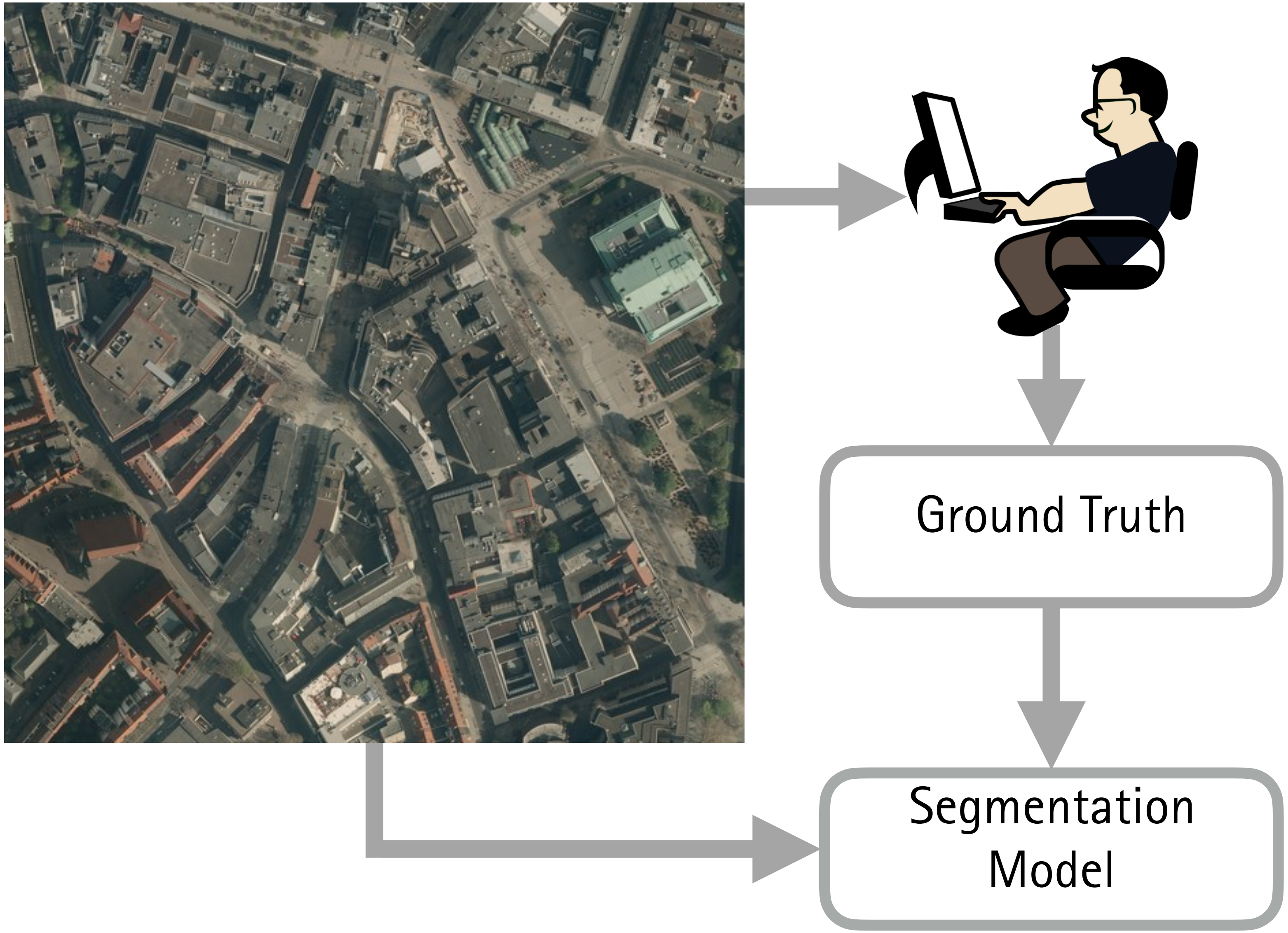
# Using Binary Space Partitioning for the Semantic Segmentation of Aerial Images

Daniel Gritzner, Jörn Ostermann

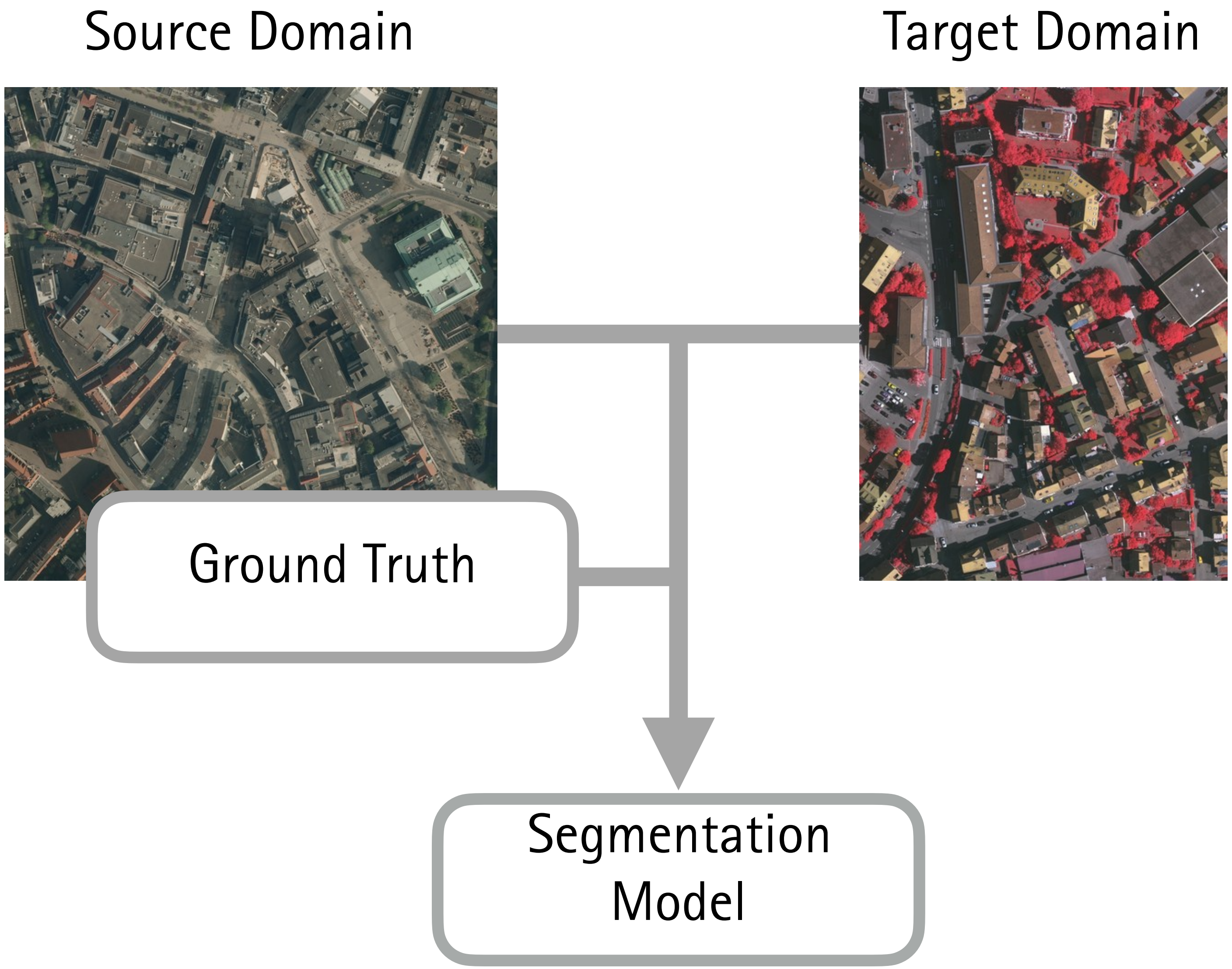
Institut für Informationsverarbeitung, Leibniz Universität Hannover

05.07.2022

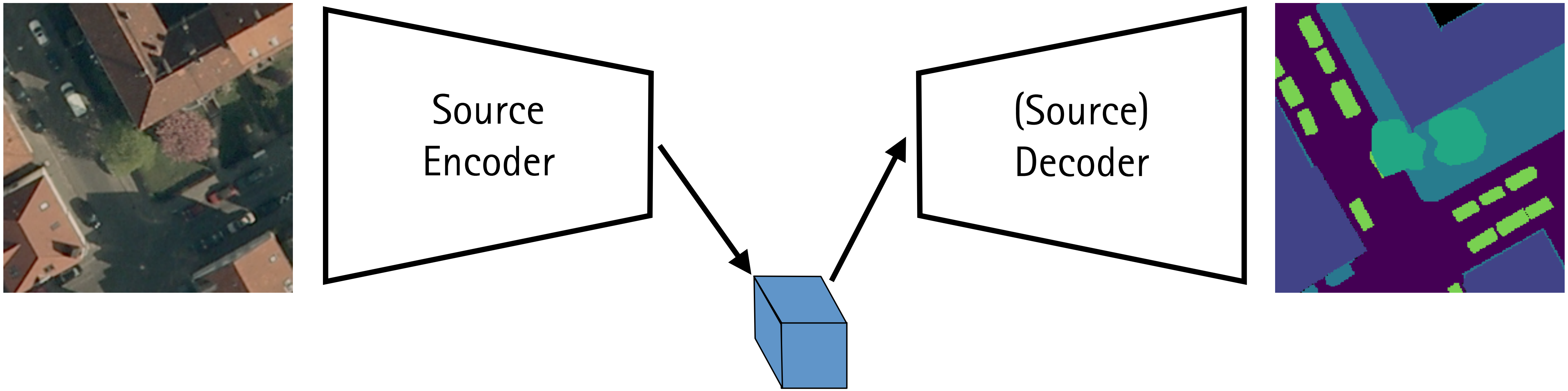
# Motivation



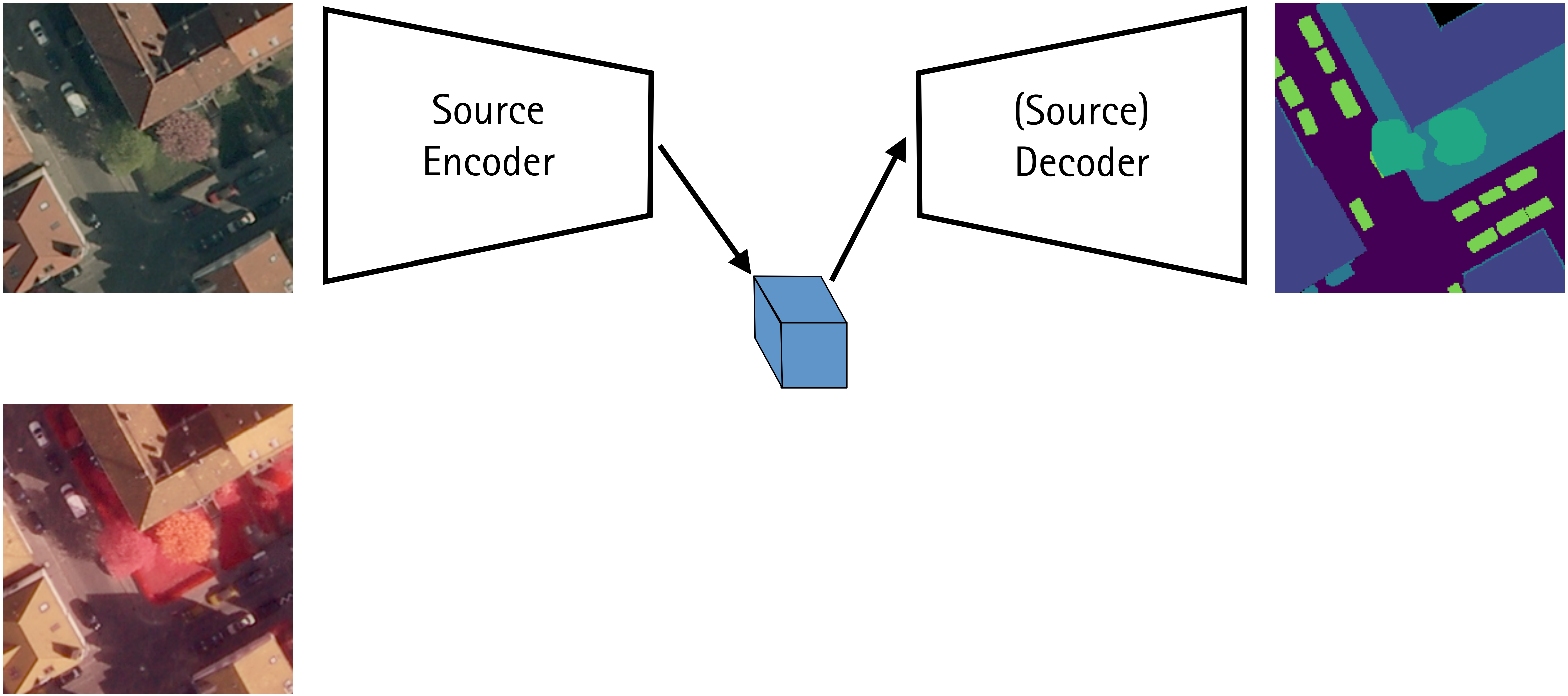
# Motivation


Source Domain
Target Domain
Ground Truth
Segmentation
Model

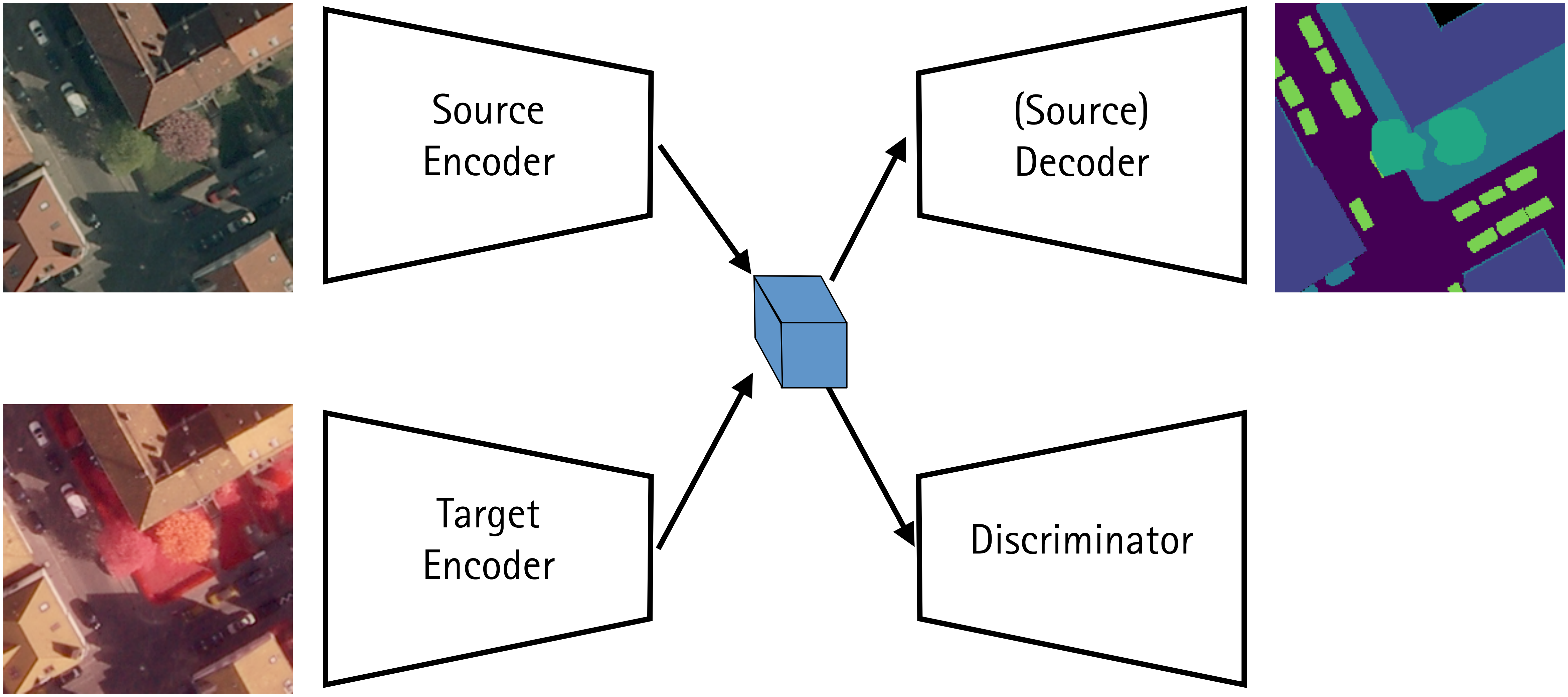
# Motivation



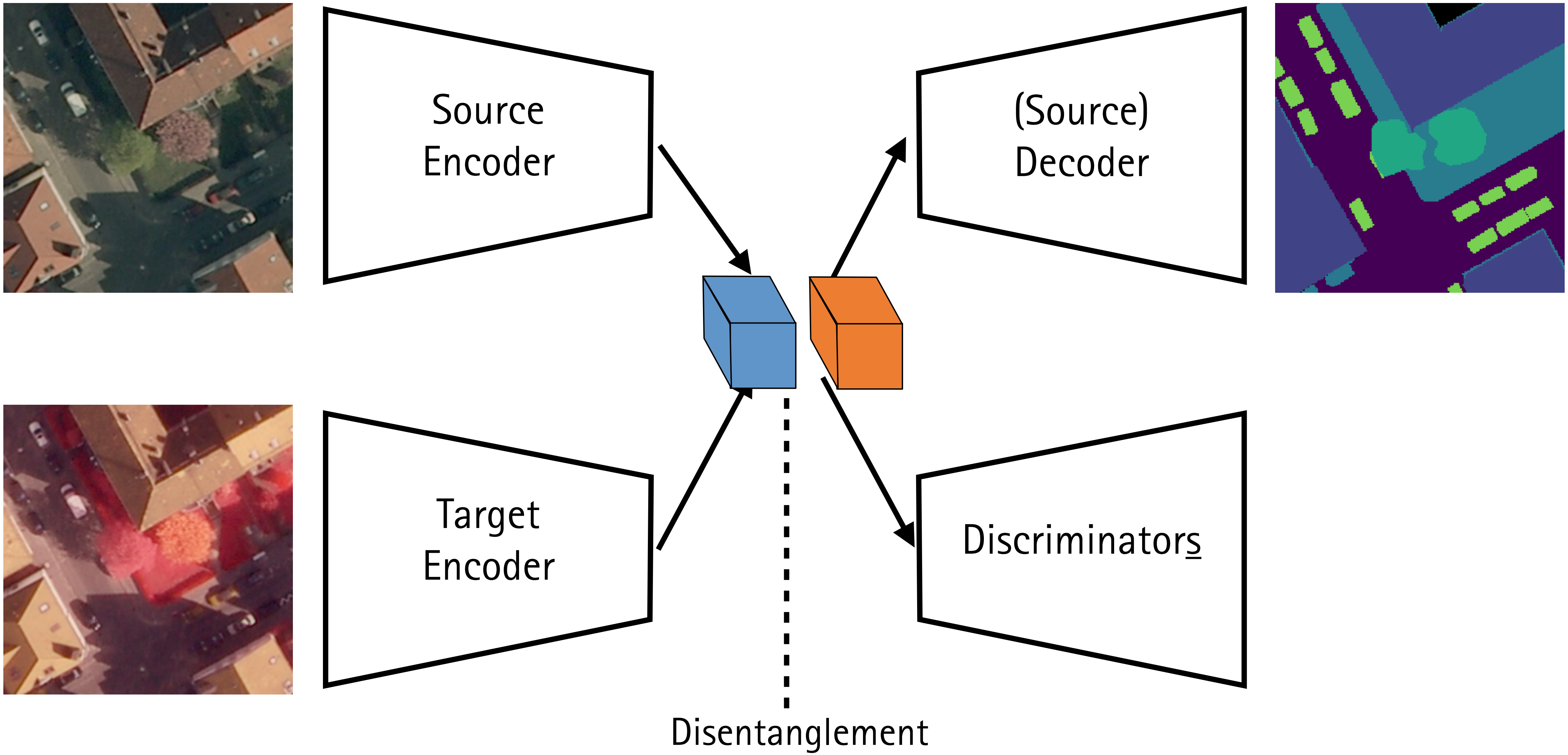
# Motivation



# Motivation

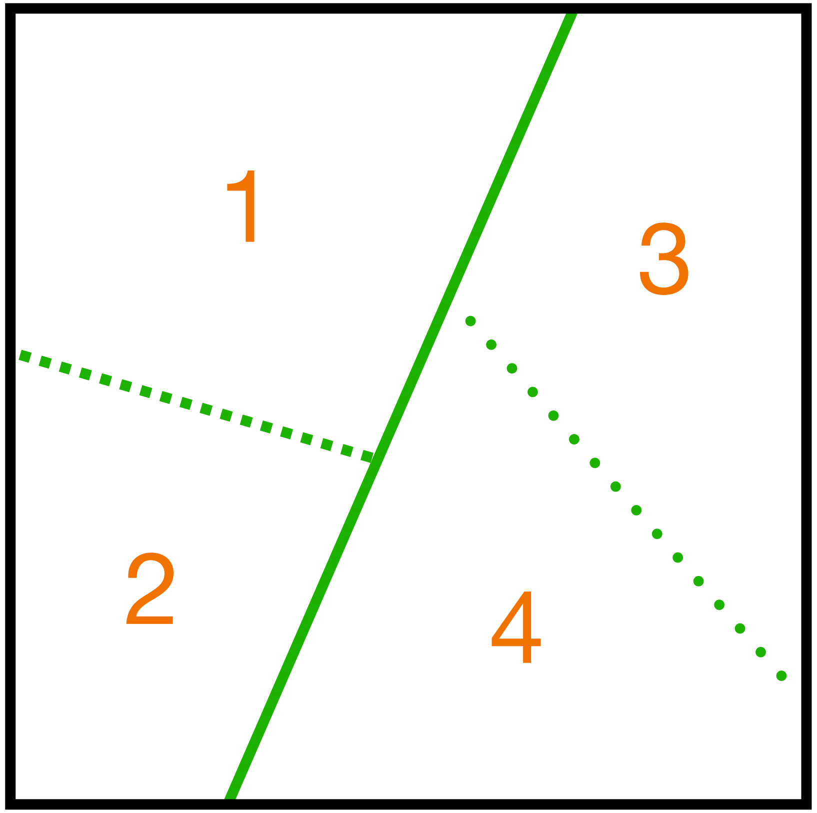

Source Encoder
(Source) Decoder
Target Encoder
Discriminator

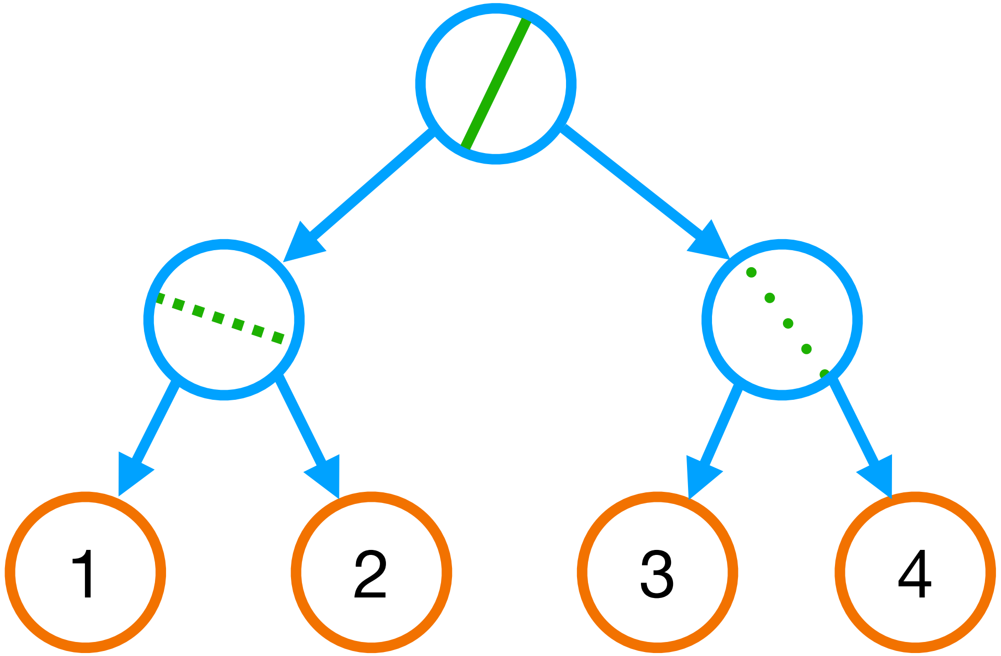
# Motivation


Source
Encoder
(Source)
Decoder
Target
Encoder
Discriminators
Disentanglement

# Binary Space Partitioning Trees

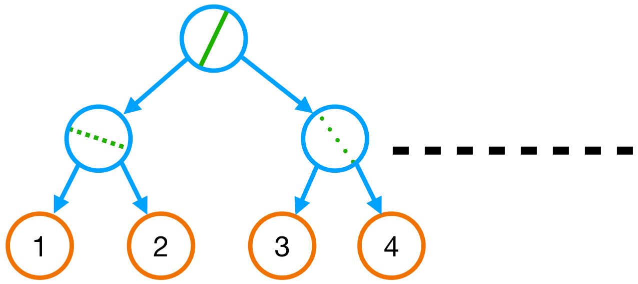


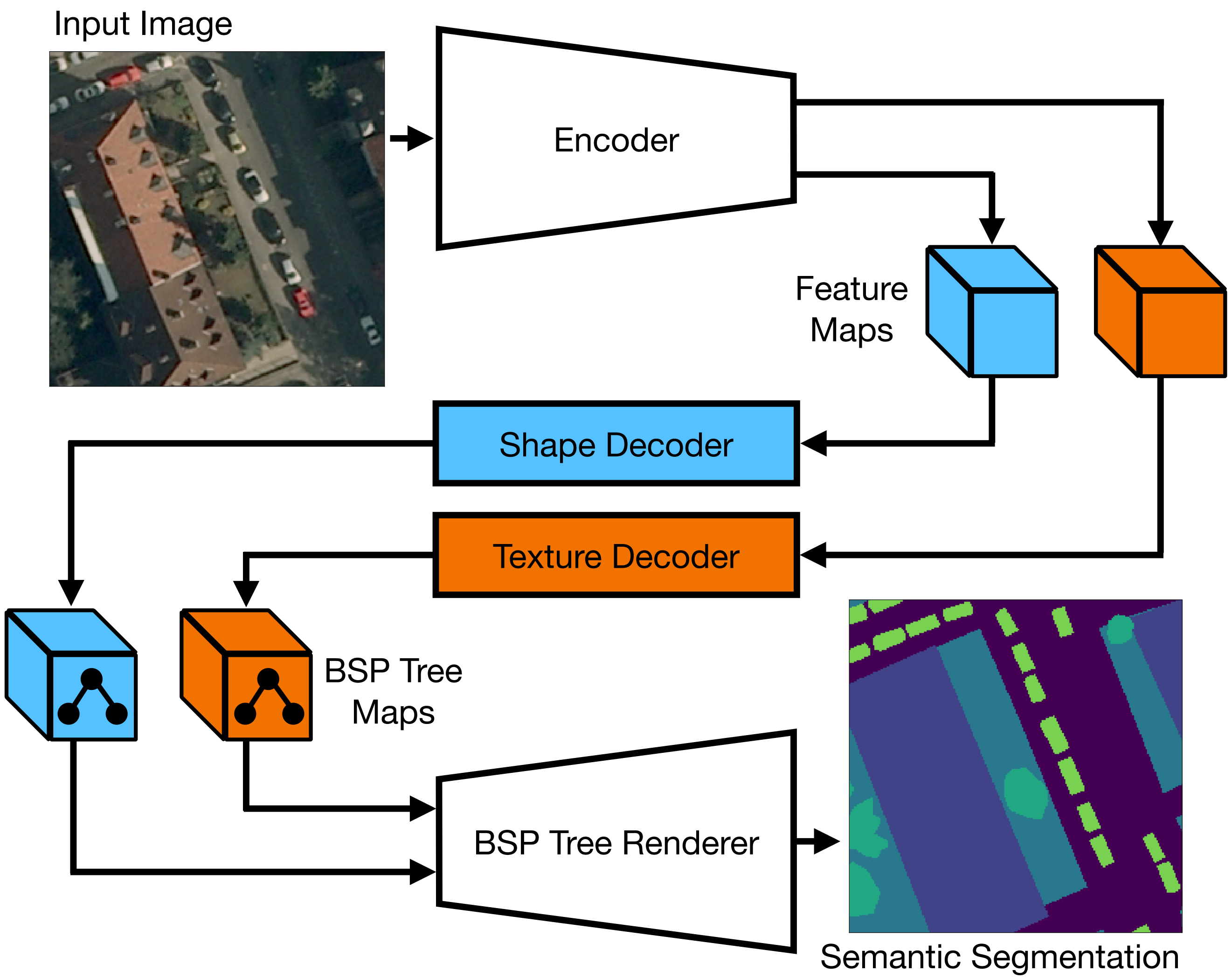


# BSPSegNet

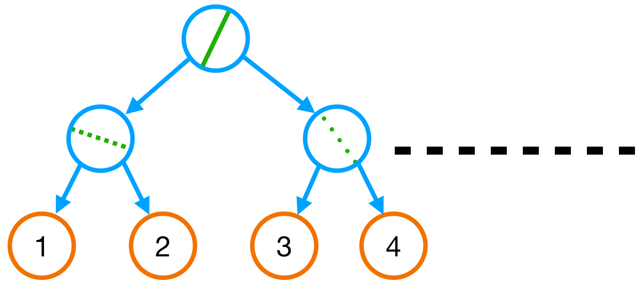
- Predict and render multiple BSP trees
  - Inherently separating shape and texture
- BSP tree renderer is differentiable
  - End-to-end training

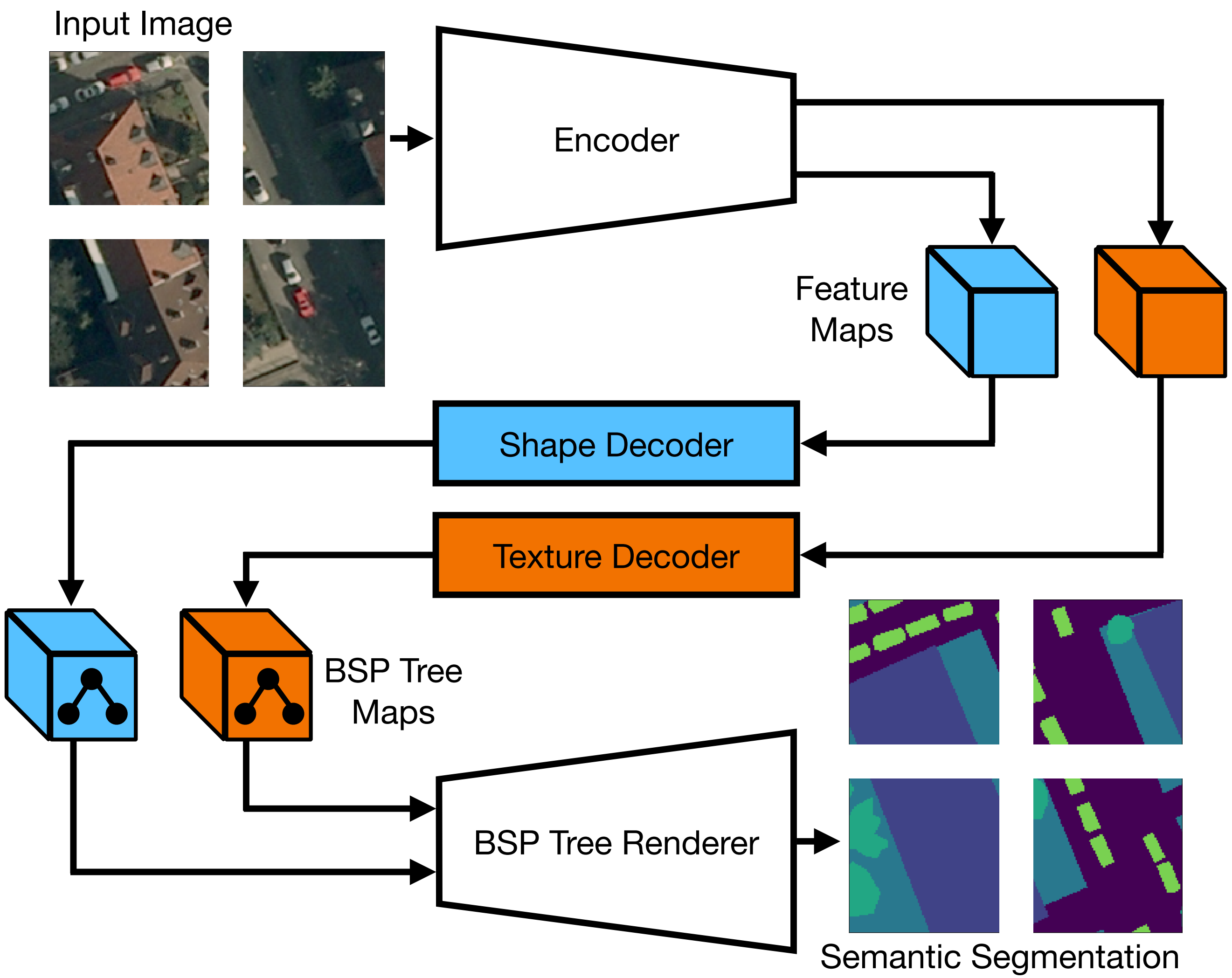


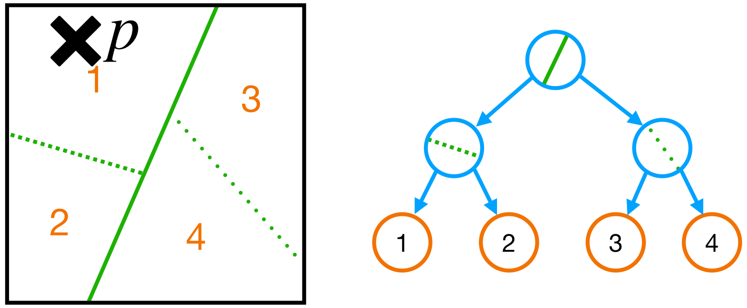
# BSPSegNet

- Predict and render multiple BSP trees
  - Inherently separating shape and texture
- BSP tree renderer is differentiable
  - End-to-end training



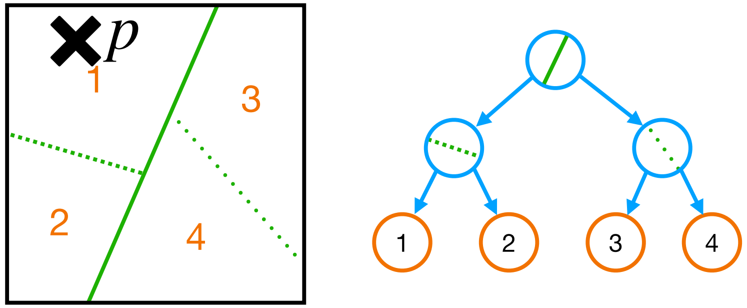



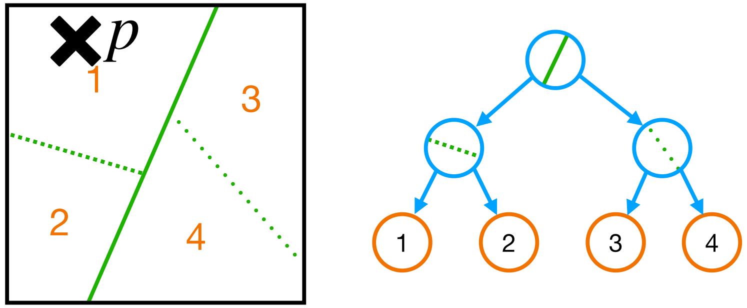
# Differentiable BSP Tree Rendering – Overview



# Differentiable BSP Tree Rendering – Overview

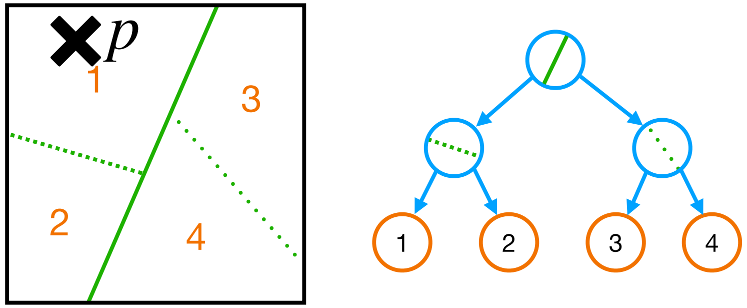
- Compute region map $r'$ that assigns a one-hot vector $r'_p$ to each pixel $p$ defining the region $p$ belongs to
  - $r'_p = (1{,}0{,}0{,}0)$, e.g., means $p$ belongs to the region associated with the first leaf node





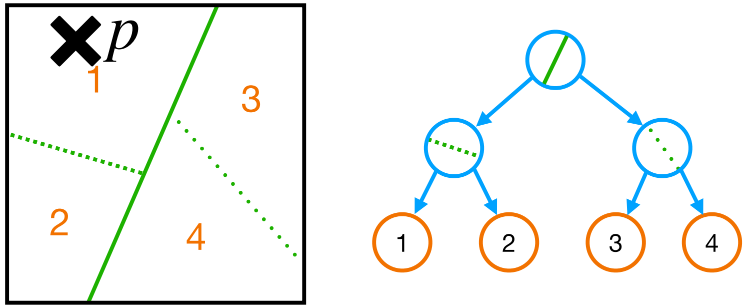

- Compute region map $r'$ that assigns a one-hot vector $r'_p$ to each pixel $p$ defining the region $p$ belongs to
  - $r'_p = (1,0,0,0)$, e.g., means $p$ belongs to the region associated with the first leaf node
- Then use the region map $r'$ and the predicted class logits $v_i$ for each leaf node $i$ to determine each pixel $p$'s class logits

# Differentiable BSP Tree Rendering – Region Map

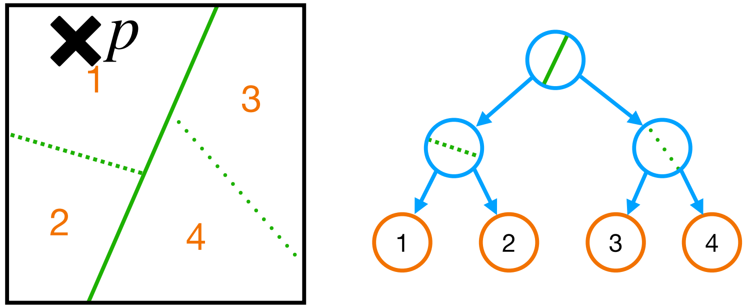


# Differentiable BSP Tree Rendering – Region Map

- Signed distance function: $f(p) = n \cdot p - d$
  - $n$ and $d$ are predicted, one fixed sample point $p$ per pixel
  - $|f(p)|$ is the distance to the predicted line
  - $\mathrm{sign}(f(p))$ indicates on which side of the line $p$ lies

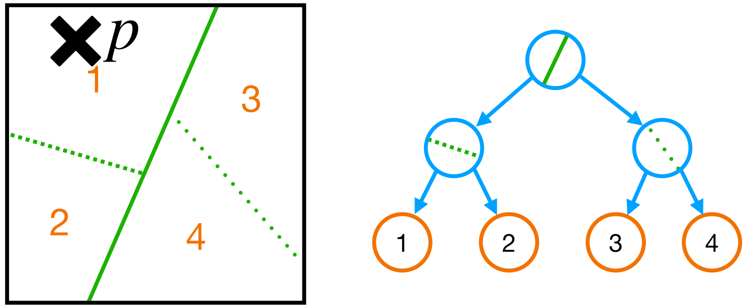


# Differentiable BSP Tree Rendering – Region Map



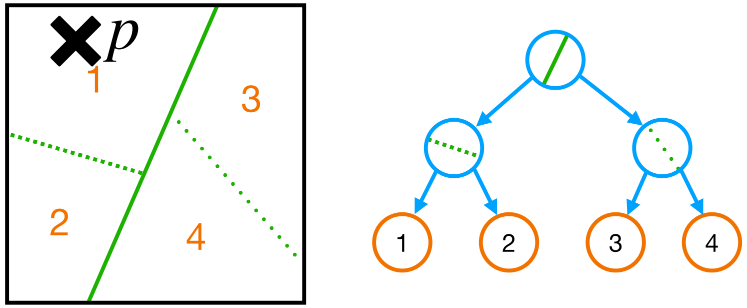

- Signed distance function: $f(p) = n \cdot p - d$
  - $n$ and $d$ are predicted, one fixed sample point $p$ per pixel
  - $|f(p)|$ is the distance to the predicted line
  - $\text{sign}(f(p))$ indicates on which side of the line $p$ lies
- Initial region map: for each pixel store a vector $r_p$ of $1$s, e.g., $r_p = (1,1,1,1)$

# Differentiable BSP Tree Rendering – Region Map




- Signed distance function: $f(p) = n \cdot p - d$
  - $n$ and $d$ are predicted, one fixed sample point $p$ per pixel
  - $|f(p)|$ is the distance to the predicted line
  - $\text{sign}(f(p))$ indicates on which side of the line $p$ lies
- Initial region map: for each pixel store a vector $r_p$ of $1$s, e.g., $r_p = (1,1,1,1)$
- For each inner node:
  - $r_p[\text{left}] = r_p[\text{left}] \cdot \lambda_R \cdot \sigma(\lambda_C f(p))$
  - $r_p[\text{right}] = r_p[\text{right}] \cdot \lambda_R \cdot (1 - \sigma(\lambda_C f(p)))$

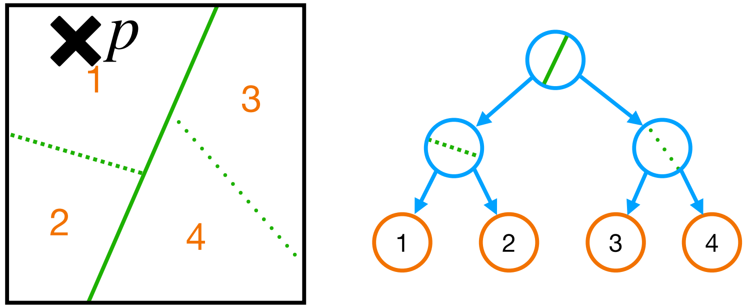
# Differentiable BSP Tree Rendering – Region Map




- Signed distance function: $f(p) = n \cdot p - d$
  - $n$ and $d$ are predicted, one fixed sample point $p$ per pixel
  - $|f(p)|$ is the distance to the predicted line
  - $\text{sign}(f(p))$ indicates on which side of the line $p$ lies
- Initial region map: for each pixel store a vector $r_p$ of $1$s, e.g., $r_p = (1,1,1,1)$
- For each inner node:
  - $r_p[\text{left}] = r_p[\text{left}] \cdot \lambda_R \cdot \sigma(\lambda_C f(p))$
  - $r_p[\text{right}] = r_p[\text{right}] \cdot \lambda_R \cdot (1 - \sigma(\lambda_C f(p)))$
- $r_p$ converges to one-hot-like vector, e.g., $r_p = (\lambda_R^D,0,0,0)$

# Differentiable BSP Tree Rendering – Per-pixel Prediction

- Compute final per-pixel prediction $g(p)$:
  - $v_i \leftarrow$ predicted class logits in leaf node $i$
  - $r'_p = \text{softmax}(r_p)$
  - final prediction: $g(p) = \sum_i r'_p[i] \cdot v_i$



# Datasets

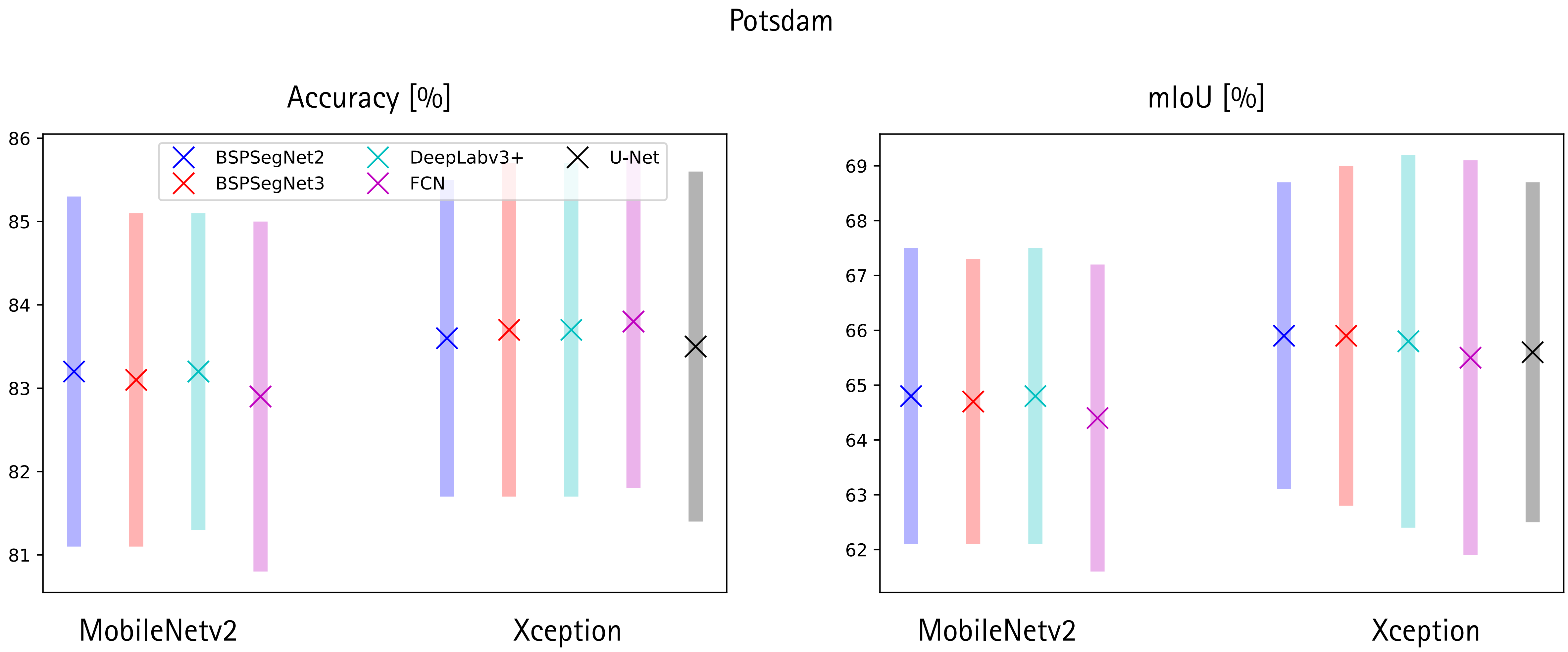
- Five datasets used for evaluation:
  - Vaihingen, Potsdam, Hannover, Nienburg, Buxtehude
- 16 images per dataset
  - 10 images used for training, the rest as validation and test sets
- Size: 2336x1281 to 6000x6000 pixels
  - 224x224 image patches used as model input
  - Random translation, rotation and shearing used for augmentation to 8000 training patches
- Ground sampling distance: 5 to 20cm
- Channels: Near-infrared, red, green, (blue), depth
- Classes: Impervious Surface, Building, Tree, Low Vegetation, Car, Clutter

# Autoencoder

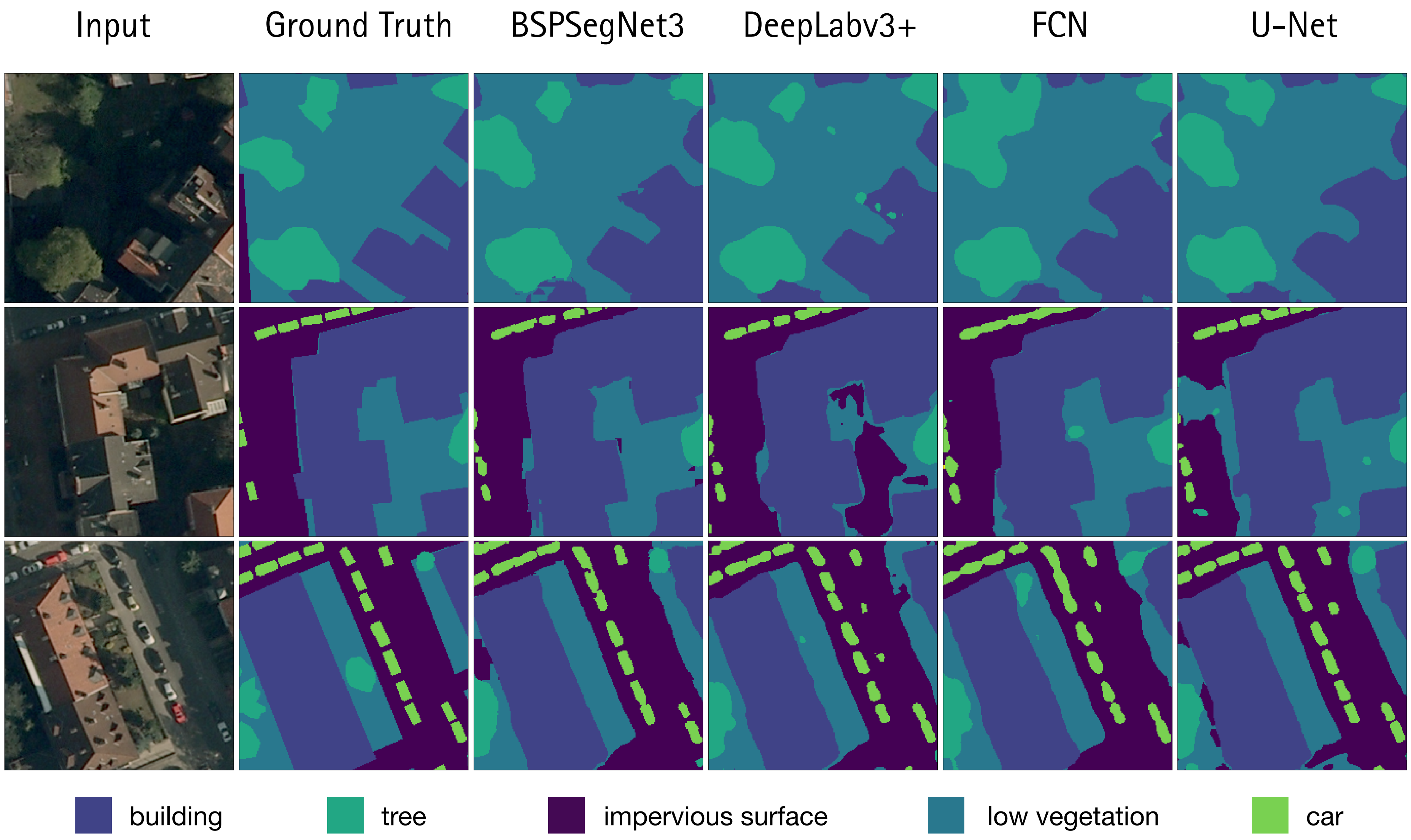
- Use BSPSegNet as ground truth autoencoder
  - Encoder: MobileNetv2
  - Two BSP tree depths: 2 and 3
  - Each BSP tree encodes a 8x8 pixel block

- 99.7% to 99.8% accuracy
- 97.4% to 99.4% mIoU
- Almost all configurations have a standard deviation of less than 0.7%

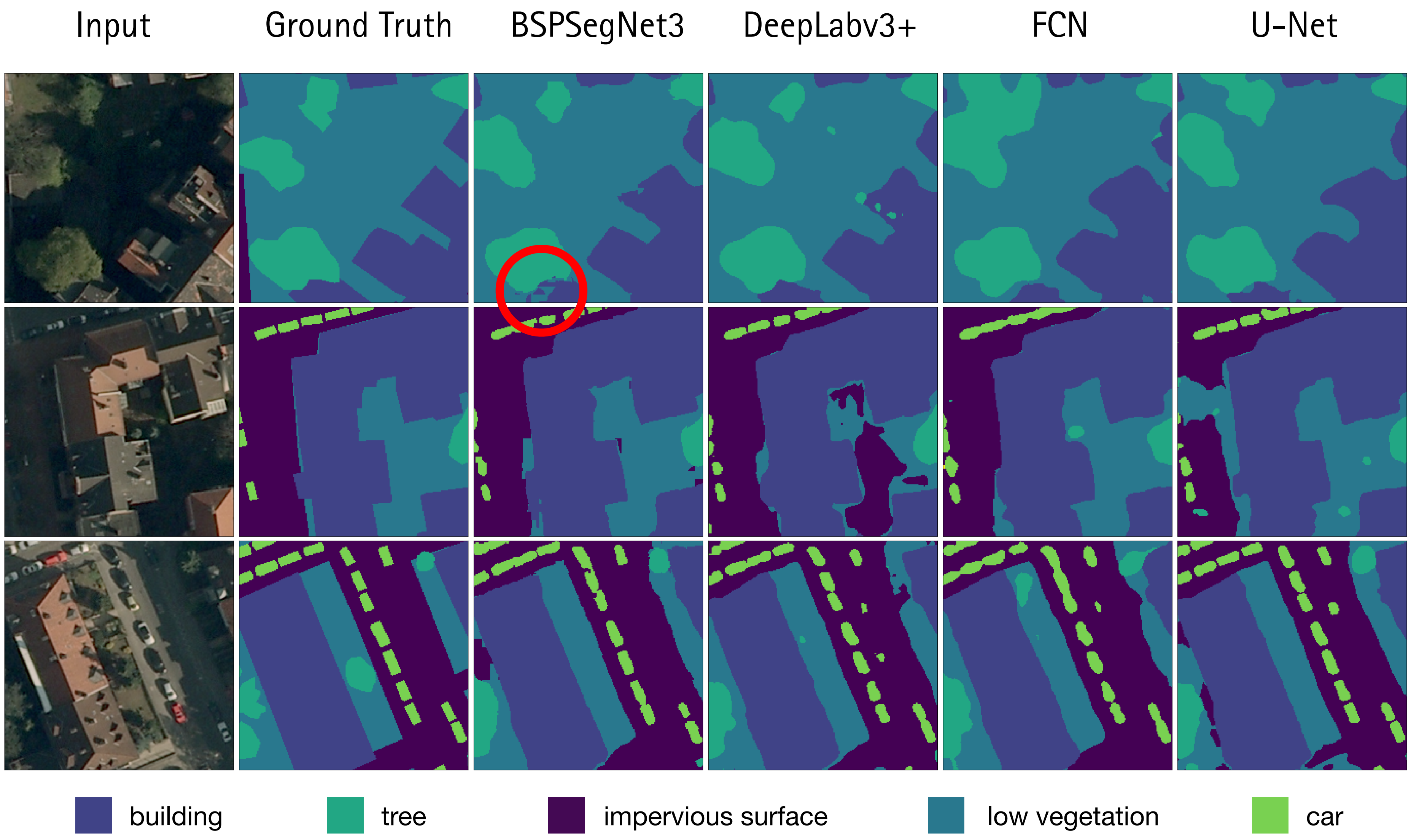
# Semantic Segmentation



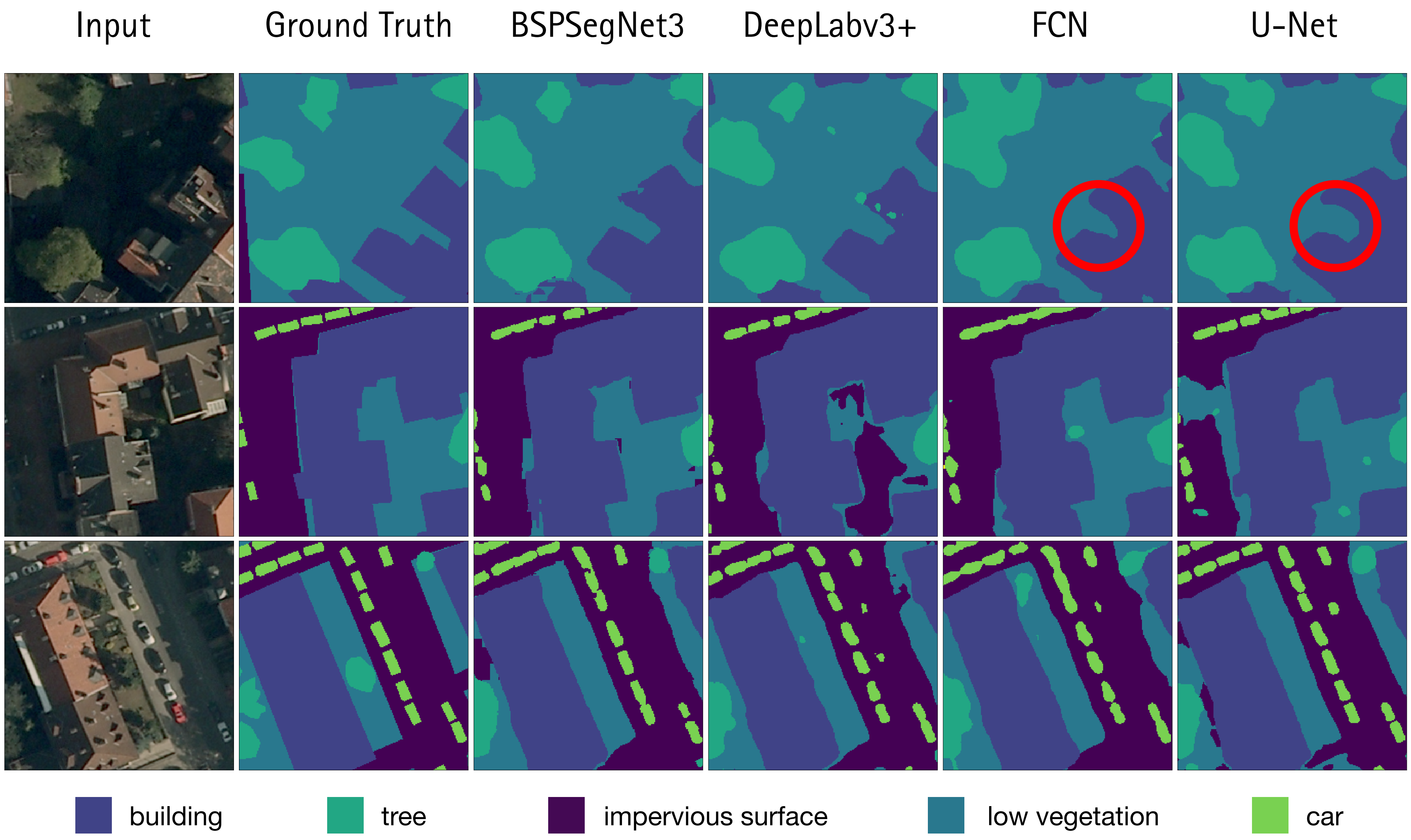
# Semantic Segmentation



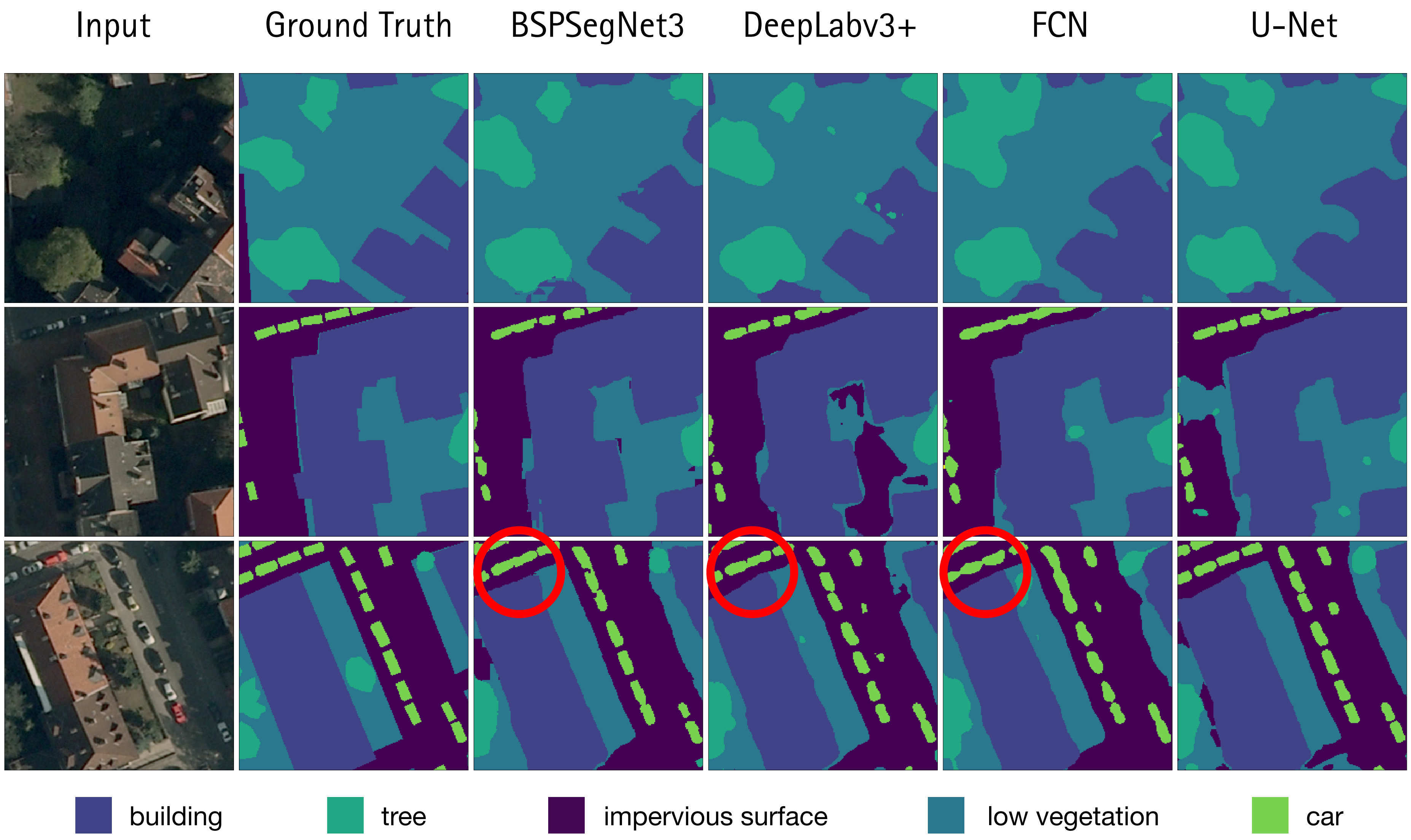
# Semantic Segmentation



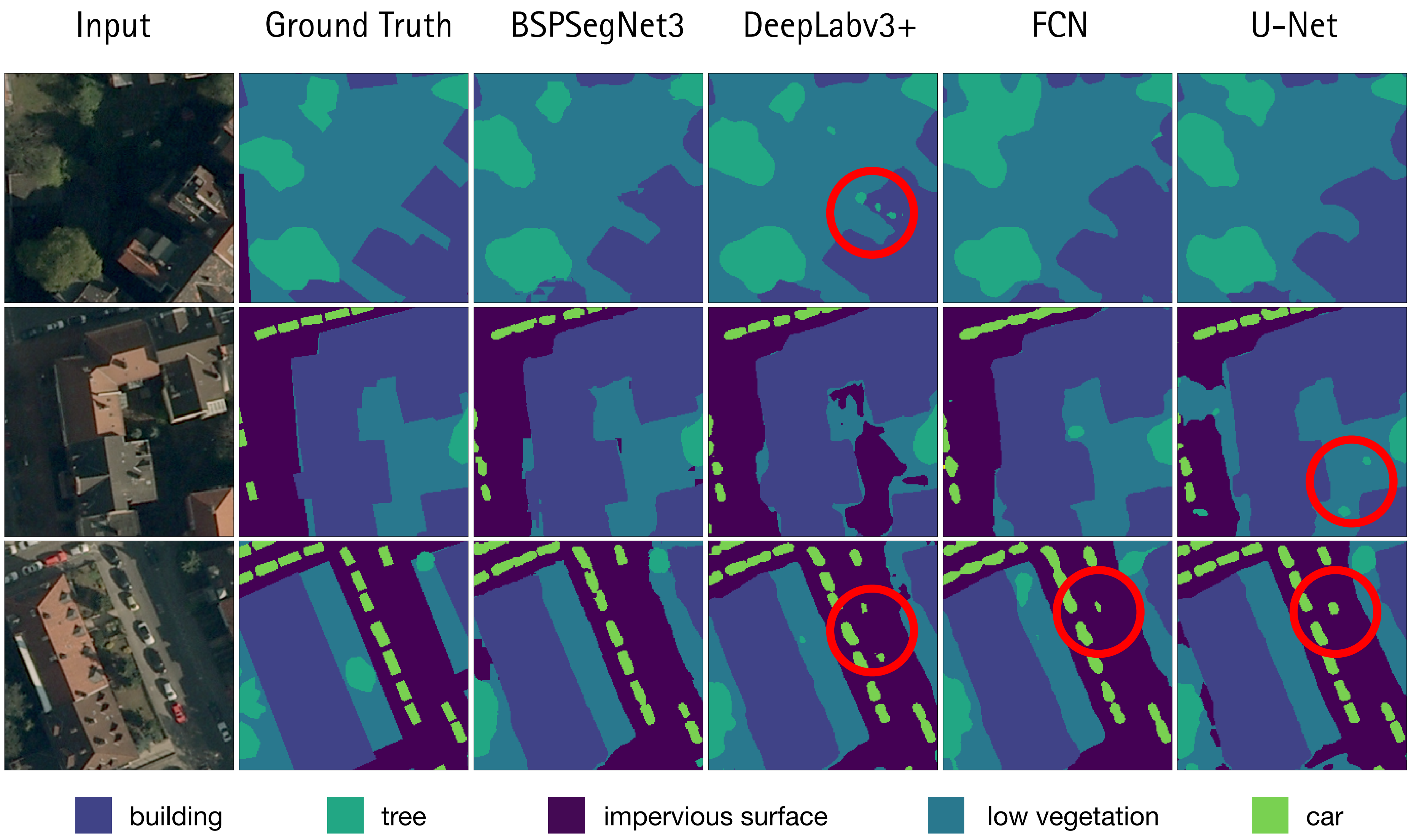
# Semantic Segmentation



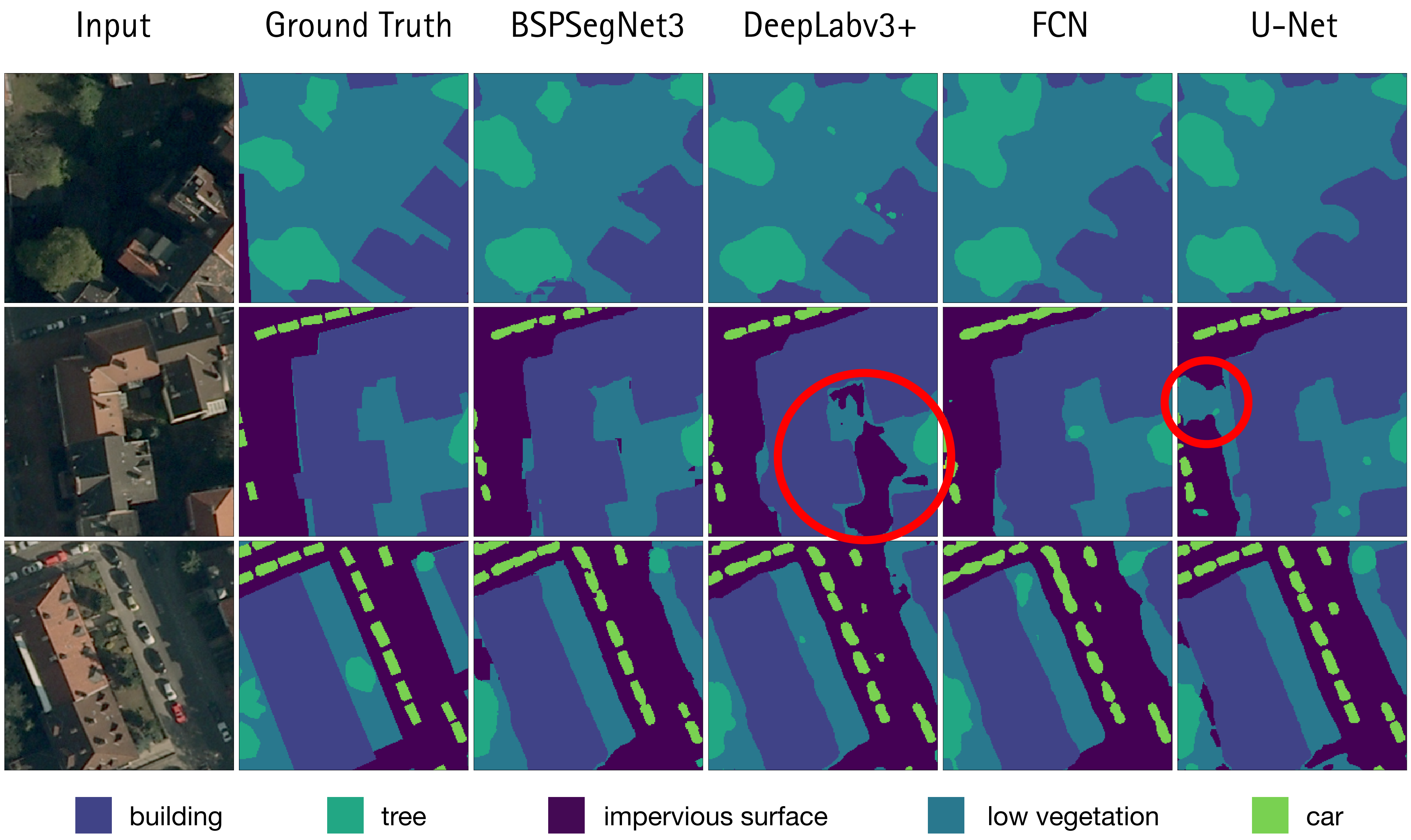
# Semantic Segmentation



# Semantic Segmentation



# Semantic Segmentation

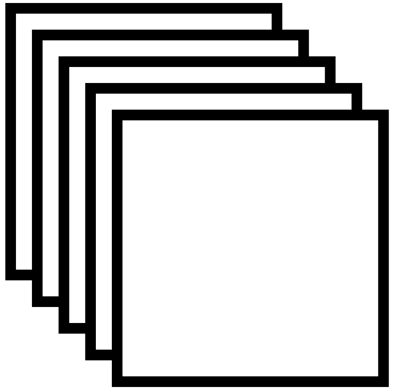


# Conclusion

- Semantic segmentation can be successfully performed by predicting binary space partitioning trees:
  - Our model inherently disentangles shape and texture features
  - It is end-to-end trainable by using differentiable BSP tree rendering
  - It can be used to map an existing segmentation to a BSP tree representation
  - It delivers state-of-the-art performance

- Future Research:
  - Use our model for domain adaptation
  - Expand our model to instance and panoptic segmentation

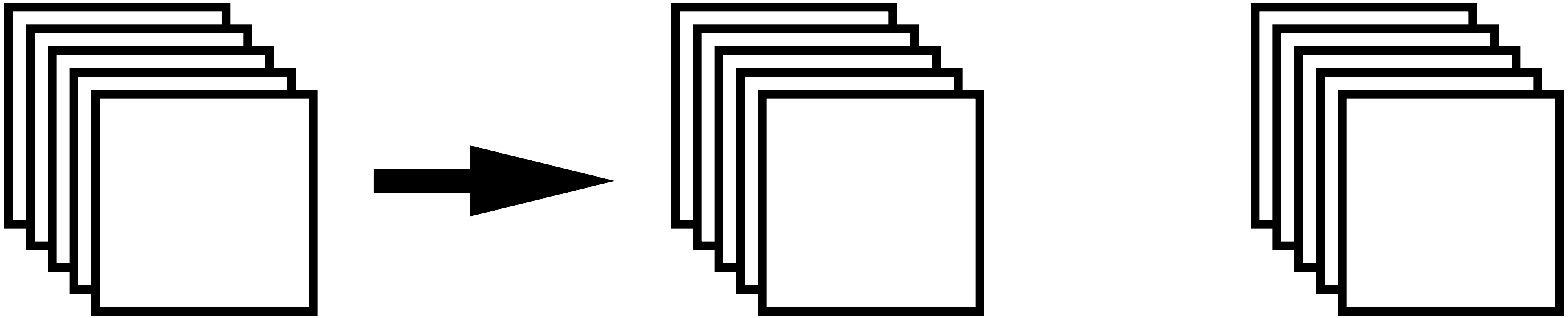
# Data Augmentation



16 images

# Data Augmentation

Leibniz Universität Hannover

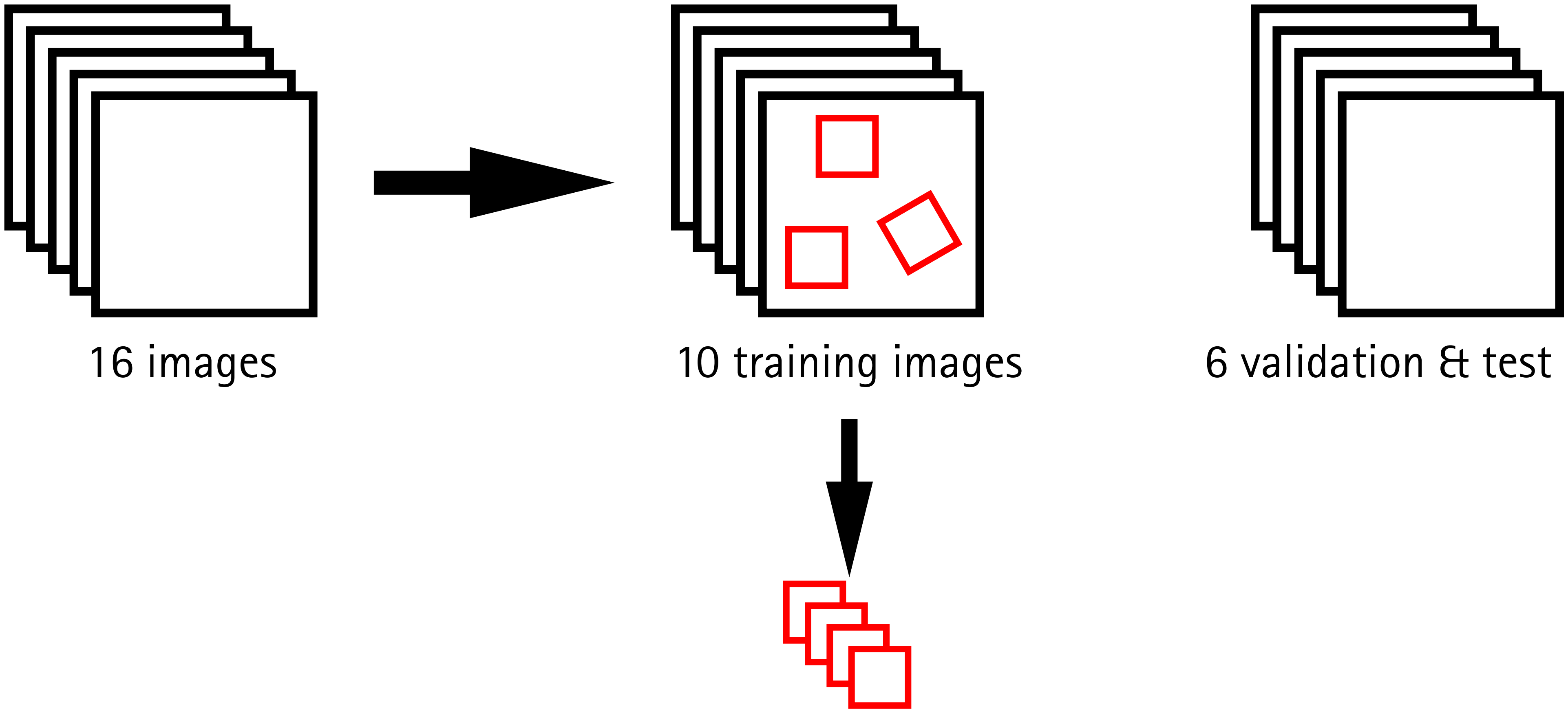


16 images

10 training images

6 validation & test images

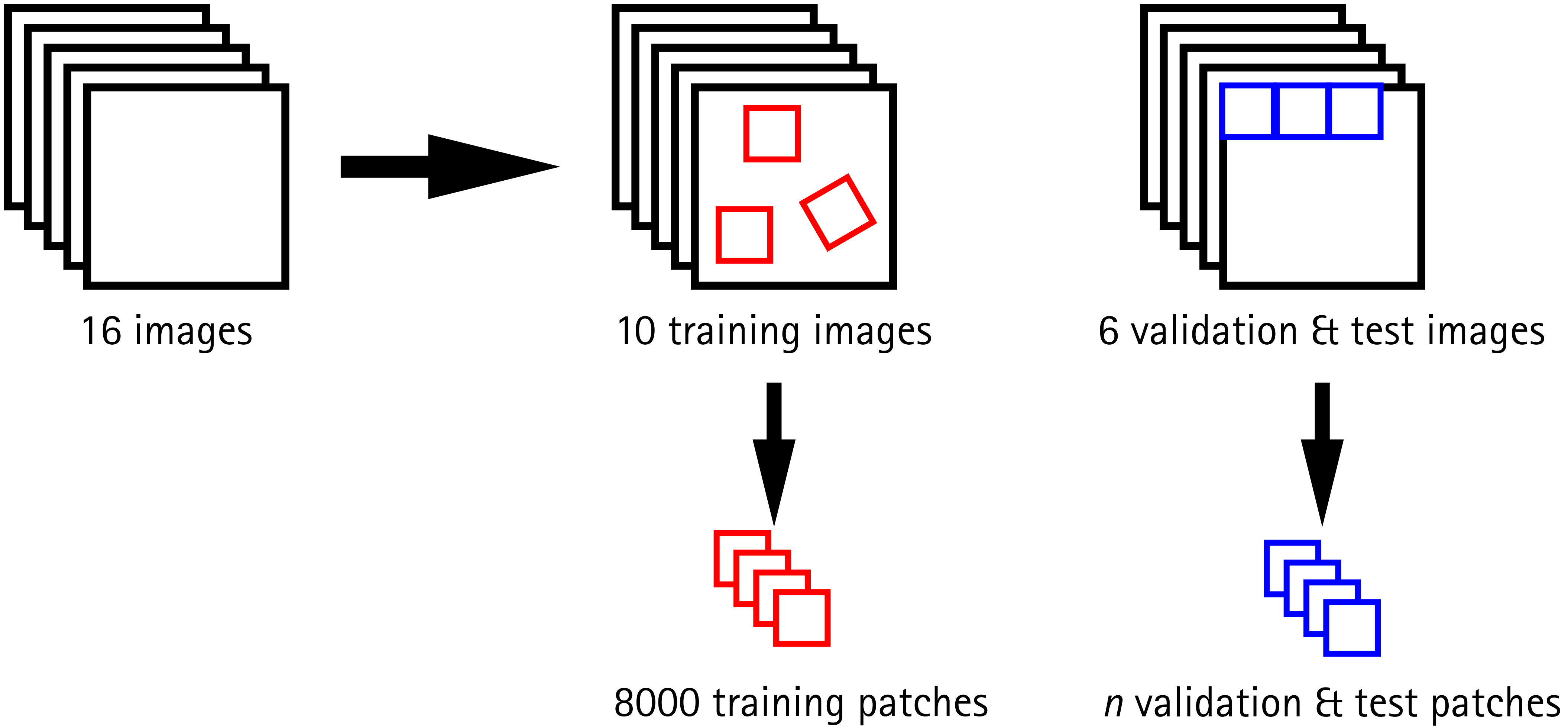
# Data Augmentation




8000 training patches

patch size = 224x224

# Data Augmentation




patch size = 224x224