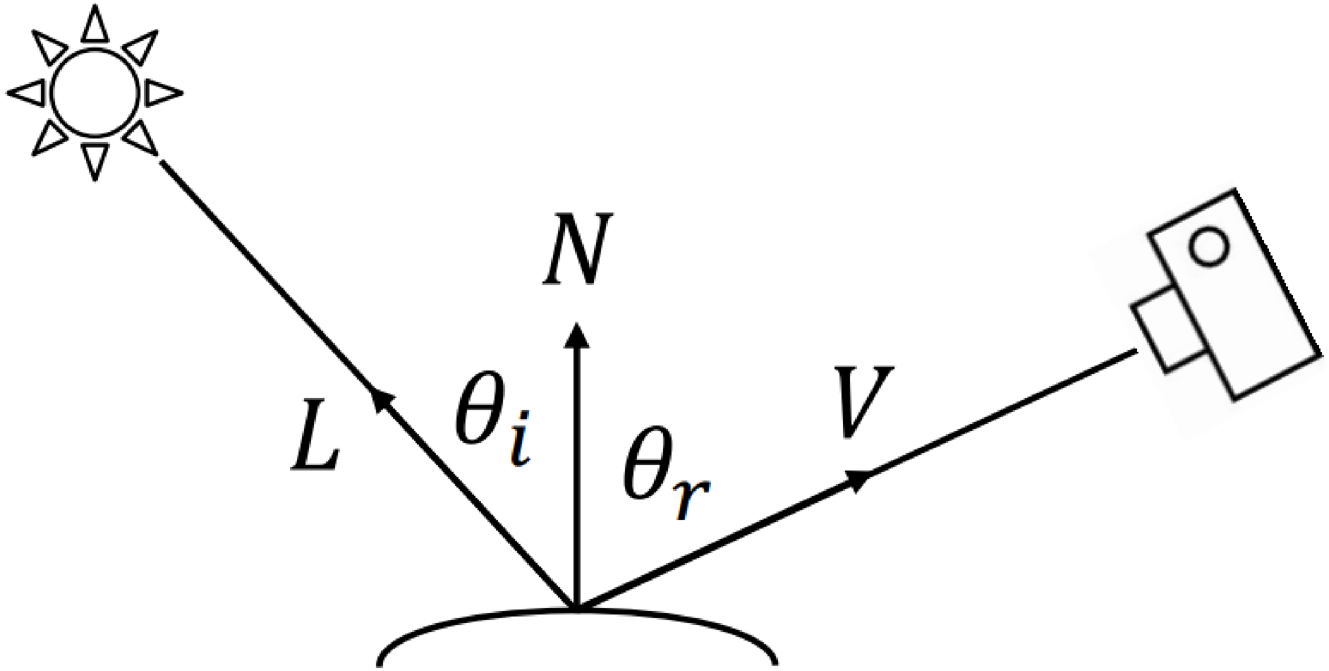
Programming Assignment #3

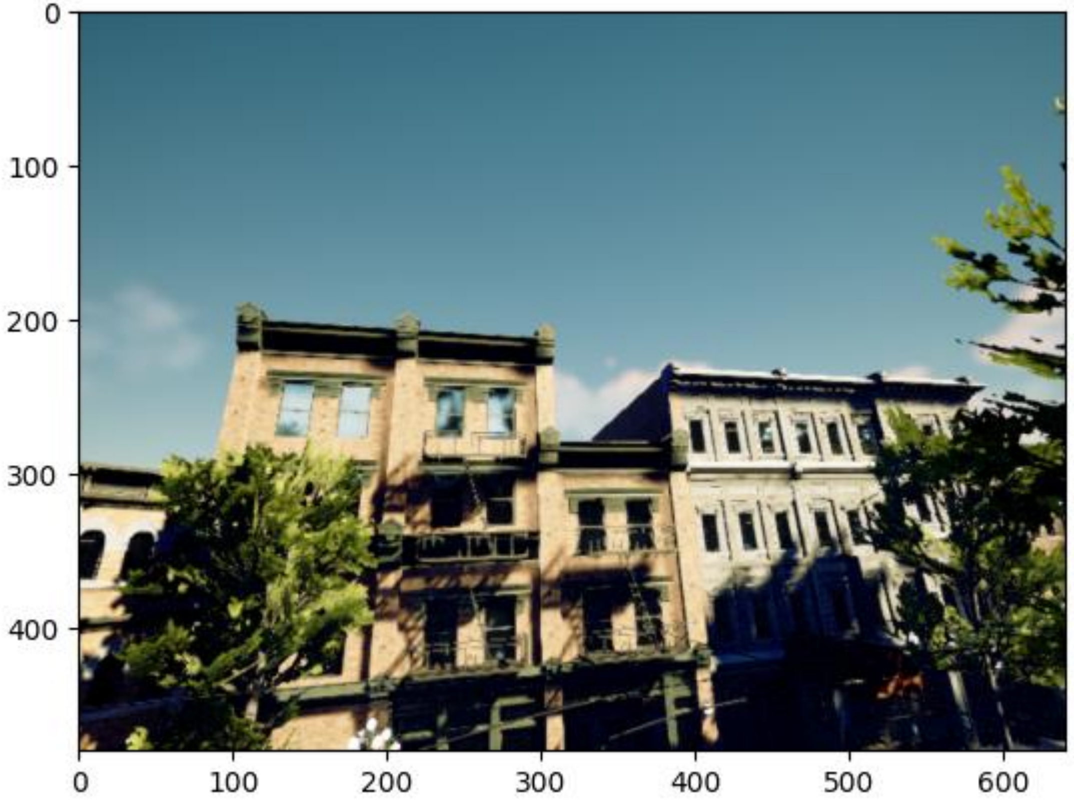
Computer vision

In PA1 (Normal) ...

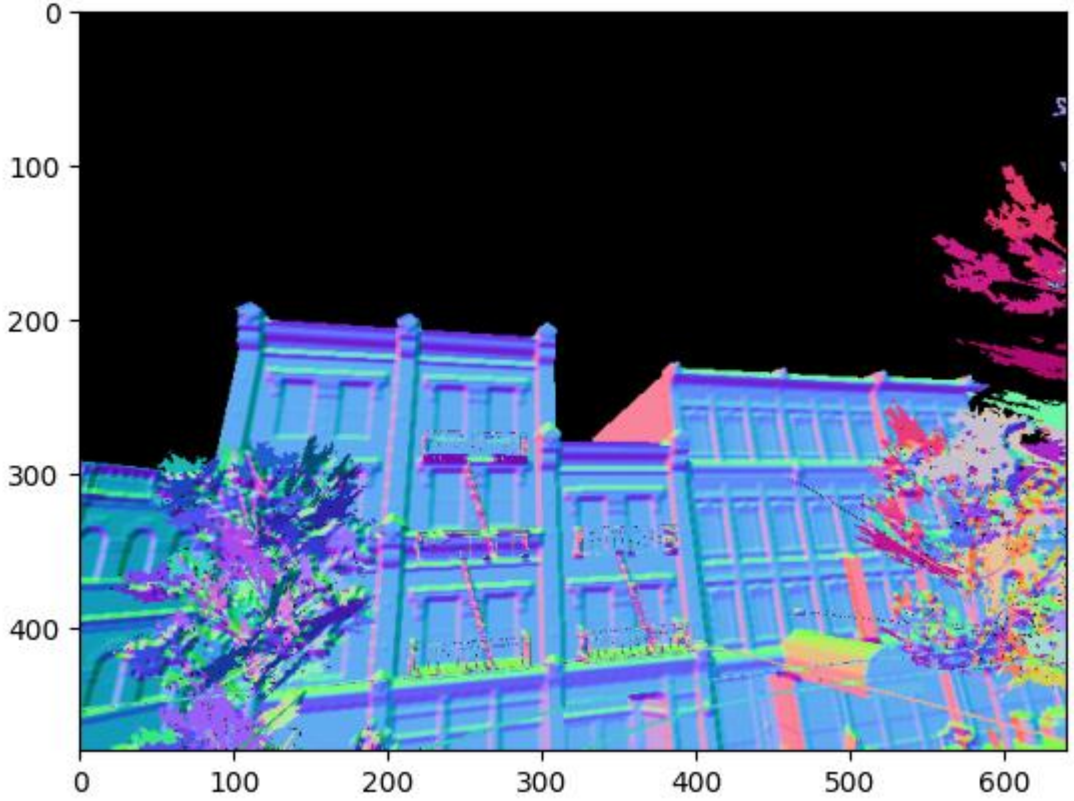


N: Surface Normal
L: Light Direction
V: Viewing direction

In PA1 (Normal) ...



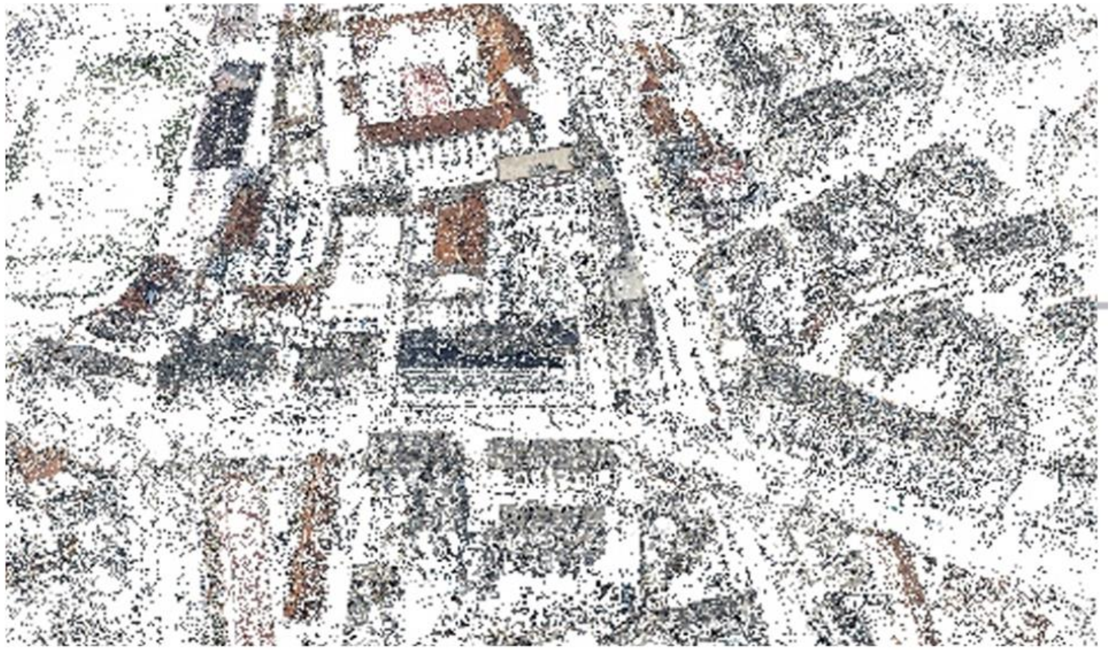
RGB



Normal

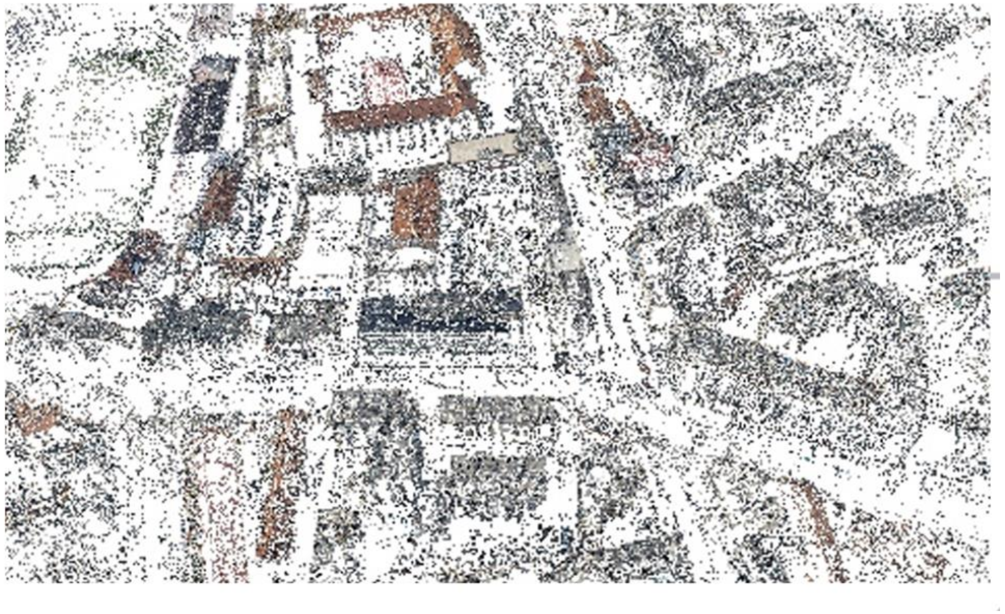
In PA2 (SFM) …

Input images

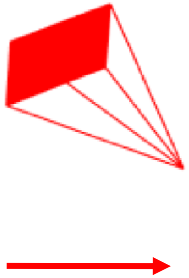


Structure from Motion

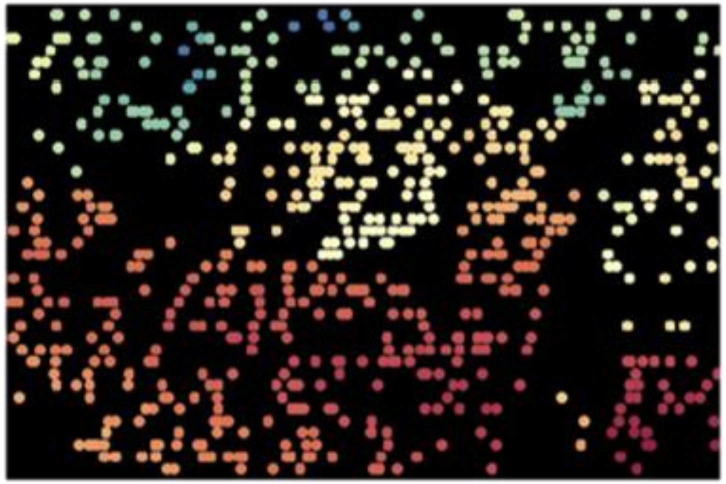
In PA2 (SFM) …



Structure from Motion

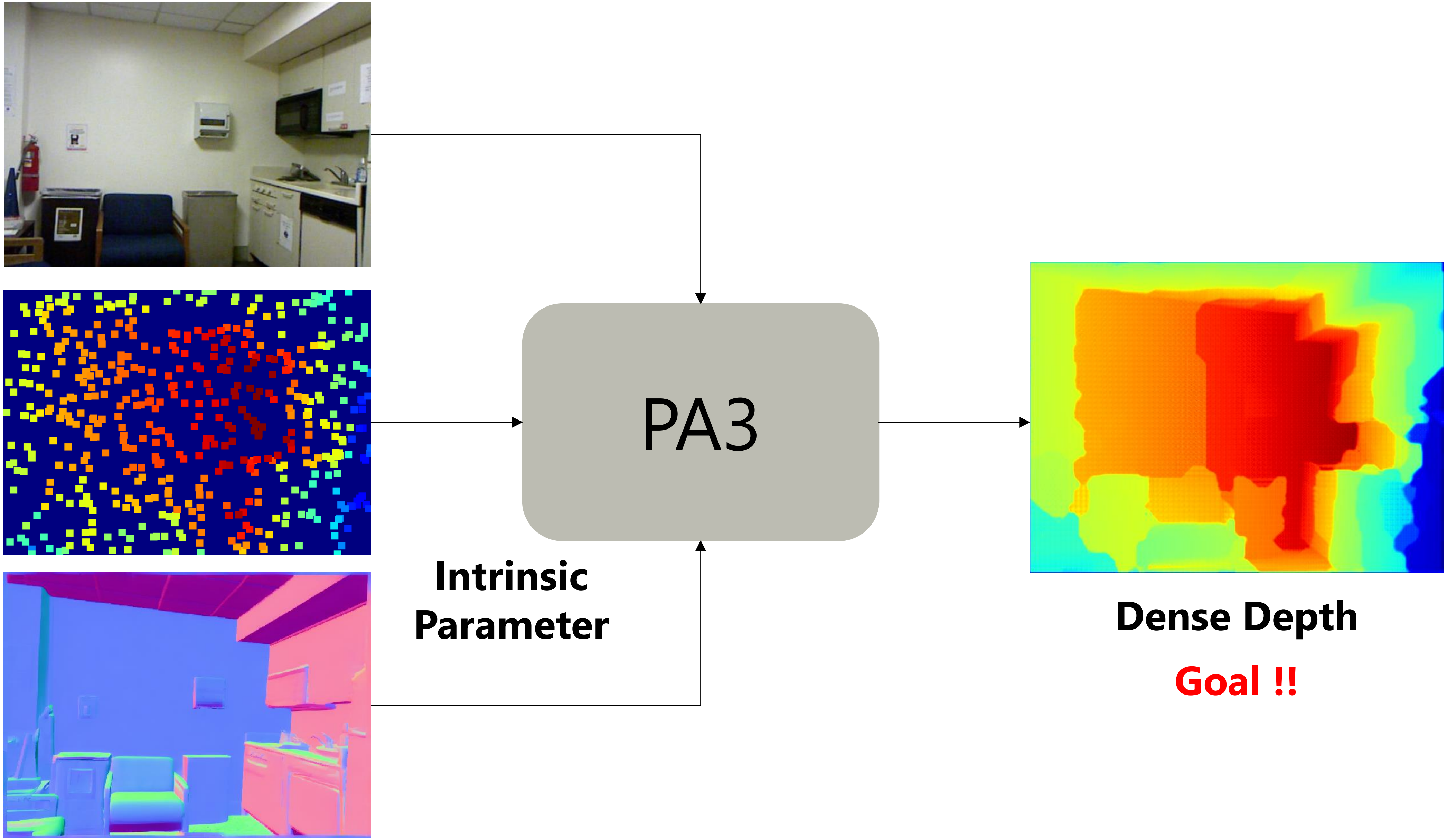


Projection to
Image coordinate



SFM Sparse Depth

What we will do in PA3 (Overview) …



RGB

Sparse Depth

Surface Normal

PA3

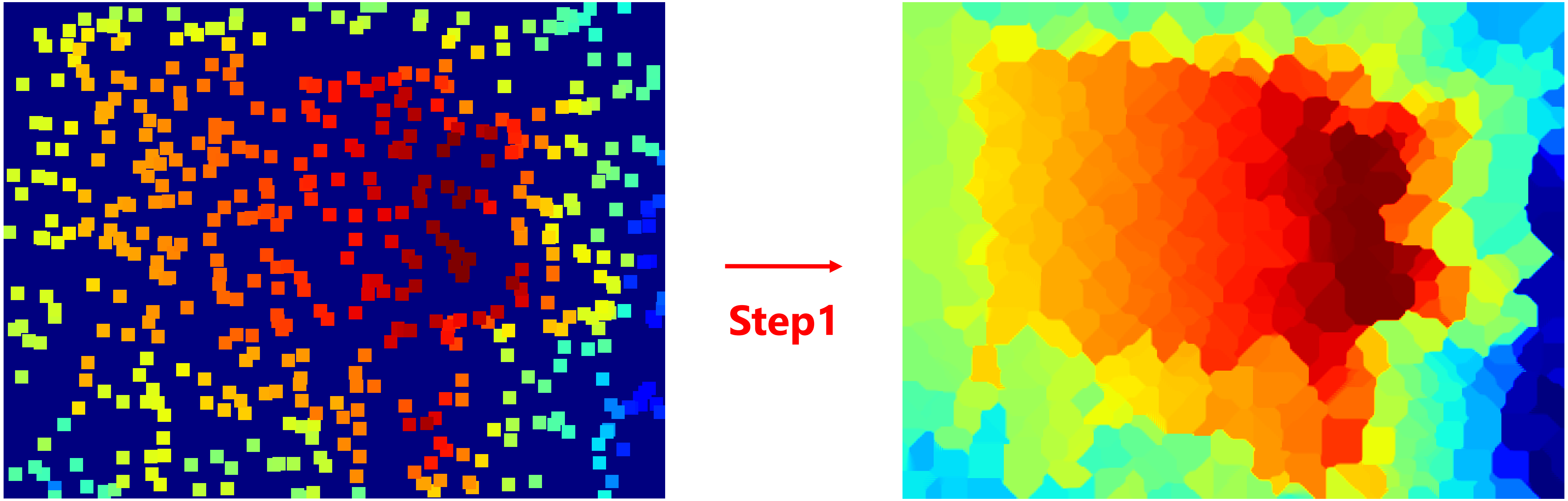
Intrinsic Parameter

Dense Depth

Goal !!

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**



Sparse Depth

Initial Depth

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

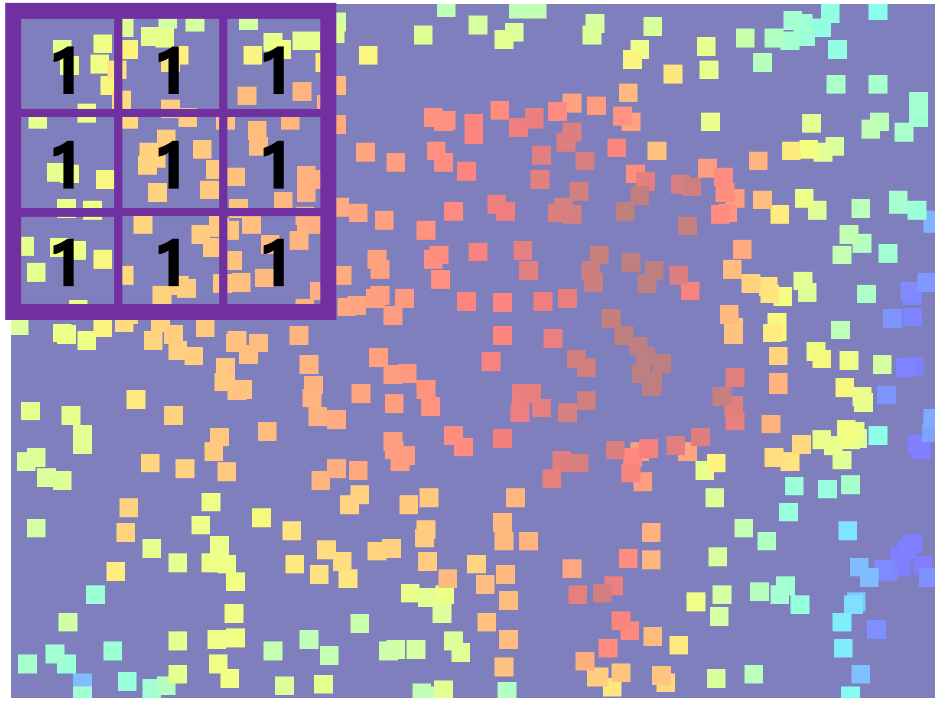
Make a kernel consisting only of ones

1	1	1
1	1	1
1	1	1

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

Move the kernel and perform convolution operation



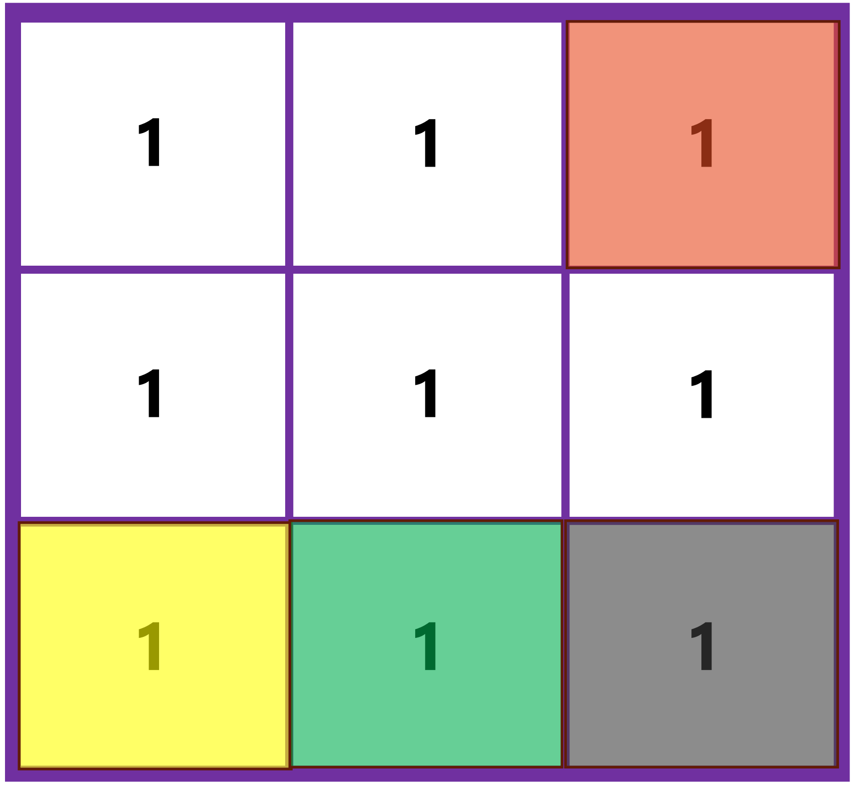
1	1	1
1	1	1
1	1	1

Sparse Depth

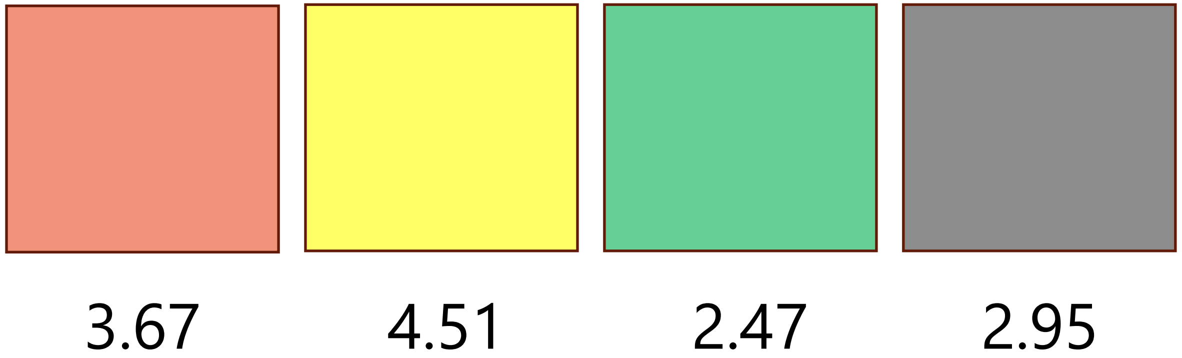
Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

Divide the total value (from convolution) by the number of valid pixels



1	1	🟥 1
1	1	1
🟨 1	🟩 1	⬛ 1



🟥	🟨	🟩	⬛
3.67	4.51	2.47	2.95

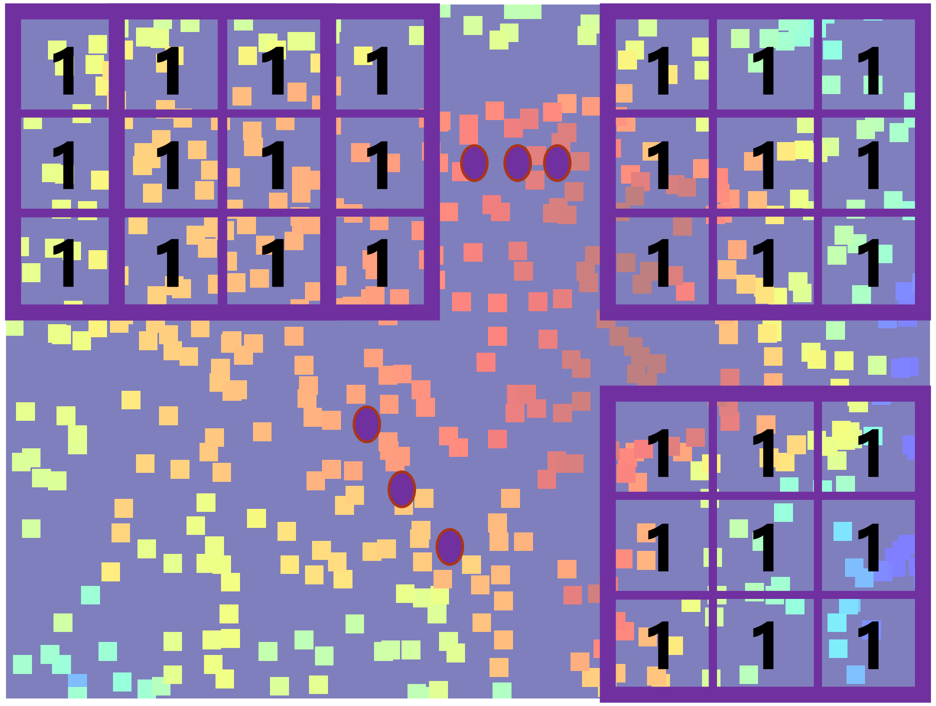
$(3.67 + 4.51 + 2.47 + 2.95) / 4 = \mathbf{3.4}$

Fill the empty pixel with this value 3.4

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

Fill the empty pixel with this value and slide the kernel

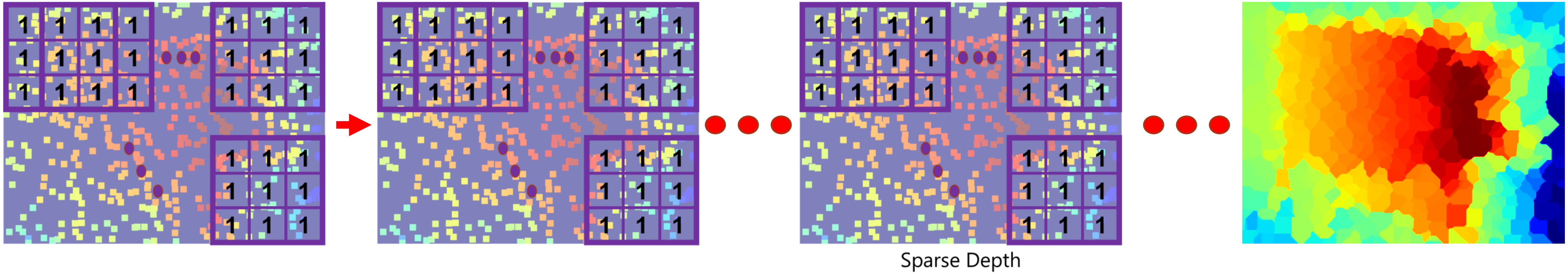


Sparse Depth

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

Iterate the previous process



Iter 1

Depth

Iter 2

Depth

Iter K

Depth

Iter N

Depth

Implementation Details (Step1)

- **Hole Filling (Initial Depth)**

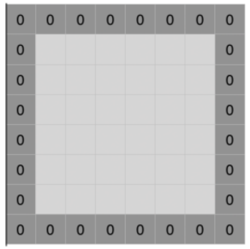
1. This is just my guideline. You can fill in the holes in your own way.

2. However, in that case, be sure to clearly specify in the report how you filled in the holes.

3. You should decide on appropriate values for variables like kernel size and the number of iterations. (3x3, 5x5, 7x7, …. / 7,8,9 iteration …)

4. Proper zero padding of appropriate size is also recommended.

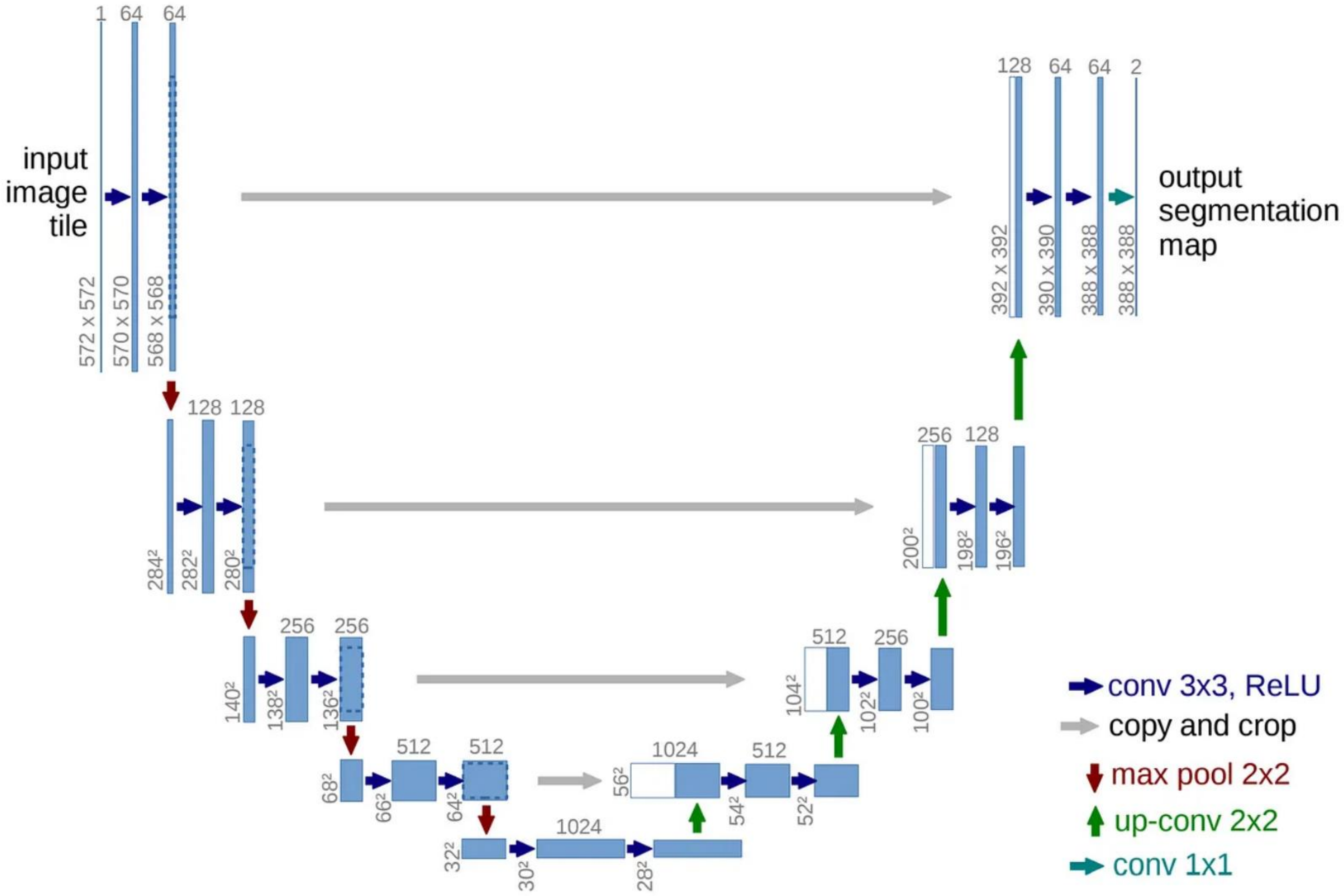
Padding Example!



0	0	0	0	0	0	0	0
0							0
0							0
0							0
0							0
0							0
0							0
0	0	0	0	0	0	0	0

Implementation Details (Step2)

- **UNet Architecture (Example)**



MICCAI'15 Ronneberger. et al U-Net: Convolutional Networks for Biomedical Image Segmentation

Implementation Details (Step2)

- **UNet Architecture**

1. Keep in mind that **slide_14** is just an example. Please design your own UNet by referencing relevant papers or resources.

2. Try your best to implement it while studying papers for learning purposes. This PA3 task doesn't require a very complex UNet structure or a large network size.

3. We don't do plagiarism checks on this part, but please be very specific in your report about the architecture you used and its characteristics.

4. You must design the input channel to include RGB images and Sparse depth, and the output channel must be the same as the Depth channel.

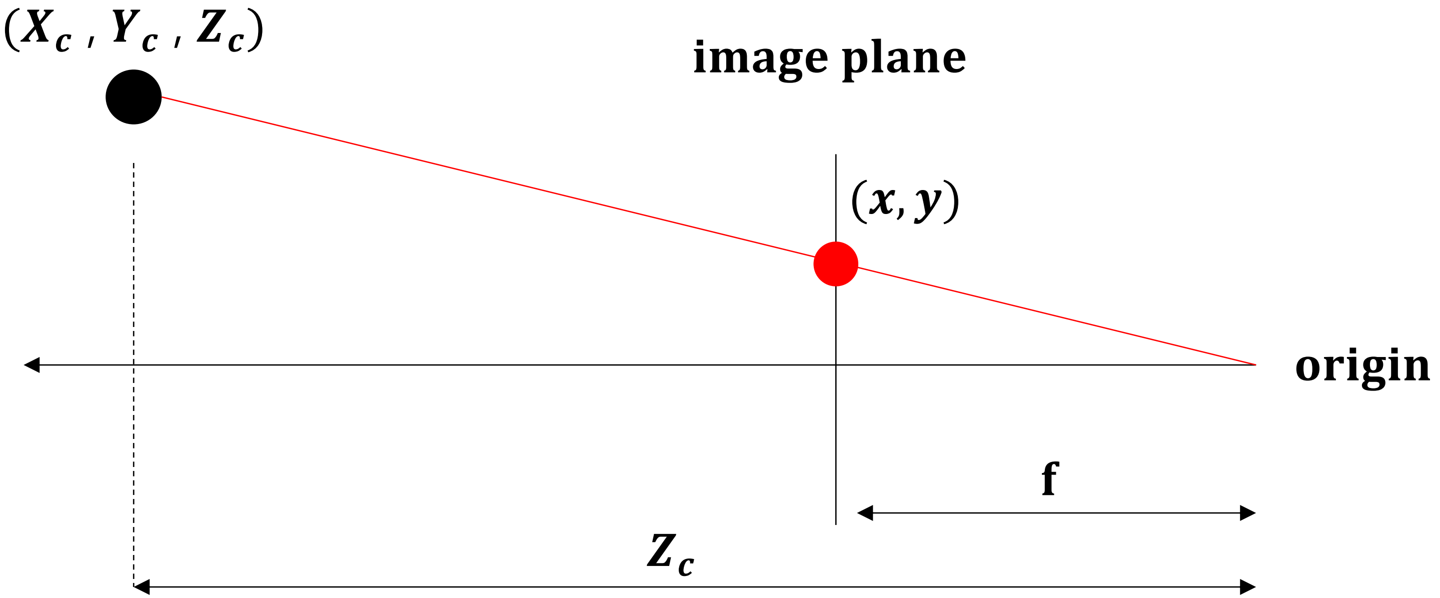
Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

Camera Coordinate System → Image Coordinate System

Camera Coordinate

(X_c , Y_c , Z_c)



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathrm{K} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

https://darkpgmr.tistory.com/32

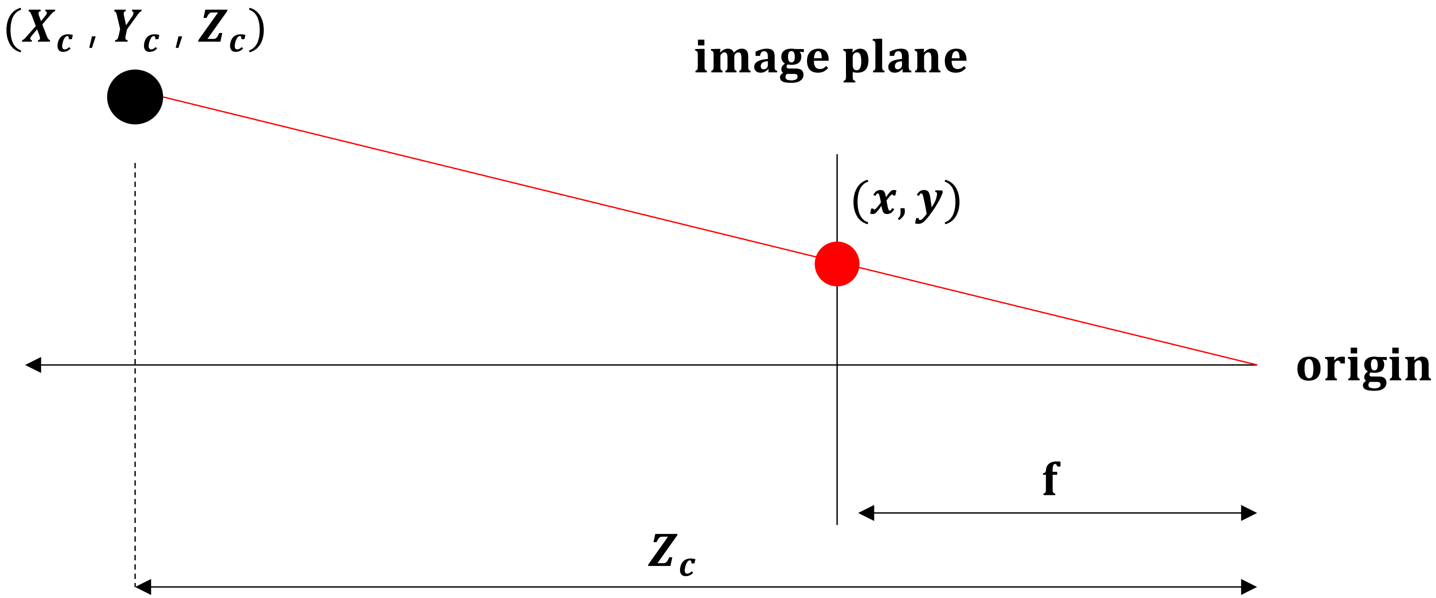
Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

Image Coordinate System → Camera Coordinate System

Camera Coordinate

$(X_c\ , Y_c\ , Z_c)$



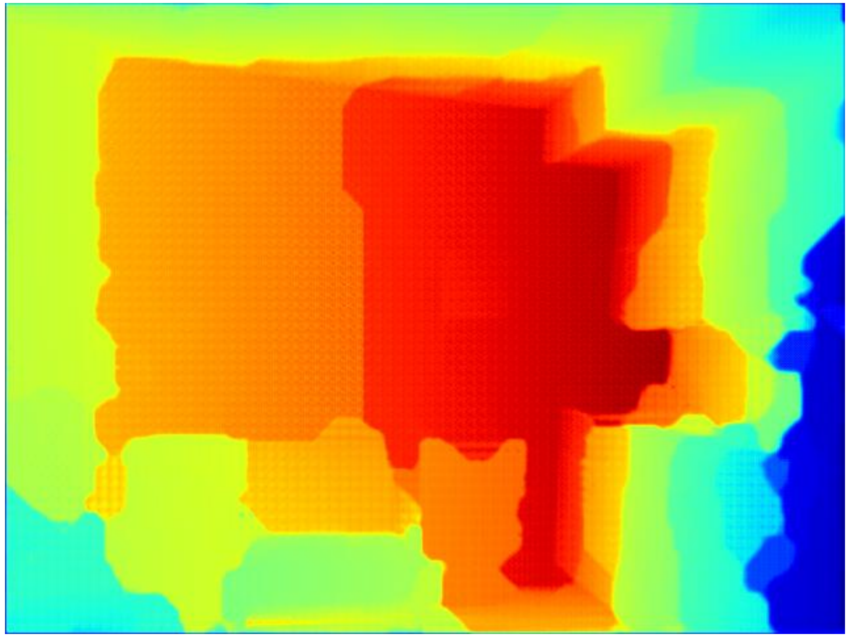
$$K^{-1}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

https://darkpgmr.tistory.com/32

Implementation Details (Step3)

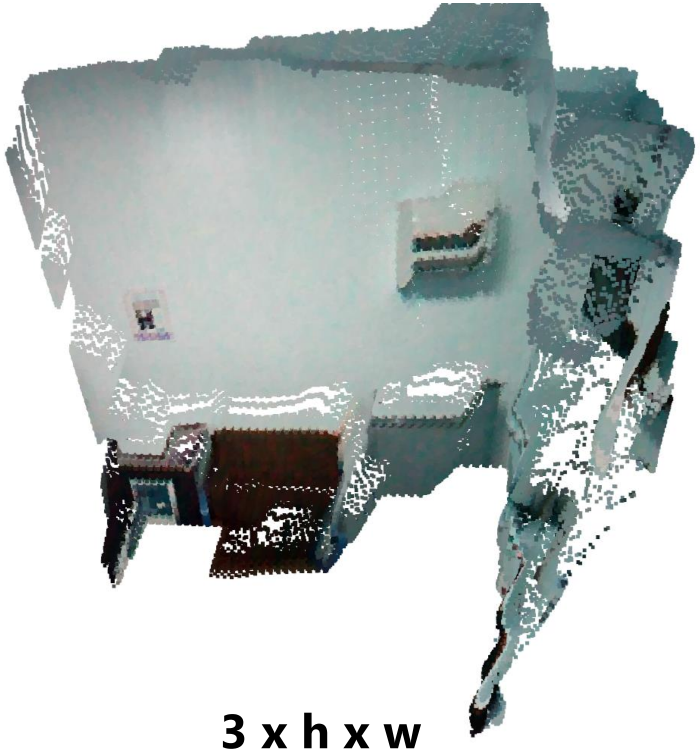
- **Conversion Depth Map to Normal Map**

$$K^{-1}$$



1 x h x w

$\rightarrow$



3 x h x w

* This visualization contains RGB-based 3D information, not accurate camera coordinate 3d points. It's just an illustrative example. Each of the 3 channels should contain x, y, and z coordinate data from camera coordinate.

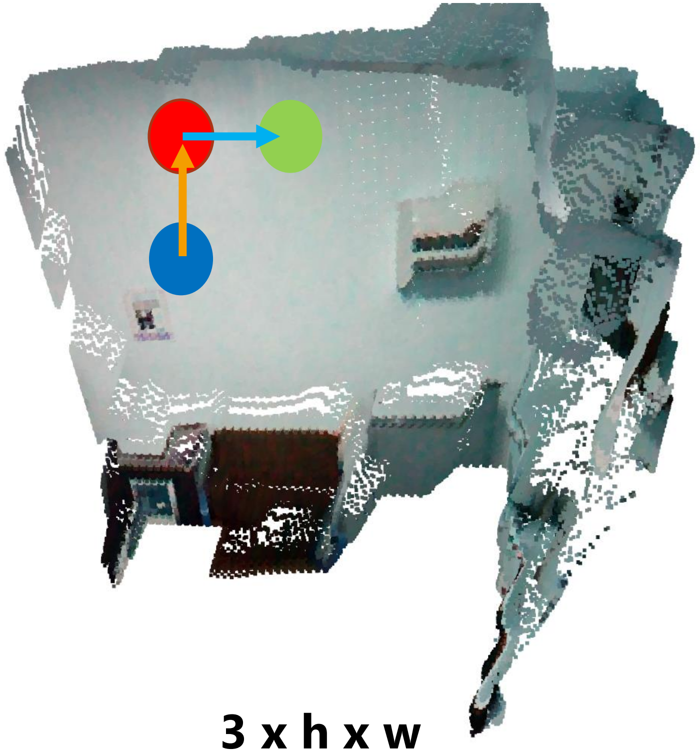
Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

Example values

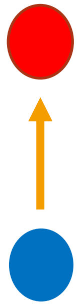
● ● ●

$$\begin{pmatrix}3.5\\4.1\\3.8\end{pmatrix}\begin{pmatrix}3.7\\4.0\\3.6\end{pmatrix}\begin{pmatrix}3.6\\4.2\\3.9\end{pmatrix}$$



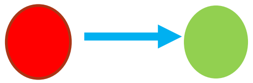
3 x h x w

Vertical vector



$$\begin{pmatrix}3.5\\4.1\\3.8\end{pmatrix} - \begin{pmatrix}3.7\\4.0\\3.6\end{pmatrix} = \begin{pmatrix}\mathbf{-0.2}\\\mathbf{0.1}\\\mathbf{0.2}\end{pmatrix}$$

Horizontal vector



$$\begin{pmatrix}3.7\\4.0\\3.6\end{pmatrix} - \begin{pmatrix}3.5\\4.1\\3.8\end{pmatrix} = \begin{pmatrix}\mathbf{0.2}\\\mathbf{-0.1}\\\mathbf{-0.2}\end{pmatrix}$$

Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

3 x h x w

→

Vertical vector **3 x h x w**

Each of the three channels represents a vertical vector between pixels.

Horizontal vector **3 x h x w**

Each of the three channels represents a horizontal vector between pixels.

Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

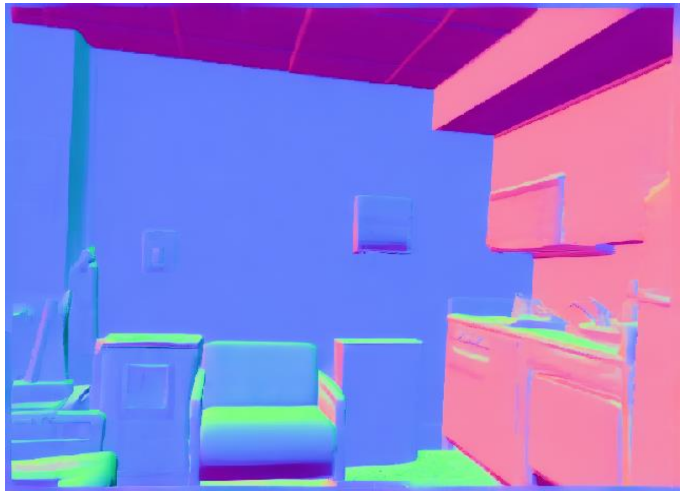
Vertical vector **3 x h x w**

Each of the three channels represents a vertical vector between pixels.

Horizontal vector **3 x h x w**

Each of the three channels represents a horizontal vector between pixels.

Cross Product



Normal vector **3 x h x w**

Implementation Details (Step3)

- **Conversion Depth Map to Normal Map**

1. The normal vector direction should be oriented toward the camera.

2. The normal should be converted to a unit vector using normalization.

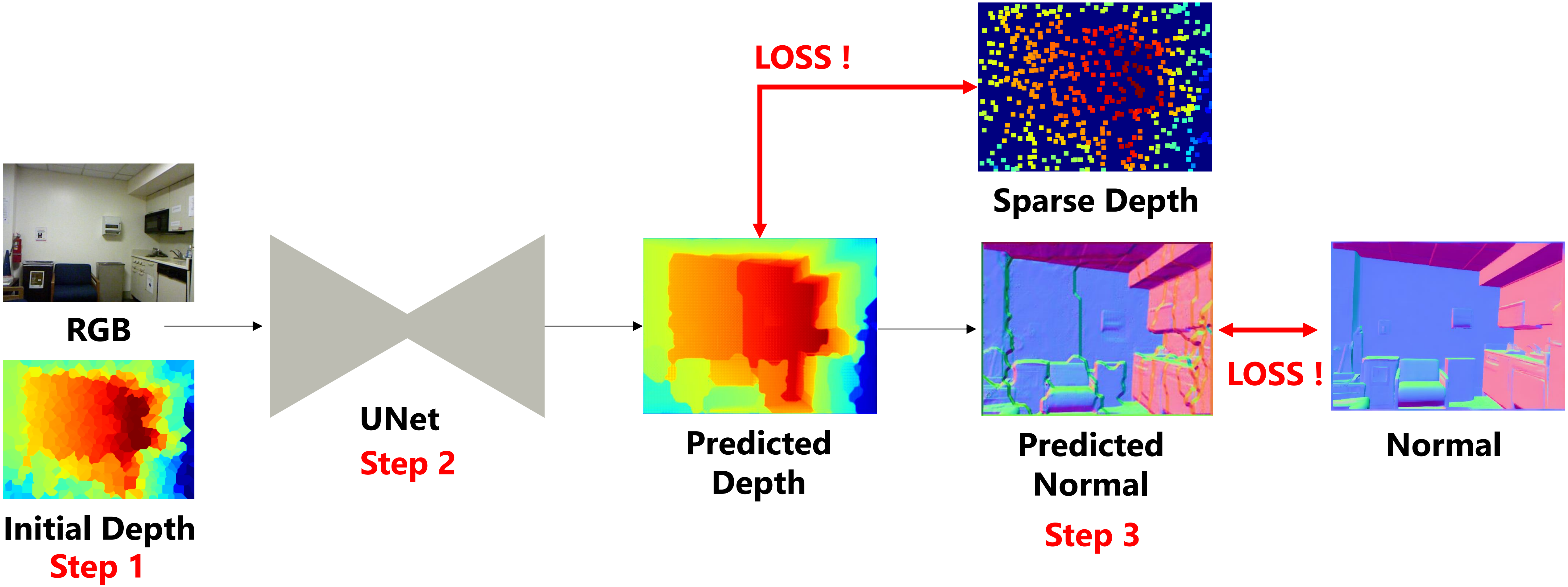
3. It is recommended to perform the operation with appropriate padding.

4. The human parameters used in the conversion should be set appropriately.

5. All steps should be implemented to allow gradients to flow for backpropagation. (use torch!!)

Implementation Details (Step4)

- **Per-Scene Optimization**



LOSS !

Sparse Depth

RGB

UNet

Step 2

Predicted Depth

Predicted Normal

Step 3

LOSS !

Normal

Initial Depth

Step 1

$$Total\ Loss = a\ L_{sparse} + b\ L_{Normal}$$

(a, b is loss weight)

Implementation Details (Step4)

- **Per-Scene Optimization**

1. loss function can be designed freely. Implement and use losses such as L1 or L2.

2. Two types of loss functions are required, and the rest can be added freely.

3. The first loss is the Sparse Depth Loss, which should be computed only for the pixels with valid values in the provided sparse depth.

4. The second loss is the loss between the predicted normal and the provided surface normal.

5. All other human parameters can be freely set. (learning rate, epochs …)

Implementation Details (Step4)

- **Per-Scene Optimization**

6. Set an appropriate number of epochs to show how the depth quality gradually improves from the initial depth.

7. The report must include both the initial depth and the final refined depth.

8. The final refined depth (1×H×W) must also be submitted in .npy format.

Additional Credit (Optional)

- **Top Performance 5 (3 points)**

1. The submitted depth will be evaluated using RMSE (Root Mean Squared Error) based on the GT that only I have. (**PA3_data_for_submission)**
2. Additional points will be awarded to the top 5 submissions with the lowest RMSE.

- **Demonstrating performance improvements through an ablation study by incorporating your own ideas and implementation such as Loss function or module (2 points).**

1. To receive credit, the report must include an explanation of the methodology, as well as the results and analysis.
2. Use the **PA3_data_for_example** data (which includes GT) for performance evaluation.

Instructions

You should implement:

- Step 1. Hole filling (3 points)
- Step 2. Unet Architecture (2 points)
- Step 3. Conversion Depth to Normal (7 points)
- Step 4. Per-Scene Optimization (5 points)
- Optional. 5 top performance (3 points)
- Optional. Additional Module or Loss … (2 points)

TA session: 2025. 06.03 / 2025.06.05
Due Date: 2025. 06.07
Any Questions: chanhwij@gm.gist.ac.kr

Good Luck!

You should write:

- A report (3 points)

Remember!

- No Plagiarism
- No delay
- No use of AI assistance(like ChatGPT)