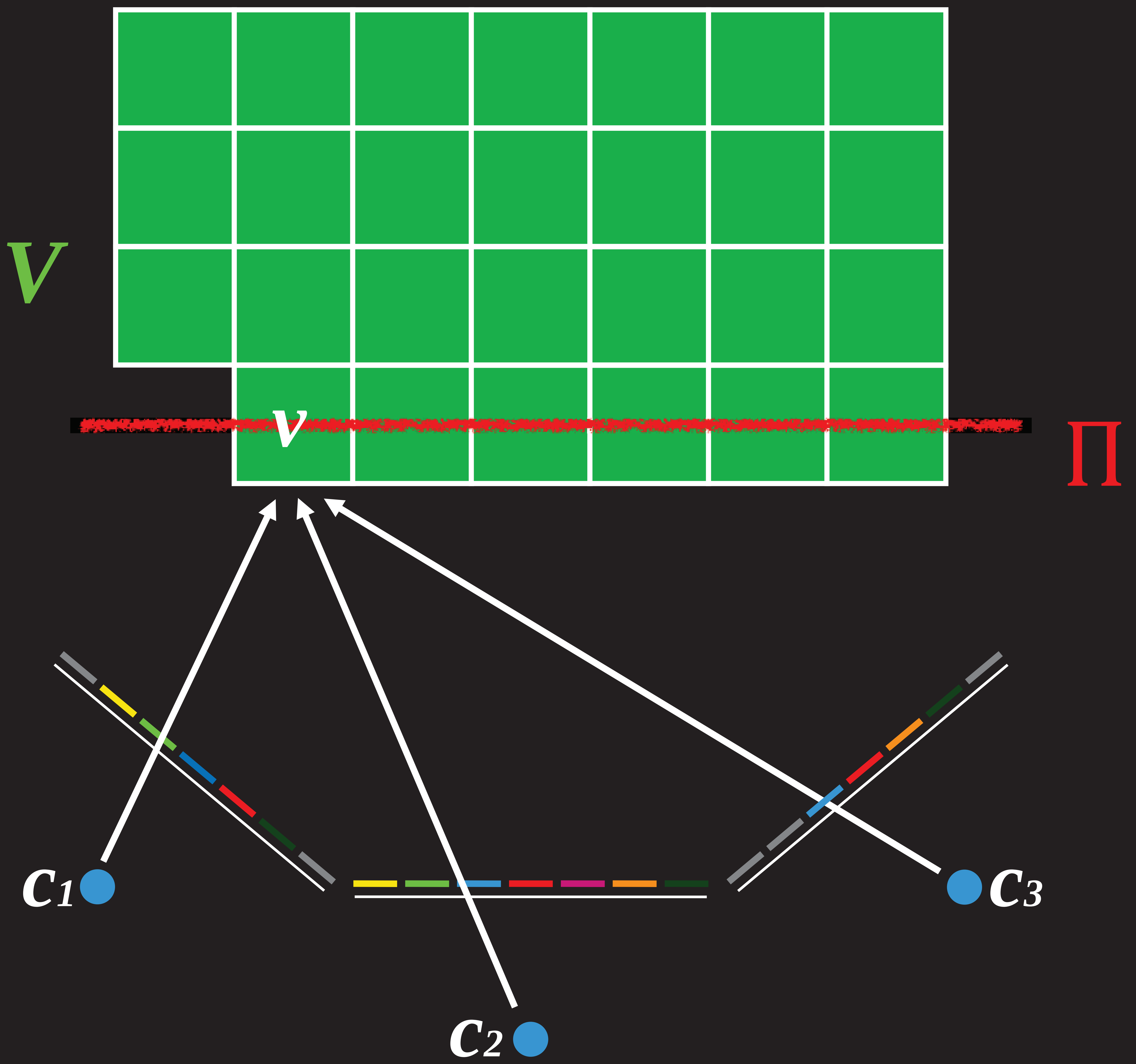
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

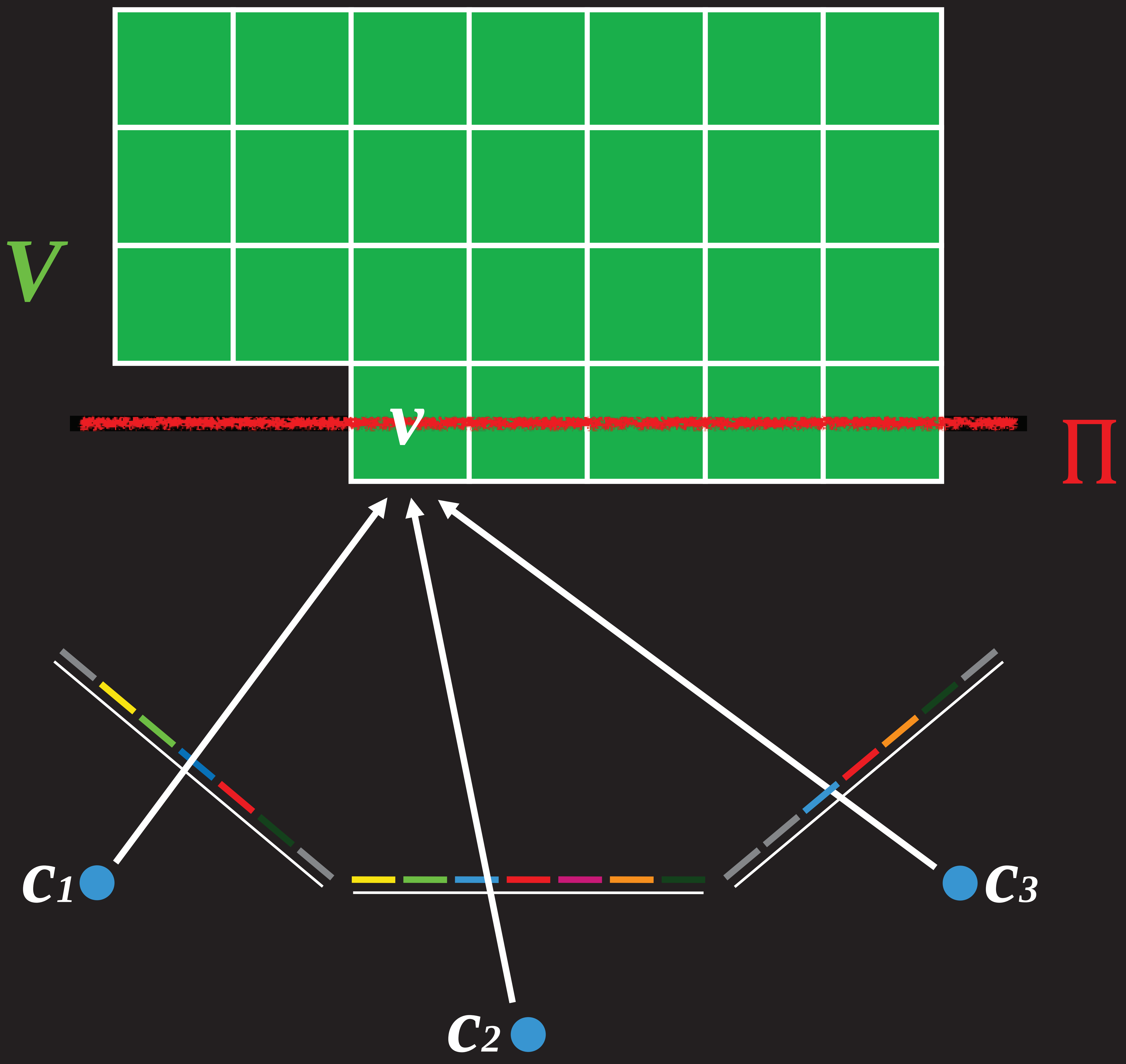
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

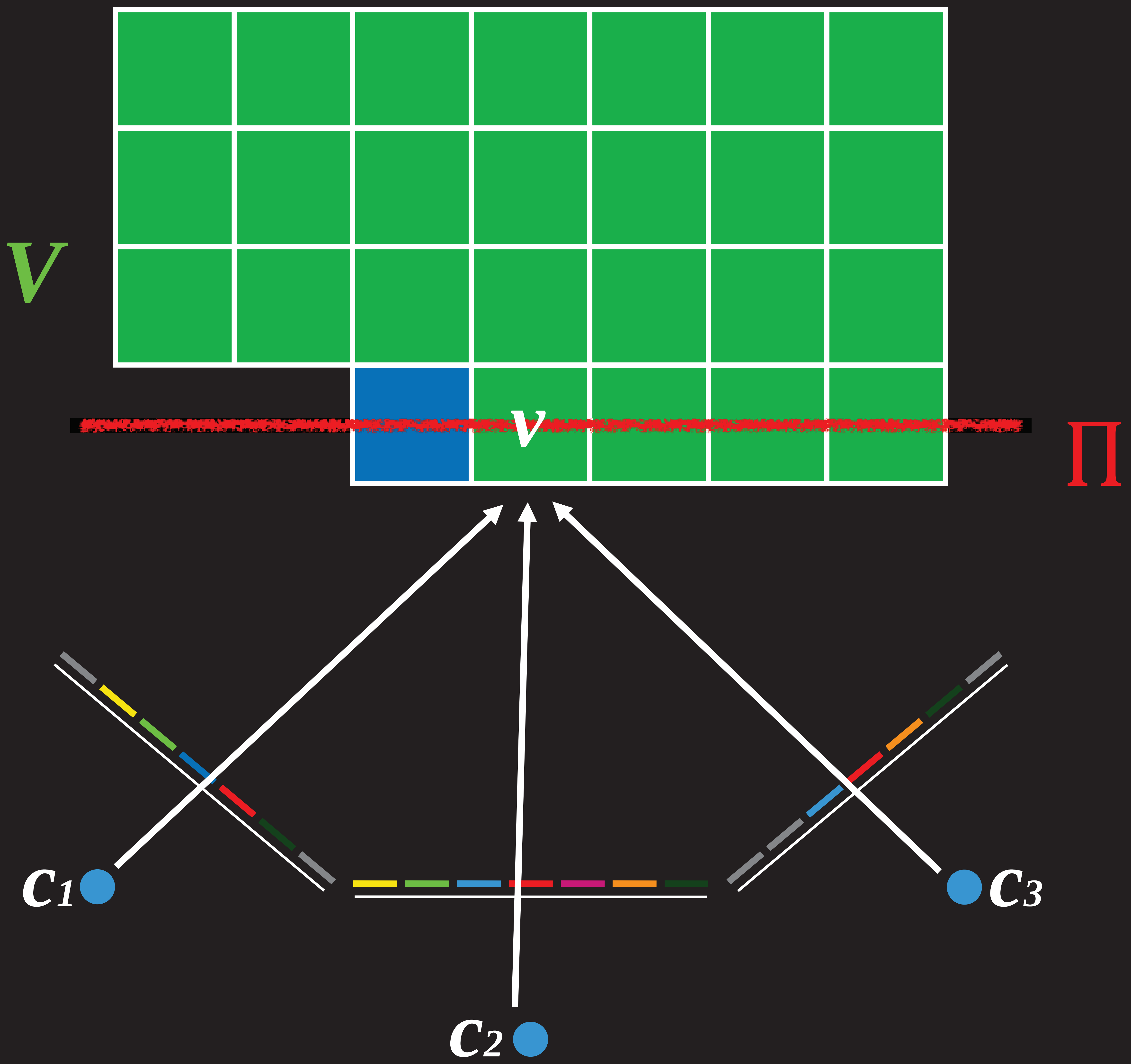
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

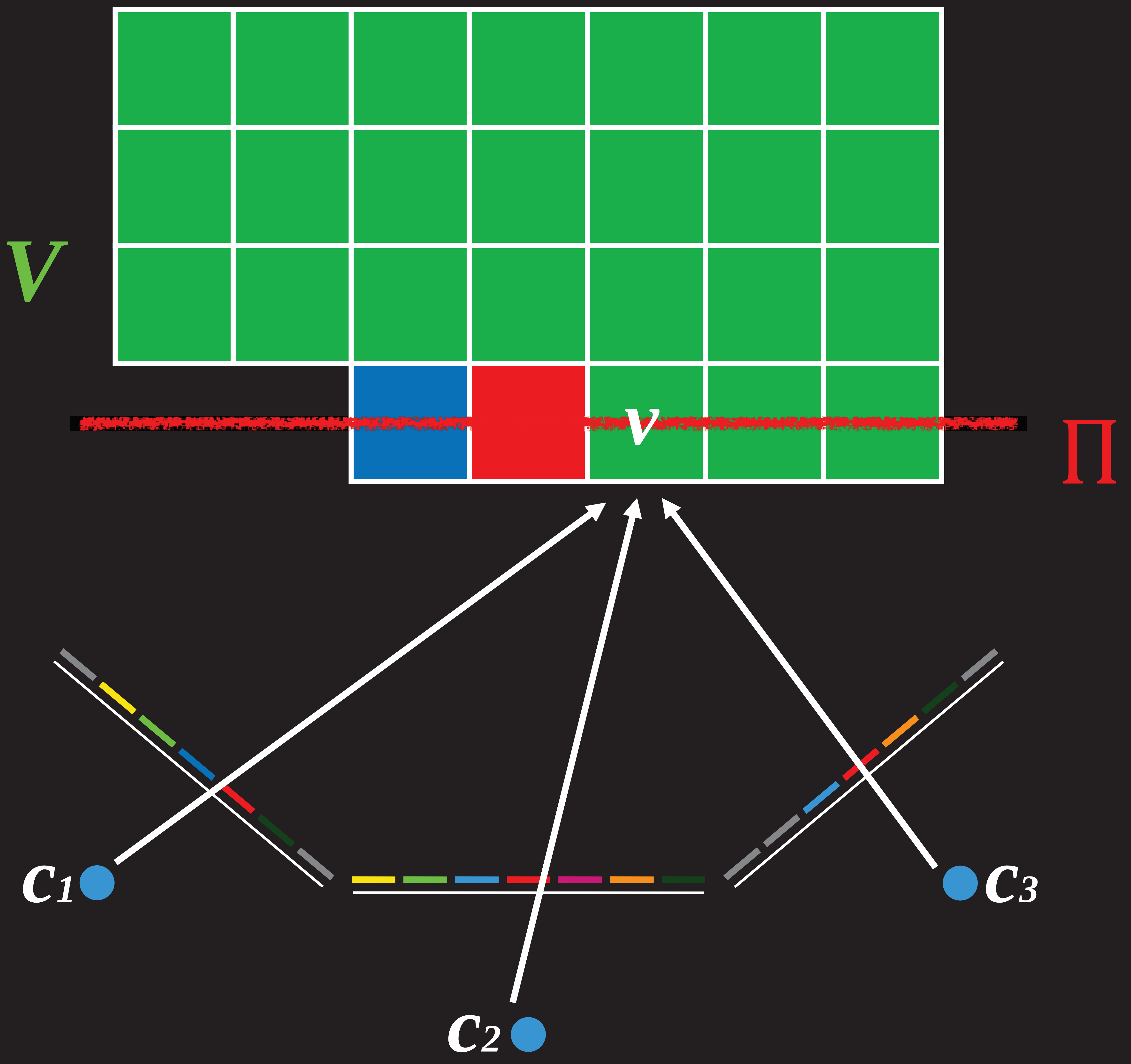
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

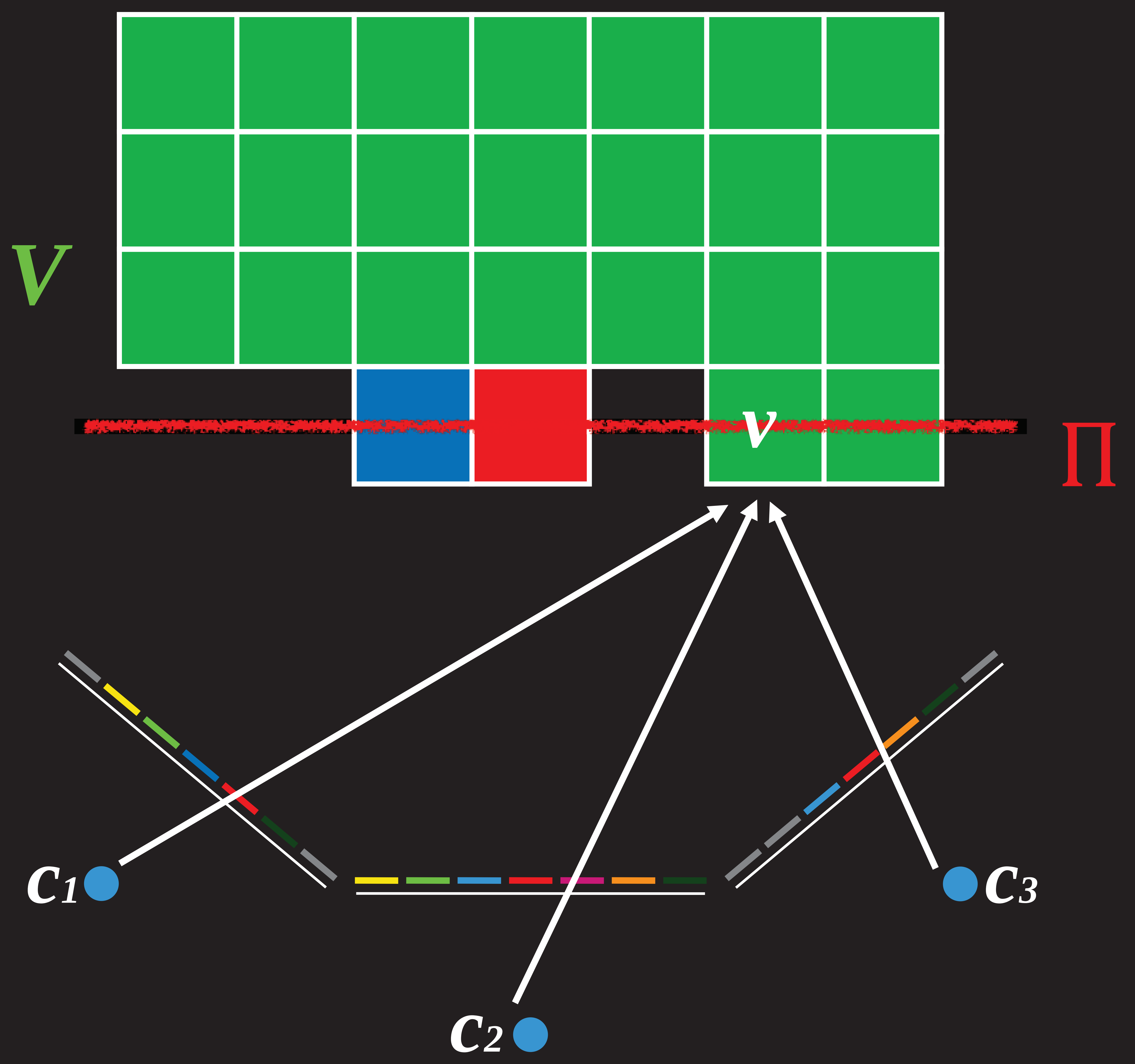
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

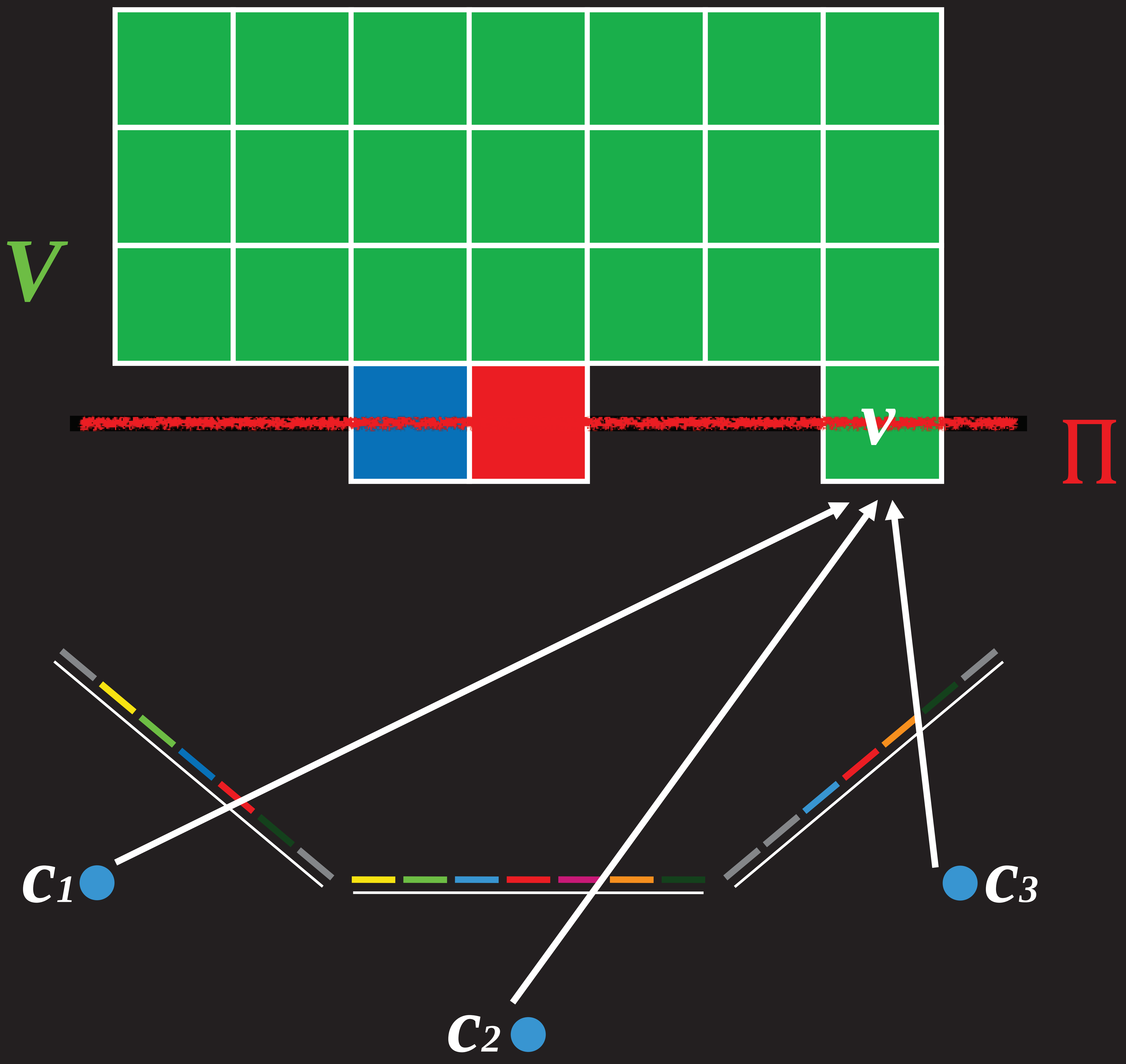
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

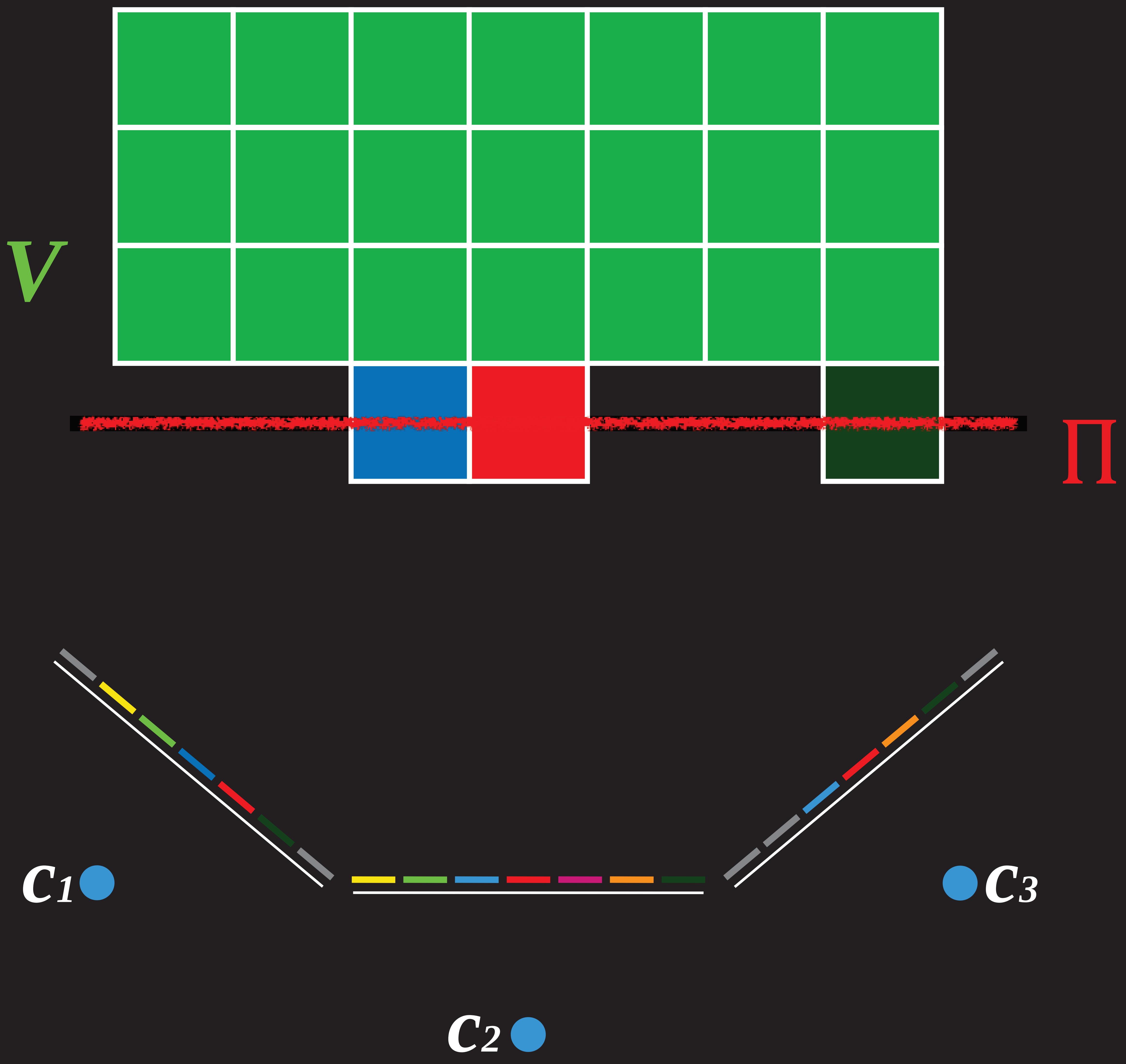
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

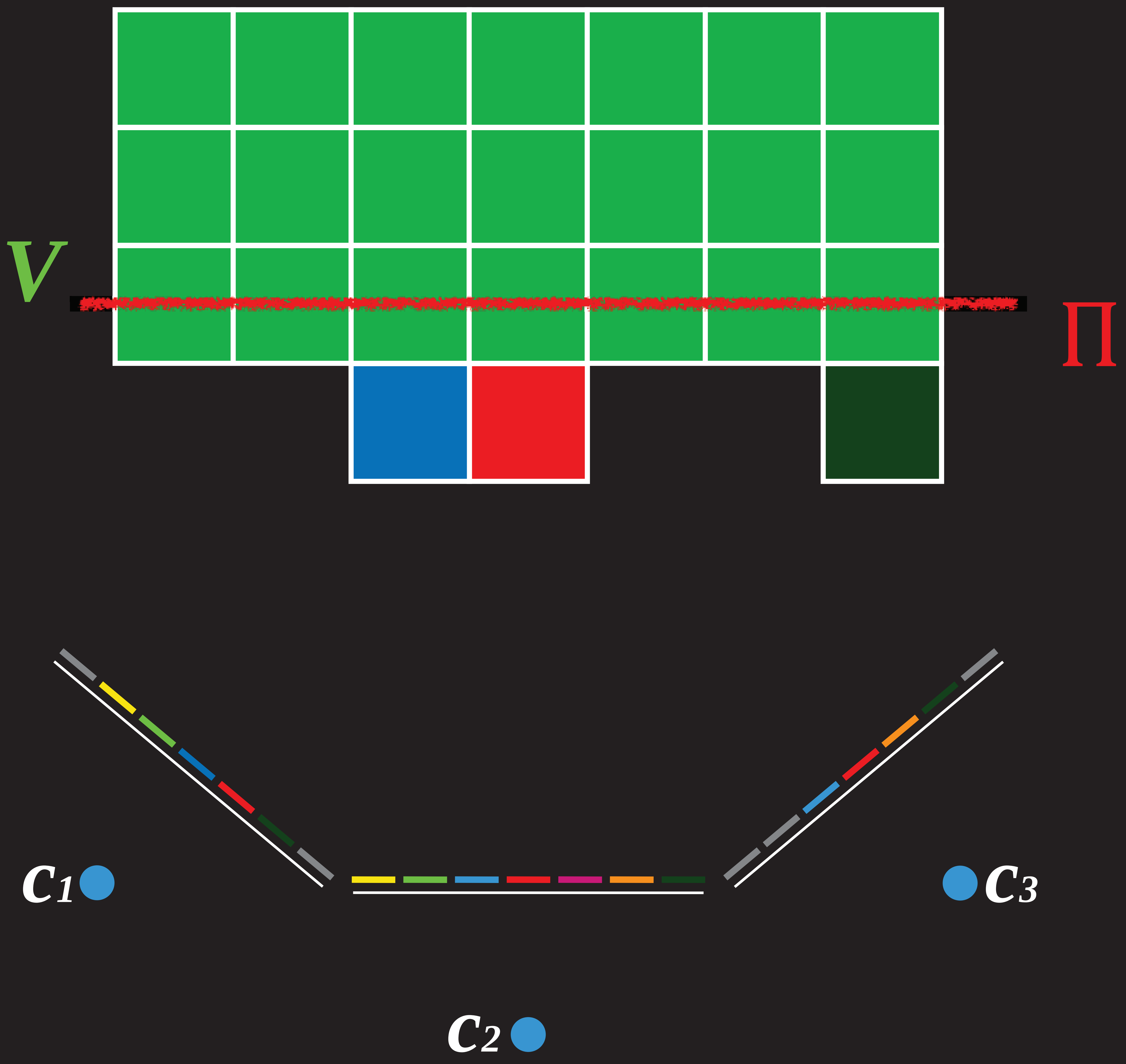
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

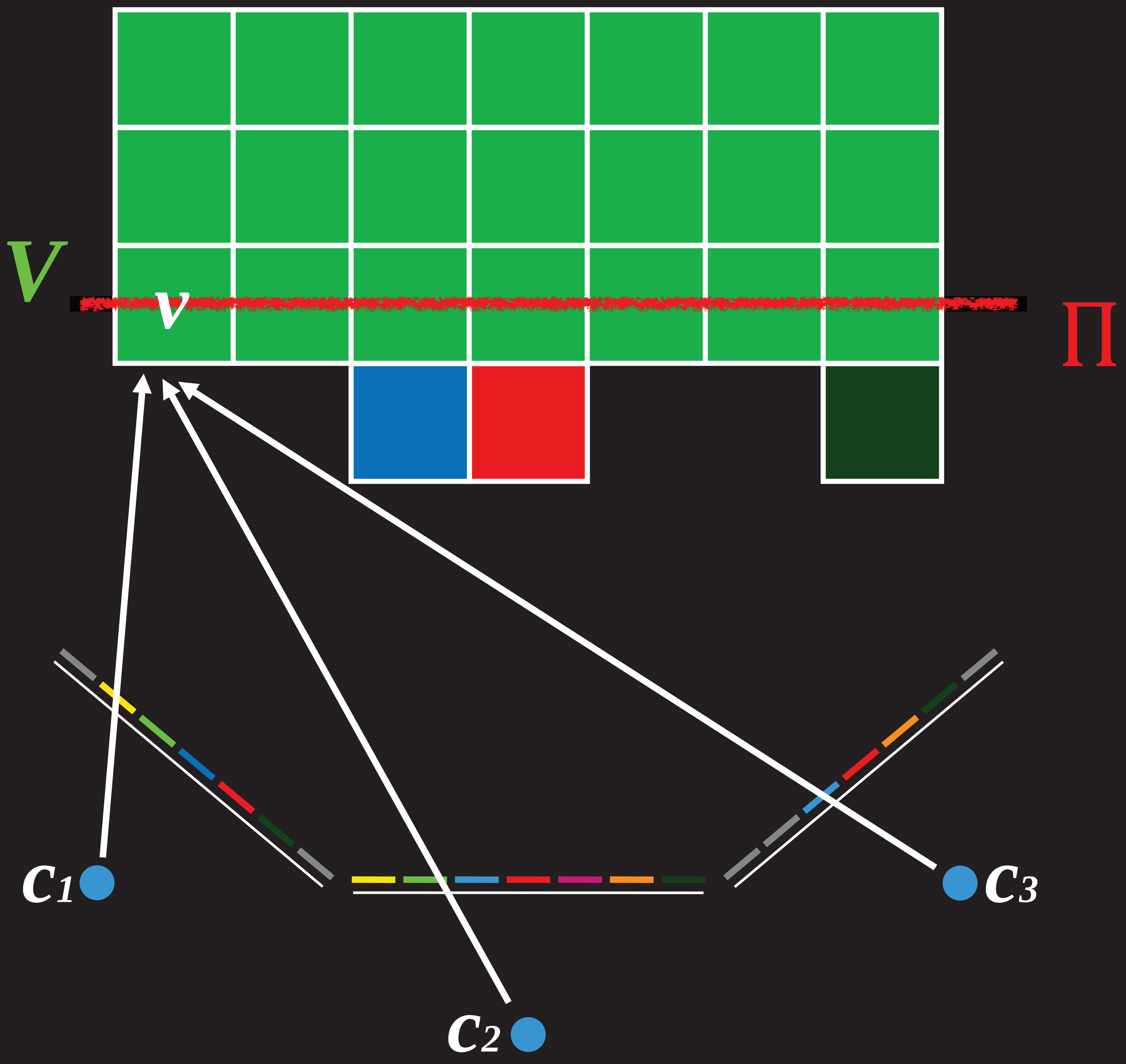
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Plane Sweep Algorithm

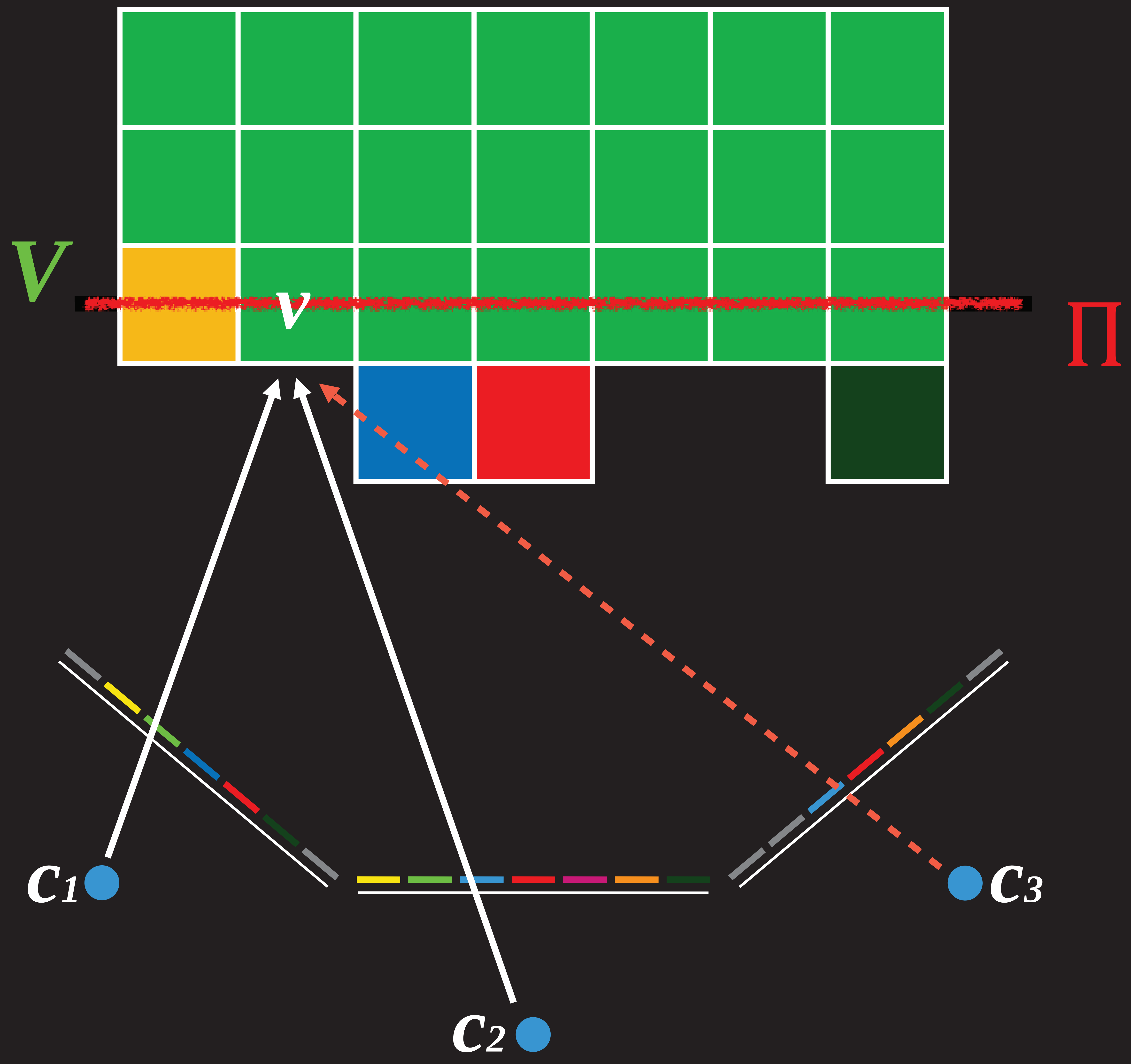
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



Space Carving: An algorithm

Pla

Ste
p
t
Ste
Ste

Ste
S

Lemma 1 (*Visibility Lemma*). *Let p be a point on $\mathcal{V}$'s surface,* $\mathrm{Surf}(\mathcal{V})$, *and let* $\mathrm{Vis}_{\mathcal{V}}(p)$ *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,* $\mathrm{Vis}_{\mathcal{V}}(p) \subseteq \mathrm{Vis}_{\mathcal{V}'}(p)$.

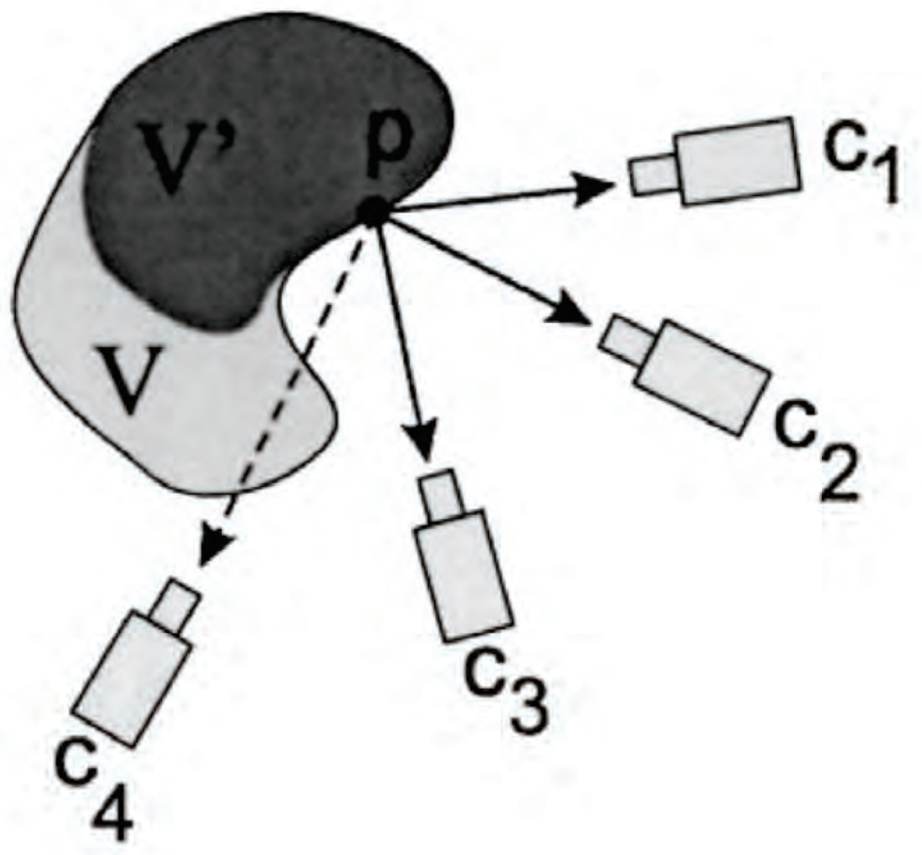
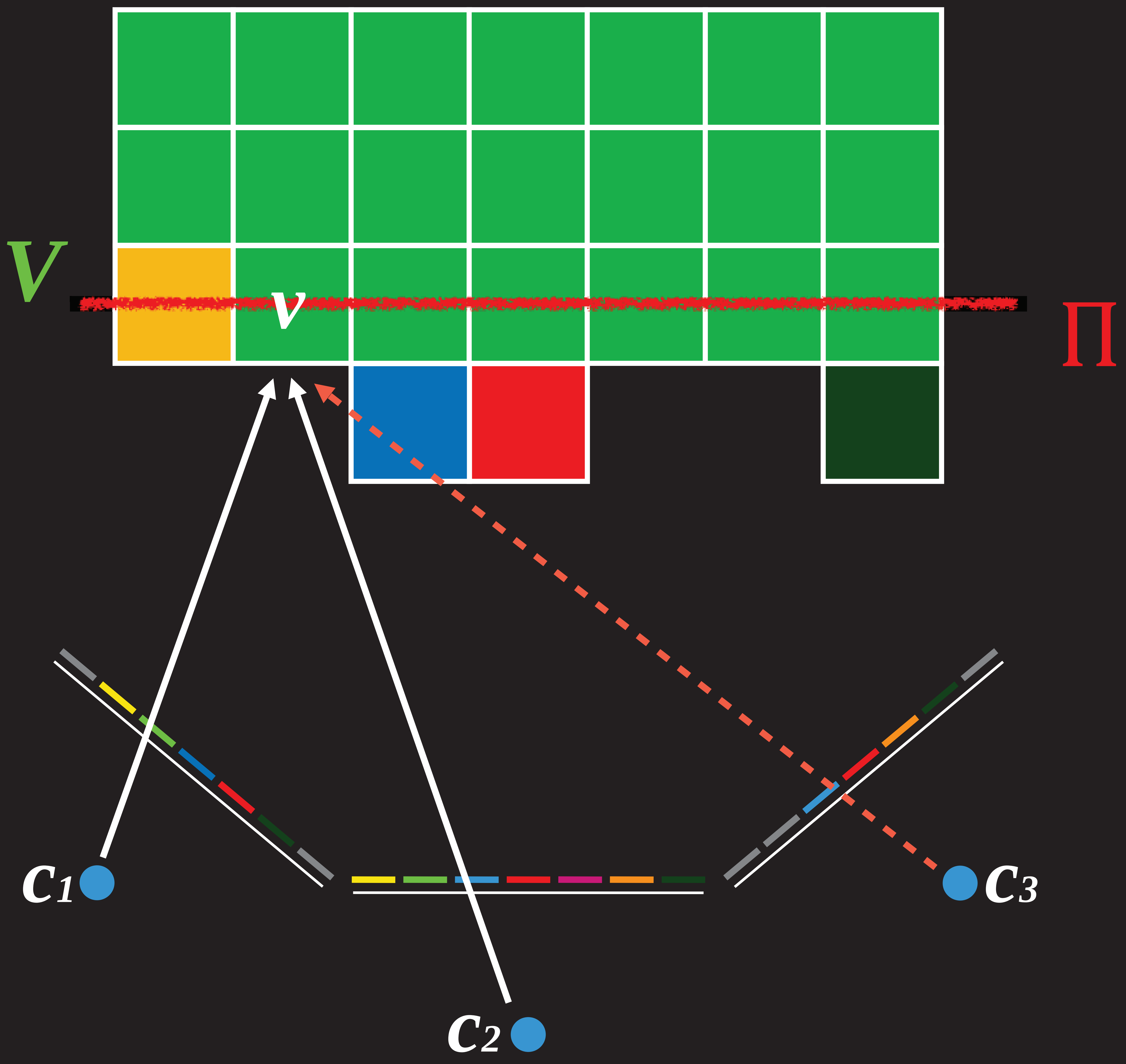


Figure 2. Illustration of the Visibility and Non-Photo-Consistency Lemmas. If p is non-photo-consistent with the photographs at c_1, c_2, c_3, it is non-photo-consistent with the entire set $Vis_{\mathcal{V}'}(p)$, which also includes c_4.



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

Space Carving: An algorithm

Plane Sweep Algorithm

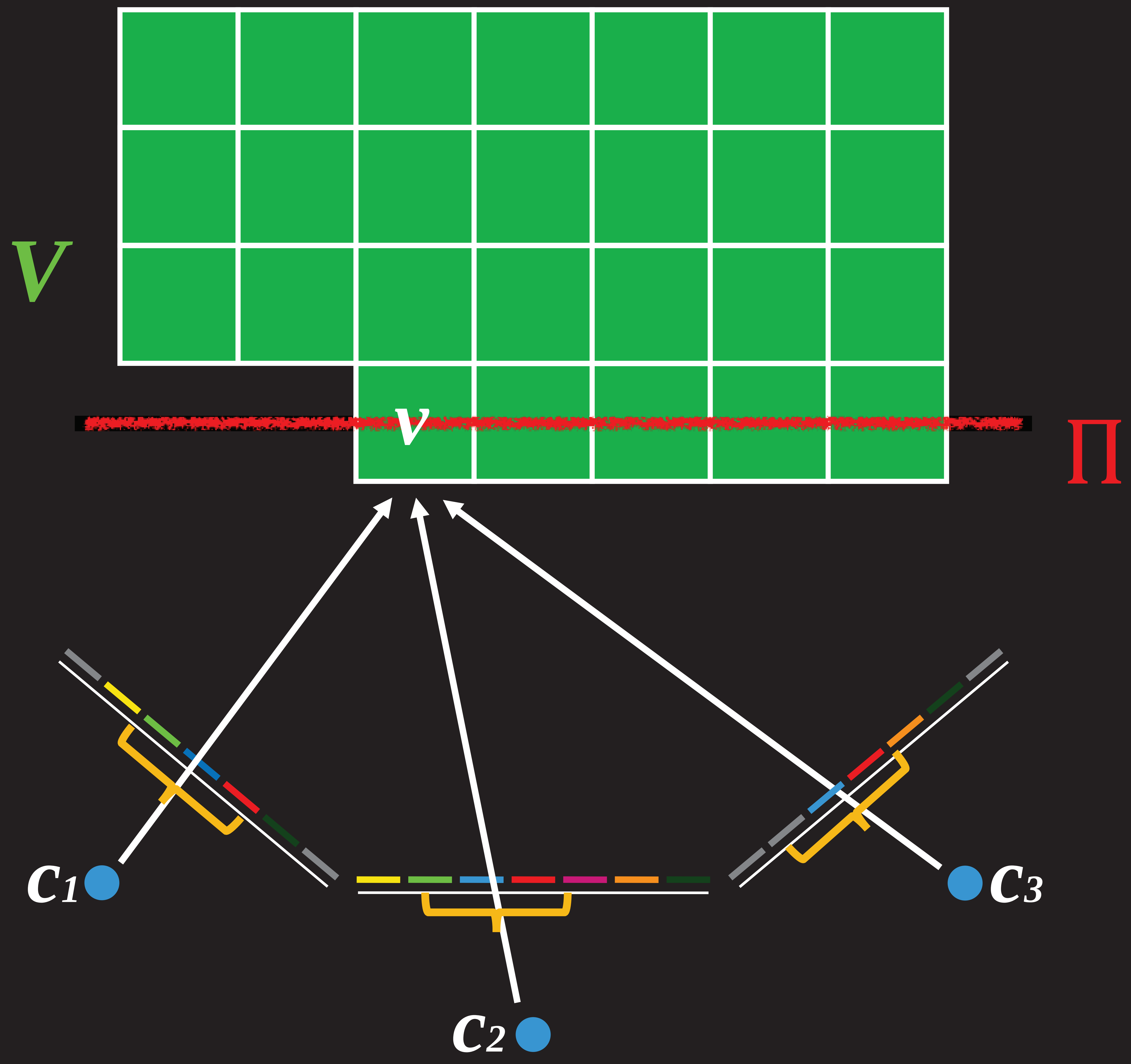
Step 1: Given an initial volume $\mathcal{V}$, initialize the sweep plane Π such that $\mathcal{V}$ lies below Π (i.e., Π is swept towards $\mathcal{V}$).

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

Step 3: For each surface voxel v on Π:

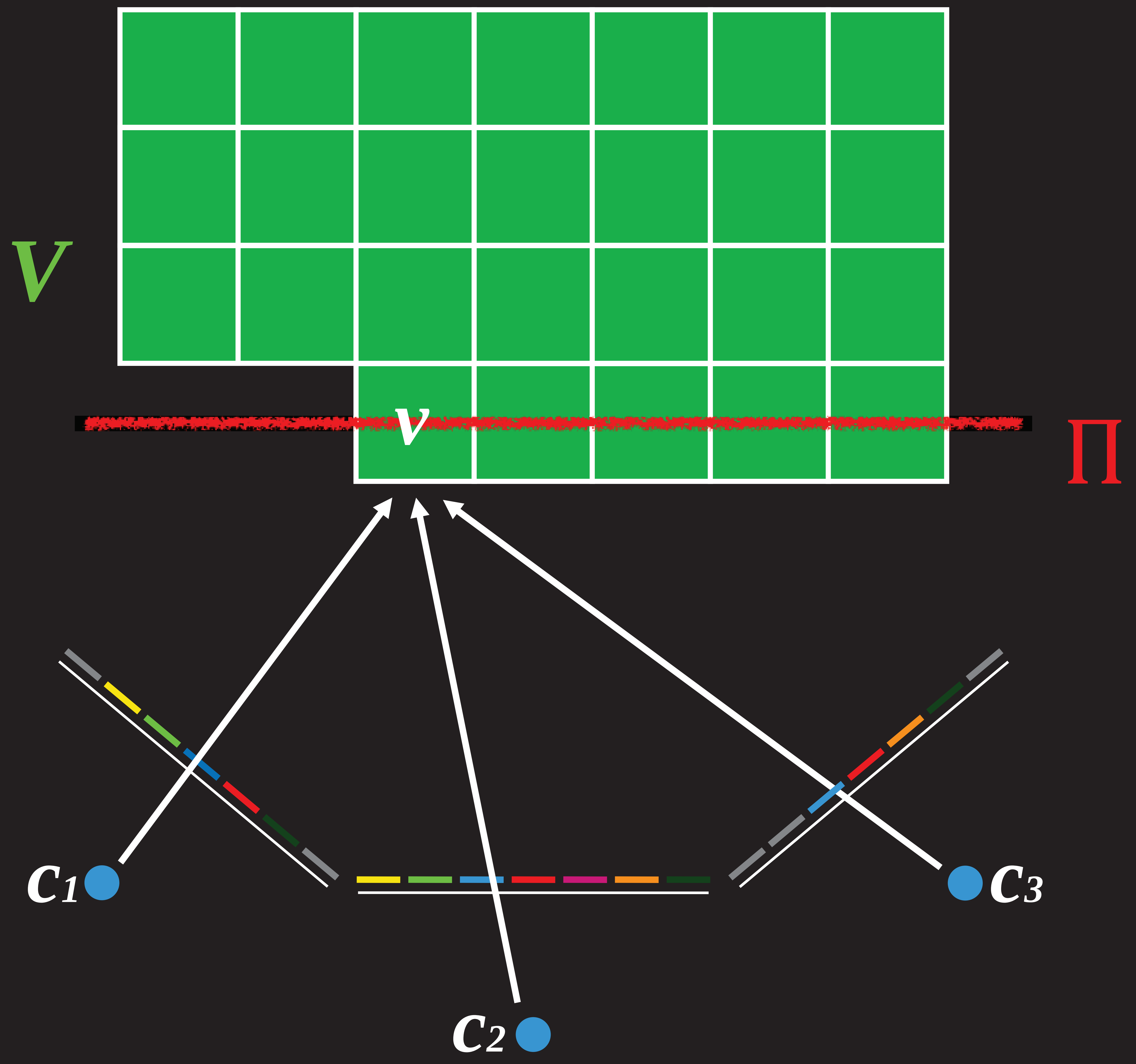
- **a.** let $c_1, \ldots, c_j$ be the cameras above Π for which v projects to an *unmarked* pixel;
- **b.** determine the photo-consistency of v using $\texttt{consist}_K(col_1, \ldots, col_j, \xi_1, \ldots, \xi_j)$;
- **c.** if v is inconsistent then set $\mathcal{V} = \mathcal{V} - \{v\}$, otherwise mark the pixels to which v projects.

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



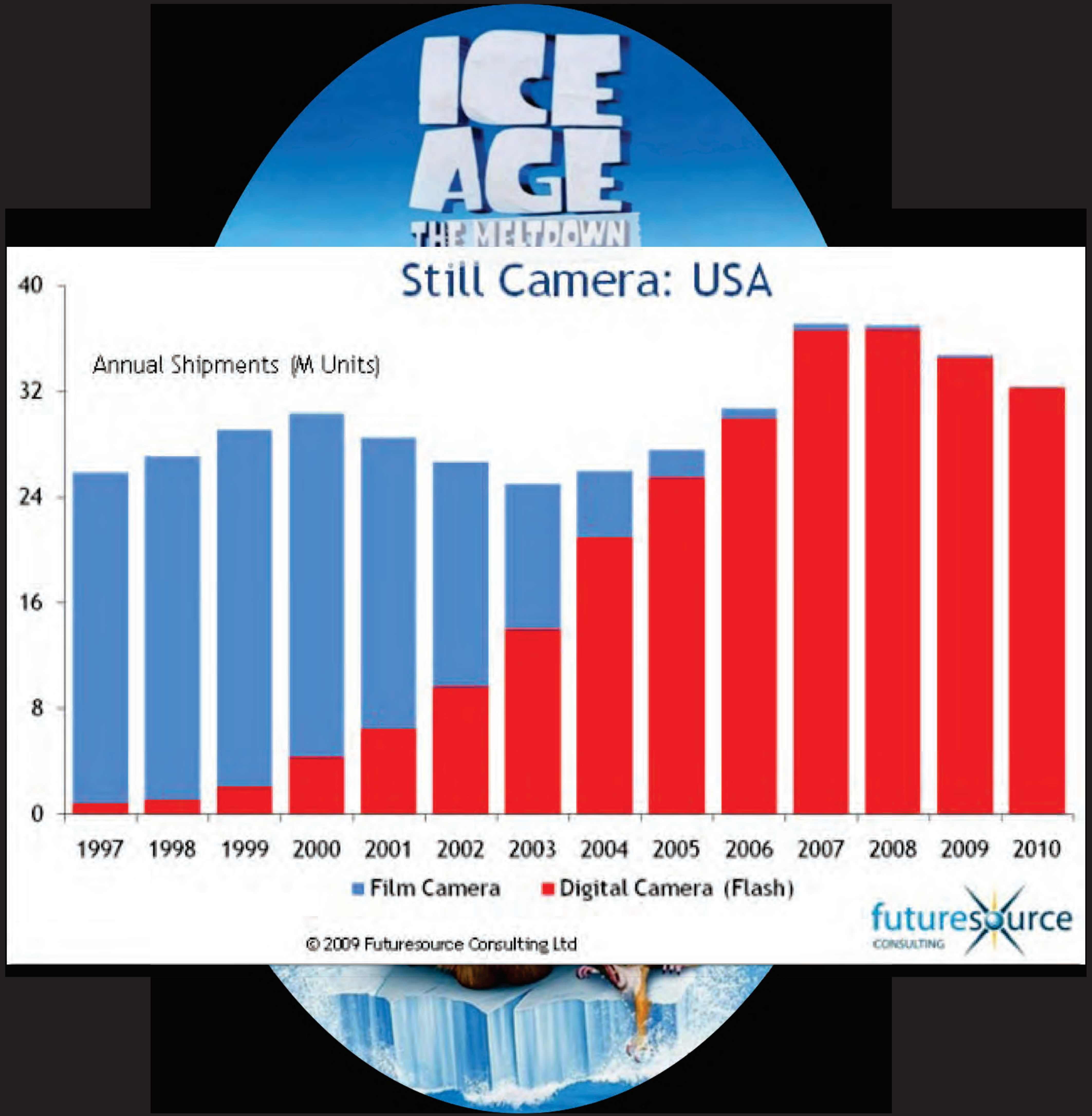
Space Carving: An algorithm

- 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
The Metropolitan Museum of Art
2012
Perception - ice-age/revolution?
2014
2015
bedroom
bedroom
bathroom
restroom
balcony
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.

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).

[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 *3DPVT'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. Ginzton, 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 MMSP*, 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. Wingbermuhle. 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 adaptative 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 modelling. *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.

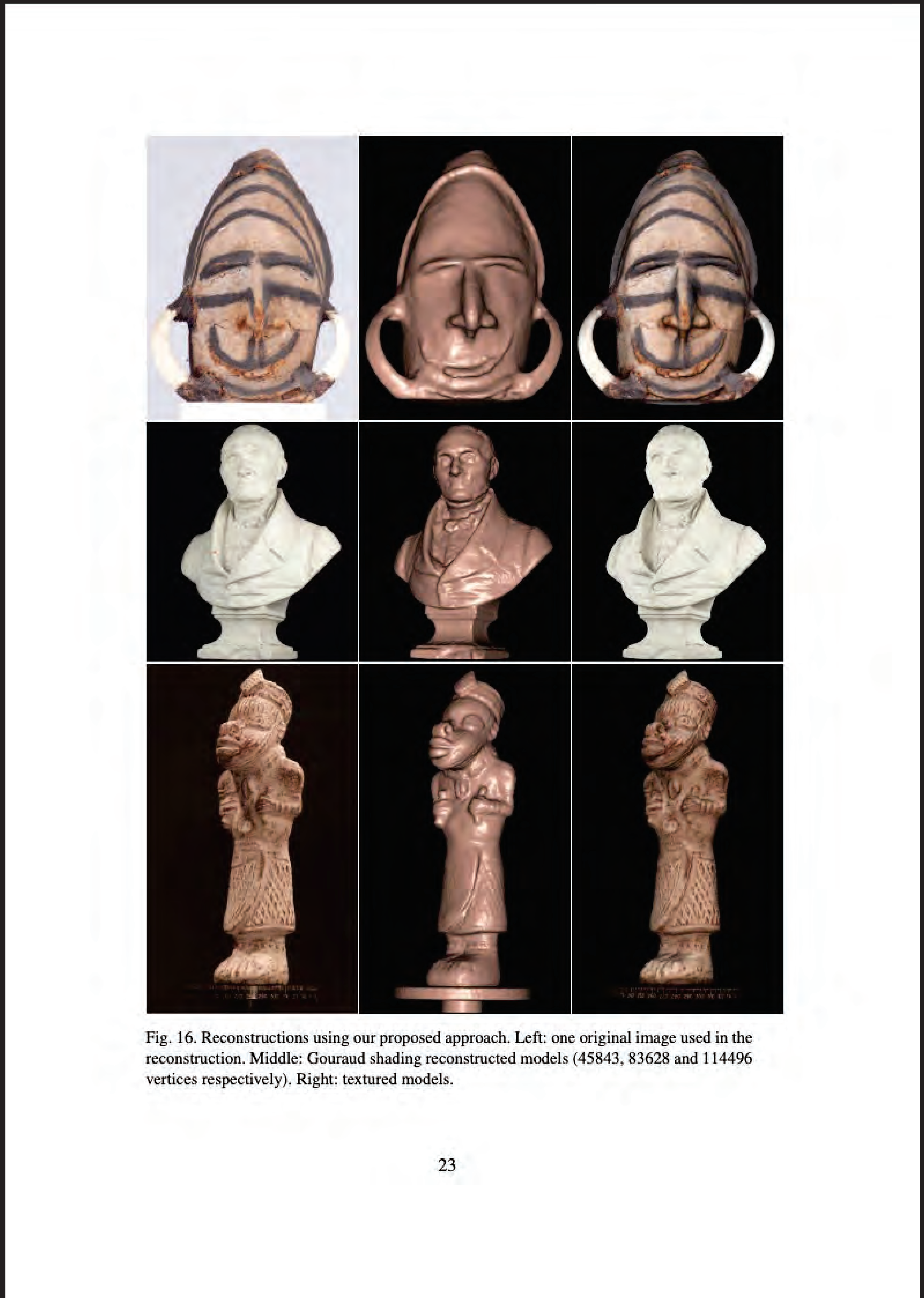
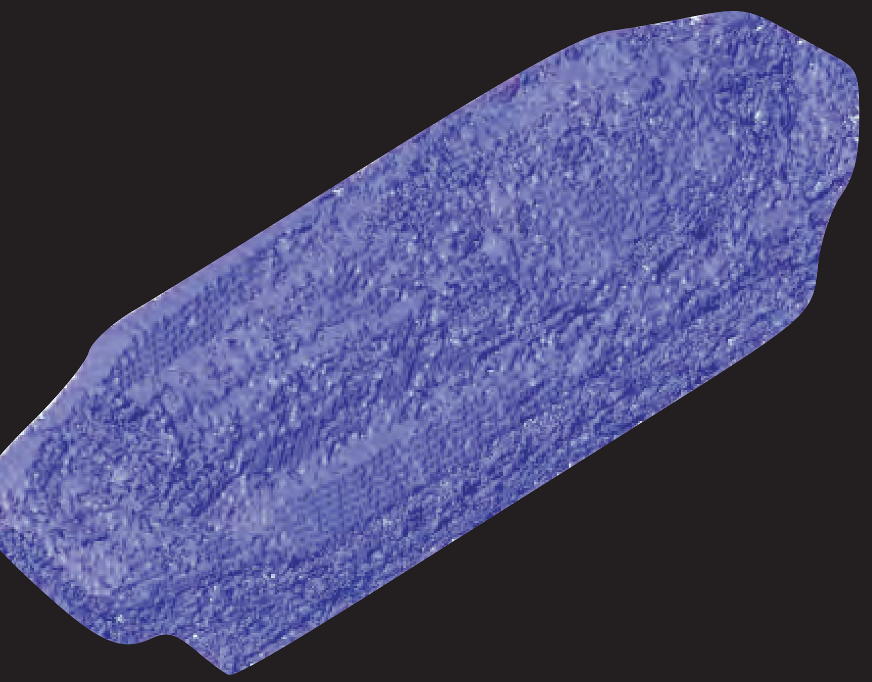


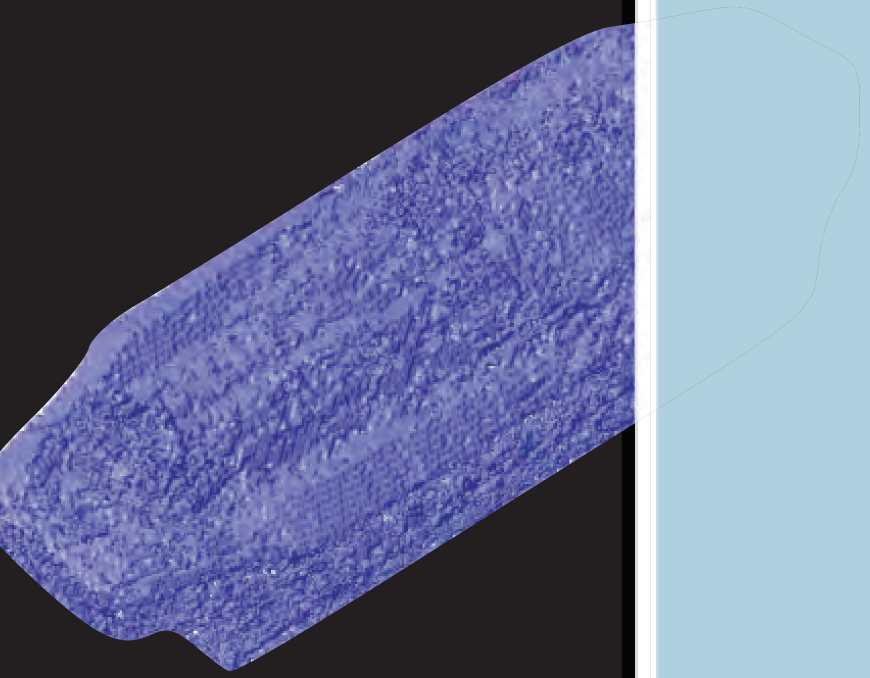
Fig. 16. Reconstructions using our proposed approach. Left: one original image used in the reconstruction. Middle: Gouraud shading reconstructed models (45843, 83628 and 114496 vertices respectively). Right: textured models.

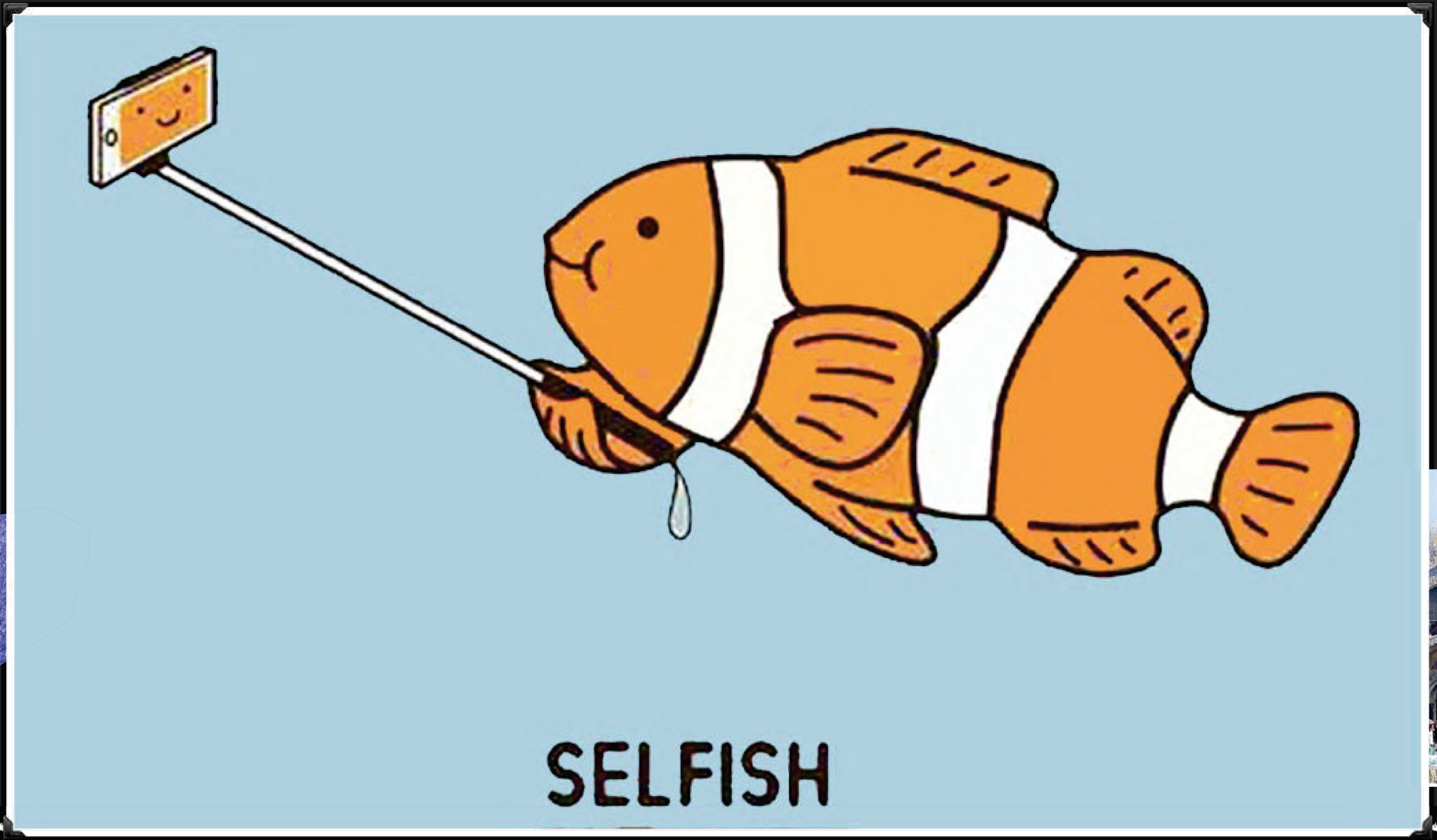
Sensing revolution





2000 2005 2010 2011 2012

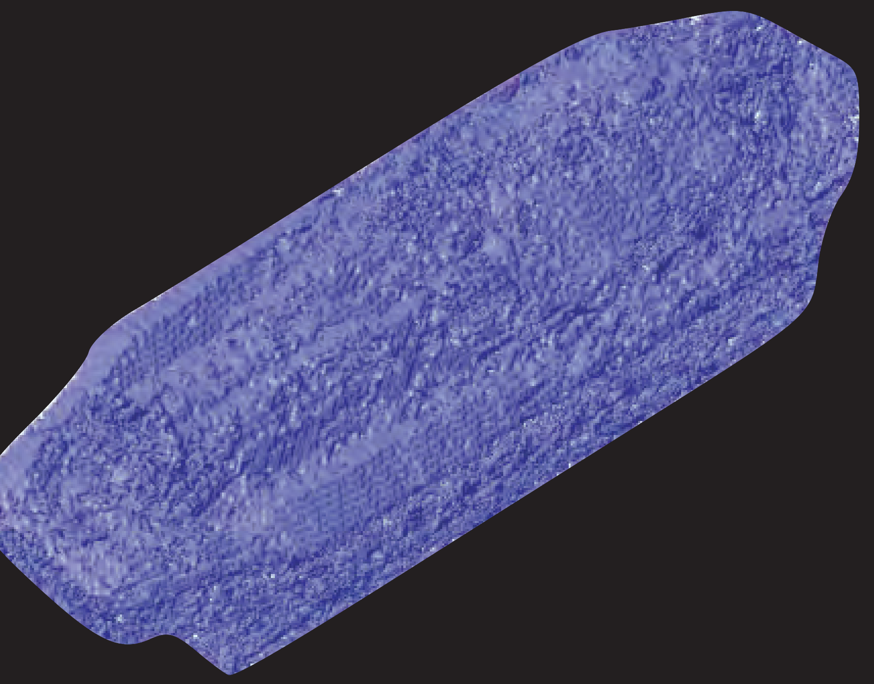




2000 2005 2010 2011 2012

Sensing revolution

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

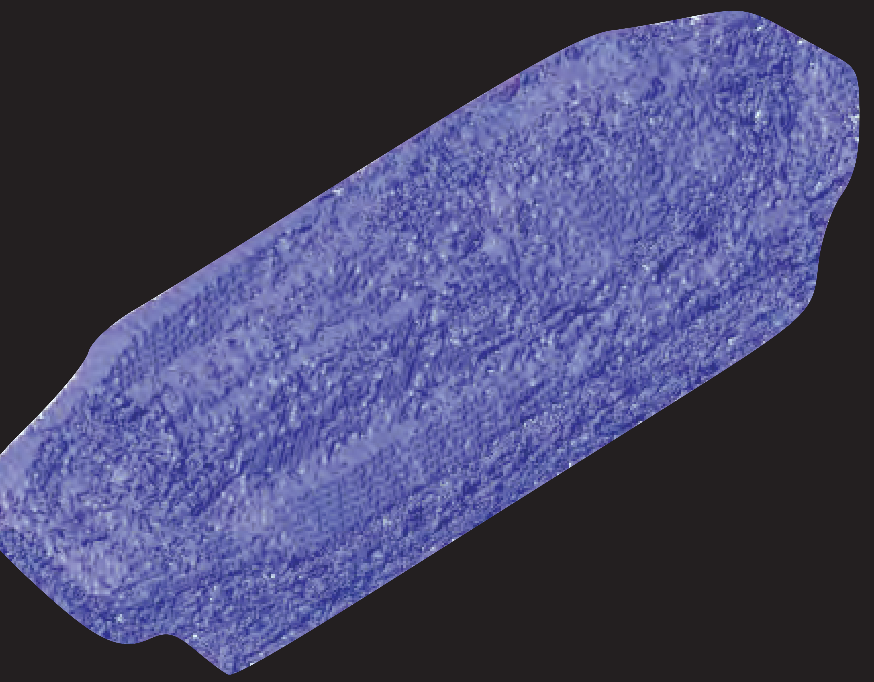


2000 2005 2010 2011 2012

Sensing revolution

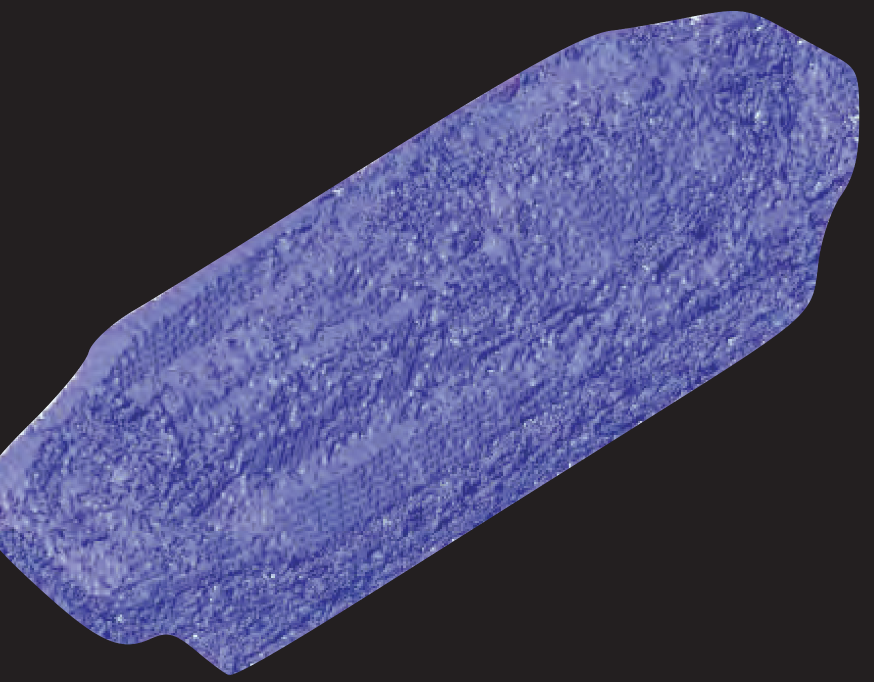
Intel Developper's Conference, 2011

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



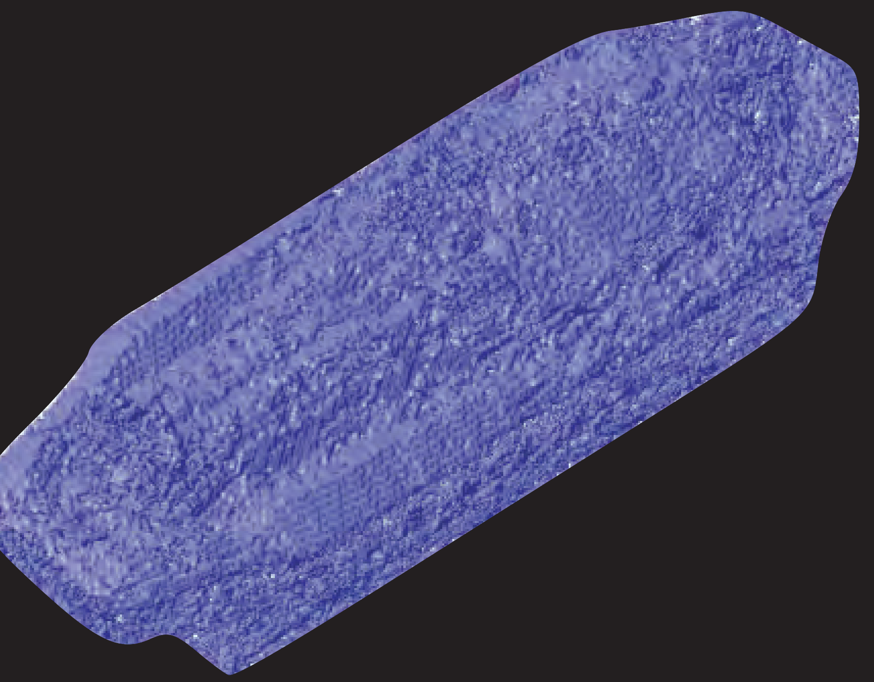
2000 2005 2010 2011 2012

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
Interntional Conference on 3D Vision 2011


Ice age
ICE AGE
THE MELTDOWN
2000
Sensing revolution
2007
2009
2010
2011
2012
The Metropolitan Museum of Art
2012
Perception - ice-age/revolution?
2014
2015
bedroom
bedroom
bathroom
washroom
corridor
restroom
balcony
2017
2018
2018
2019


2'10"
6'6"
8'
5'6"
Geometric Elements
3D primitive
2D primitive
1D primitive
0D primitive

Bottom-up

2D primitive

1D primitive

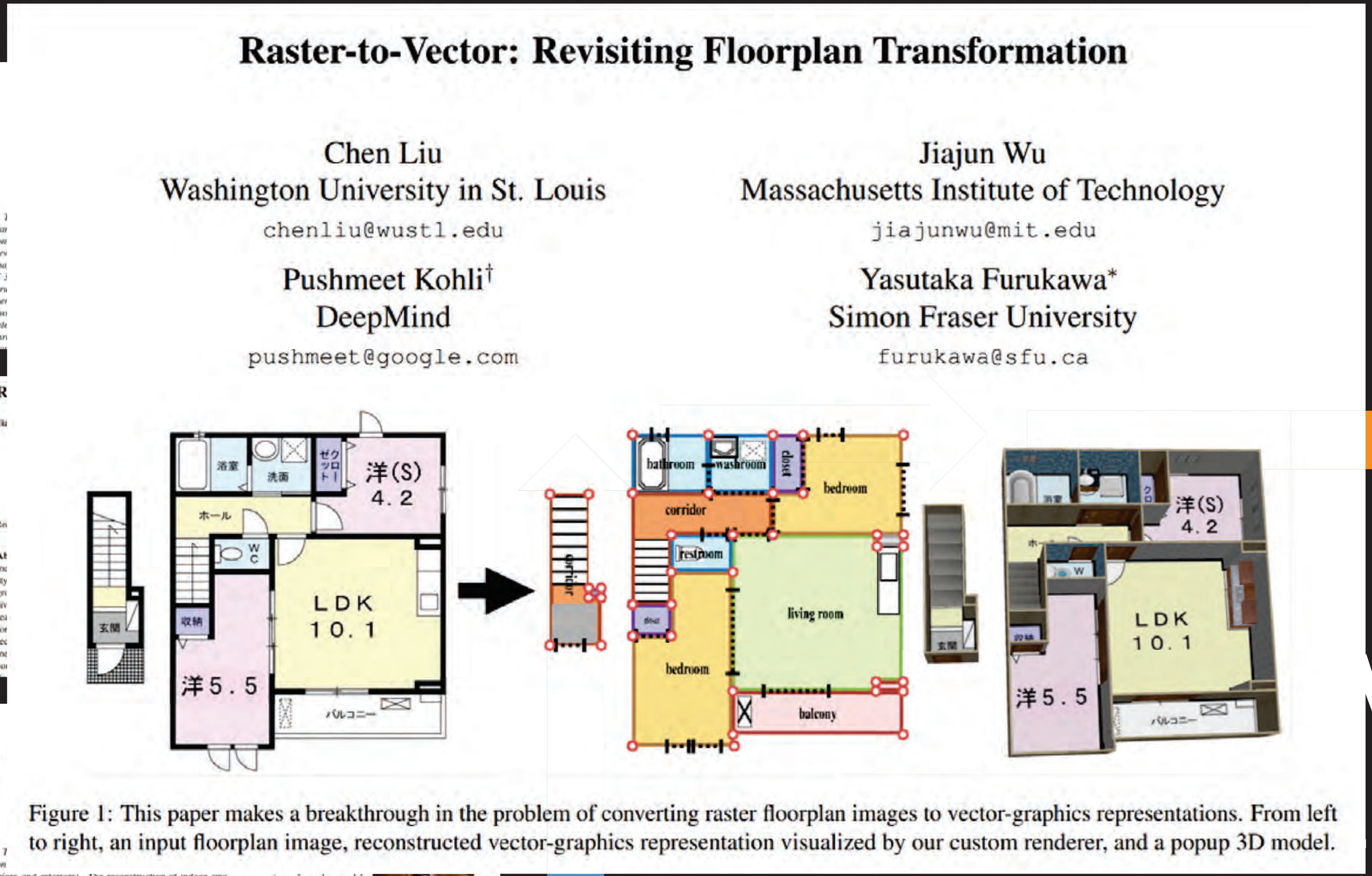
0D primitive


2'10"
8'
5'6"
Bottom-up
2D primitive
1D primitive
0D primitive


2'10"
8'
5'6"
Bottom-up
2D primitive
1D primitive
0D primitive


Bottom-up
2D primitive
1D primitive
0D primitive


Piecewise Planar and Compact Floorplan Reconstruction from Images
Structured Indoor Modeling
Raster-to-Vector: Revisiting Floorplan Transformation
FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans
Neural Procedural Reconstruction for Residential Buildings
PlaneNet: Piece-wise Planar Reconstruction from a Single RGB Image
Reconstructing the World's Museums
PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image
Reconstructing Building Interiors from Images
Floor-SP: Inverse CAD for Floorplans by Sequential Room-wise Shortest Path
Manhattan-world Stereo
Vectorizing World Buildings: Planar Graph Reconstruction by Primitive Detection and Relationship Classification
CVPR-ICCV-ECCV papers
for geometry perception…
Sensing
Ice age
Revolution
Perception
Ice age
?Revolution?
1985
1990
1995
2000
2005
2010
2015
2020



Raster-to-Vector: Revisiting Floorplan Transformation

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

Jiajun Wu
Massachusetts Institute of Technology
jiajunwu@mit.edu

Pushmeet Kohli†
DeepMind
pushmeet@google.com

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.

NII 国立情報学研究所 National Institute of Informatics
IDR 情報学研究データリポジトリ
English

HOME | データ一覧 | 研究成果一覧 | ユーザフォーラム | 組織 | 関連リンク | 問い合わせ

> HOME > データ一覧 > LIFULL HOME'Sデータセット

- 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，特定の物件に紐付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」といった画像の種別や，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

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

1985 1990 1995 2000 2005 2010 2015 2020


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-
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
Jiajun Wu
Massachusetts Institute of Technology
jiajunwu@mit.edu
Pushmeet Kohli†
DeepMind
pushmeet@google.com
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.
arXiv [cs.CV] 31 Mar 2018
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

Reconstructing Building Interiors from Images

Raster-to-Vector: Revisiting Floorplan Transformation

Manhattan-world Stereo

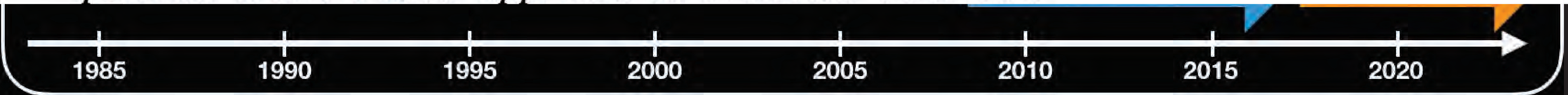
Yasutaka Furukawa Brian Curless Steven M. Seitz
Department of Computer Science & Engineering
University of Washington, USA
{furukawa,curless,seitz}@cs.washington.edu

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

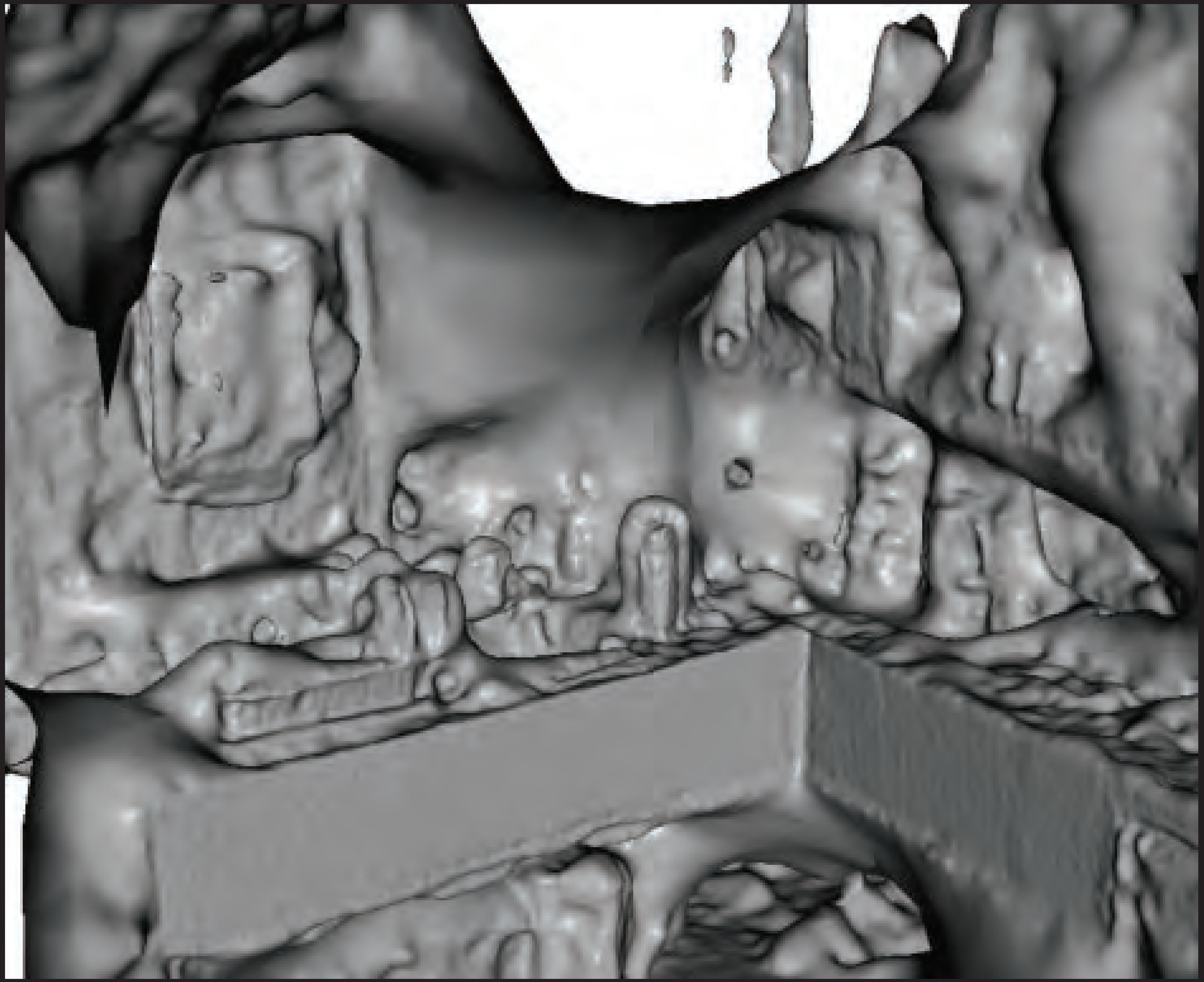
Abstract

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.

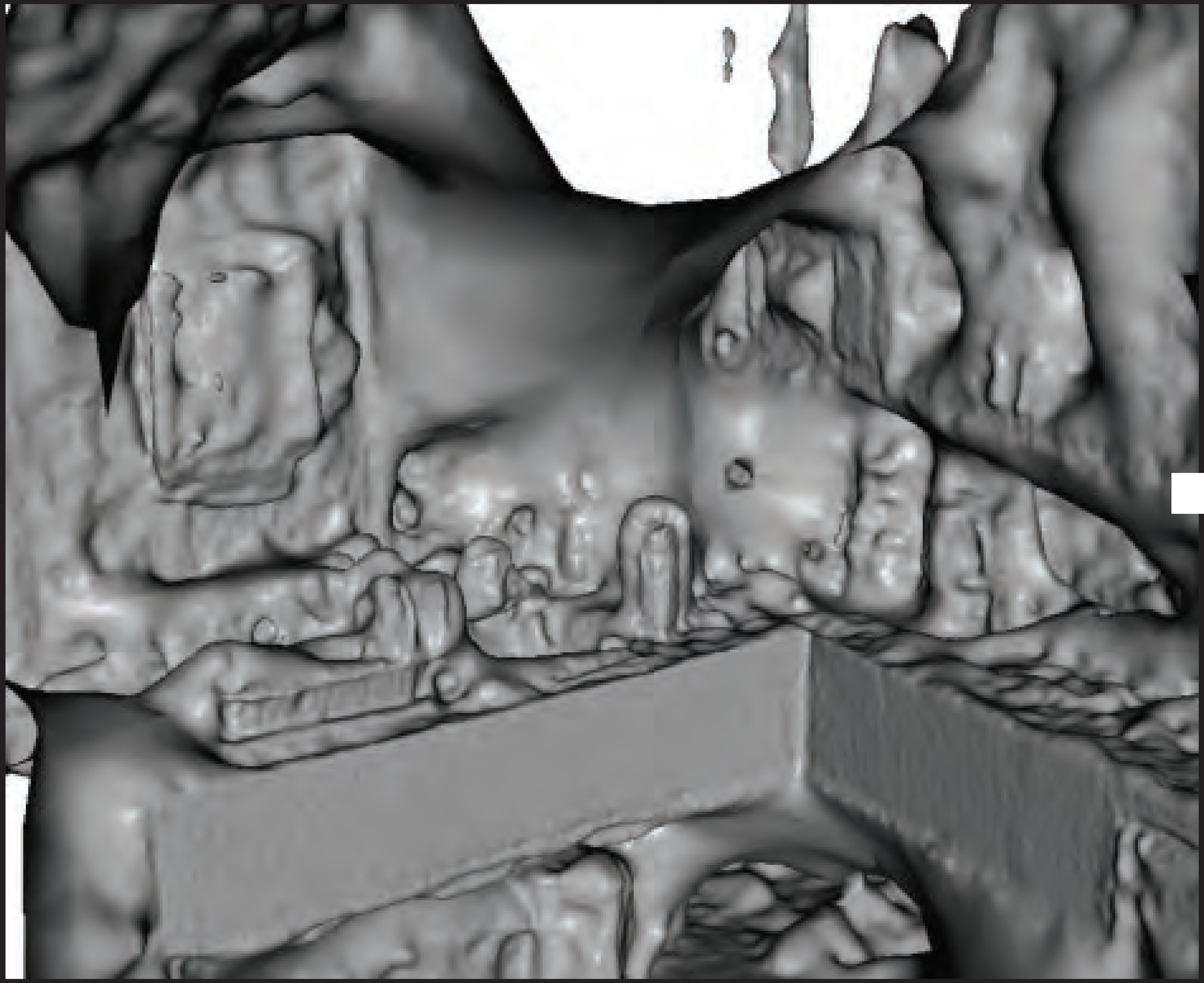


Geometry sensing



[Furukawa and Ponce, 2007]

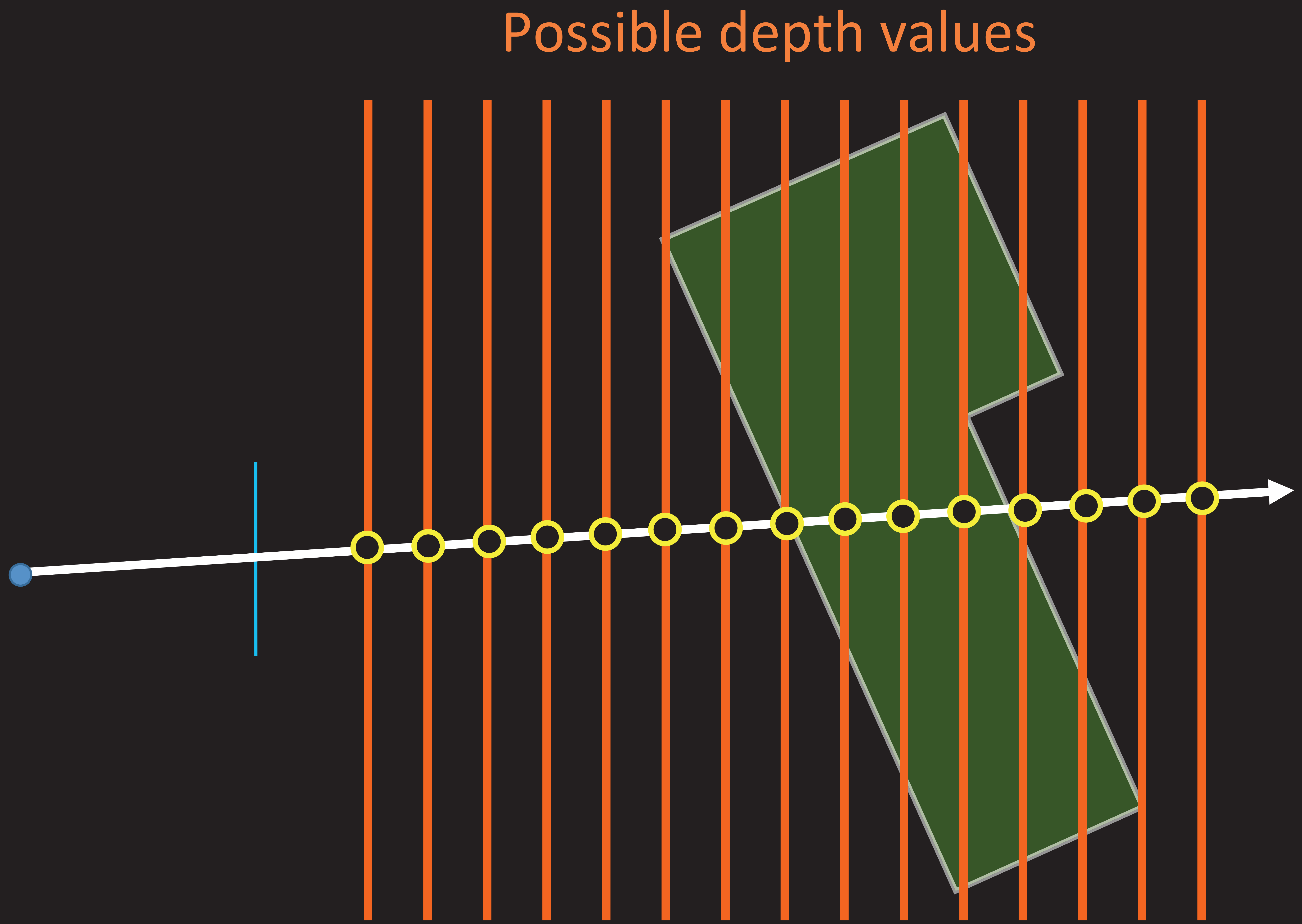
Geometry perception



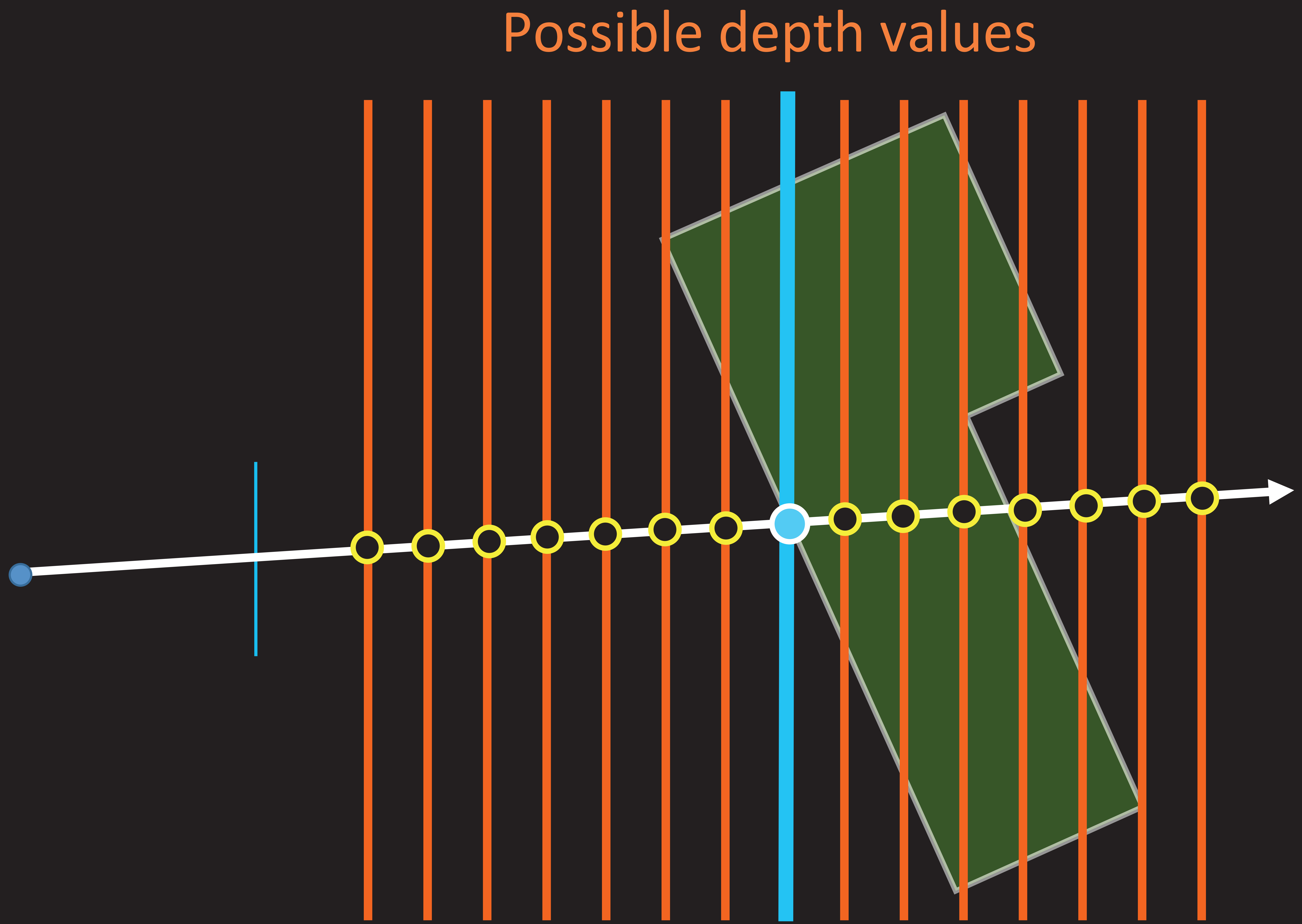
[Furukawa and Ponce, 2007]

Planarity/orthogonality enforcement

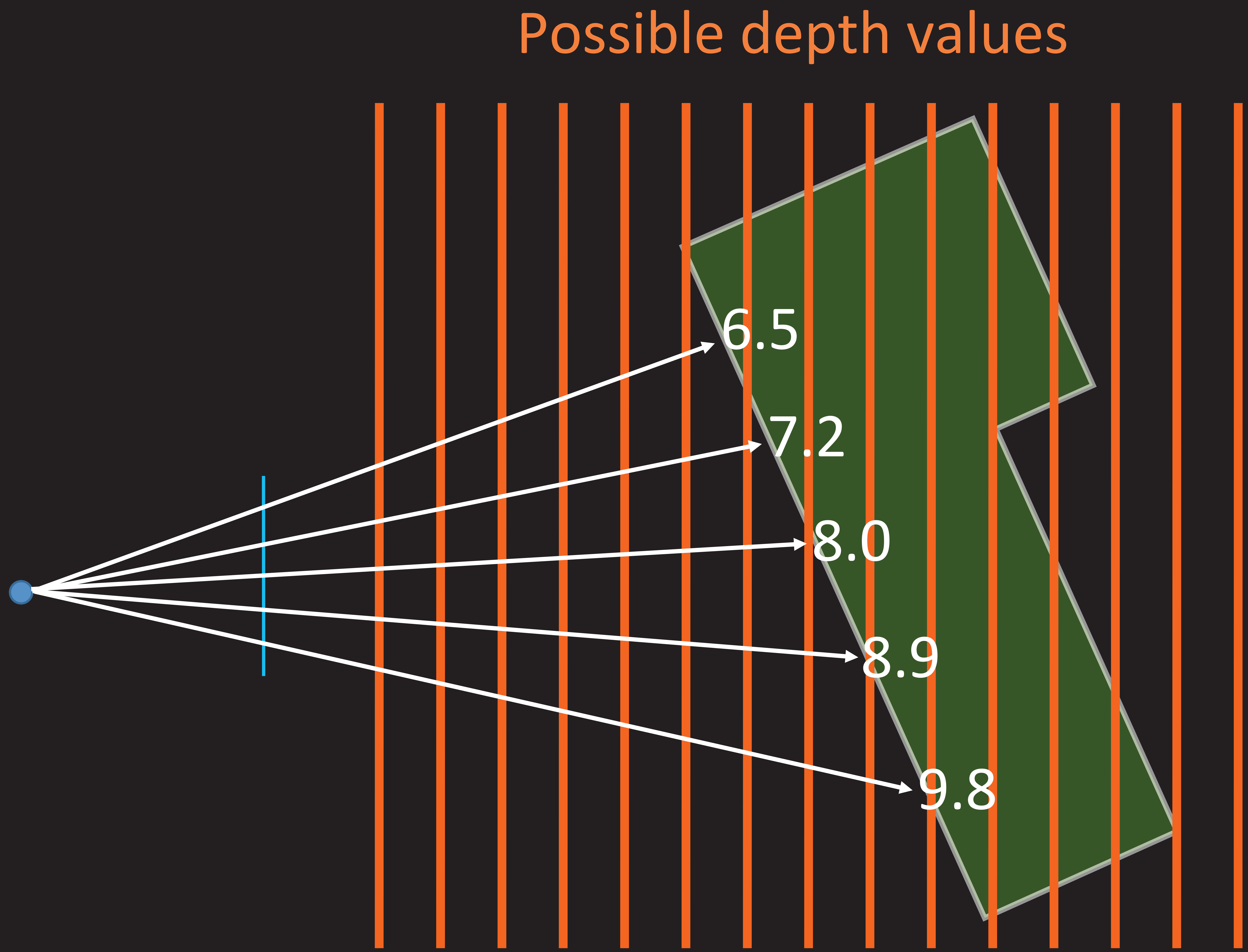
(Sensing) Depthmap estimation



(Sensing) Depthmap estimation



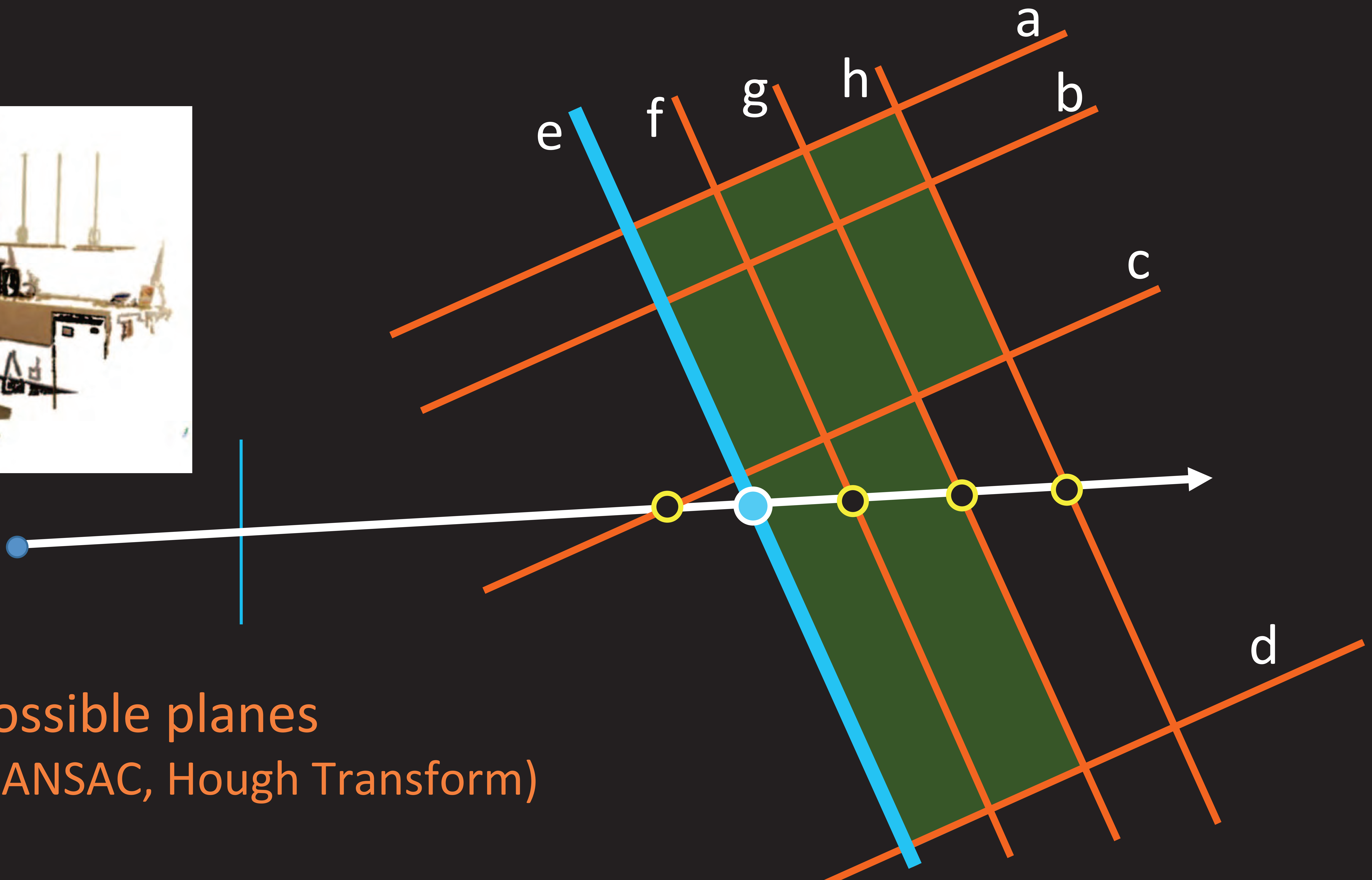
(Sensing) Depthmap estimation



~~(Sensing)~~ Depthmap estimation

(Perception)

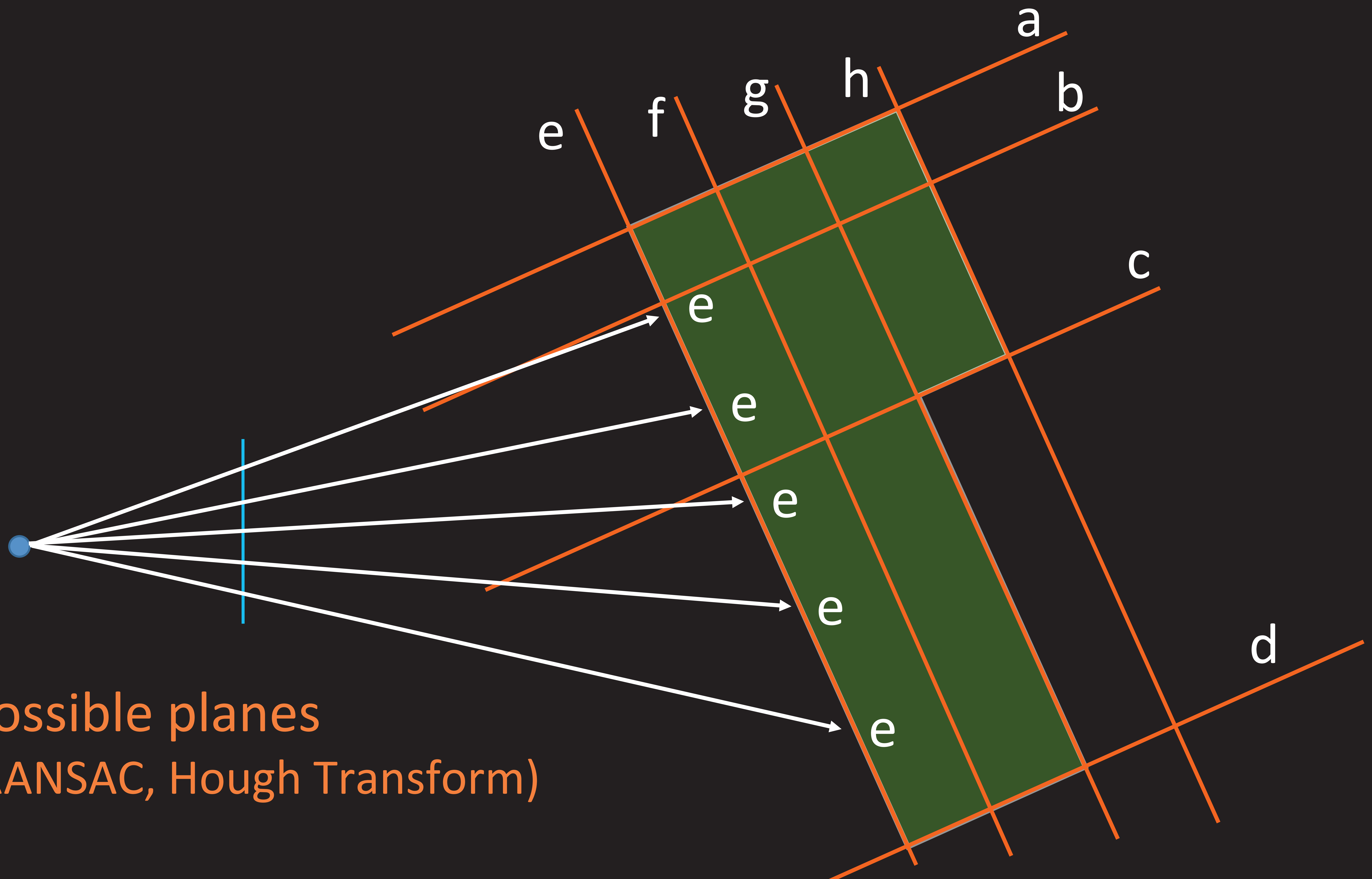
(Perception) Depthmap estimation



Possible planes

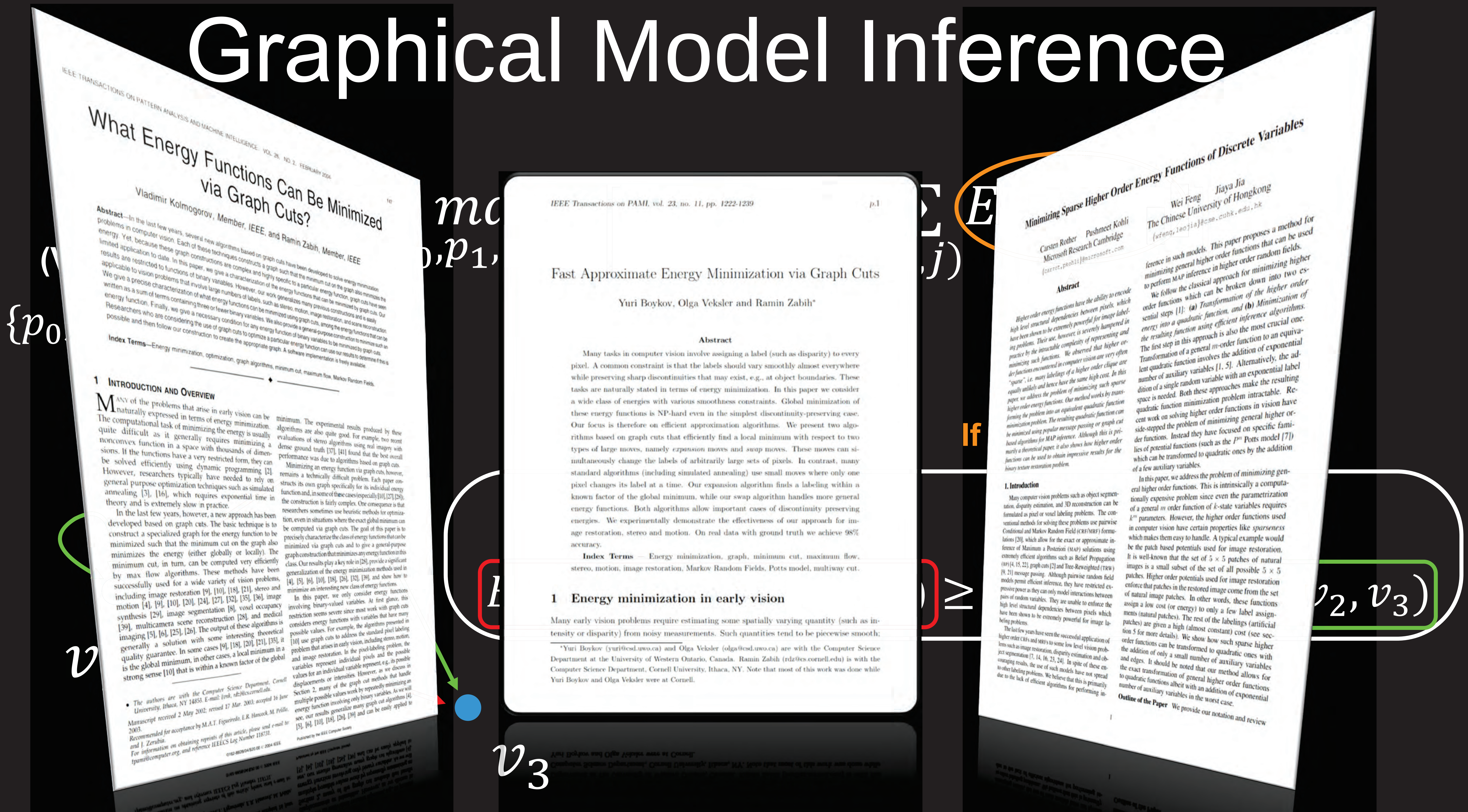
(RANSAC, Hough Transform)

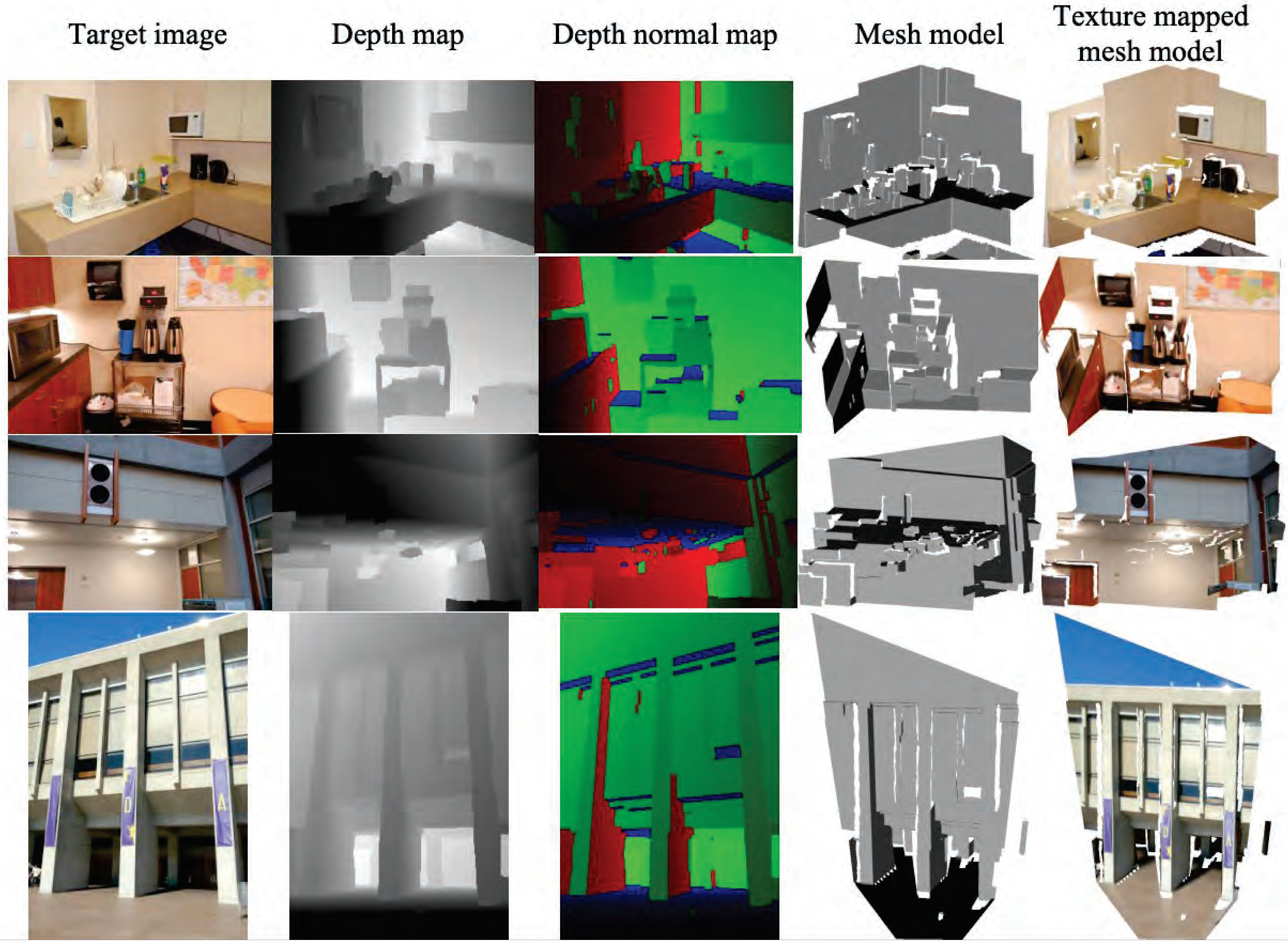
(Perception) Planemap estimation



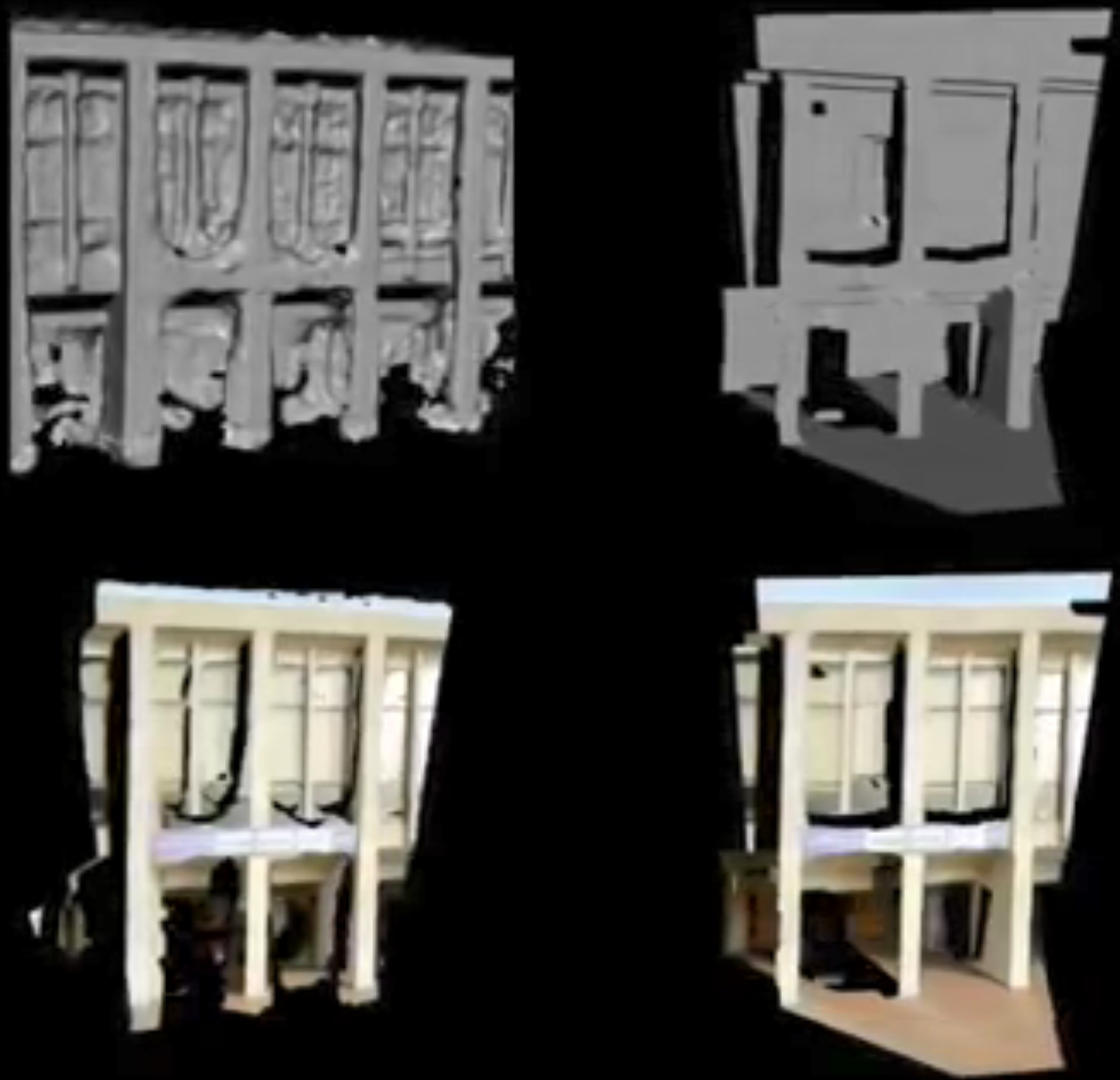
Possible planes
(RANSAC, Hough Transform)

Graphical Model Inference




Target image
Depth map
Depth normal map
Mesh model
Texture mapped mesh model

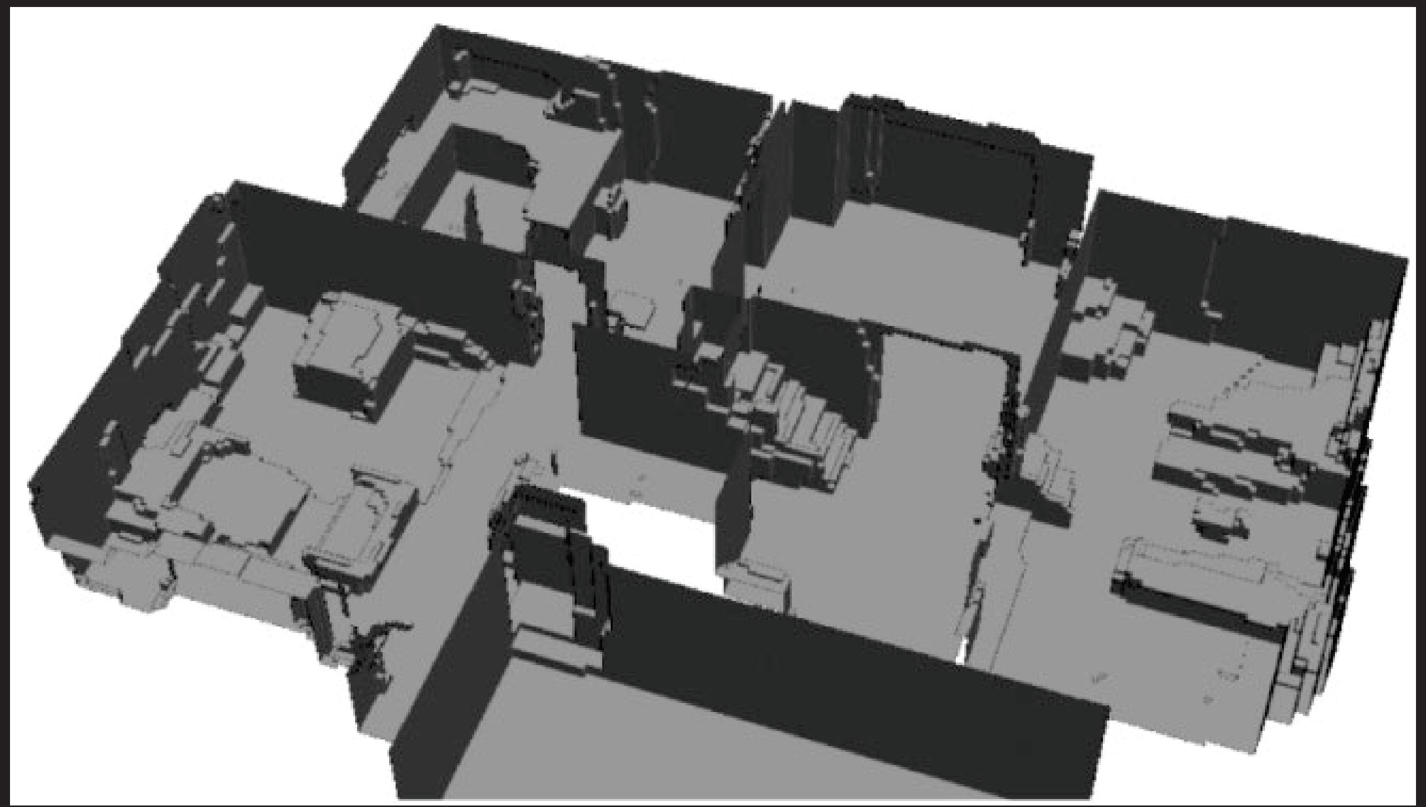
PMVS+Poisson | Our Result



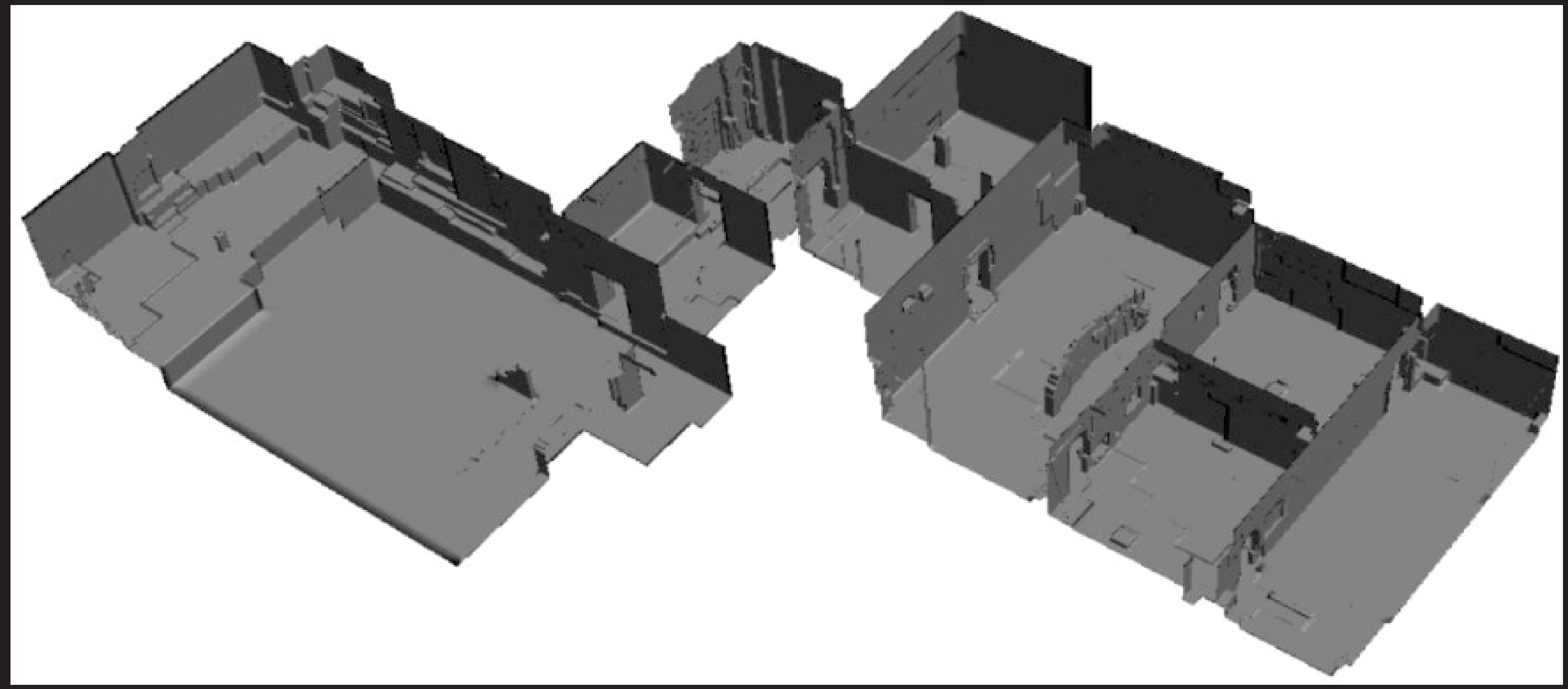
Reconstructing building interiors from images
Yasutaka Furukawa, Rick Szeliski, Brian Curless, and Steve Seitz
International Conference on Computer Vision 2009

Kitchen - 22 images

house - 148 images

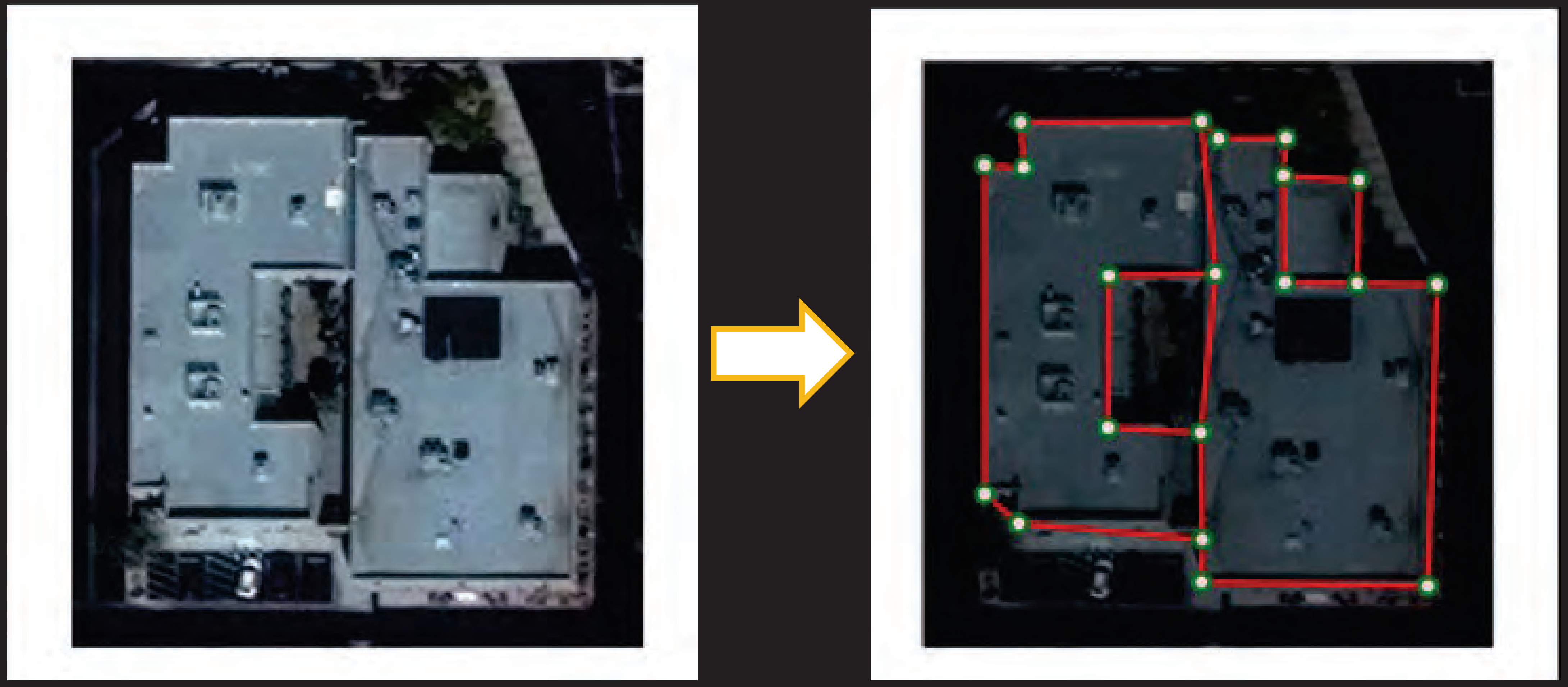


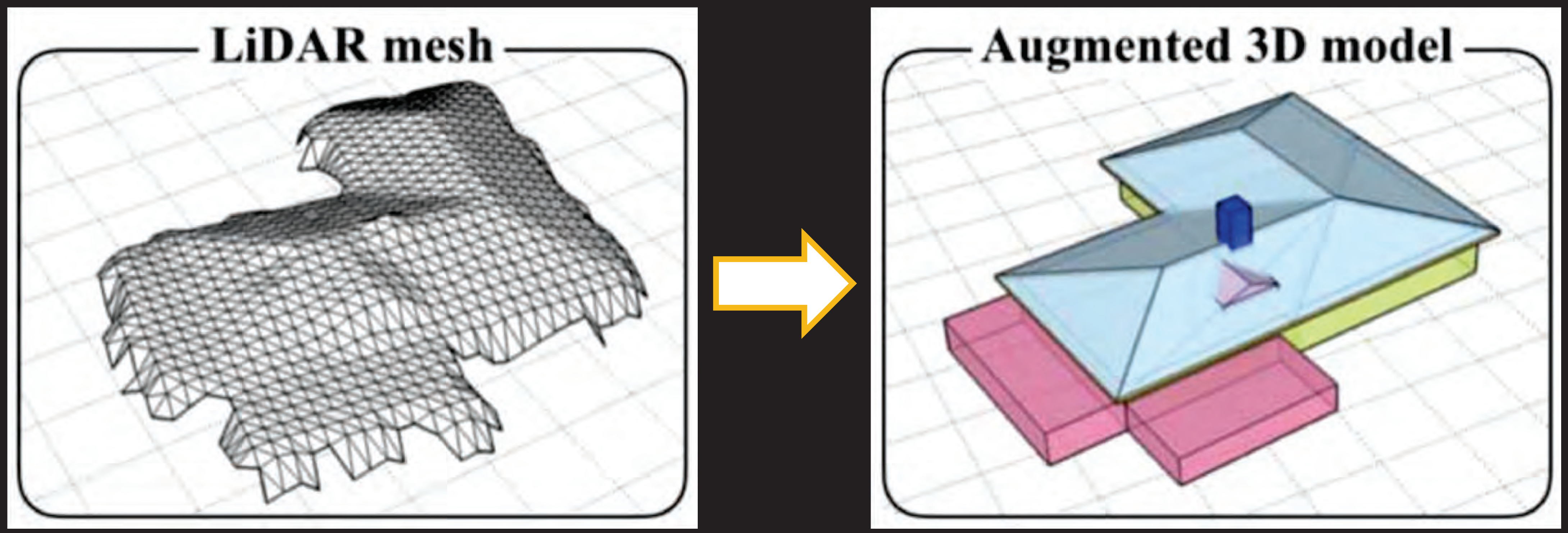
gallery - 492 images

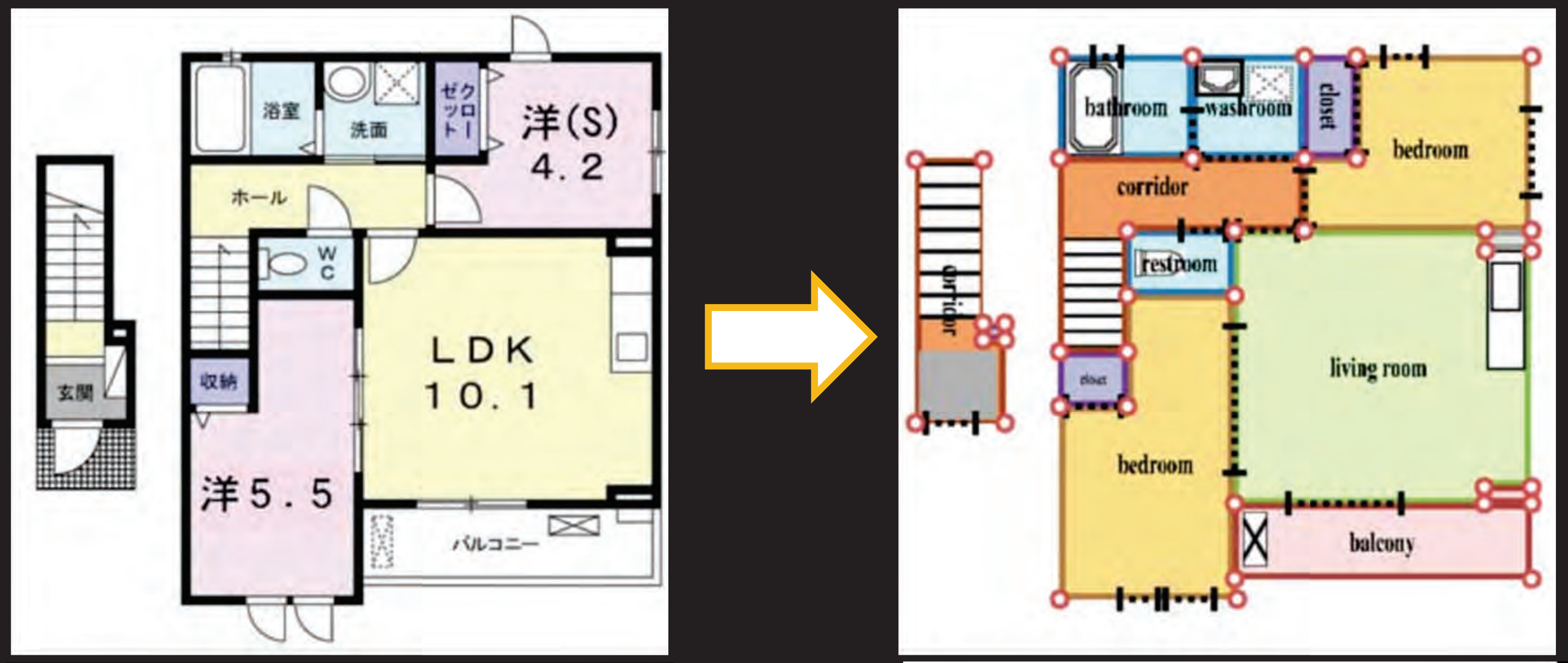


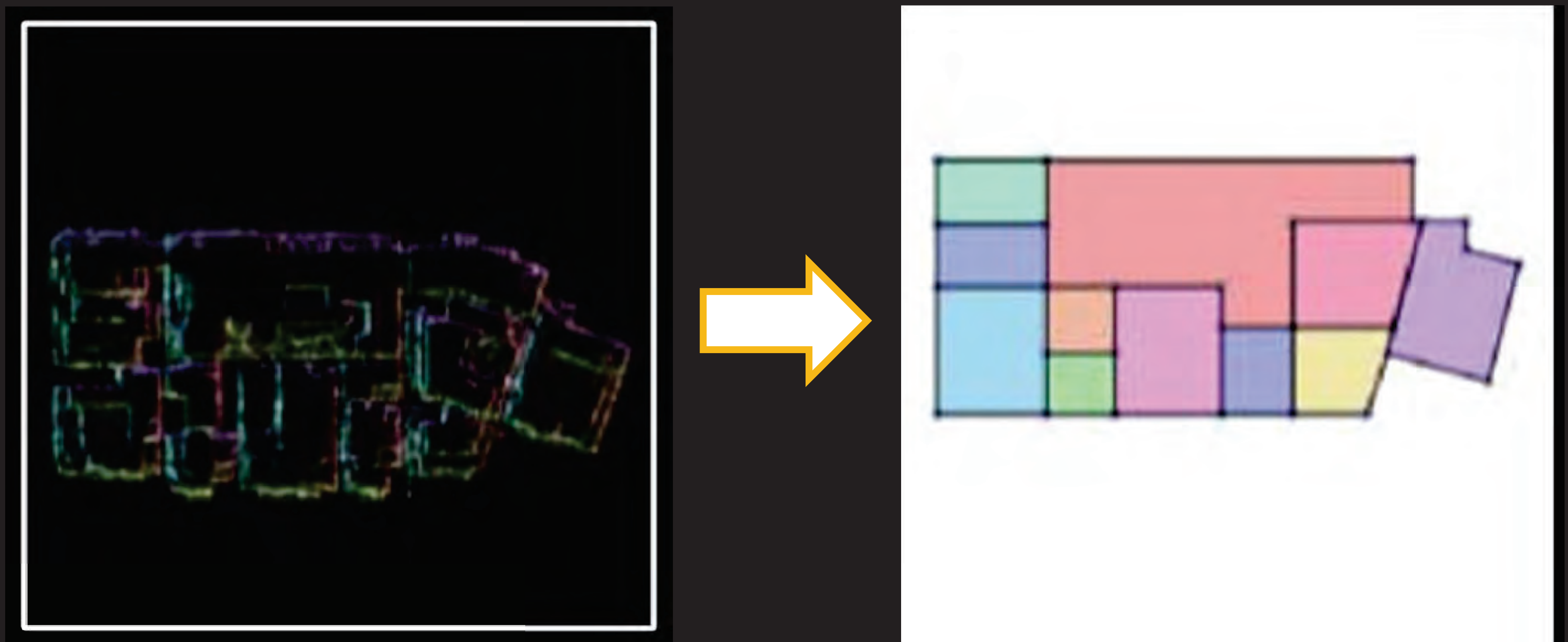

Reconstructing Building Interiors from Images
Yasutaka Furukawa, Brian Curless, Steven M. Seitz
University of Washington, Seattle, USA
{furukawa,curless,seitz}@cs.washington.edu
Richard Szeliski
Microsoft Research, Redmond, USA
szeliski@microsoft.com
Abstract
This paper proposes a fully automated 3D 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-
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
Jiajun Wu
Massachusetts Institute of Technology
jiajunwu@mit.edu
Pushmeet Kohli†
DeepMind
pushmeet@google.com
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.
[cs.CV] 31 Mar 2018
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

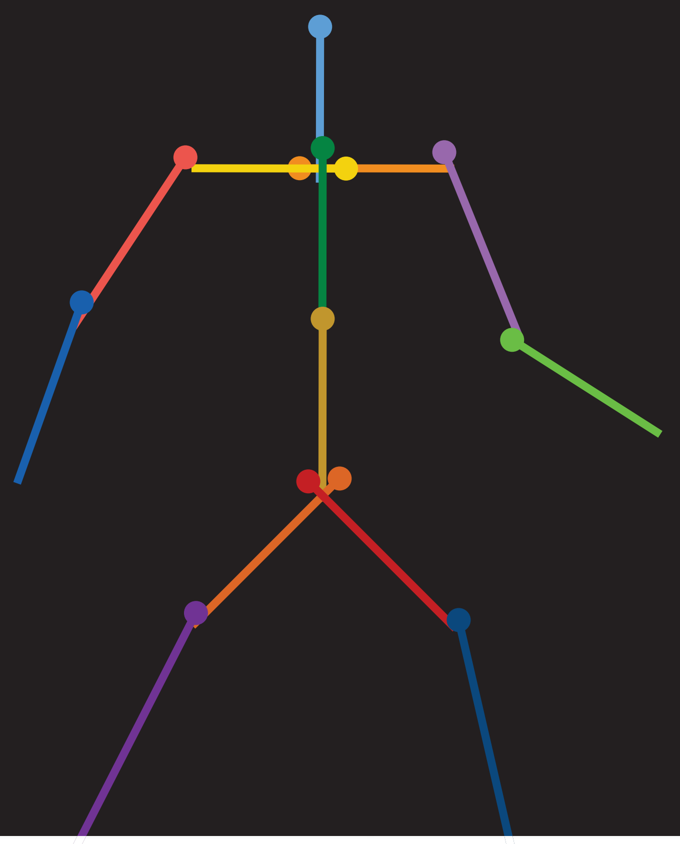




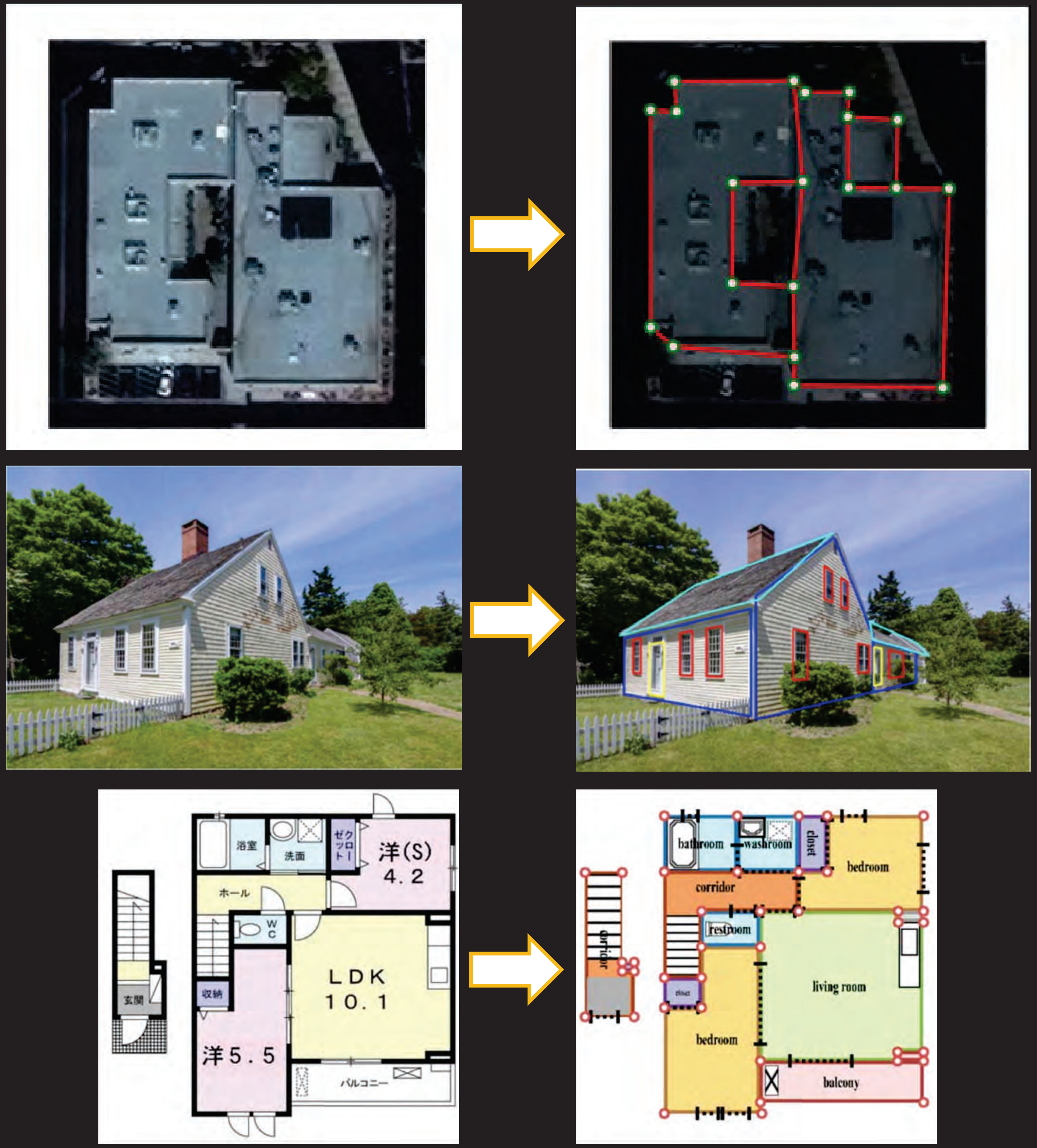




Why is this hard?



Fixed topology

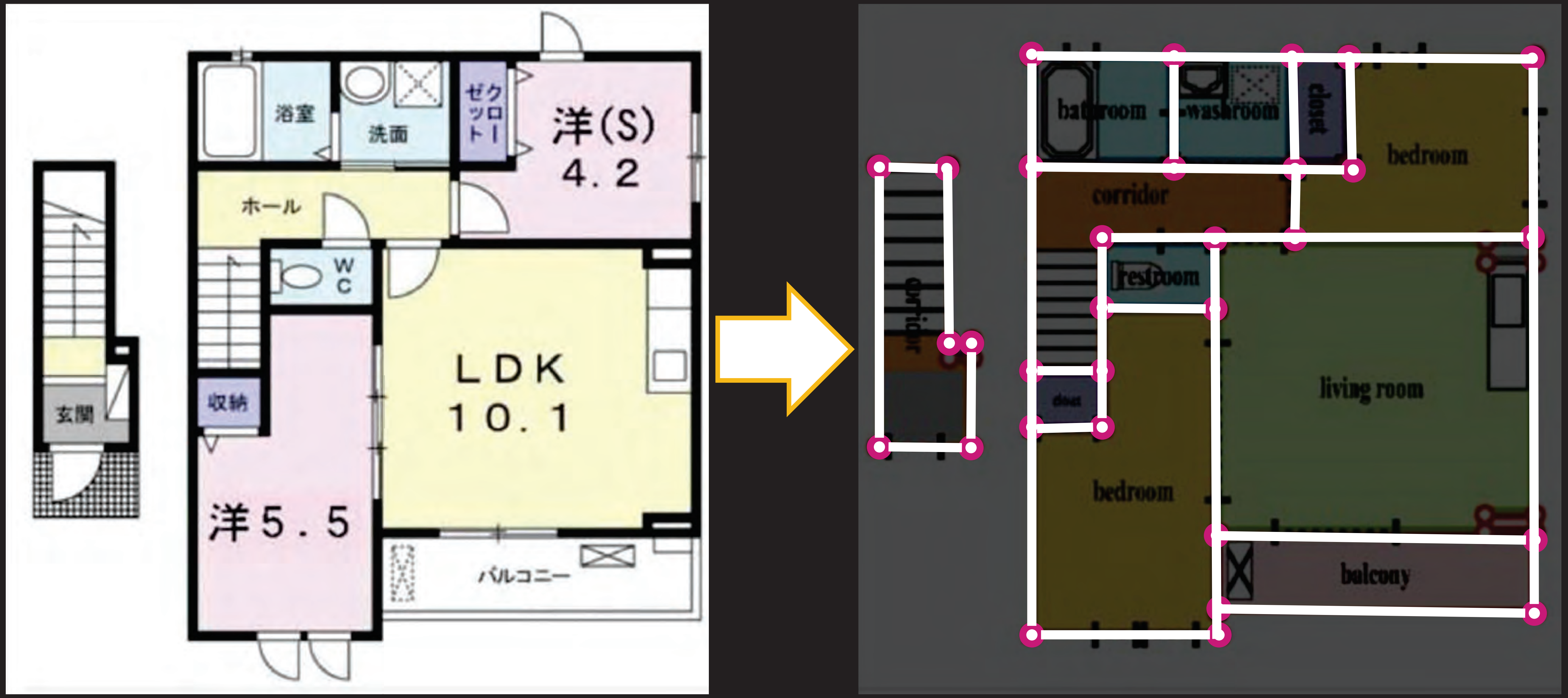


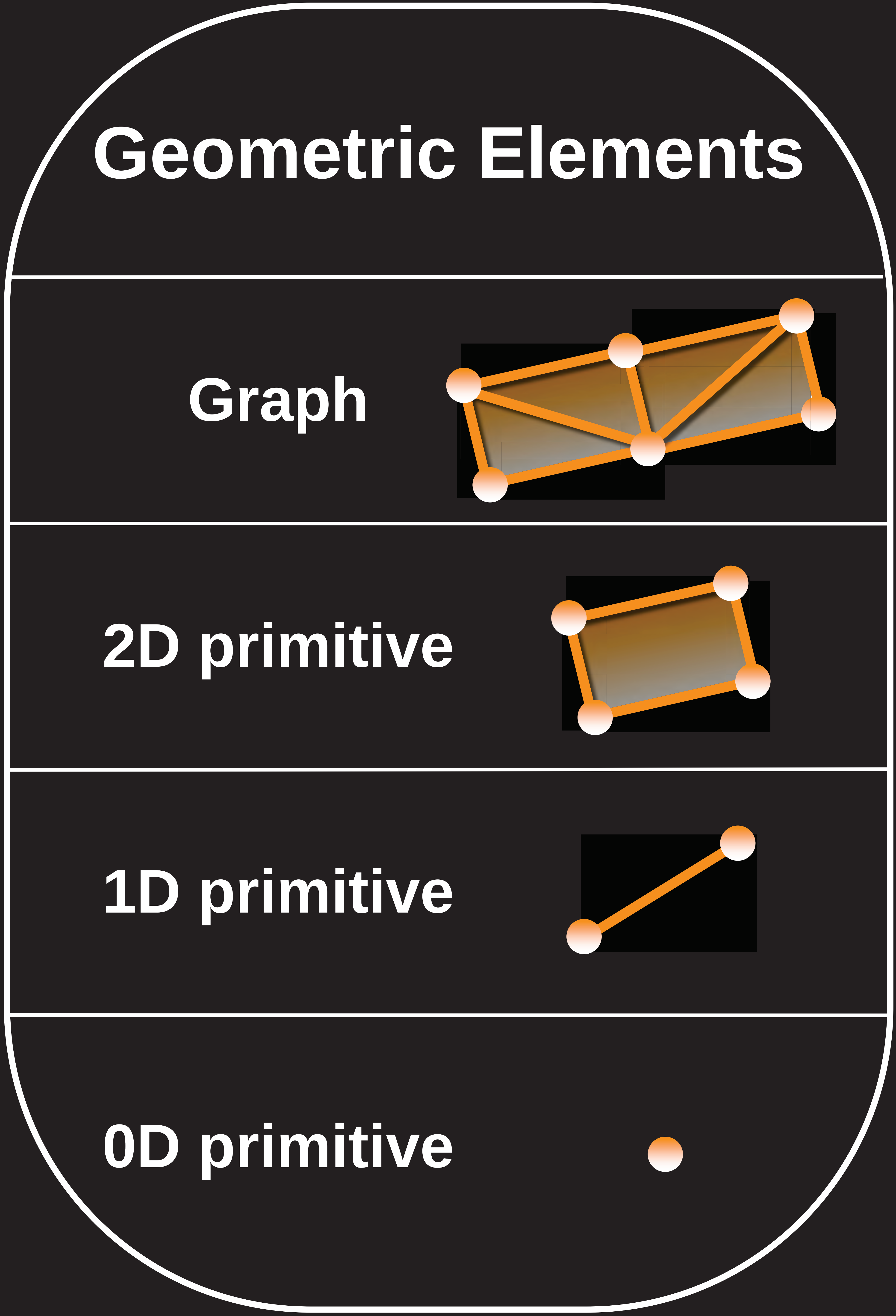
Arbitrary topology

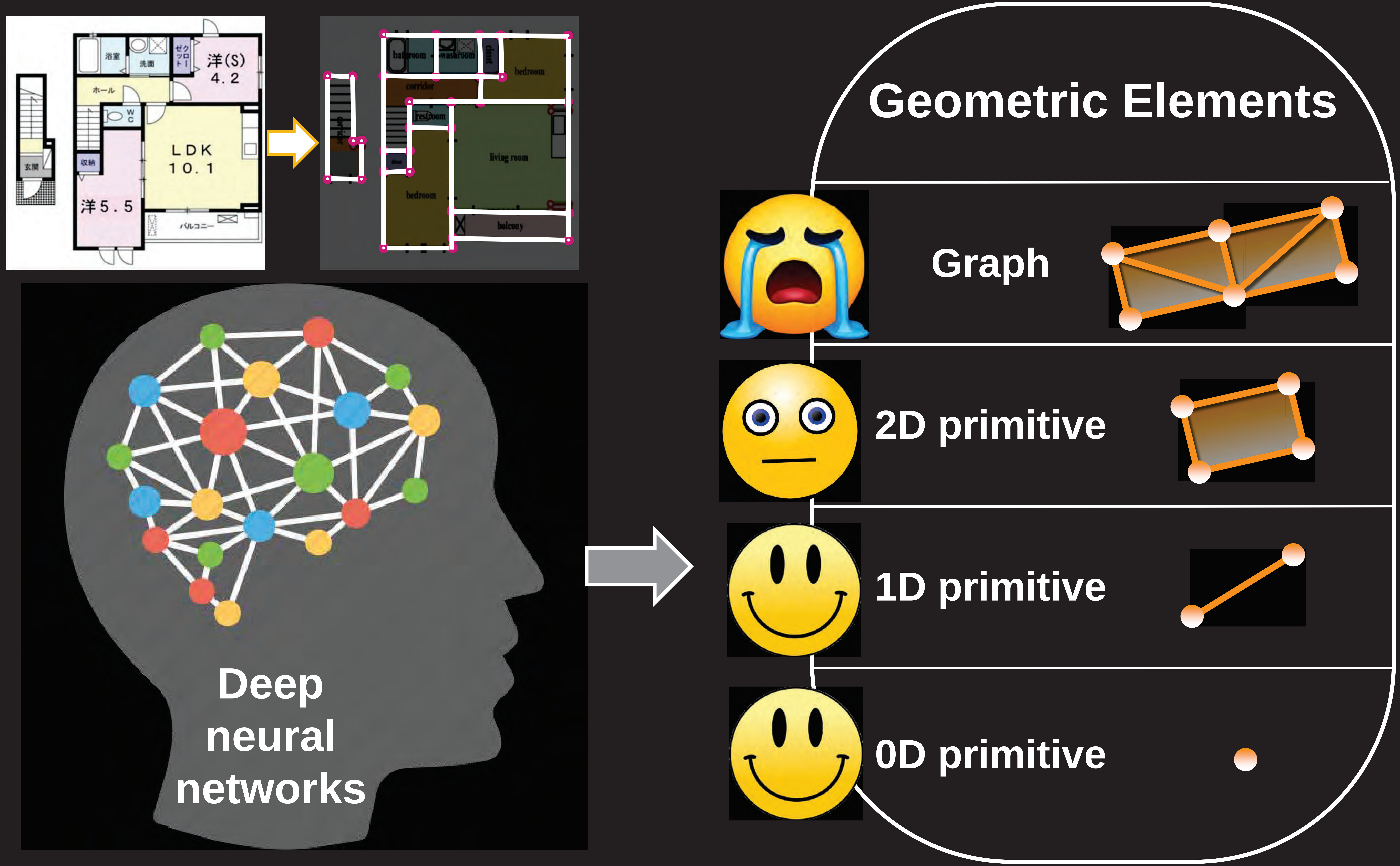
Planar graph inference

Planar graph inference

Planar graph inference



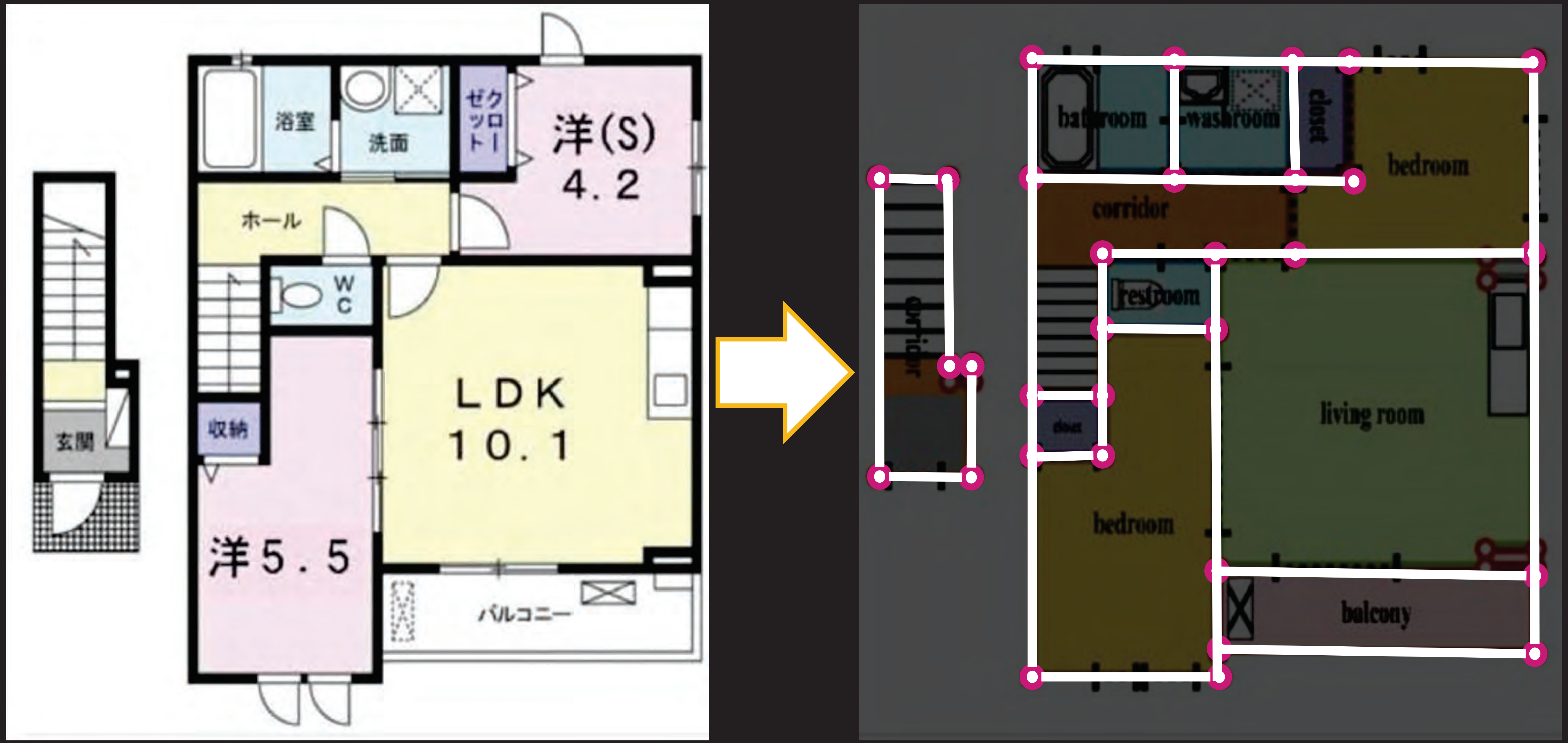


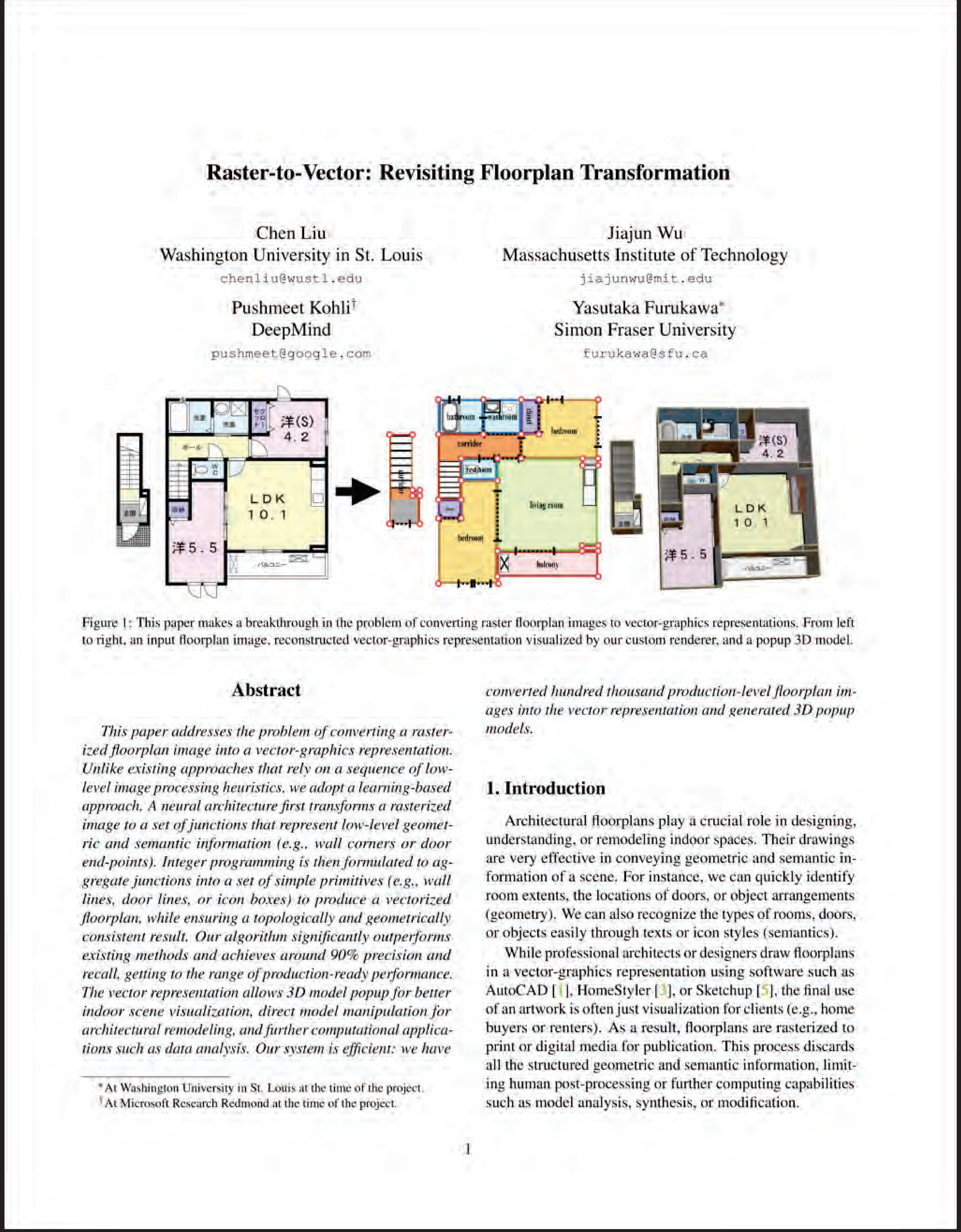


浴室
洗面
洋(S)
4.2
ホール
LDK
10.1
洋5.5
収納
玄関
バルコニー
bathroom
washroom
closet
bedroom
corridor
restroom
living room
bedroom
balcony
Deep
neural
networks
Geometric Elements
Graph
2D primitive
1D primitive
0D primitive


Geometric Elements
Optimization
(Integer Programming)
Graph
2D primitive
1D primitive
0D primitive
Deep
neural
networks

Planar graph inference





Raster-to-Vector: Revisiting Floorplan Transformation

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

Jiajun Wu
Massachusetts Institute of Technology
jiajunwu@mit.edu

Pushmeet Kohli†
DeepMind
pushmeet@google.com

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

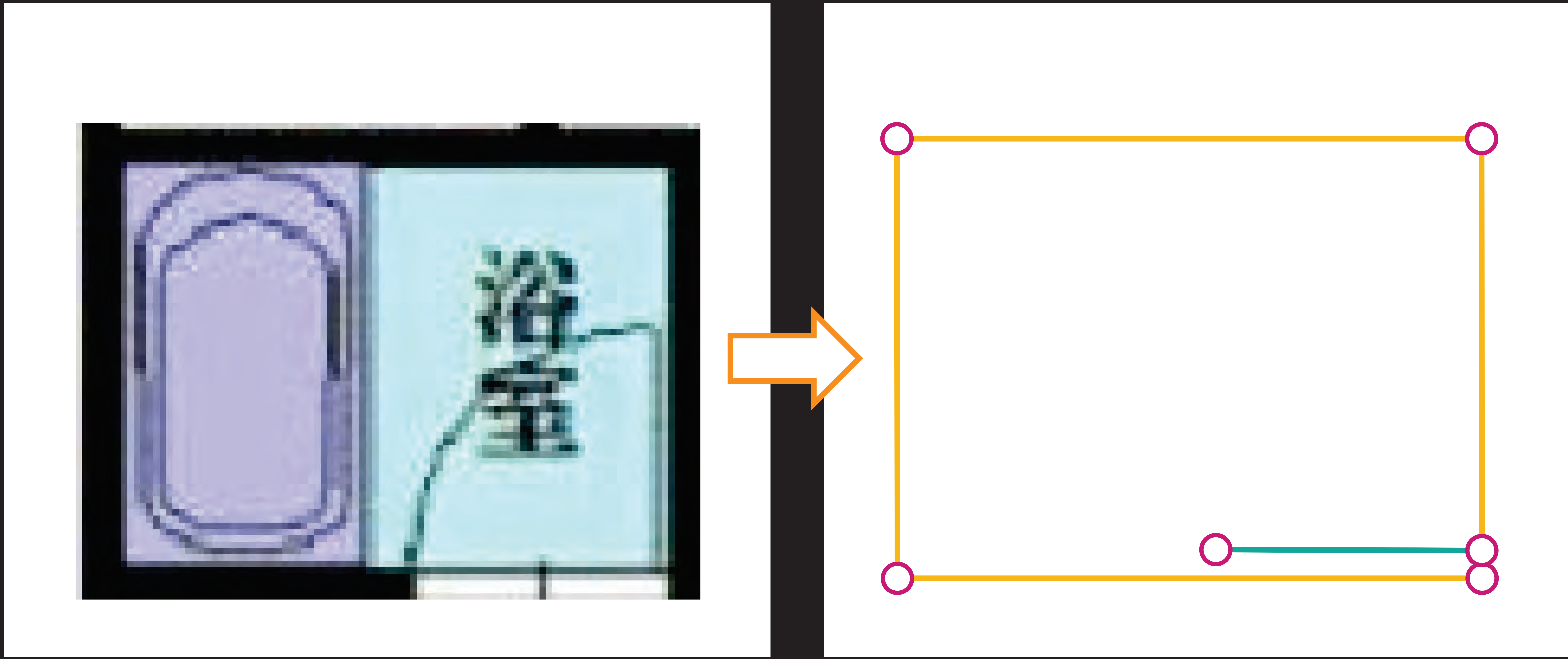
Architectural floorplans play a crucial role in designing, understanding, or remodeling indoor spaces. Their drawings are very effective in conveying geometric and semantic information of a scene. For instance, we can quickly identify room extents, the locations of doors, or object arrangements (geometry). 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.

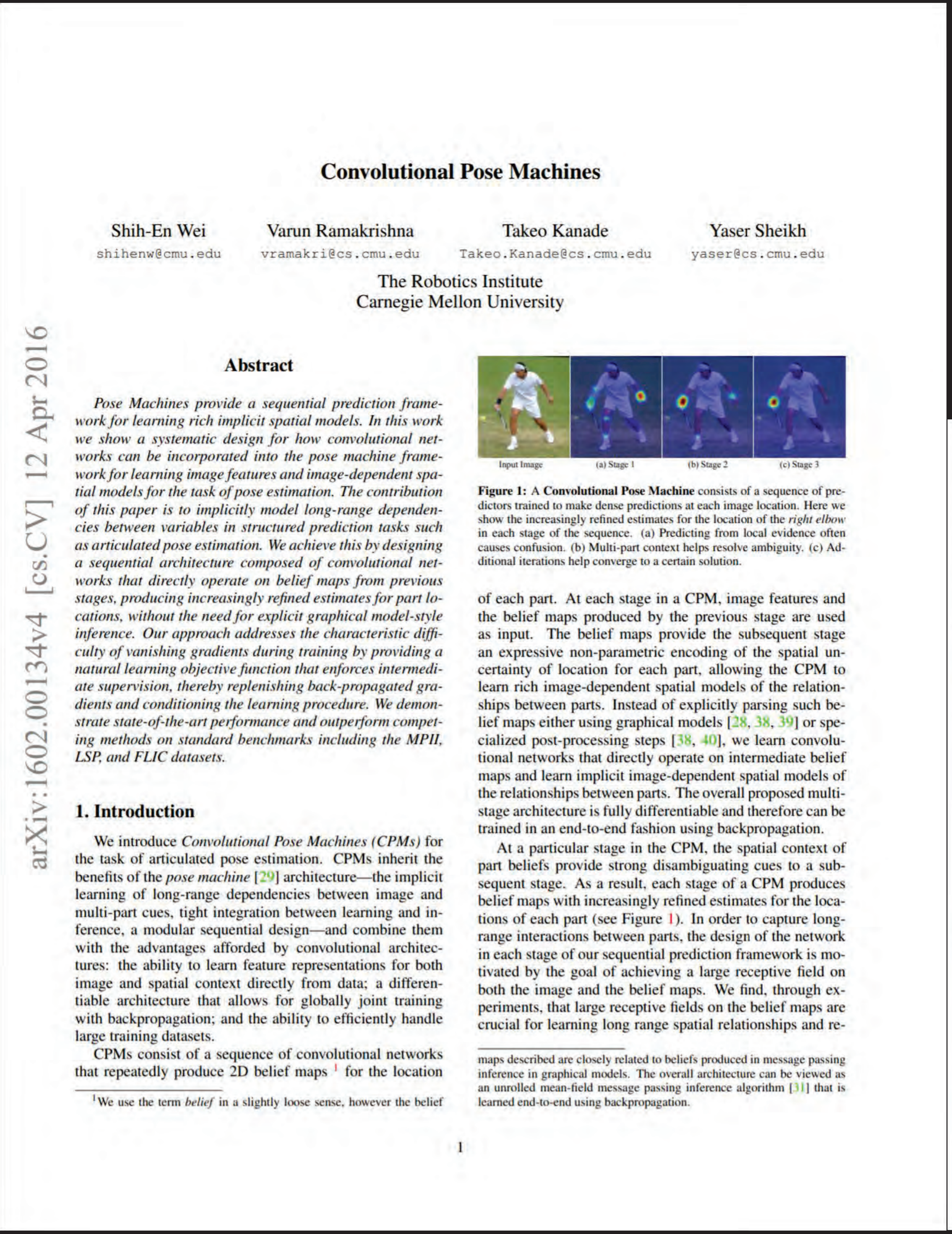
*At Washington University in St. Louis at the time of the project.
†At Microsoft Research Redmond at the time of the project.

1

Toy example



Corner detection



Convolutional Pose Machines

Shih-En Wei
shihenw@cmu.edu

Varun Ramakrishna
vramakri@cs.cmu.edu

Takeo Kanade
Takeo.Kanade@cs.cmu.edu

Yaser Sheikh
yaser@cs.cmu.edu

The Robotics Institute
Carnegie Mellon University

arXiv:1602.00134v4 [cs.CV] 12 Apr 2016

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.

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 [1] for the location 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-

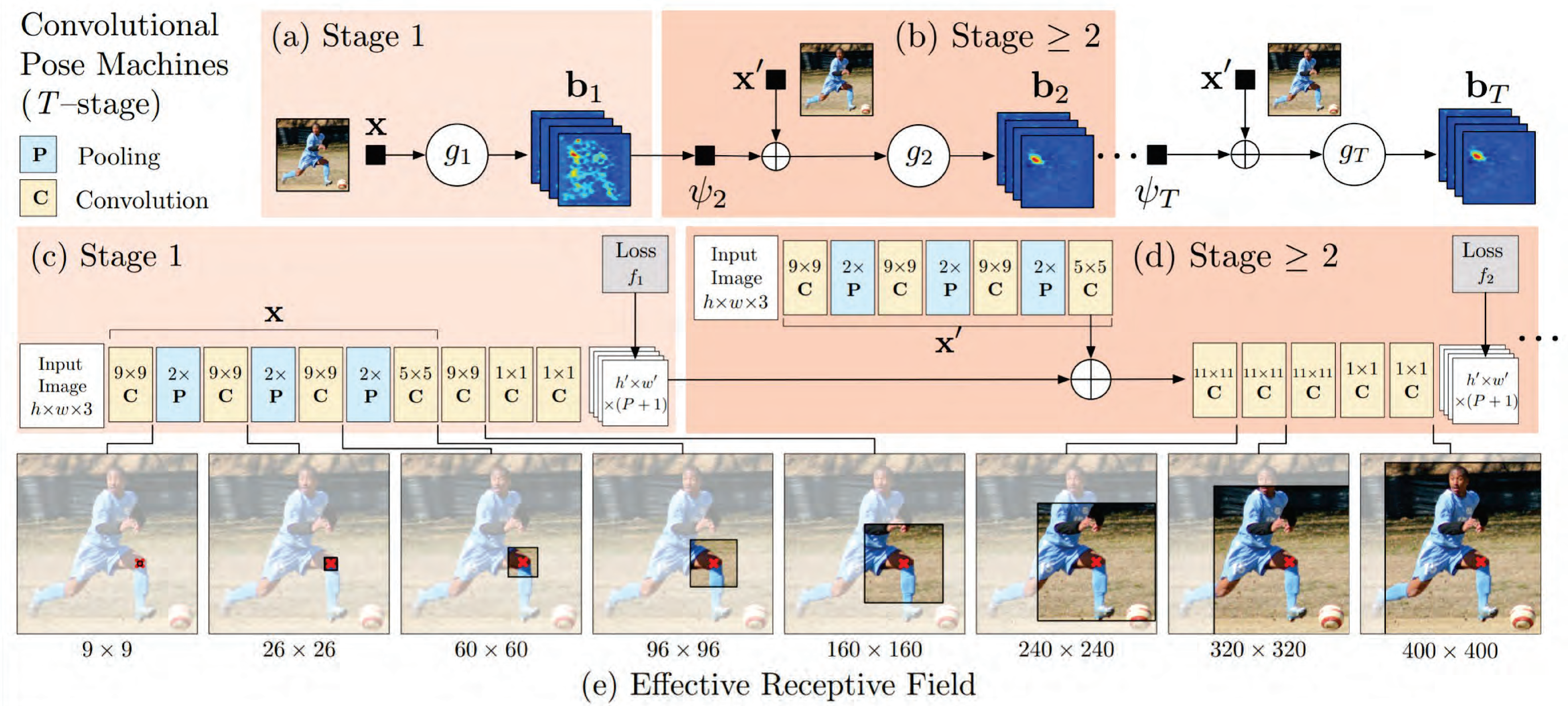
Input Image (a) Stage 1 (b) Stage 2 (c) Stage 3

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.

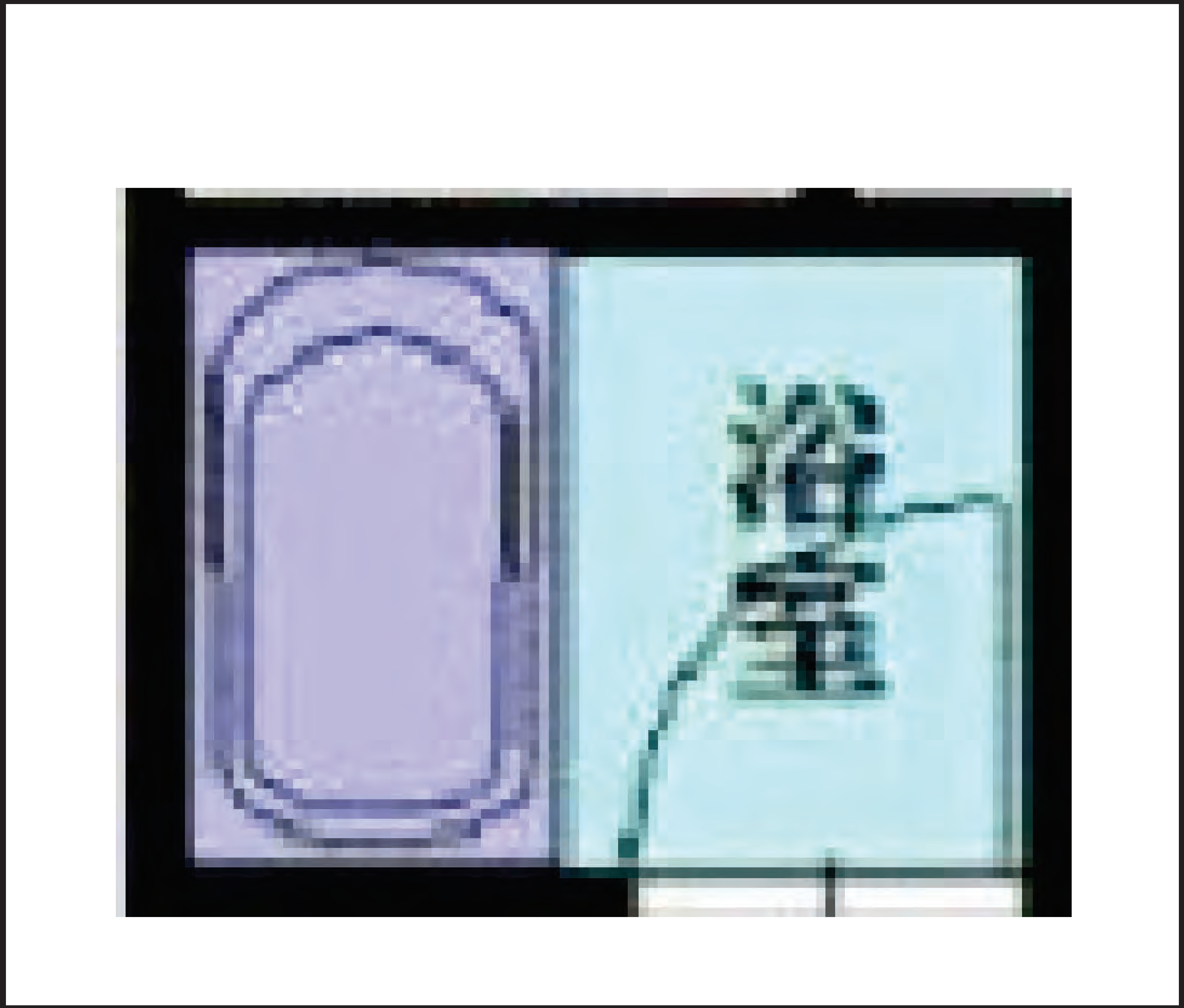
[1] We use the term *belief* in a slightly loose sense, however the belief maps described are closely related to beliefs produced in message passing inference in graphical models. The overall architecture can be viewed as an unrolled mean-field message passing inference algorithm [31] that is learned end-to-end using backpropagation.

1

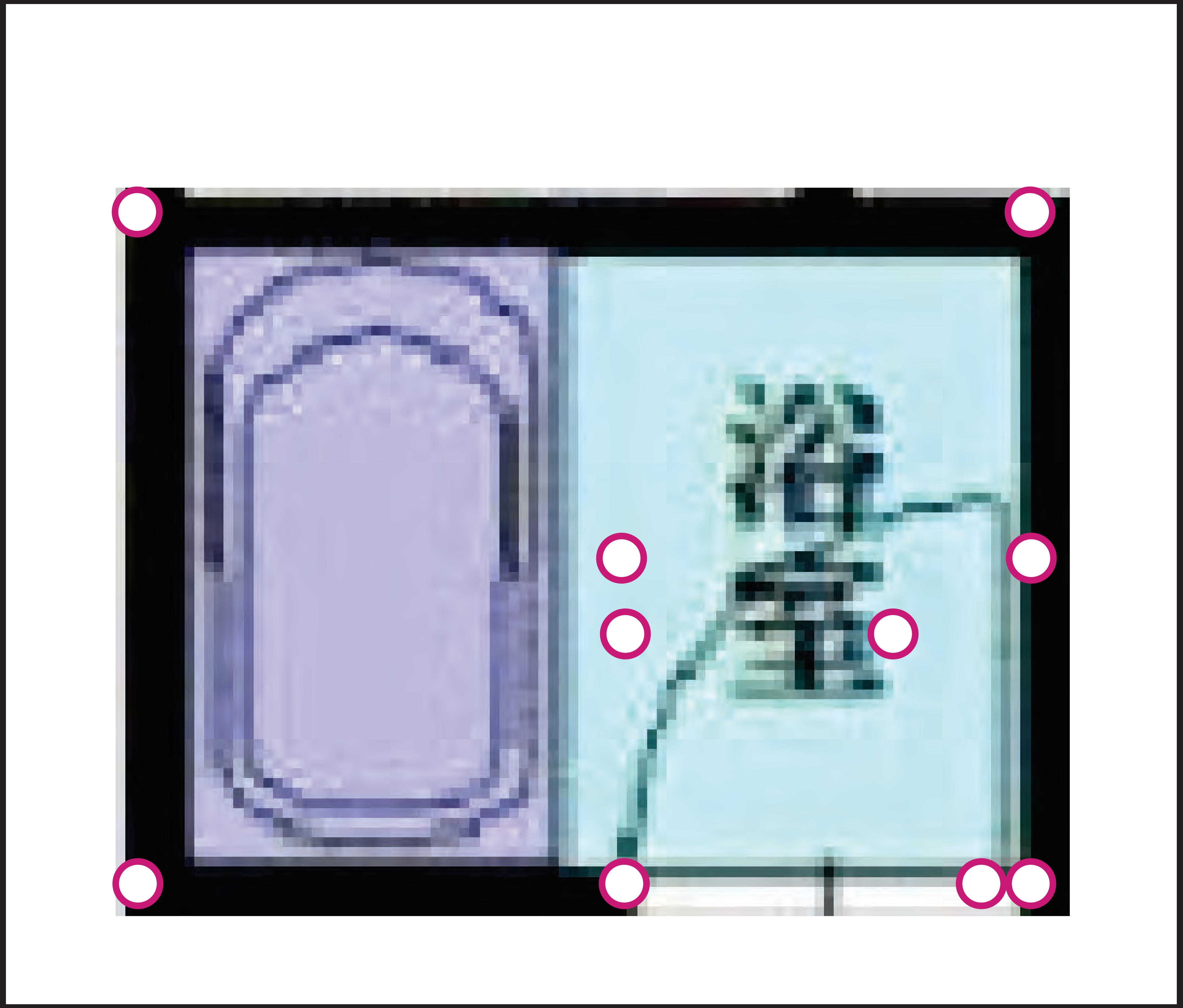
Convolutional Pose Machines
[Wei, Ramakrishna, Kanade, Sheikh, 2016]



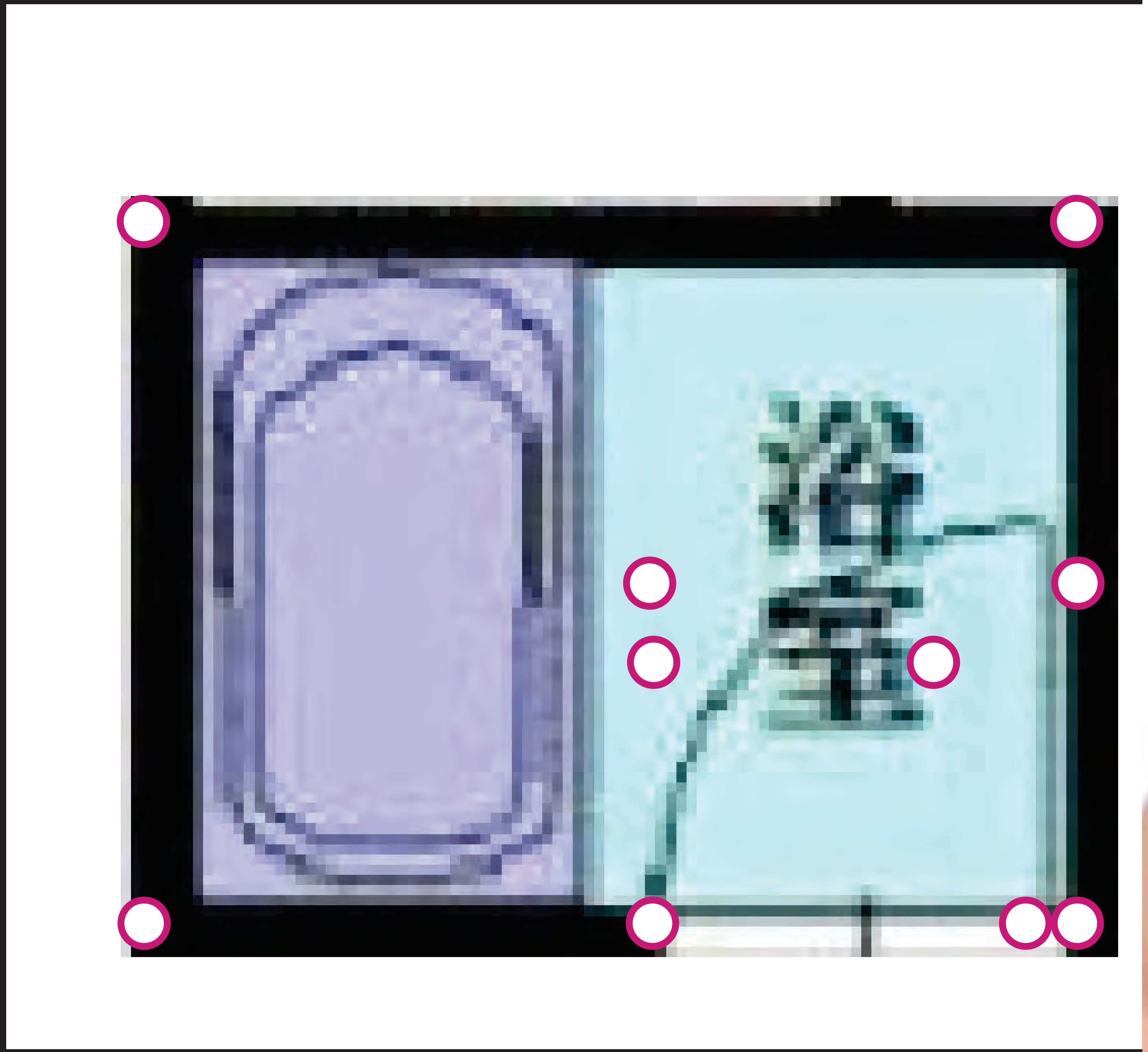
Corner candidates



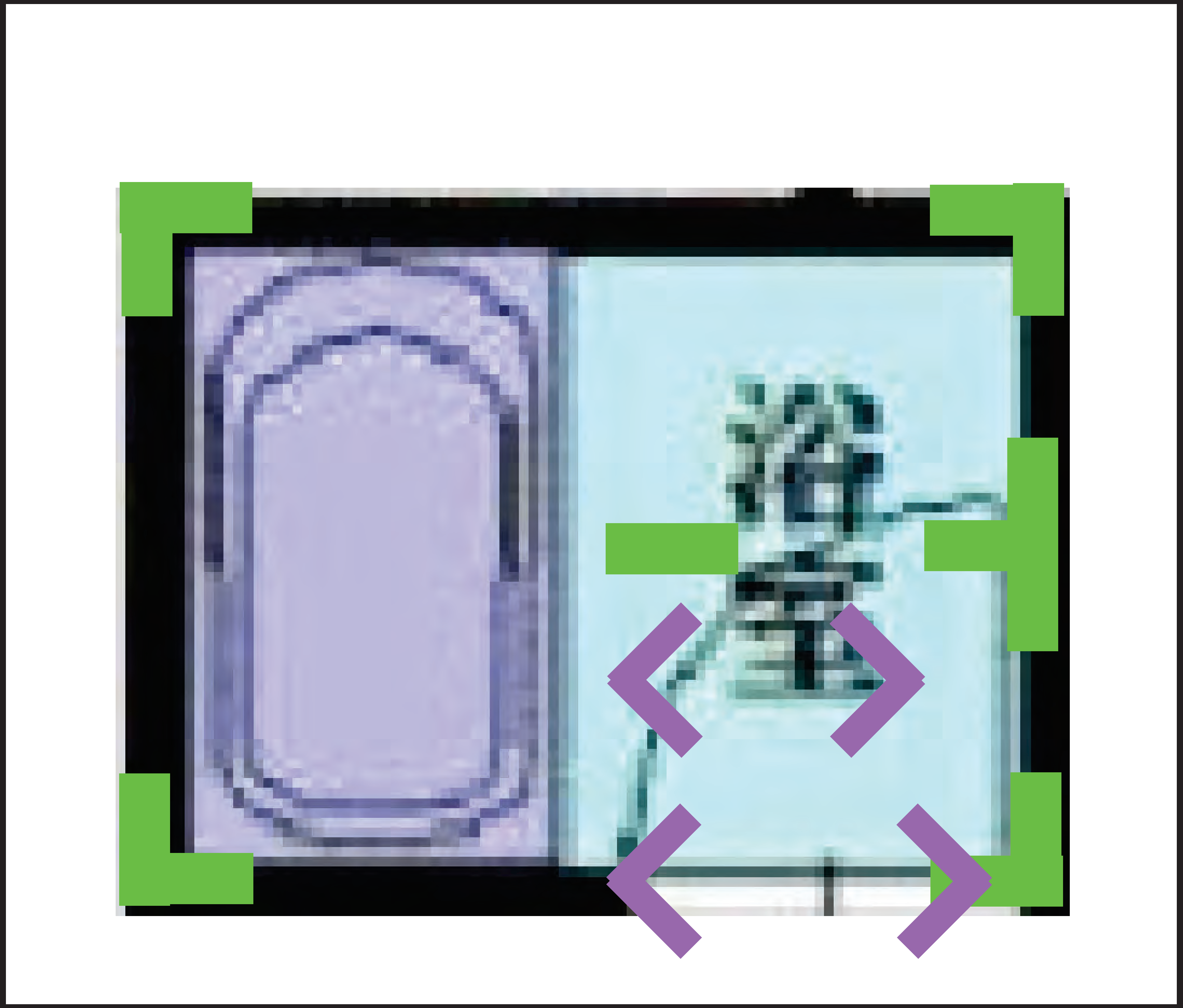
Corner candidates



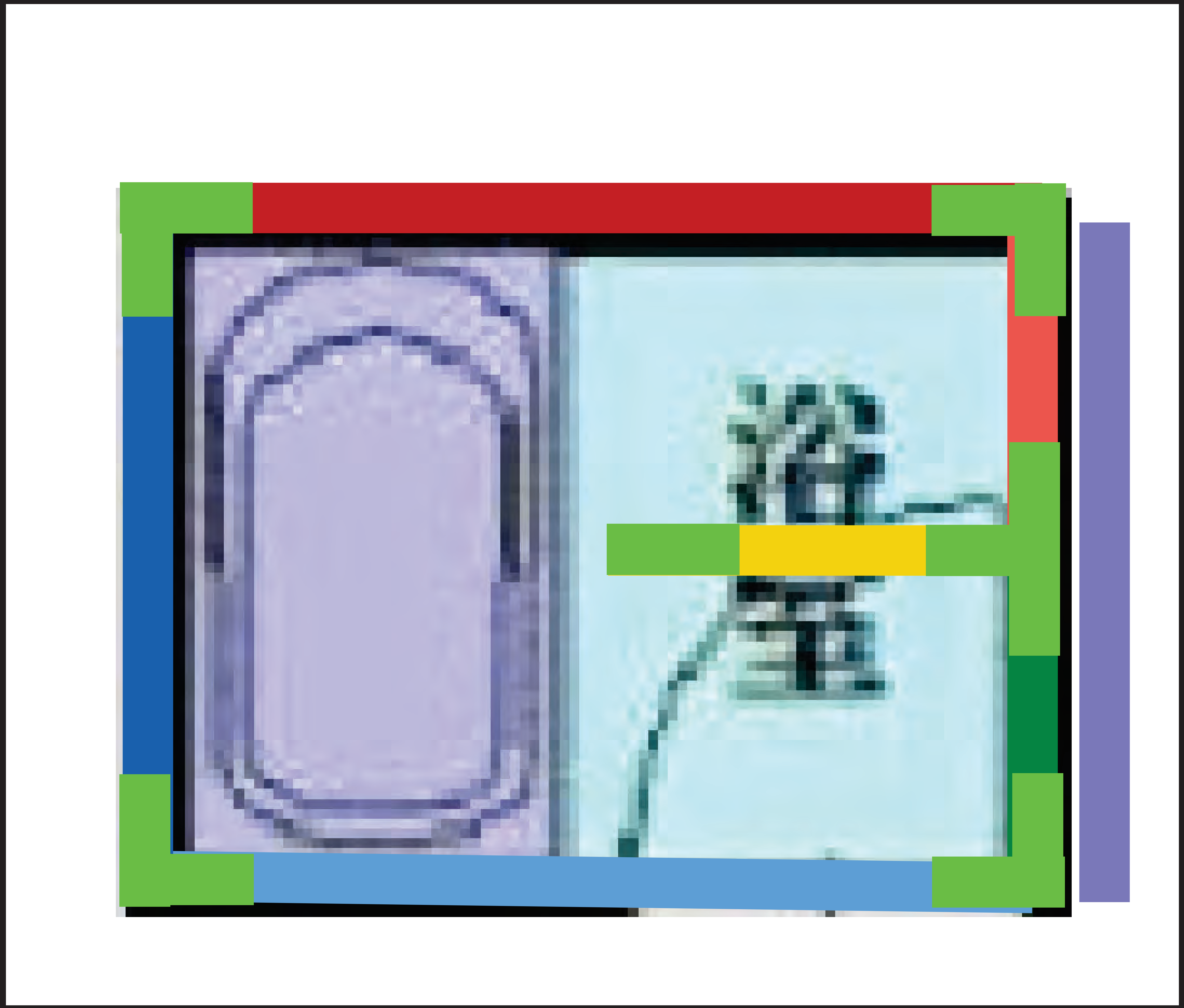
Corner ca



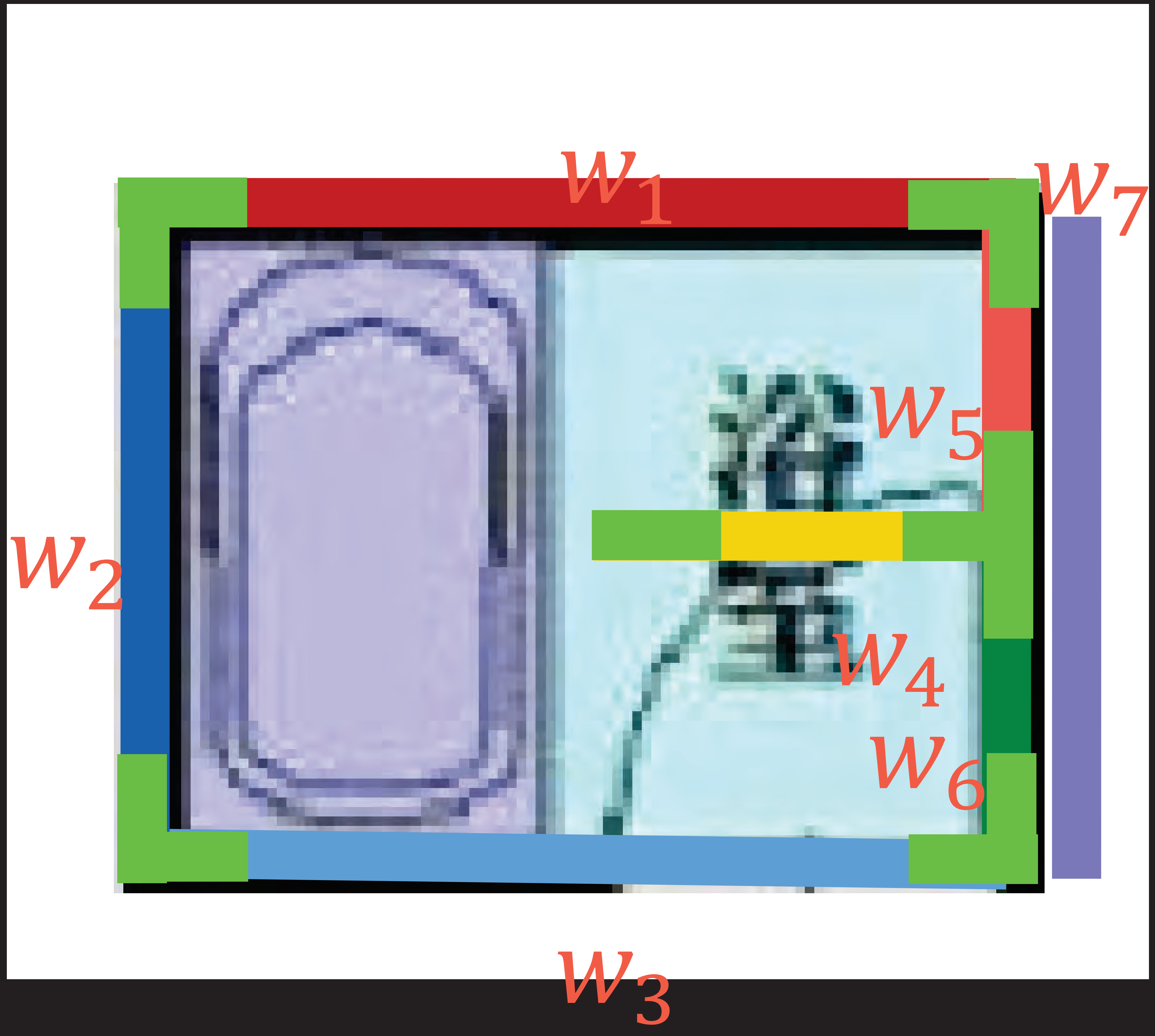
Corner candidates



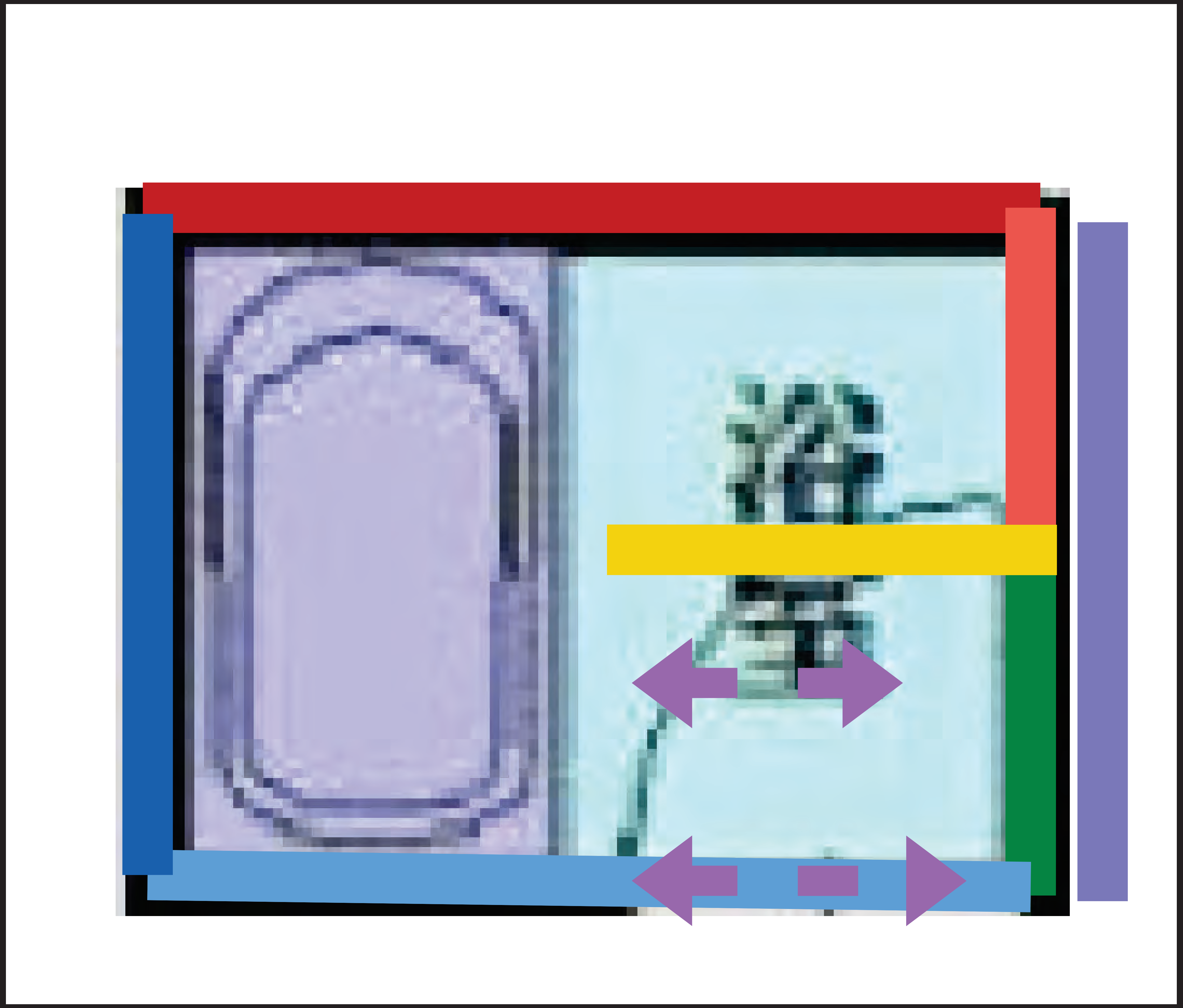
Wall candidates



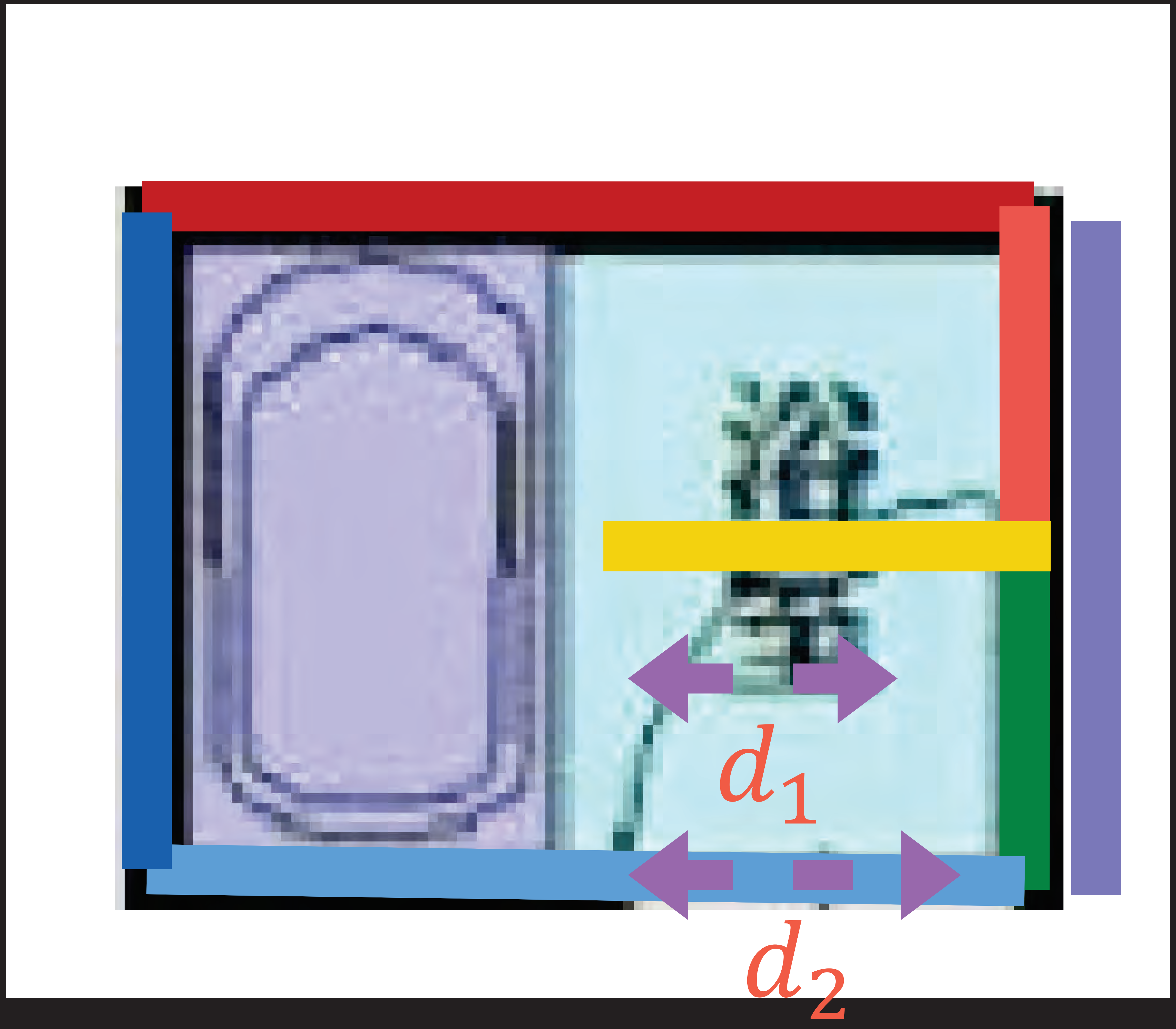
Wall candidates



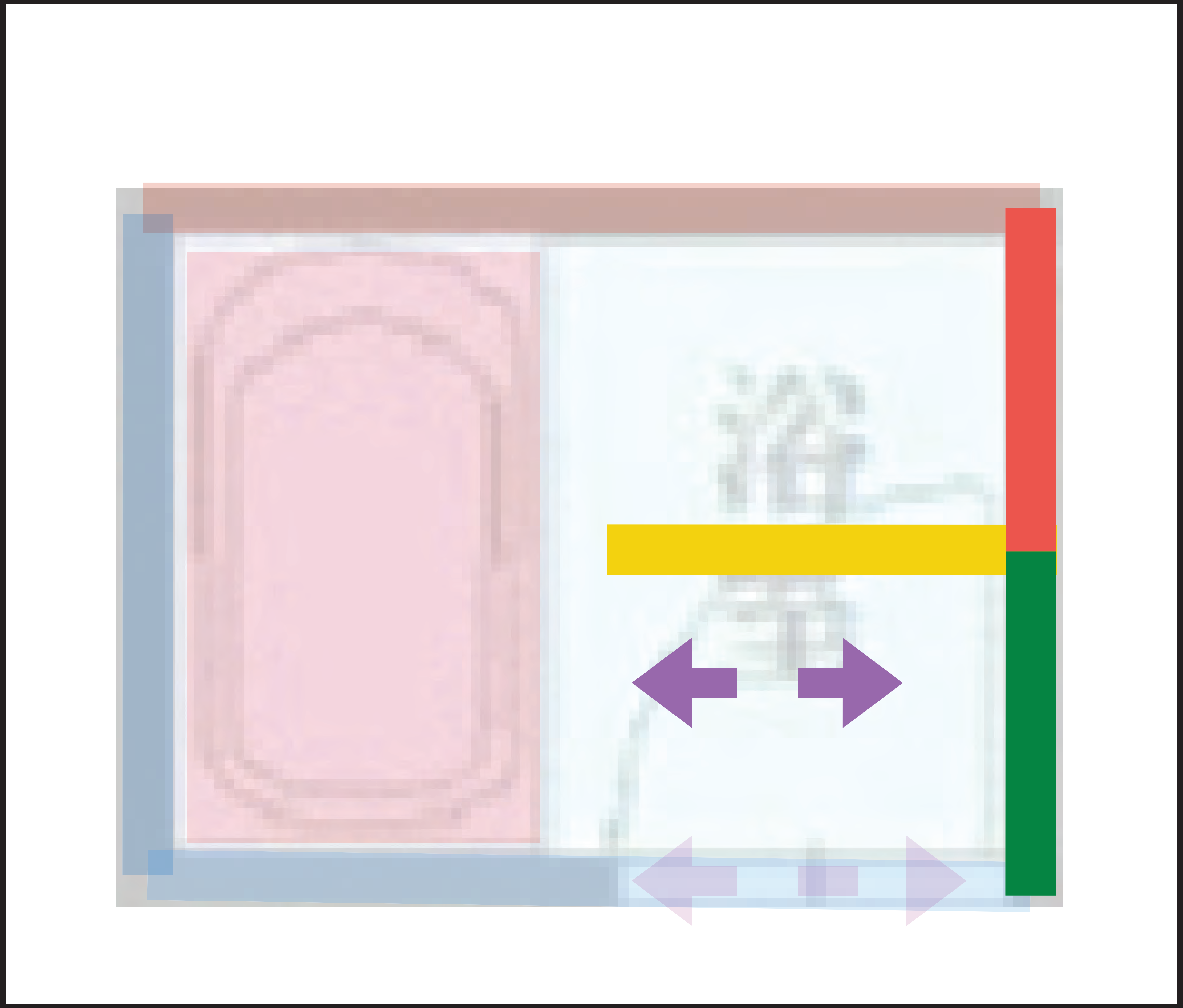
Door candidates



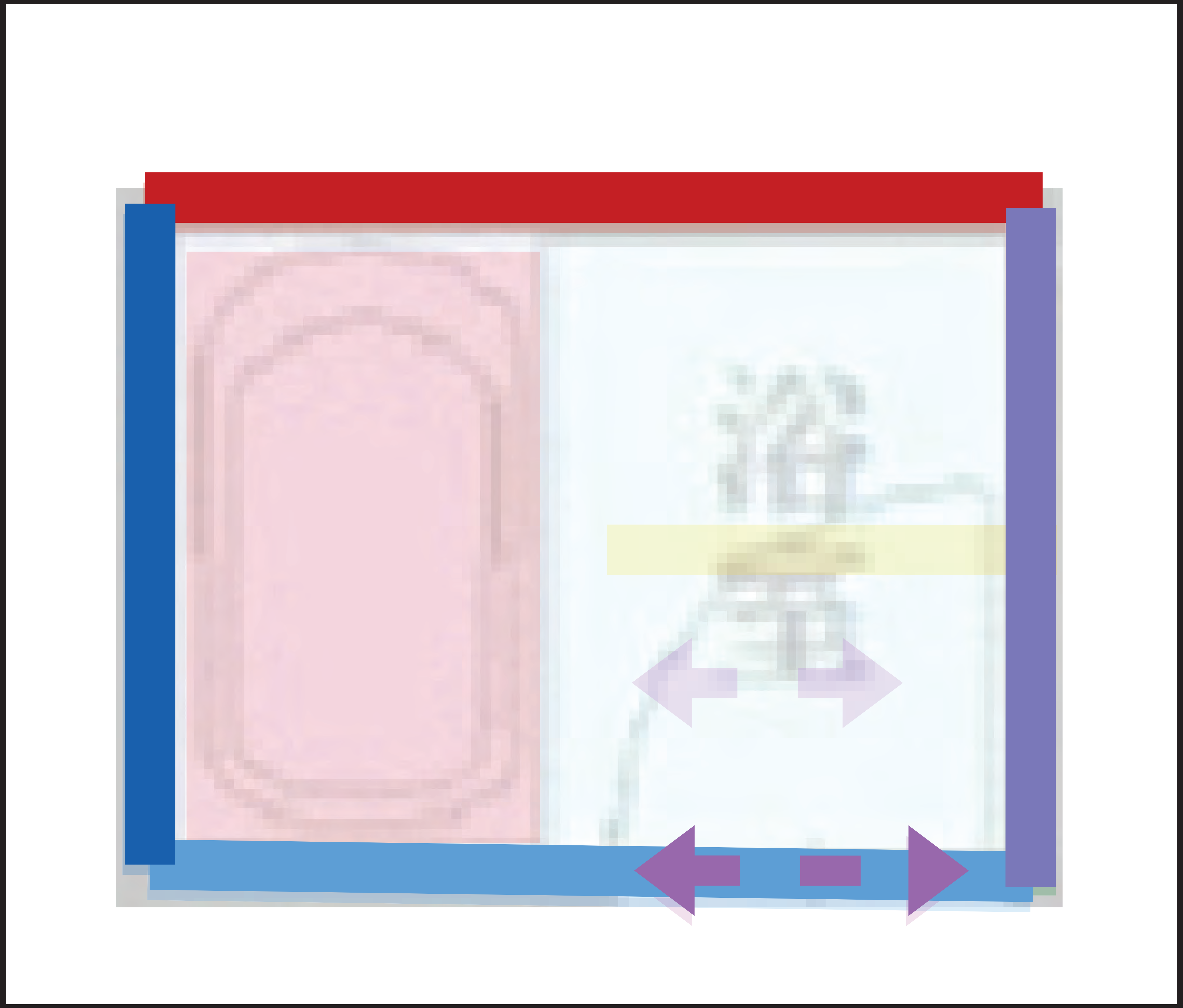
Door candidates



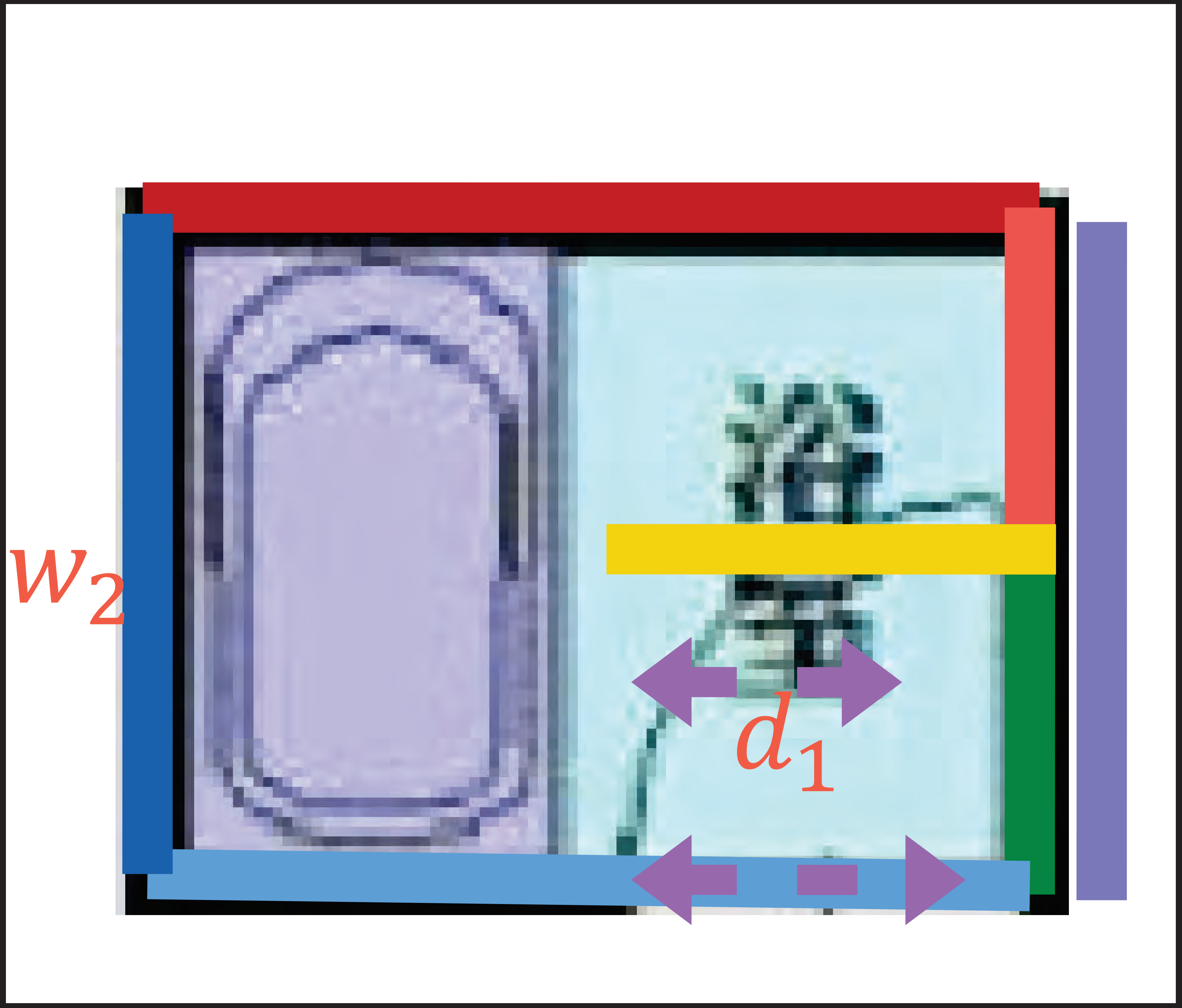
Incorrect primitive candidates



Correct primitive candidates



Optimization formulation

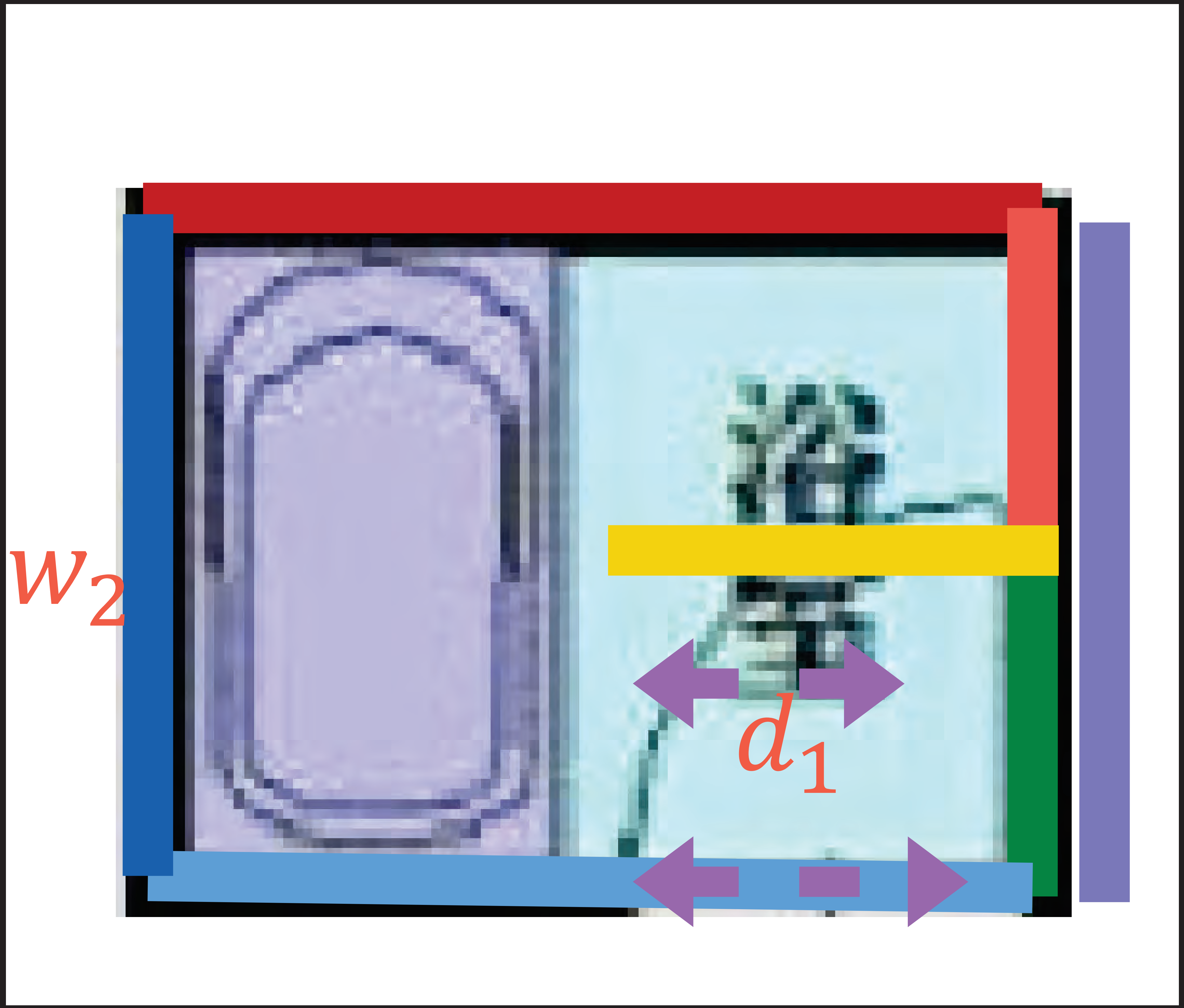


$$v \leftarrow \begin{cases} \text{1 if correct} \\ \text{0 if incorrect} \end{cases}$$

$$d_1 \leftarrow 0$$

$$w_2 \leftarrow 1$$

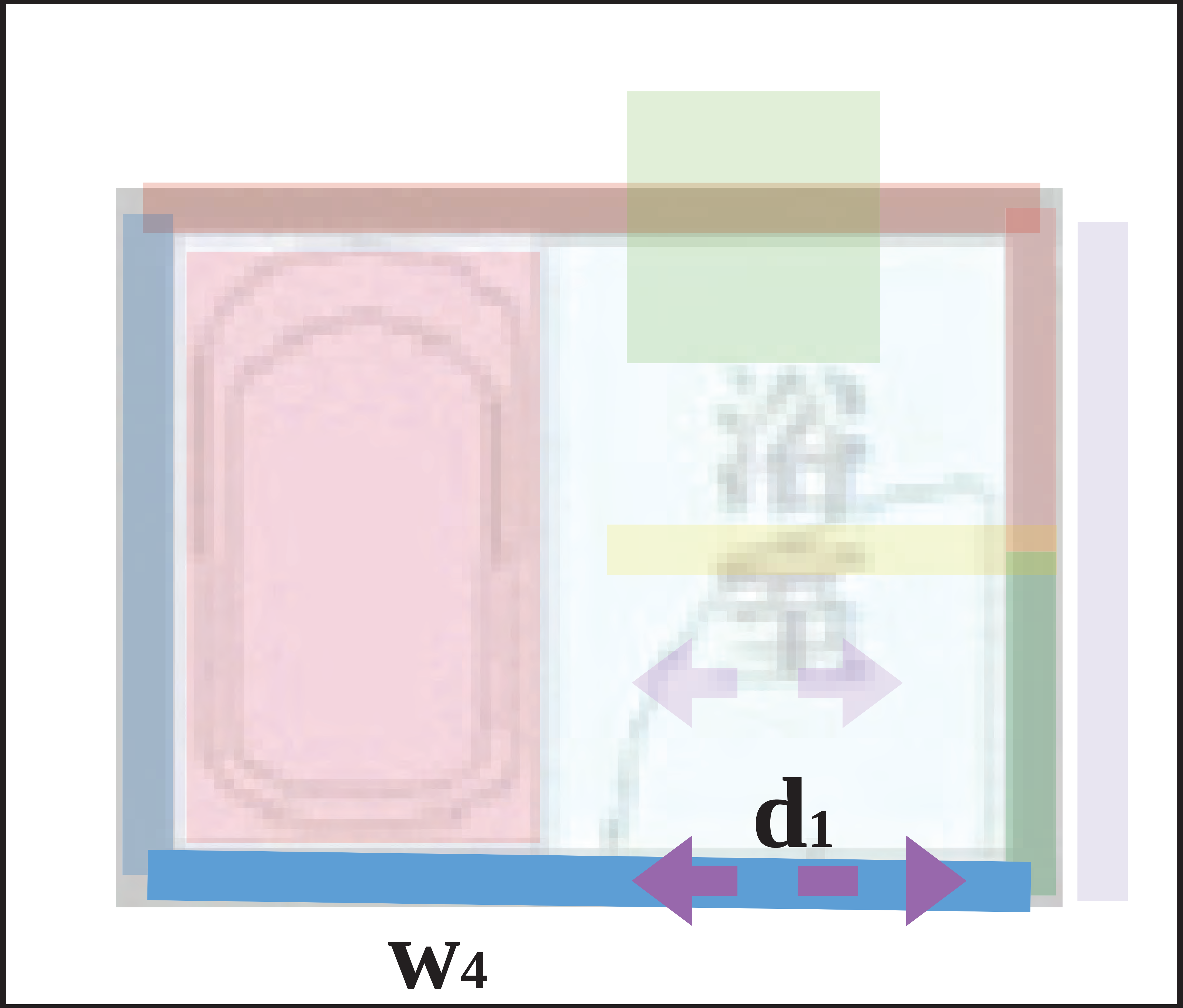
Optimization formulation



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

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

Door constraints



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

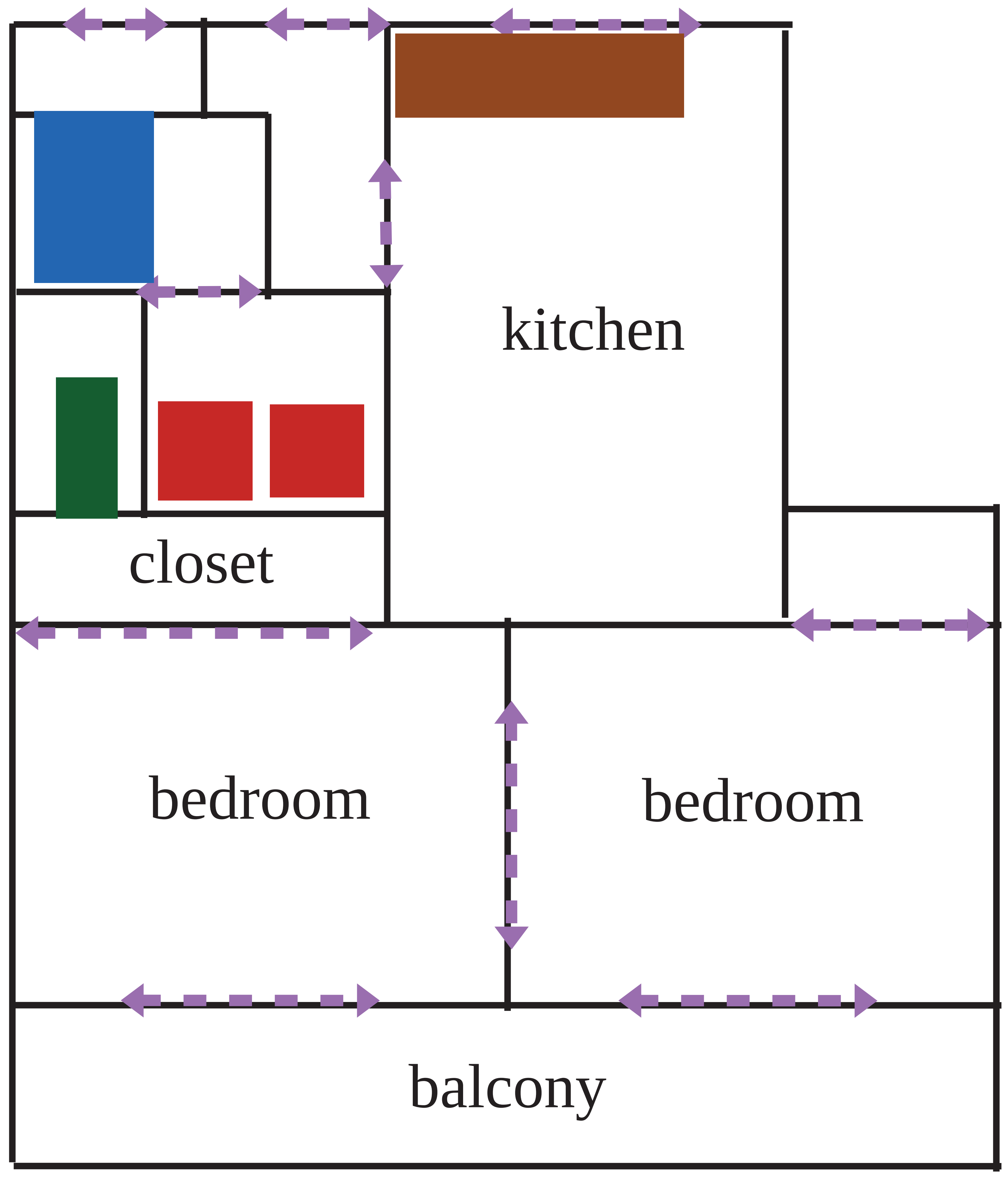
Subject to

$$d_1 \le w_4 (+w_? + w_? \cdots)$$

Loop constraints

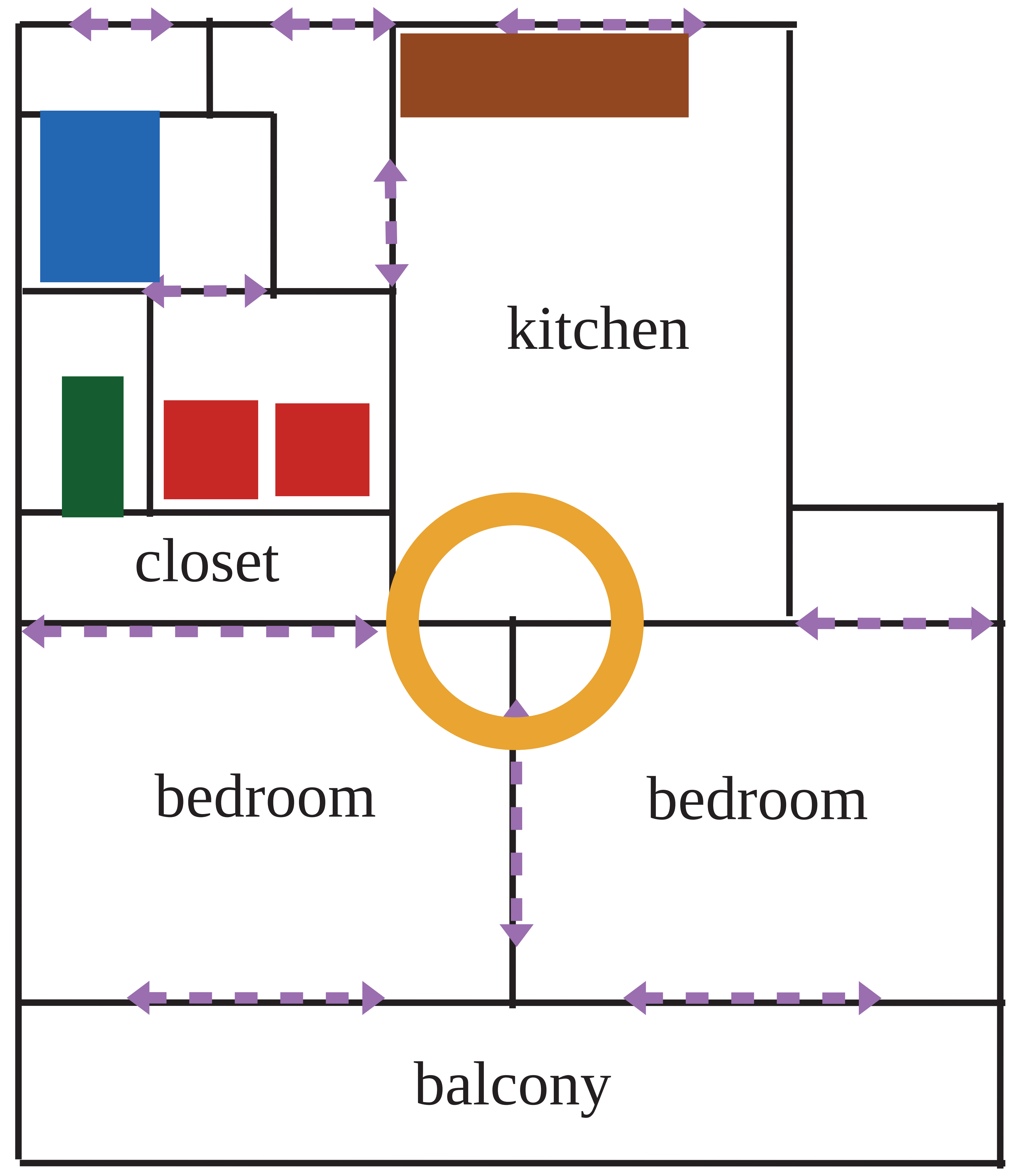
Loop constraints

Loop constraints

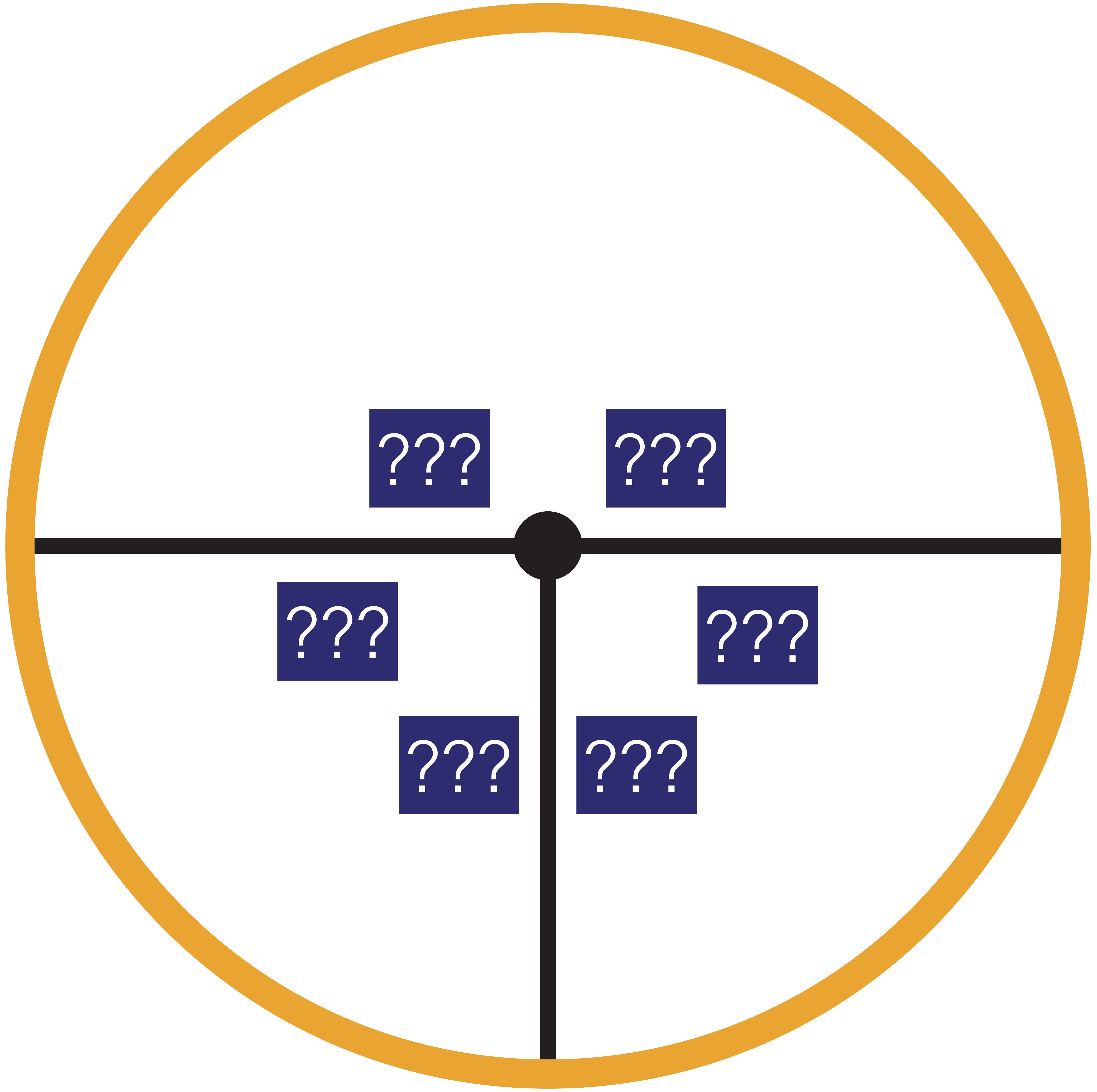


Vectorized floorplan

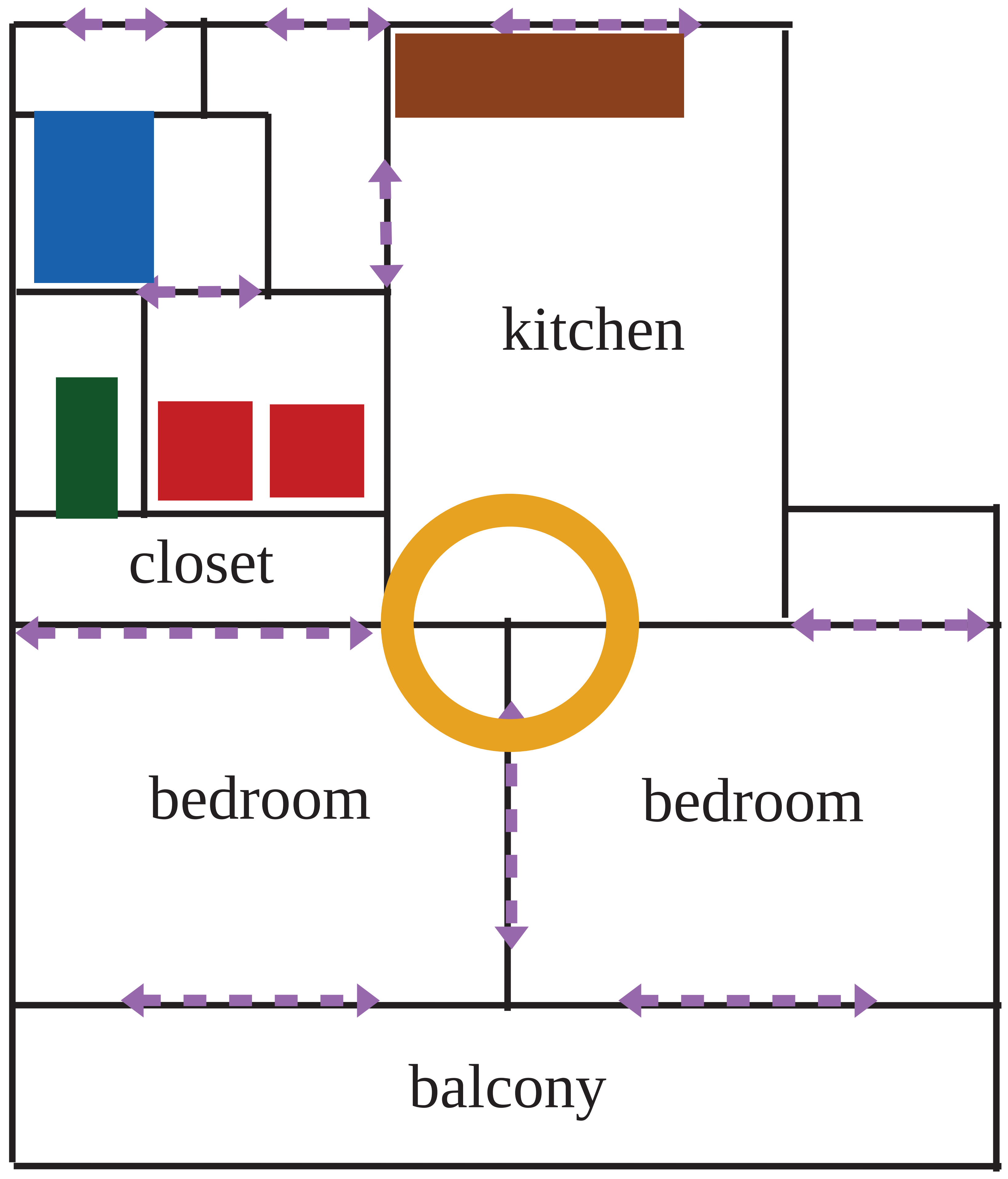
Loop constraints



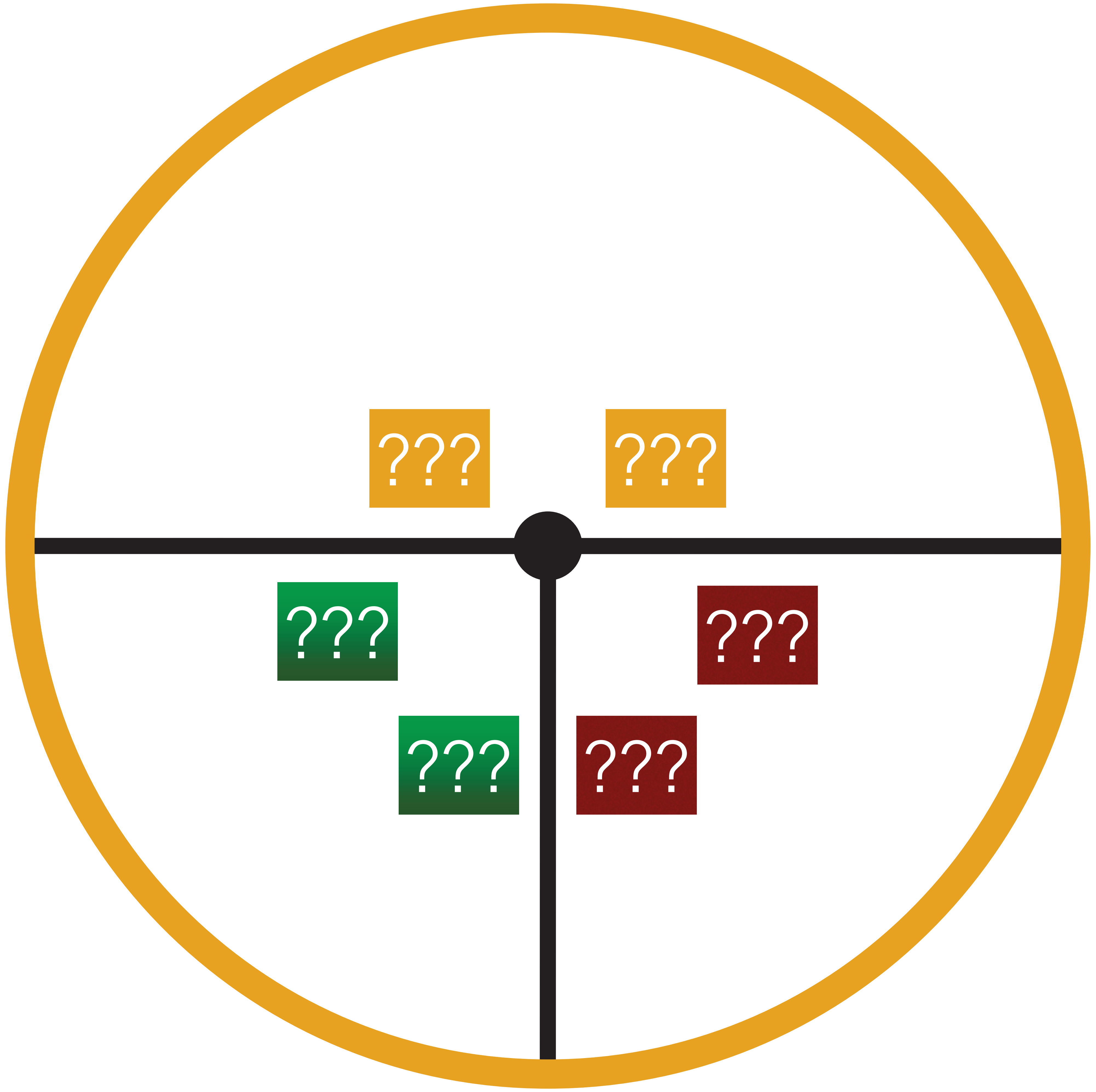
Vectorized floorplan



Loop constraints



Vectorized floorplan



Integer Programming

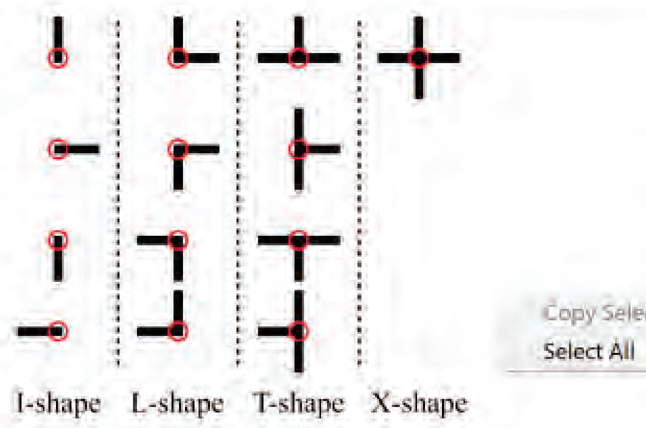


Figure 3: There are four wall junction types: I-, L-, T-, and X-shaped, depending on the degrees of incident wall segments. Considering orientations, we have in total 13 (= 4 + 4 + 4 + 1) types.

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°, 90°, 180°, and 270°. 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-classfies 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,
- $S^L_{wall}(p,s)$, $S^R_{wall}(p,s)$, $S_{icon}(p,s)$ 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, $S^L_{wall}(p,s)$ 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.

$$P_{wall}(p) = \sum_s S^L_{wall}(p,s) = \sum_s S^R_{wall}(p,s),$$

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

- 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 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.

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

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.

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 l ng

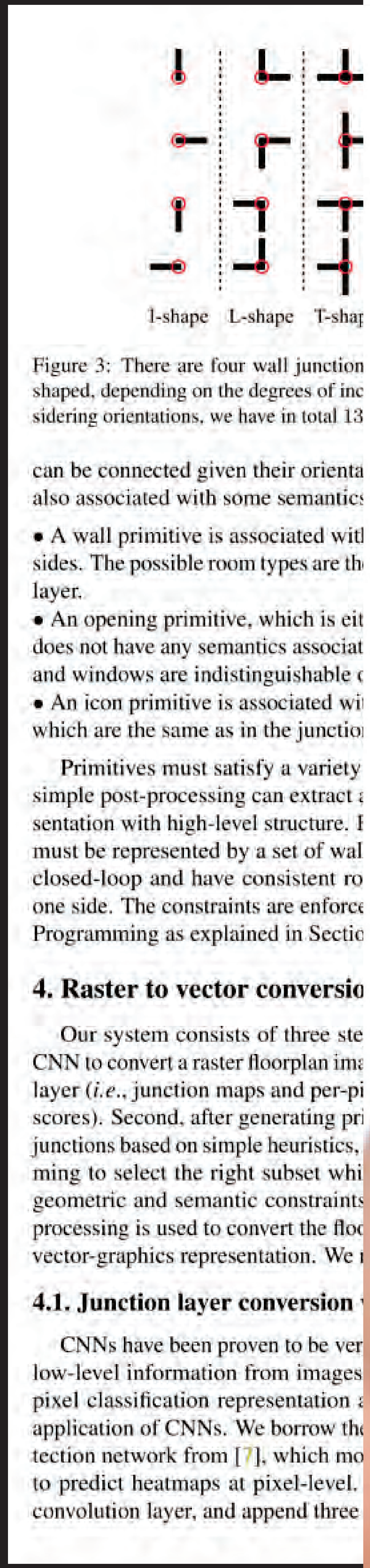

I-shape L-shape T-shap

Figure 3: There are four wall junction
shaped, depending on the degrees of inc
sidering orientations, we have in total 13

can be connected given their orienta
also associated with some semantics

• A wall primitive is associated wit
sides. The possible room types are th
layer.
• An opening primitive, which is eit
does not have any semantics associat
and windows are indistinguishable
• An icon primitive is associated wi
which are the same as in the junctio

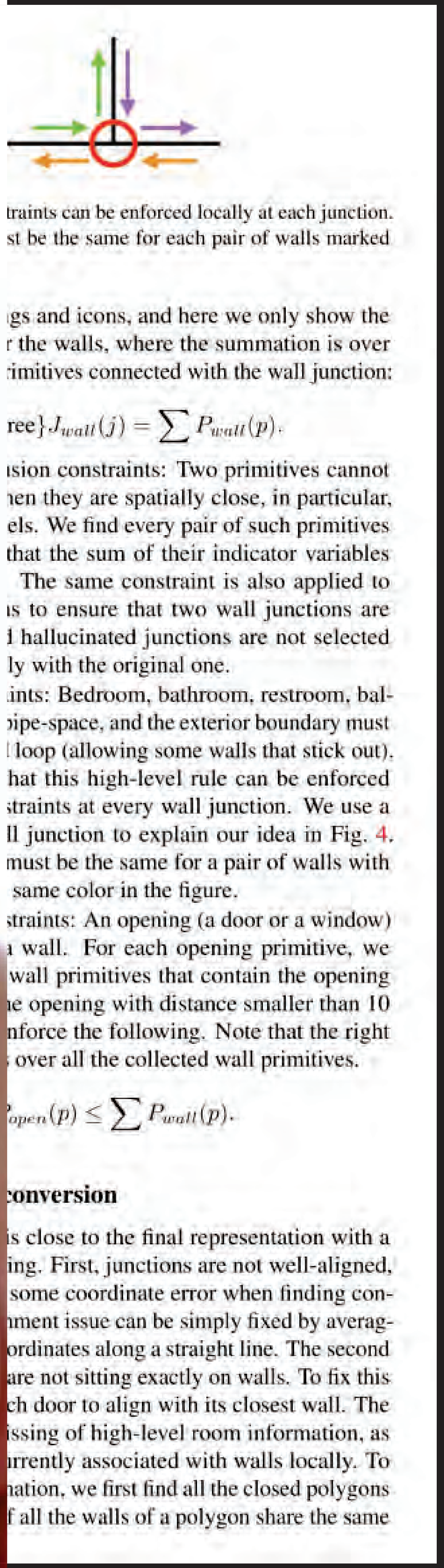
Primitives must satisfy a variety
simple post-processing can extract
sentation with high-level structure.
must be represented by a set of wal
closed-loop and have consistent ro
one side. The constraints are enforce
Programming as explained in Sectio

4. Raster to vector conversio

Our system consists of three ste
CNN to convert a raster floorplan ima
layer (*i.e.*, junction maps and per-pi
scores). Second, after generating pri
junctions based on simple heuristics,
ming to select the right subset whi
geometric and semantic constraints
processing is used to convert the floo
vector-graphics representation. We

4.1. Junction layer conversion

CNNs have been proven to be ver
low-level information from images
pixel classification representation
application of CNNs. We borrow th
tection network from [7], which mo
to predict heatmaps at pixel-level.
convolution layer, and append three


traints can be enforced locally at each junction.
st be the same for each pair of walls marked

gs and icons, and here we only show the
the walls, where the summation is over
imitives connected with the wall junction:

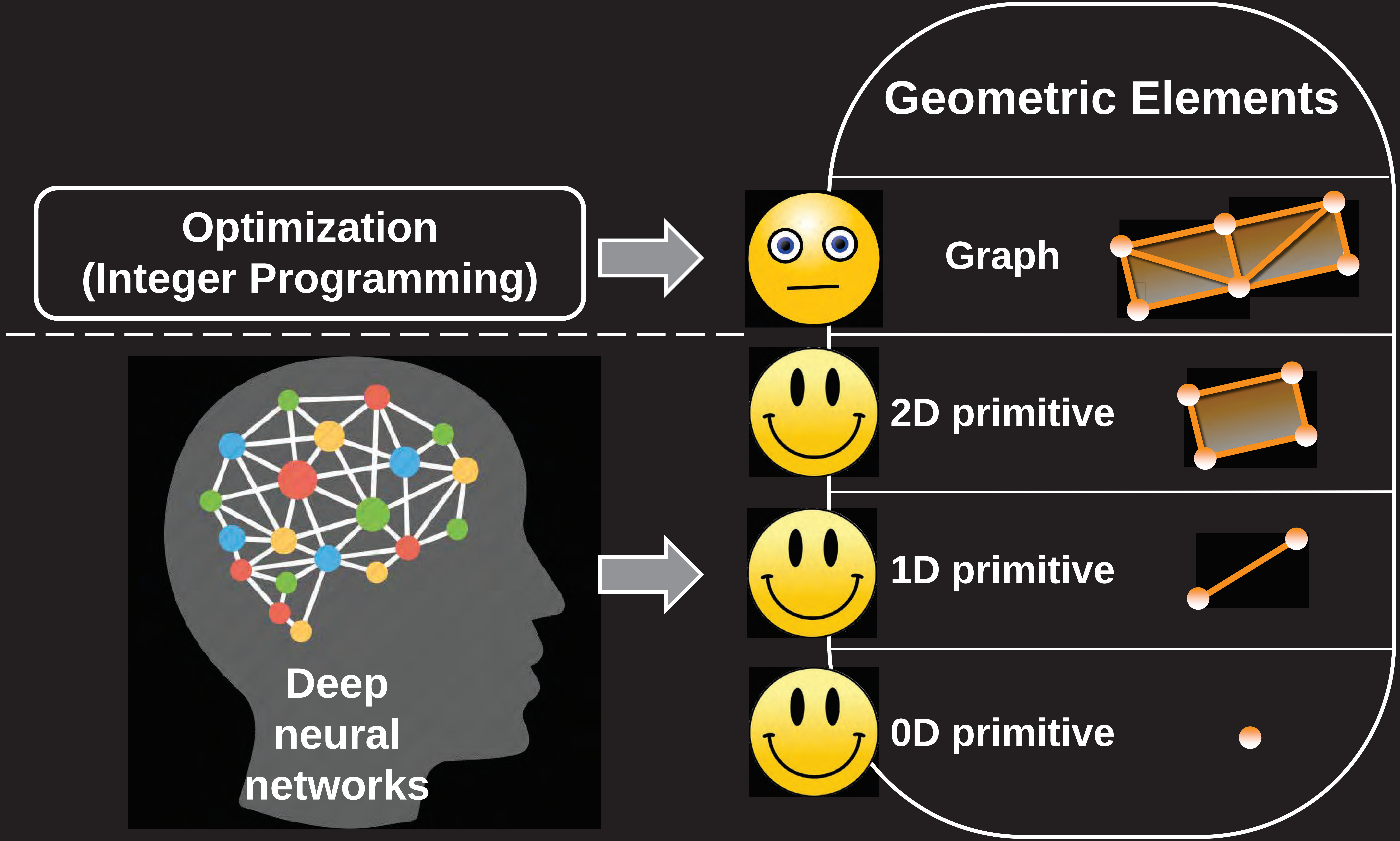
ree} $J_{wall}(j) = \sum P_{wall}(p)$.

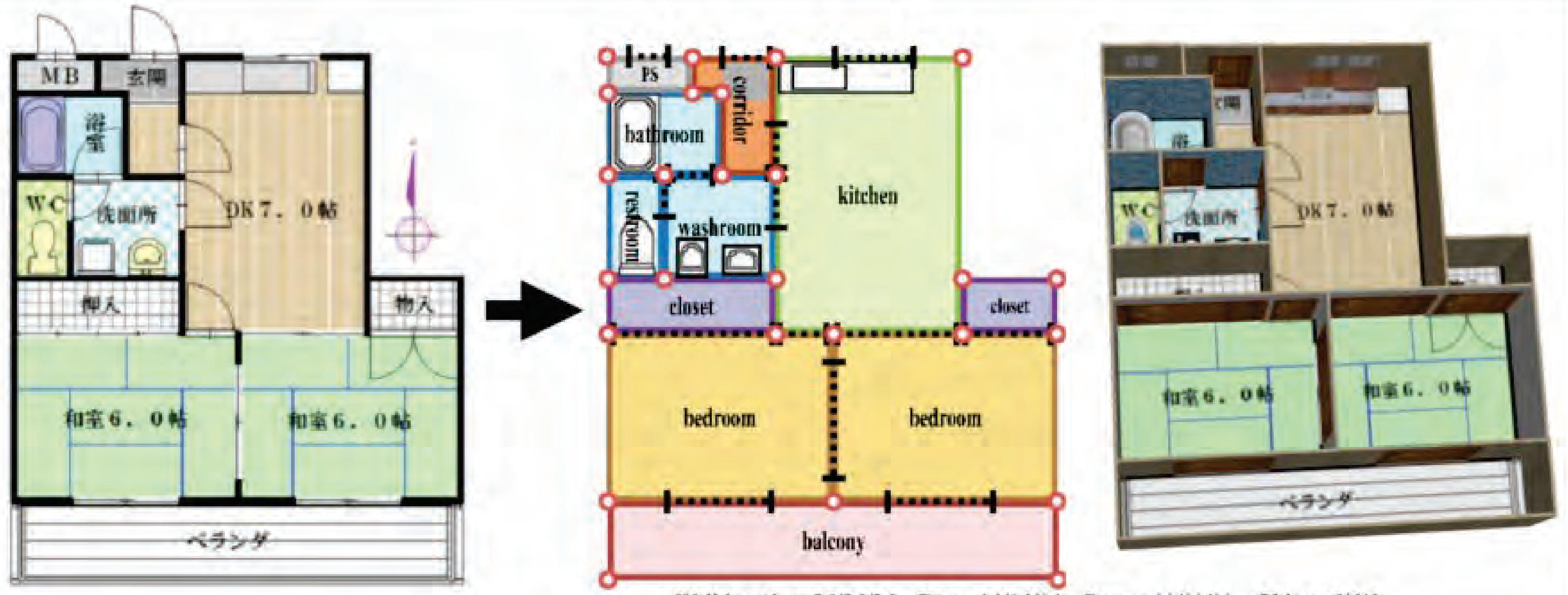
sion constraints: Two primitives cannot
en they are spatially close, in particular,
els. We find every pair of such primitives
that the sum of their indicator variables
The same constraint is also applied to
s to ensure that two wall junctions are
hallucinated junctions are not selected
ly with the original one.
ints: Bedroom, bathroom, restroom, bal-
pipe-space, and the exterior boundary must
loop (allowing some walls that stick out).
hat this high-level rule can be enforced
straints at every wall junction. We use a
ll junction to explain our idea in Fig. 4.
must be the same for a pair of walls with
same color in the figure.
straints: An opening (a door or a window)
wall. For each opening primitive, we
wall primitives that contain the opening
e opening with distance smaller than 10
nforce the following. Note that the right
over all the collected wall primitives.

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

conversion

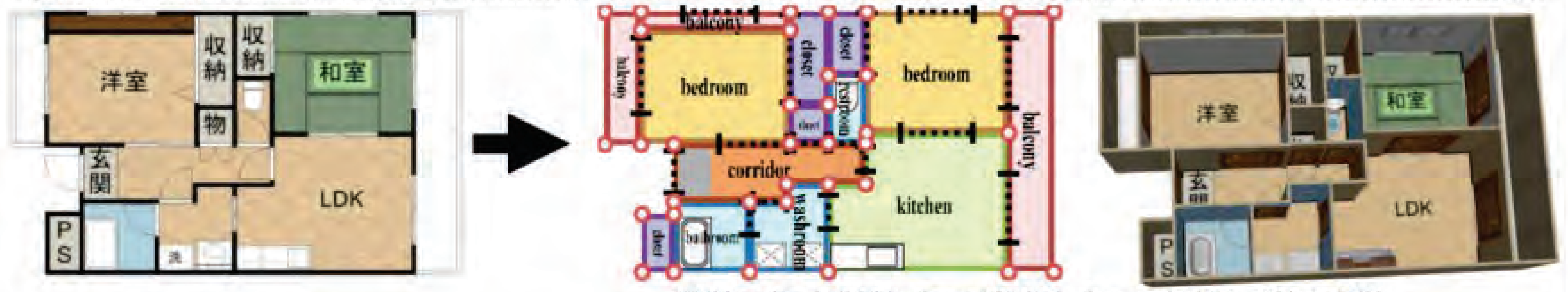
s close to the final representation with a
ing. First, junctions are not well-aligned,
some coordinate error when finding con-
nment issue can be simply fixed by averag-
ordinates along a straight line. The second
are not sitting exactly on walls. To fix this
ch door to align with its closest wall. The
issing of high-level room information, as
rrently associated with walls locally. To
ation, we first find all the closed polygons
f all the walls of a polygon share the same


Optimization
(Integer Programming)
Deep
neural
networks
Geometric Elements
Graph
2D primitive
1D primitive
0D primitive

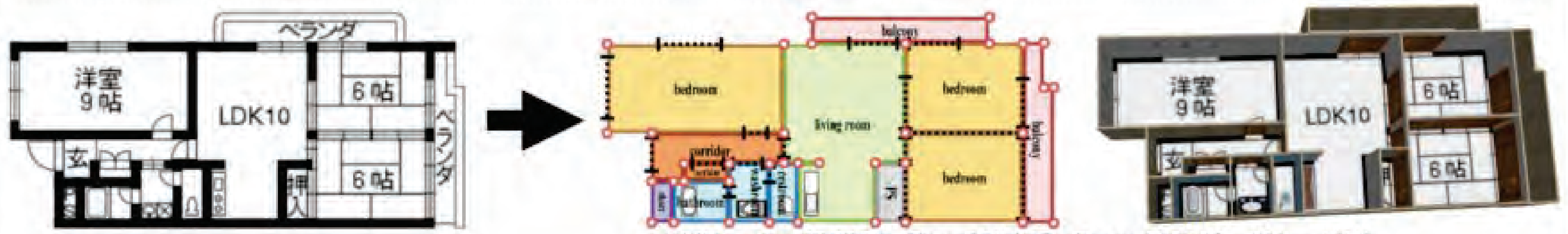


Wall junction: 26/26/26 Door: 14/14/14 Room: 11/11/11 Object: 6/6/6

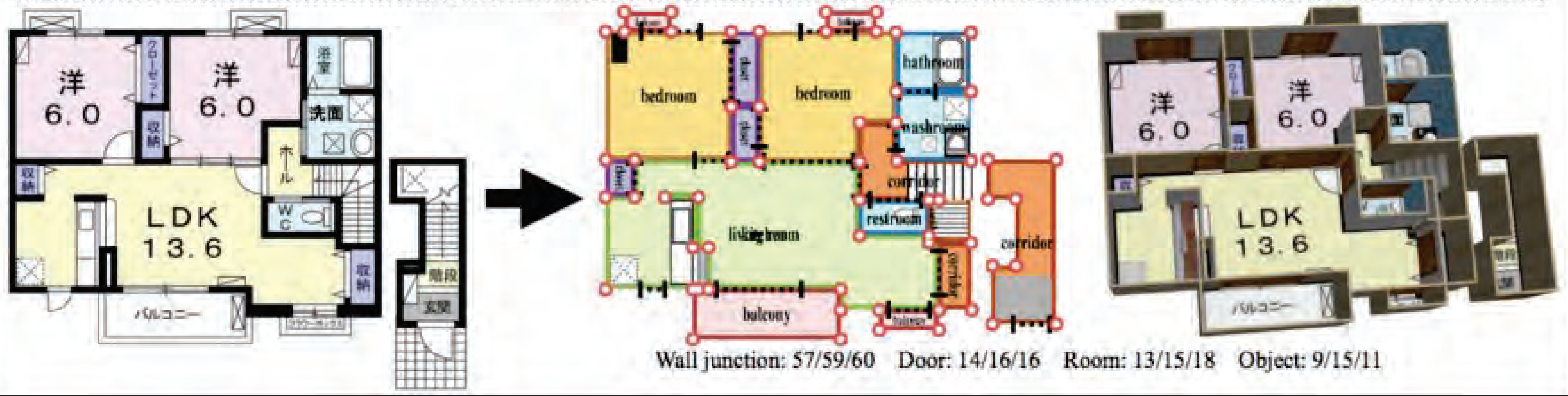
Wall junction: 26/28/26 Door: 16/17/16 Room: 10/10/10 Object: 7/7/7



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



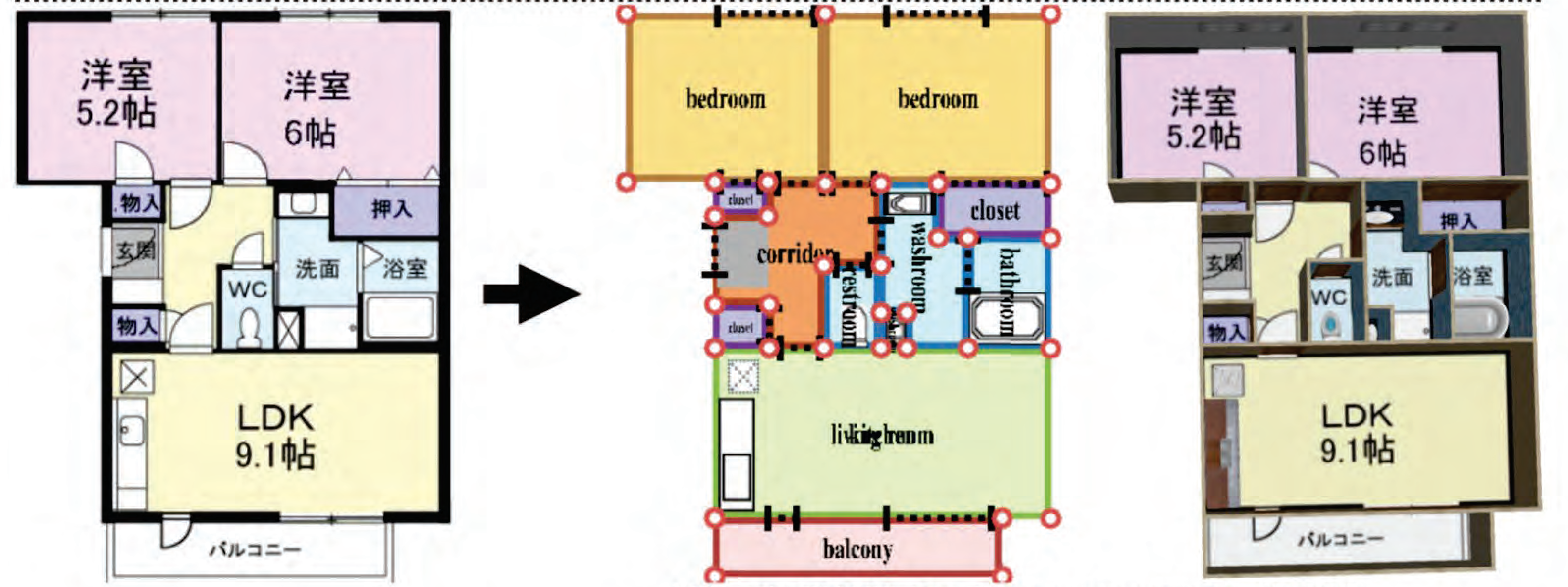
Wall junction: 37/37/37 Door: 15/16/17 Room: 12/13/13 Object: 3/6/4



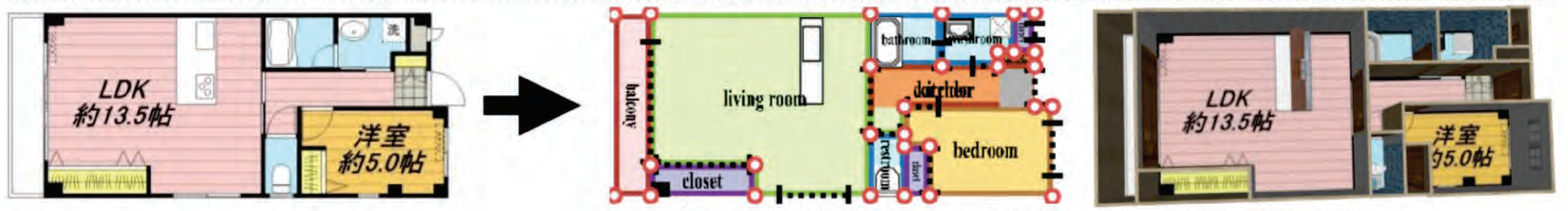
Wall junction: 57/59/60 Door: 14/16/16 Room: 13/15/18 Object: 9/15/11

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

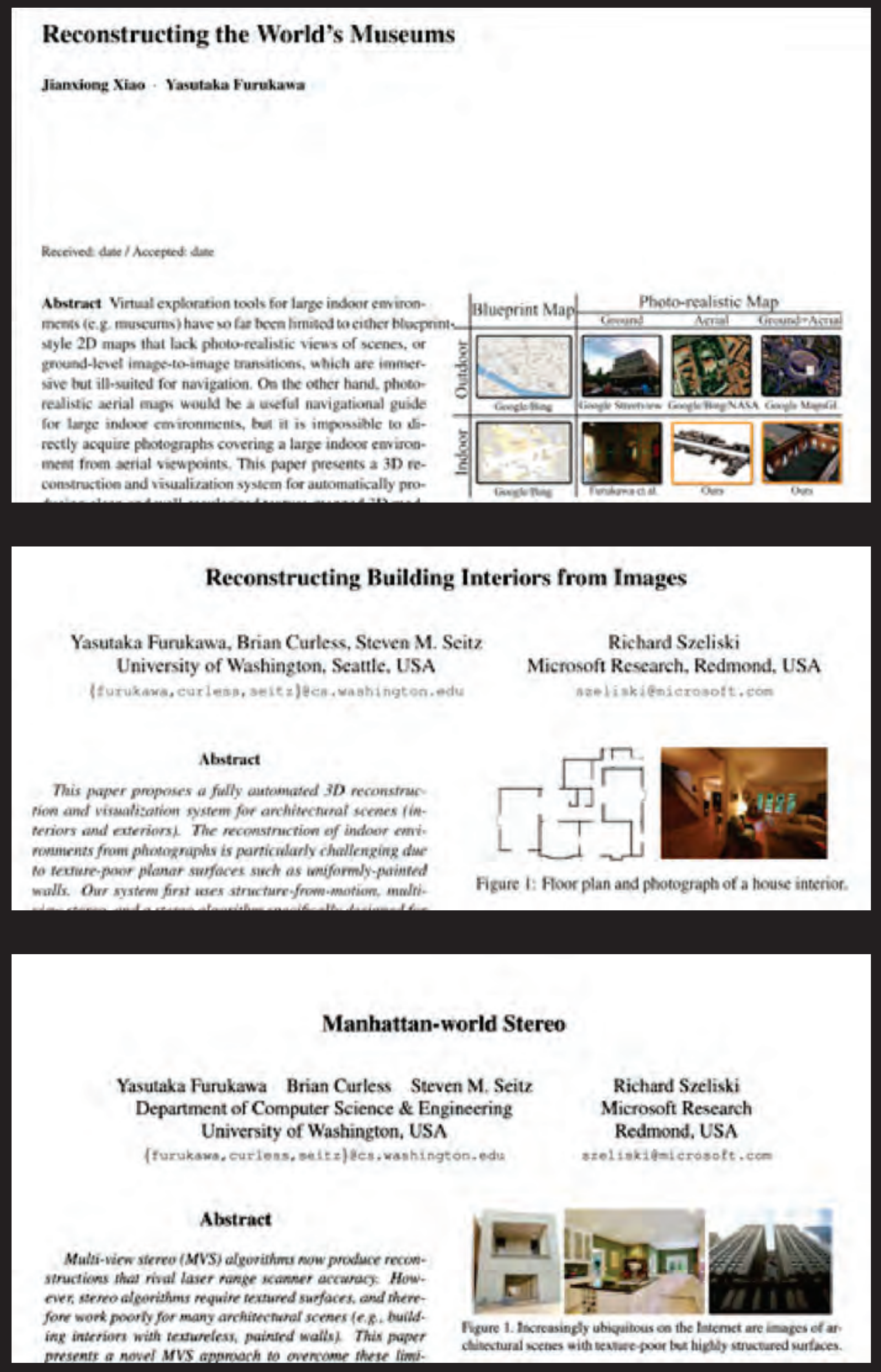

Reception Room / Dining Room
Kitchen
Bedroom
bedroom
kitchen
restroom
corridor
bedroom
balcony
bedroom
bedroom
bathroom
corridor
living room
bathroom
BALCONY
RECEPTION ROOM
BEDROOM
KITCHEN
THIRD FLOOR
BATHROOM
balcony
living room
bedroom
kitchen
kitchen
washroom
RECEPTION ROOM / OPEN PLAN KITCHEN
BEDROOM
kitchen
bathroom
corridor
bedroom

bedroom
corridor
bathroom
closet
bedroom
kitchen
balcony
Wall junction: 29/30/30 Door: 15/15/15 Room: 10/11/11 Object: 6/6/6
洋室8帖
K2帖
bedroom
closet
bathroom
Wall junction: 18/18/18 Door: 8/8/8 Room: 7/7/7 Object: 5/6/6
浴室
玄関
K
洋
12.3
バルコニー
corridor
closet
bedroom
balcony
Wall junction: 30/34/32 Door: 7/7/7 Room: 7/9/10 Object: 8/10/9
ロフト
4.5帖
バルコニー
洋室
7.5帖
浴室
玄
押
bedroom
bathroom
closet
bedroom
kitchen
closet
Wall junction: 16/18/18 Door: 6/6/6 Room: 6/7/7 Object: 4/4/4

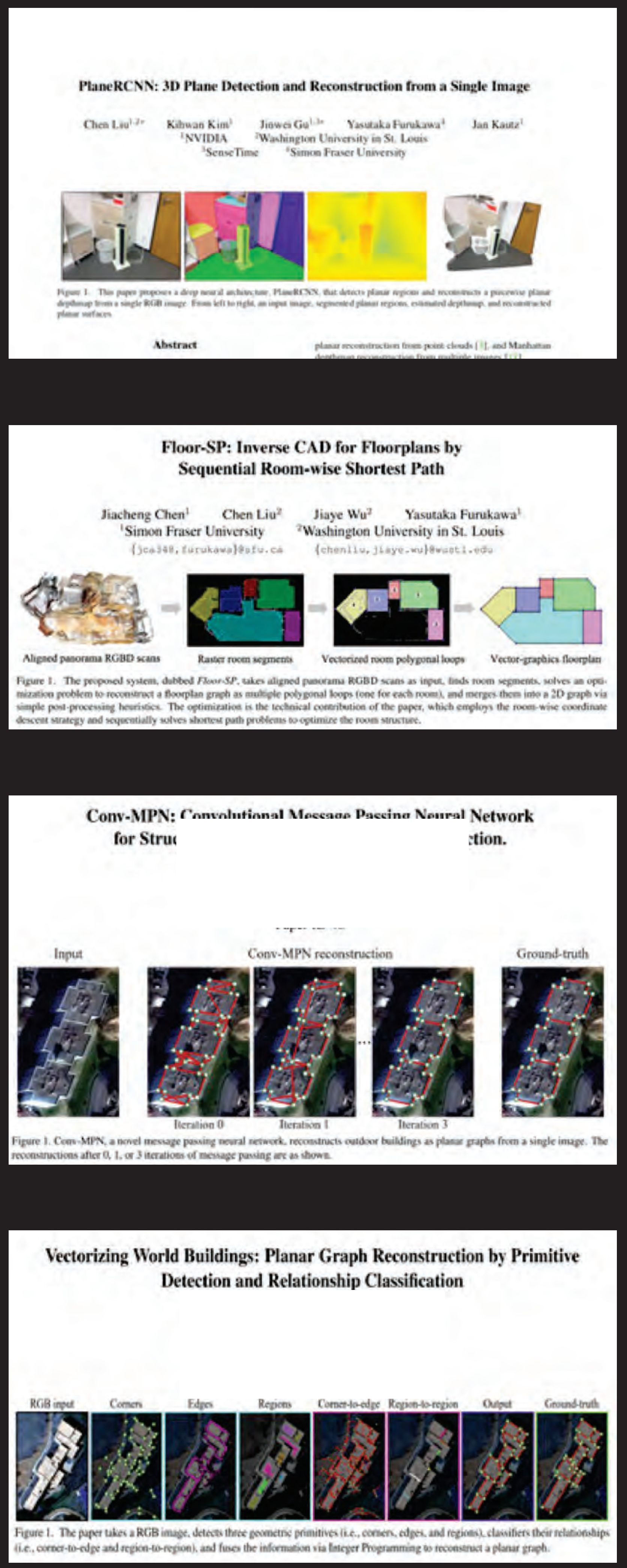
Quantitative evaluations

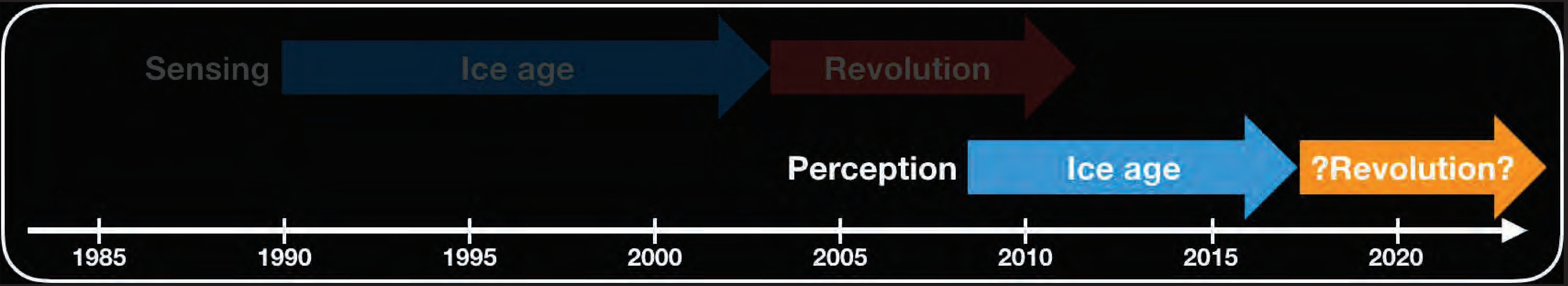
Method	Wall Junction		Opening		Icon		Room	
	Acc.	Recall	Acc.	Recall	Acc.	Recall	Acc.	Recall
Ahmed *et al.* [6]	74.9	57.5	61.3	48.7	N/A	N/A	N/A	N/A
Ours (without IP)	70.7	95.1	67.9	91.4	22.3	77.4	80.9	78.5
Ours (without mutual exclusion constraints)	92.8	91.7	68.5	91.1	22.0	76.2	82.8	87.5
Ours (without 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.

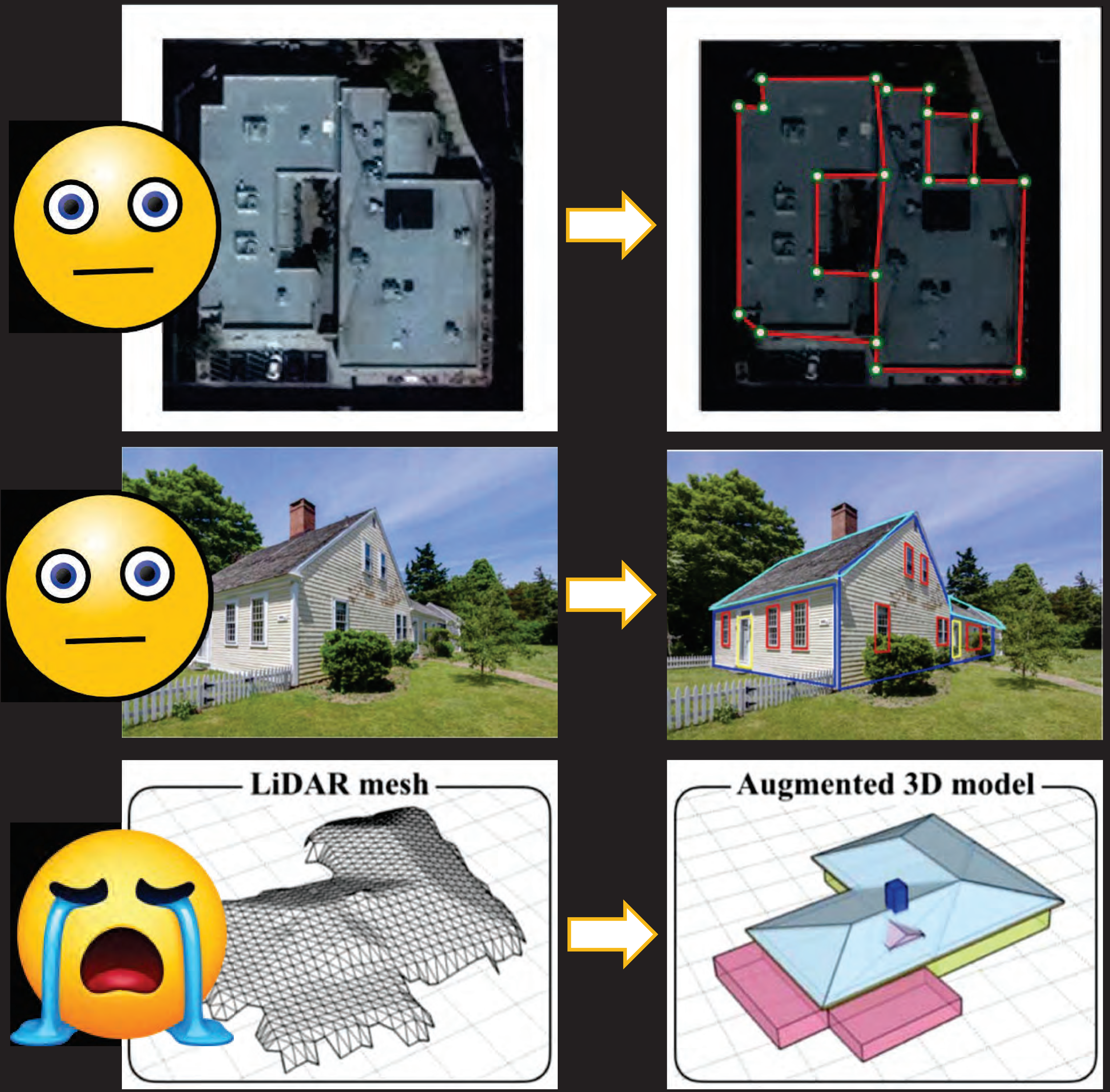


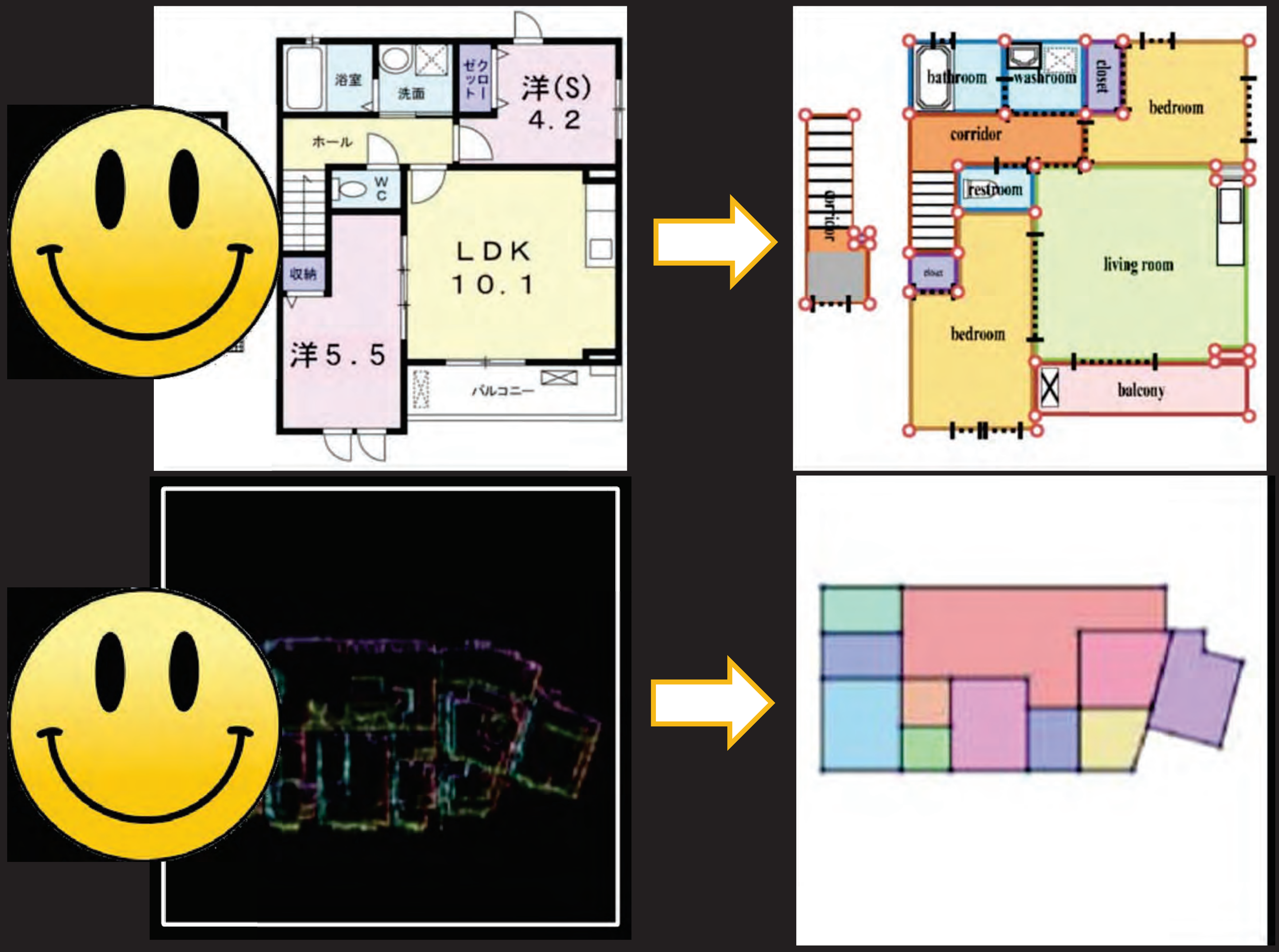
CVPR-ICCV-ECCV papers for geometry perception…



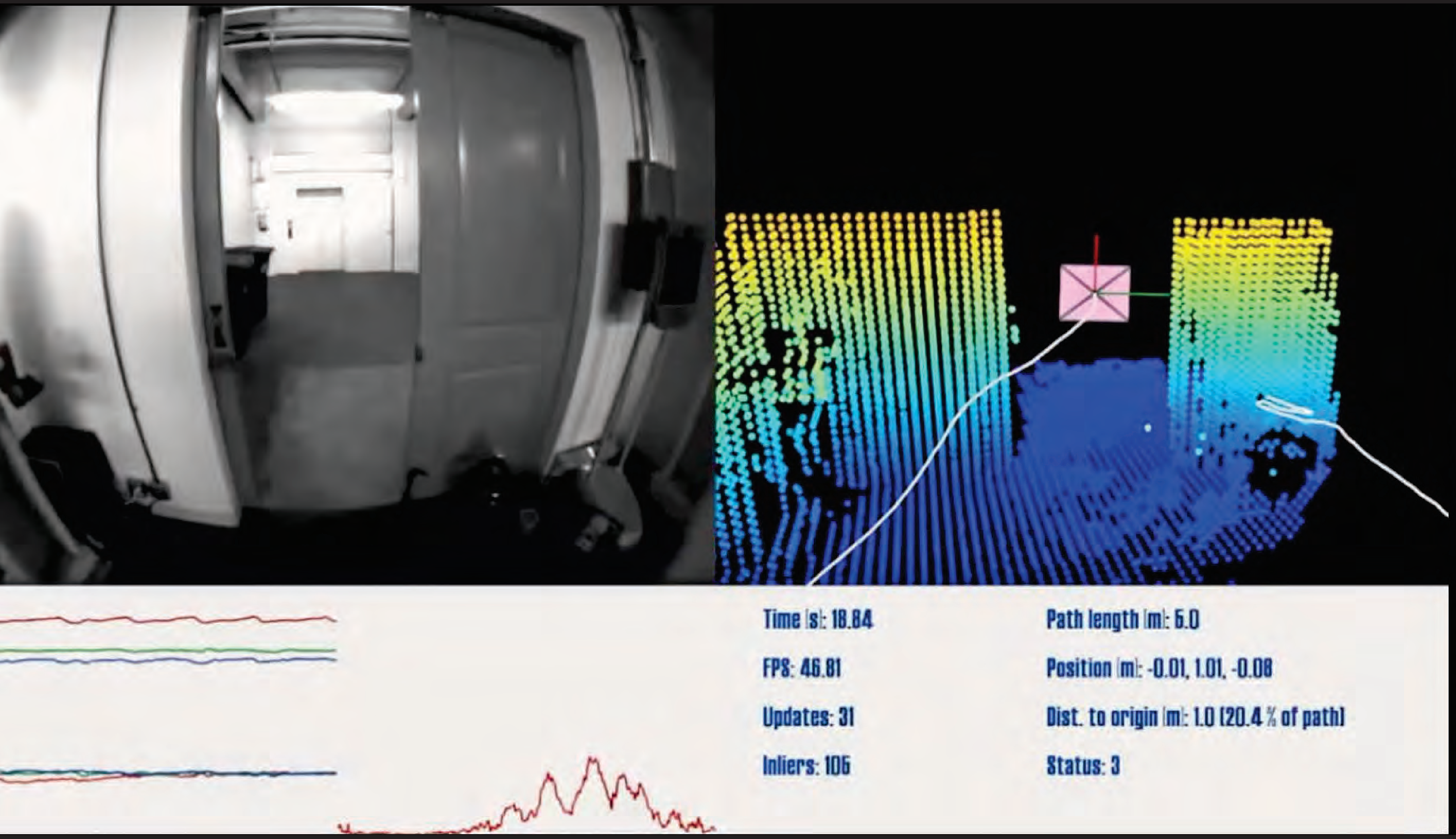


Ice-age or revolution?





Sensing

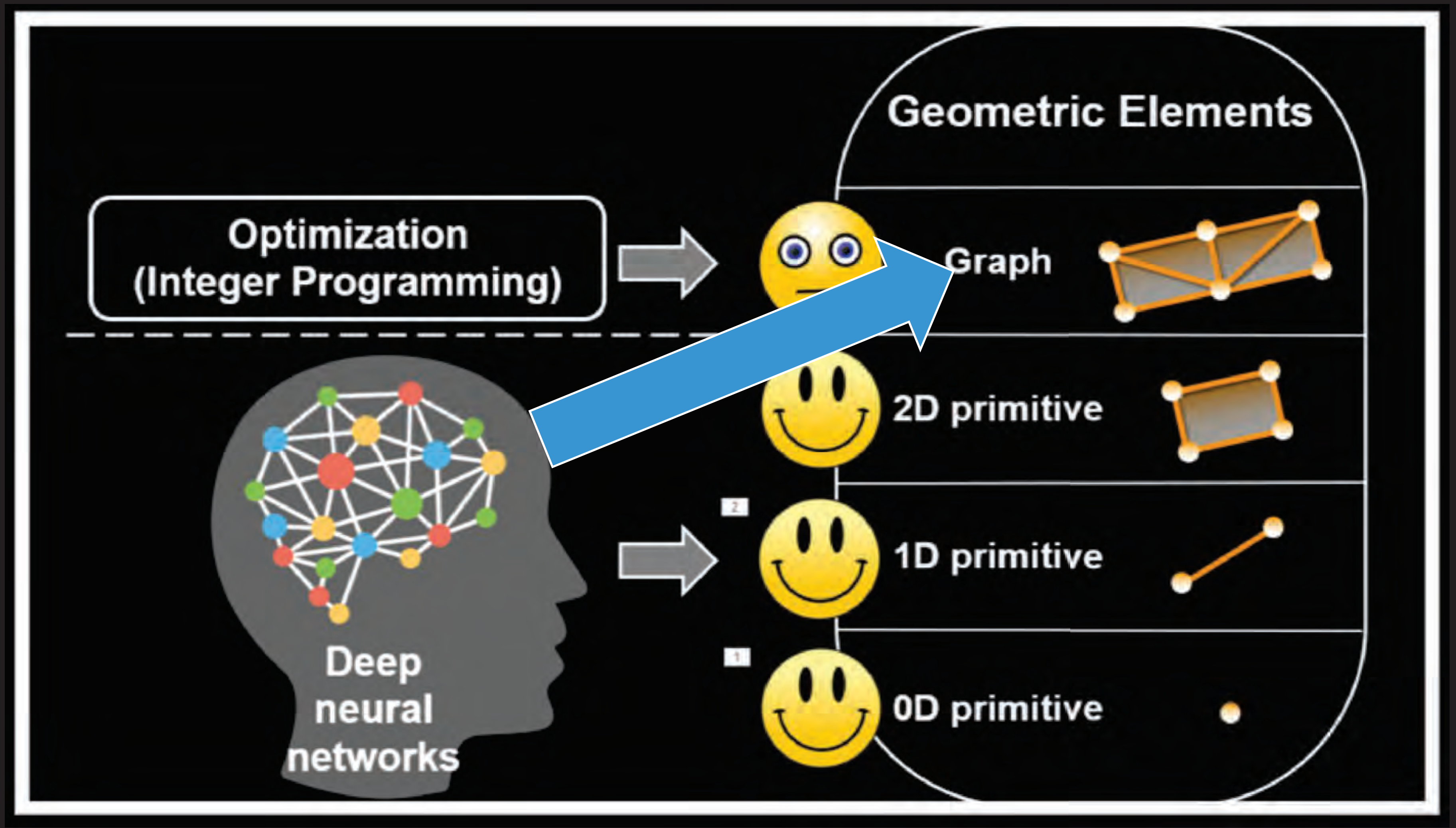


[Google Project Tango]

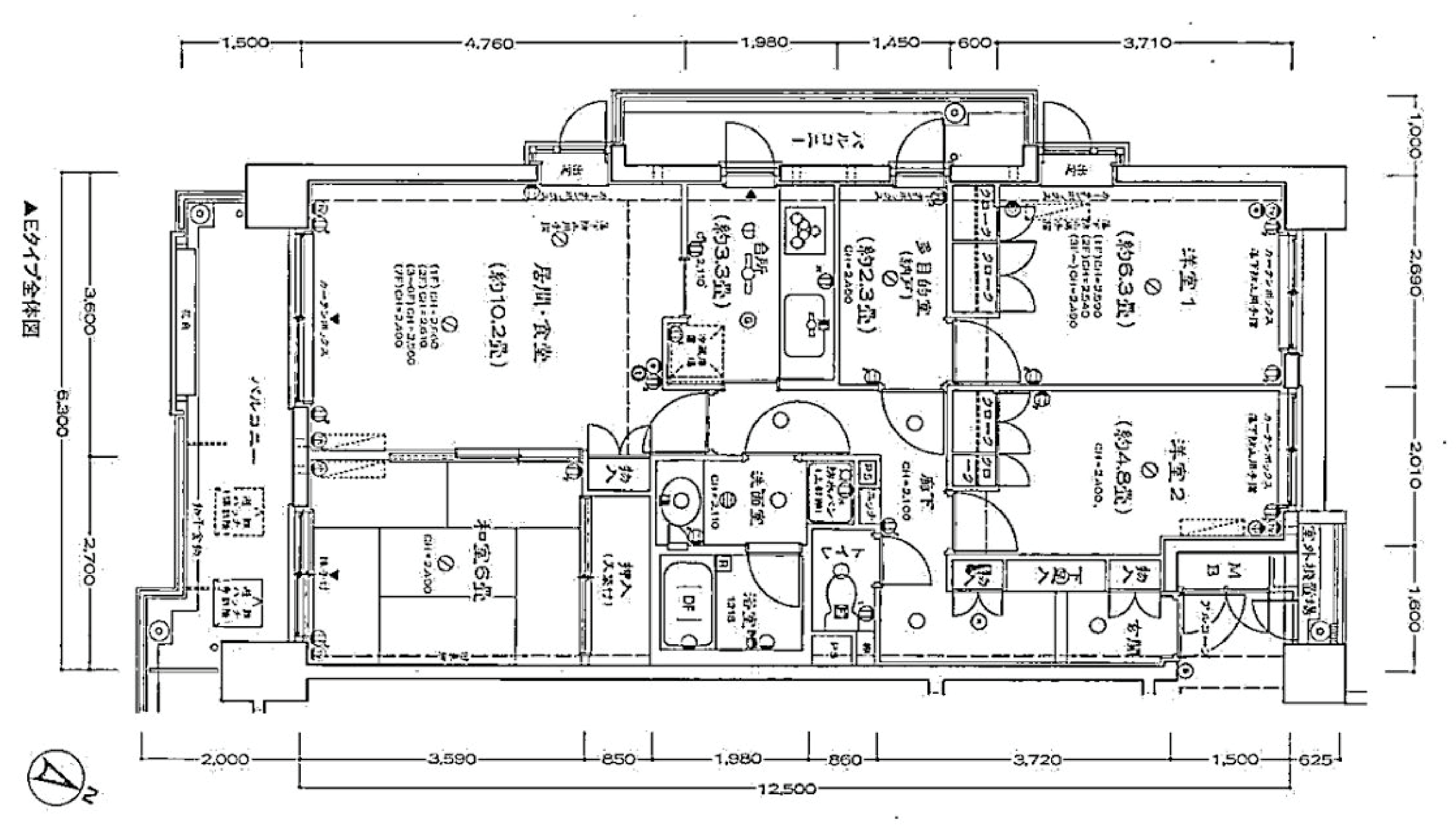
Perception

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

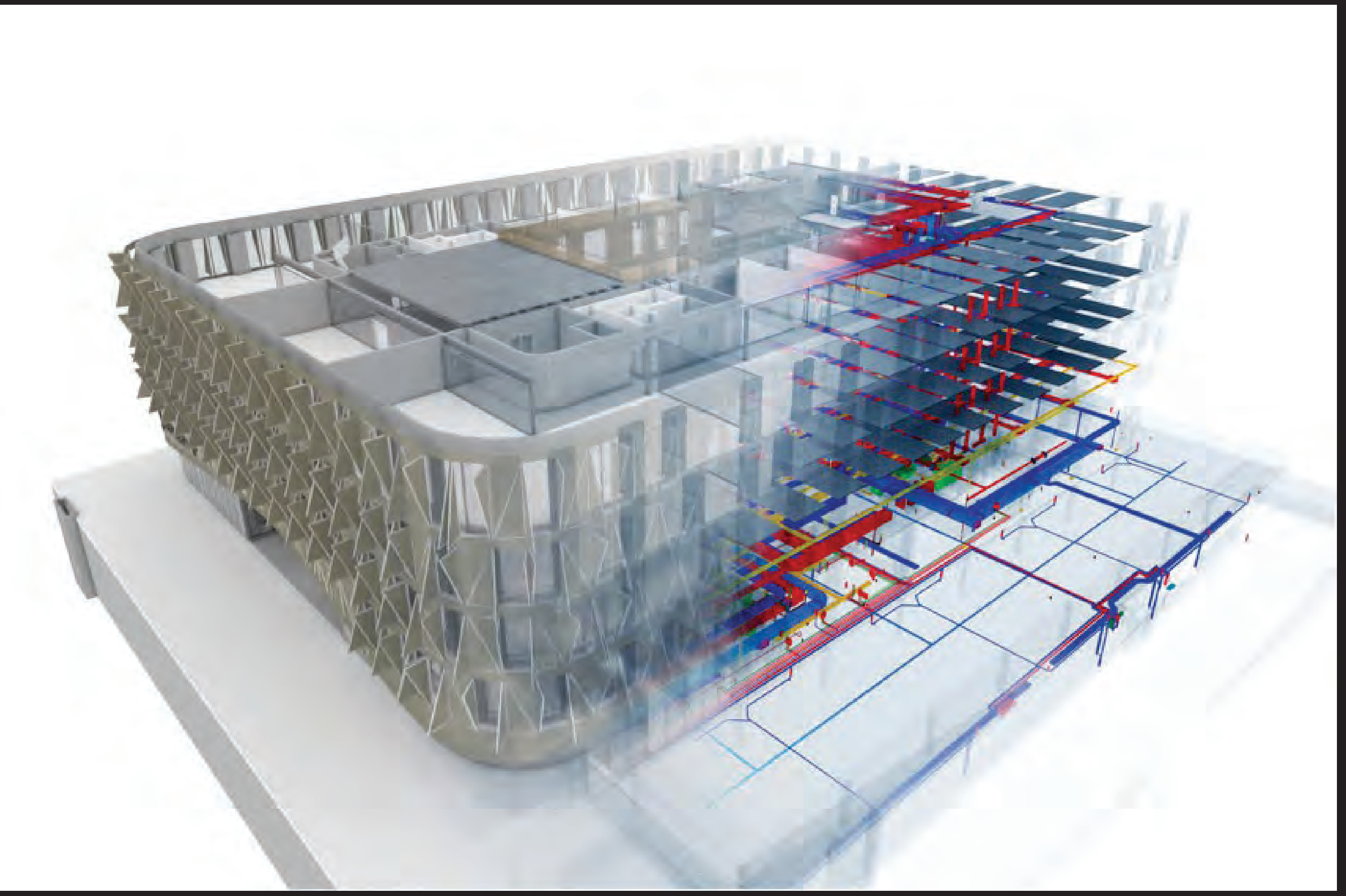
Future challenges



DNN for high-level perception



Construction-level perception



3D perception

Image (smartphone) only

Acknowledgements

IARPA BE THE FUTURE

NSERC CRSNG

Zillow

NSF

贝壳金服 KE FINANCE

Microsoft

Adobe

renoworks

Google

Jean Ponce

Steve Seitz

Brian Curless

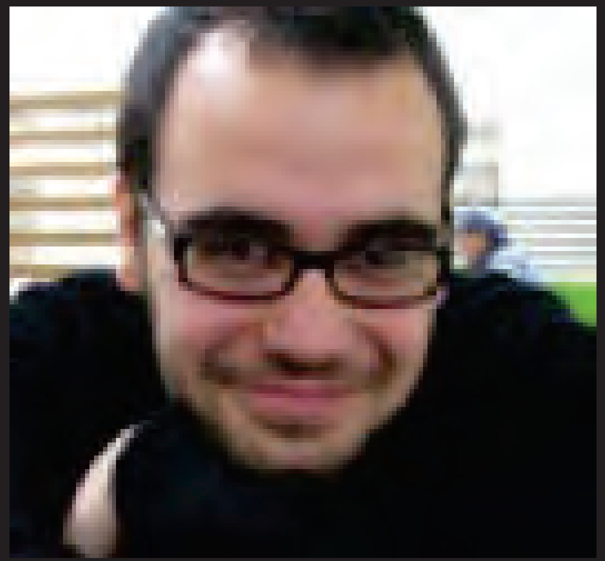
Rick Szeliski

Yoji Kiyota

Satoshi Ikehata

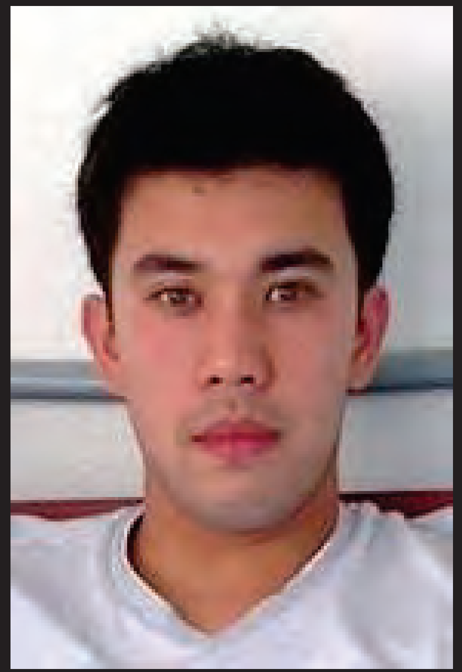
Tomoko Ohsuga

Pushmeet Kohli



Ricardo Cabral

Jiajun Wu



Nelson Nauata

Chen Liu

Hang Yan

Huayi Zeng

Jiachen Chen