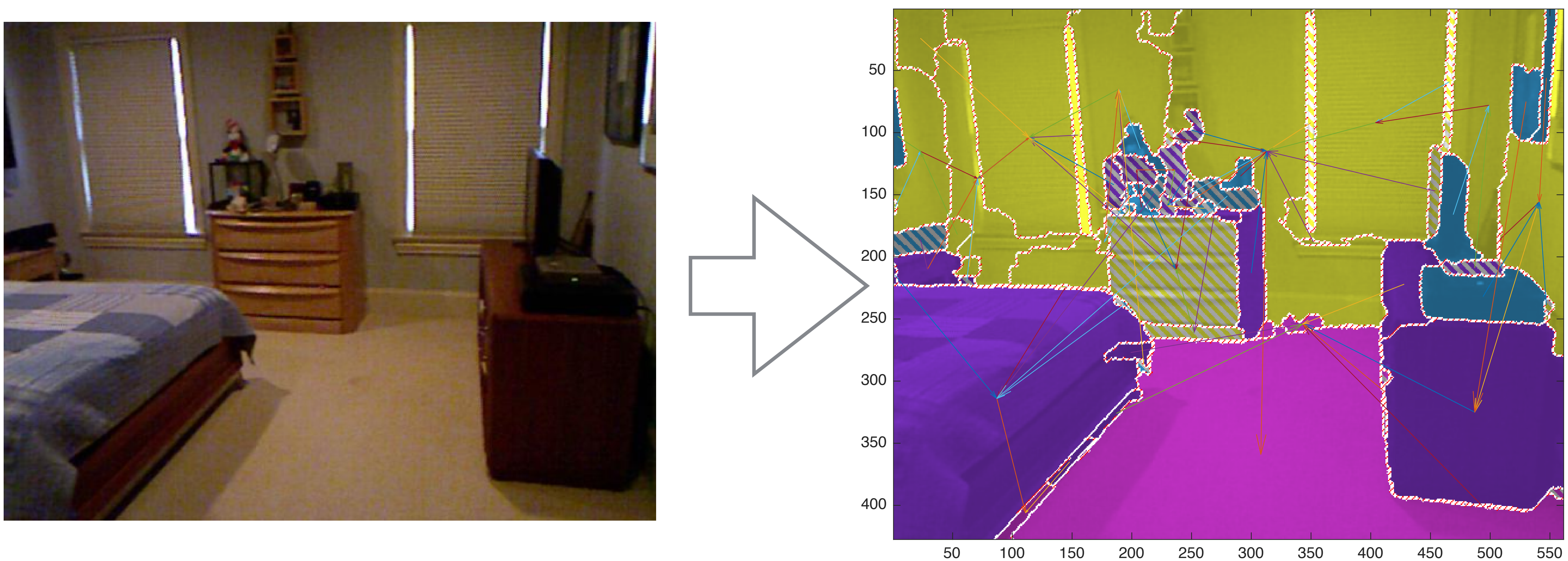
CS381V Experiment Presentation

Chun-Chen Kuo

The Paper

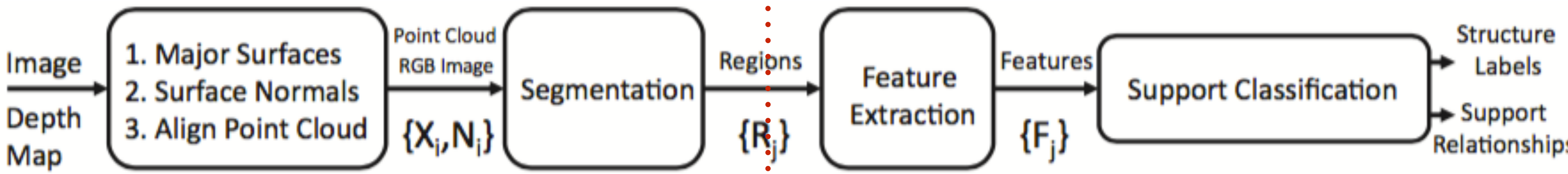
- Indoor Segmentation and Support Inference from RGBD Images. N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. ECCV 2012.



Pipeline

segmentation

support inference



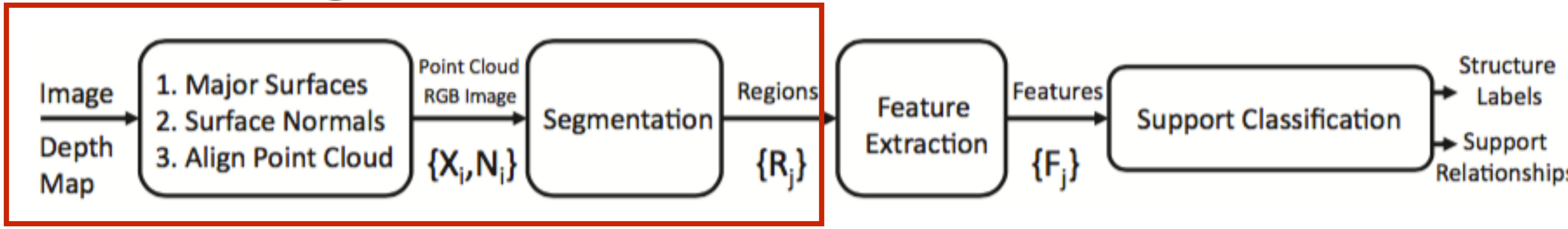
Outline

- Run the segmentation pipeline
- Experiment on the segmentation pipeline
- Run the support inference pipeline
- Address strength and weakness

Outline

- Run the segmentation pipeline
- Experiment on the segmentation pipeline
- Run the support inference pipeline
- Address strength and weakness

Segmentation Pipeline



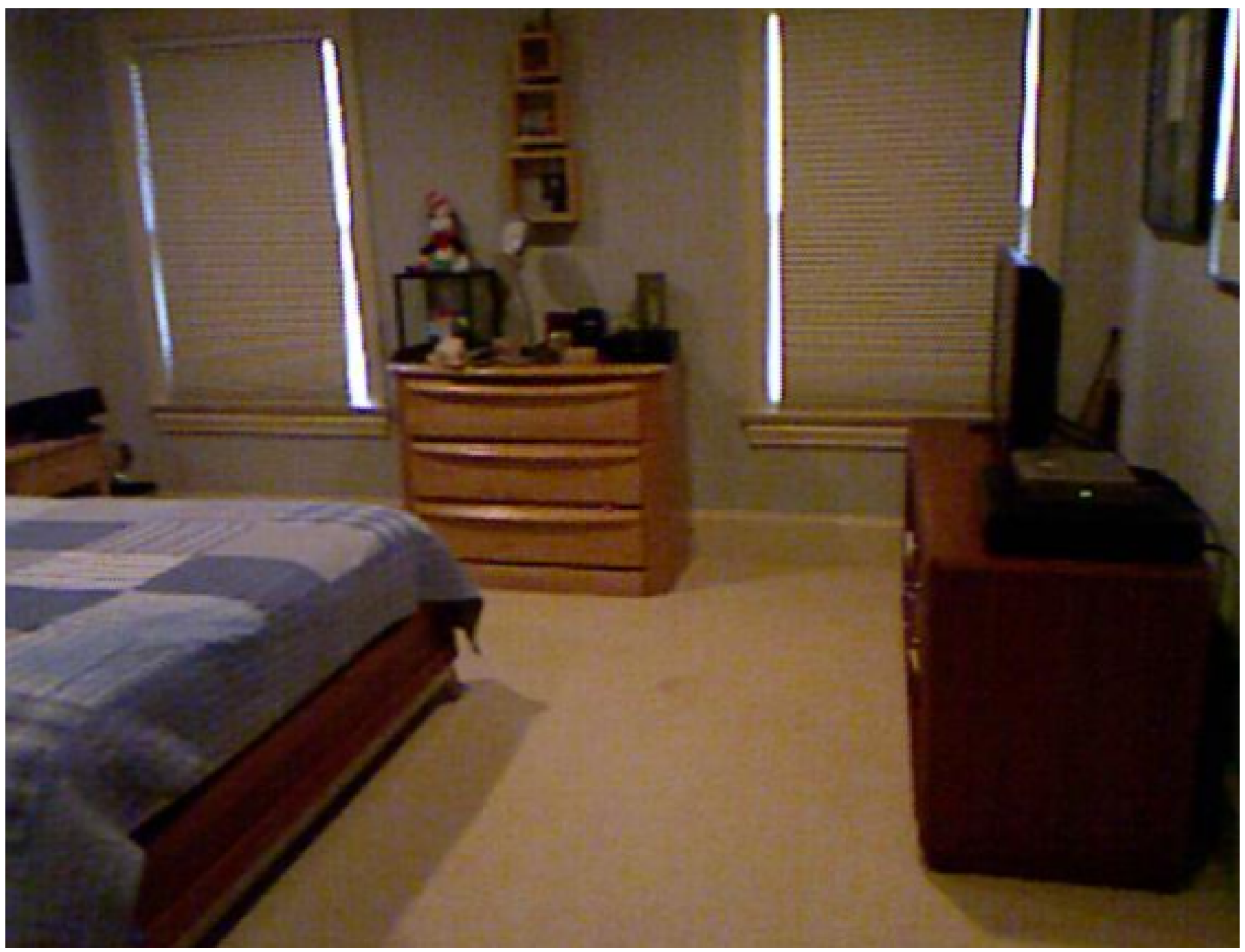
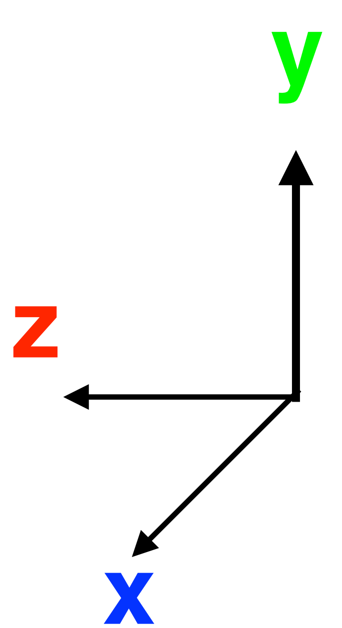
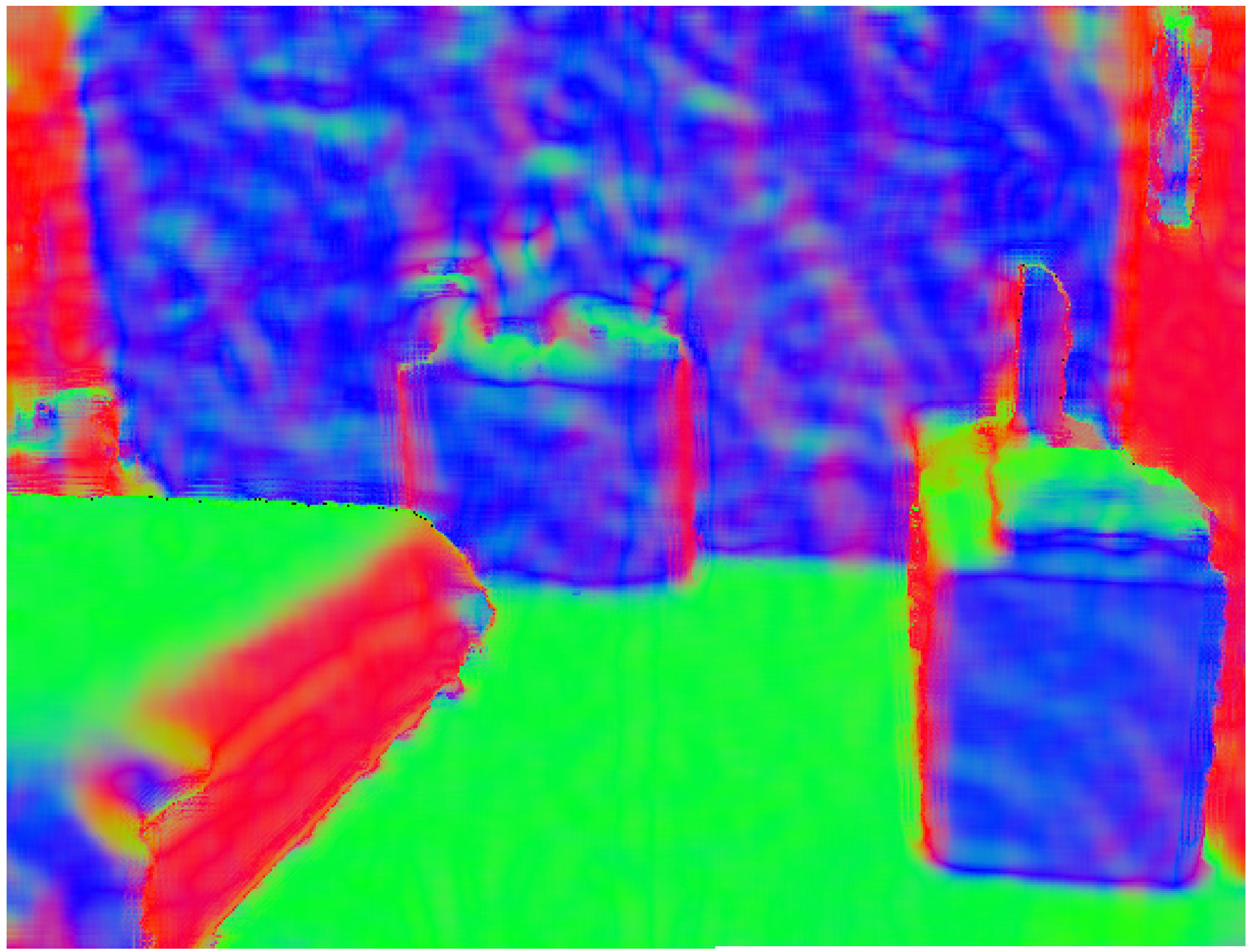


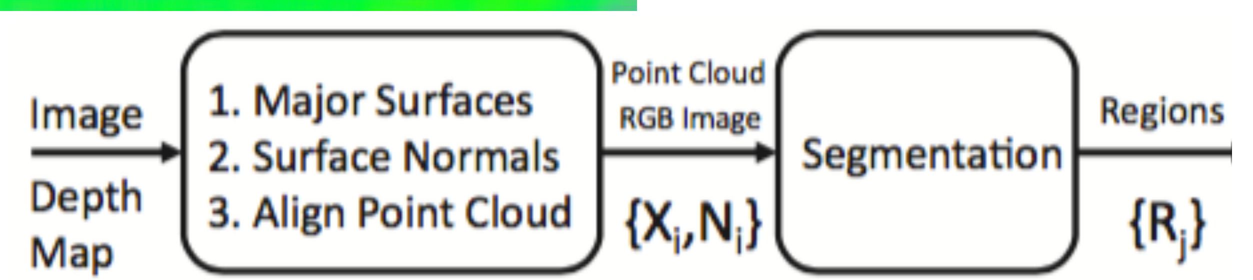
Image920, RGB

Depth Map

Compute Surface Normal







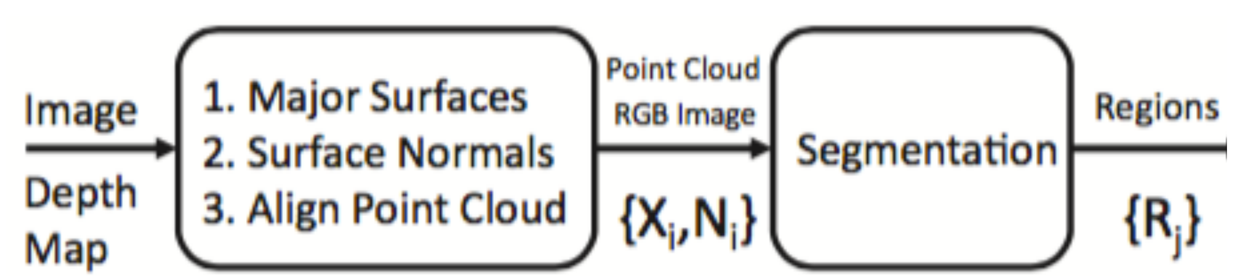
Align to room coordinates

$$S(v_1, v_2, v_3) = \sum_{j=1}^{3}\left[\frac{w_N}{N_N}\sum_{i}^{N_N}\exp\left(\frac{-(\mathbf{N}_i \cdot \mathbf{v}_j)^2}{\sigma^2}\right) + \frac{w_L}{N_L}\sum_{i}^{N_L}\exp\left(-\frac{(\mathbf{L}_i \cdot \mathbf{v}_j)^2}{\sigma^2}\right)\right]$$

$$R = [\mathbf{v}_X \quad \mathbf{v}_Y \quad \mathbf{v}_Z]$$

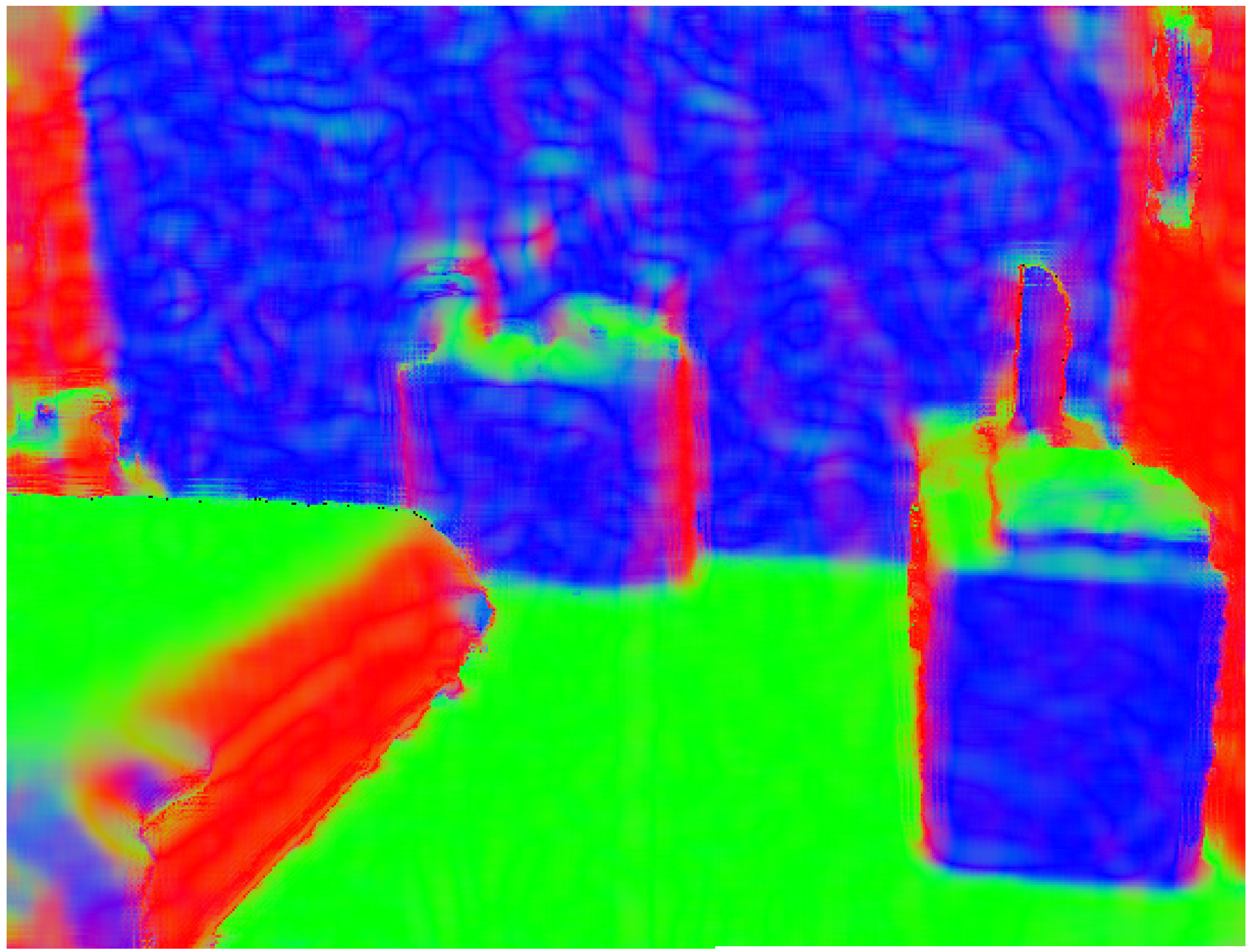
planeData.R

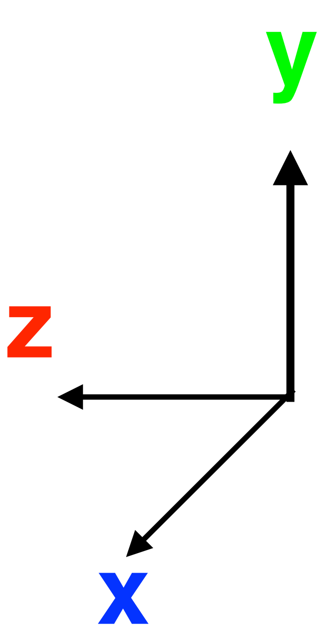
	1	2	3
1	0.9720	-0.0313	-0.2330
2	-0.0256	0.9711	-0.2373
3	0.2337	0.2366	0.9431

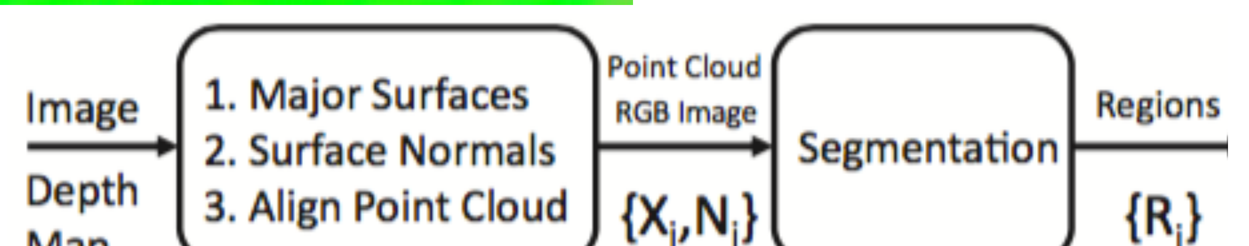


Aligned Surface Normal

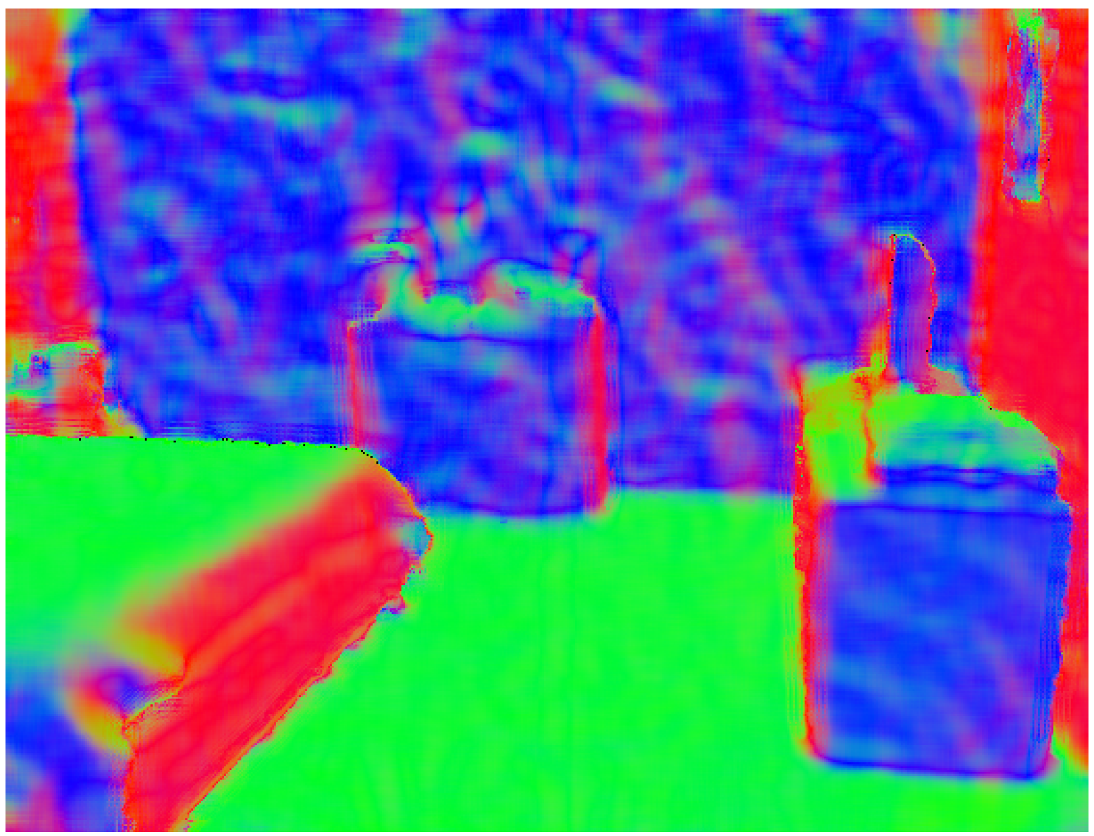
```
visNormals = reshape(planeData.normals * planeData.R', [sz(1) sz(2) 3]);
```

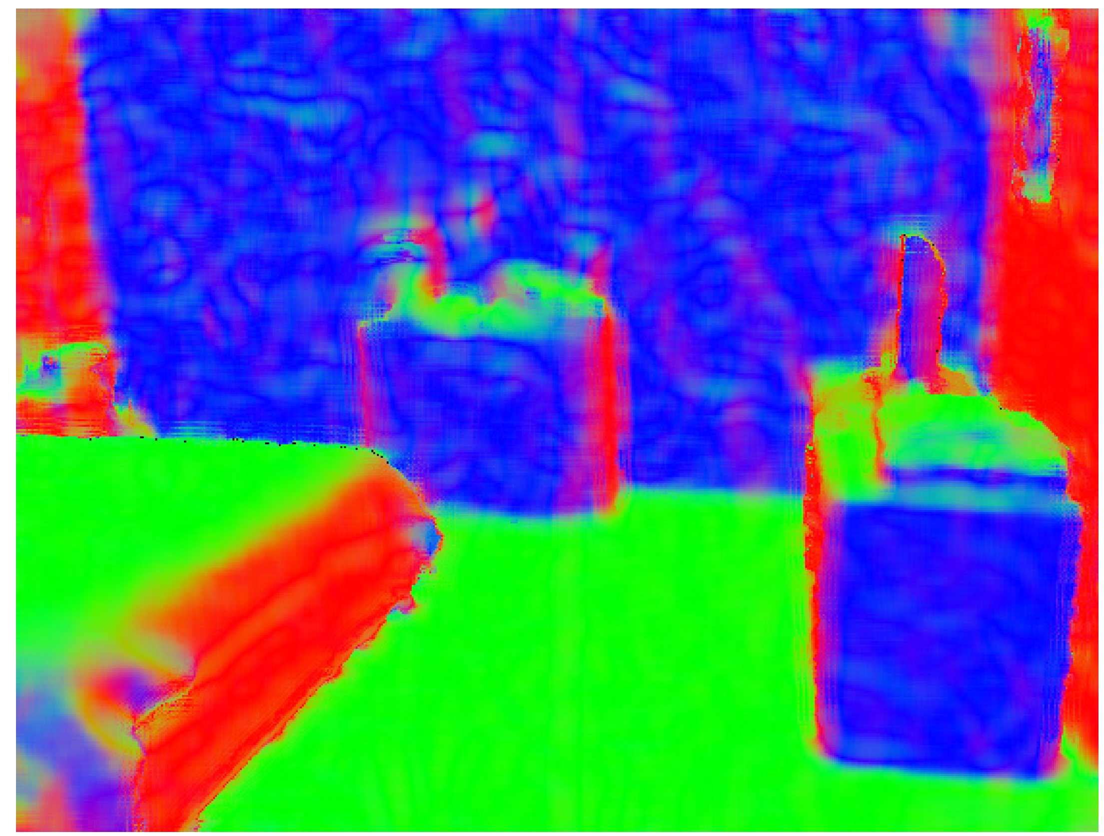


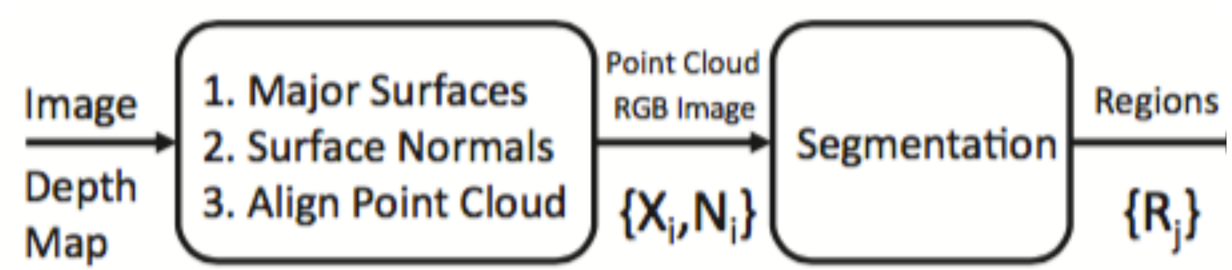




After Alignment







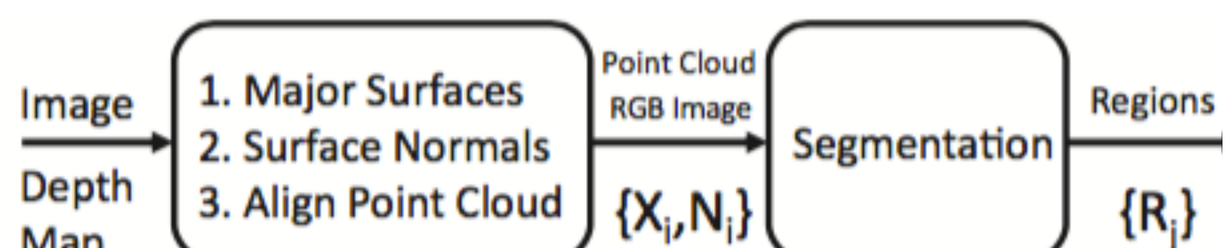
Find Major Planes by RANSAC

planes

9x4 double

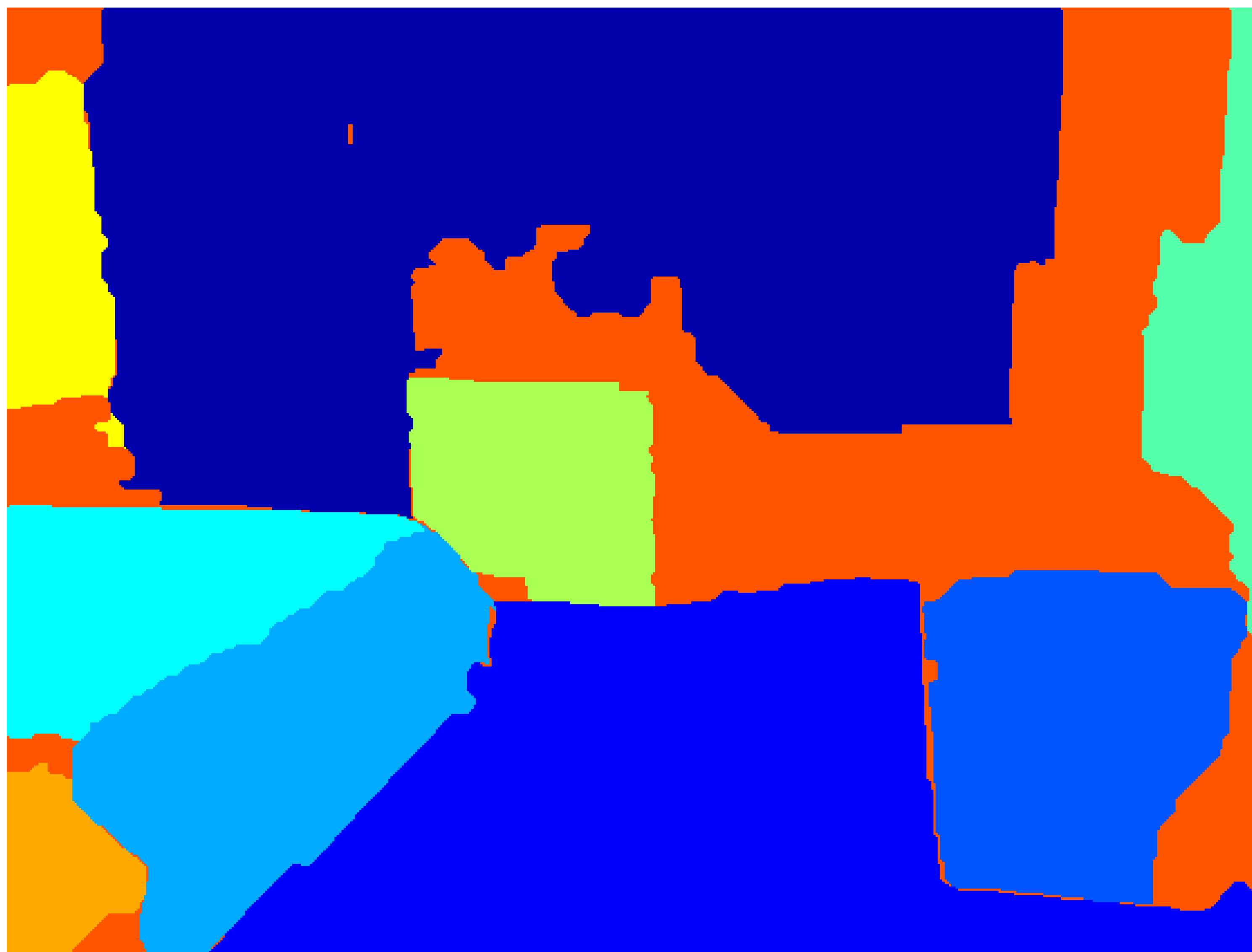
	1	2	3	4
1	0.2454	0.1918	0.9503	−4.2327
2	0.0384	−0.9718	0.2327	−1.4294
3	0.1631	0.2710	0.9487	−2.1915
4	0.9681	−0.0396	−0.2475	1.2939
5	0.0164	−0.9723	0.2333	−0.8234
6	0.9692	−0.0050	−0.2461	−0.6685
7	−0.2857	−0.2504	−0.9250	3.7373
8	−0.9197	0.0027	0.3925	−4.0064
9	0.1991	0.2986	−0.9334	2.0293

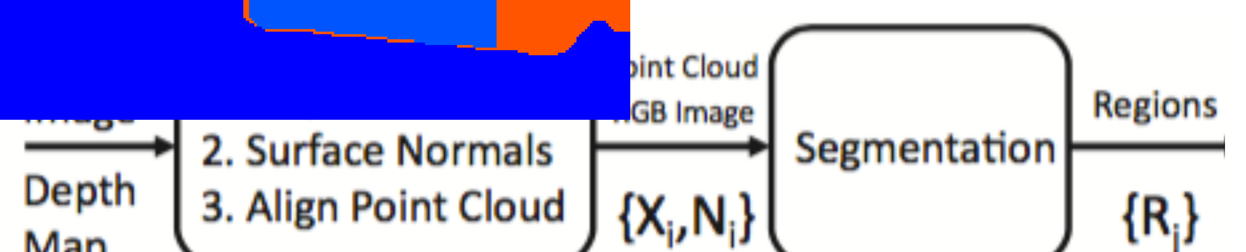
$0.2454x+0.1918y+0.9503z-4.2327$



Reassign Pixels to Planes

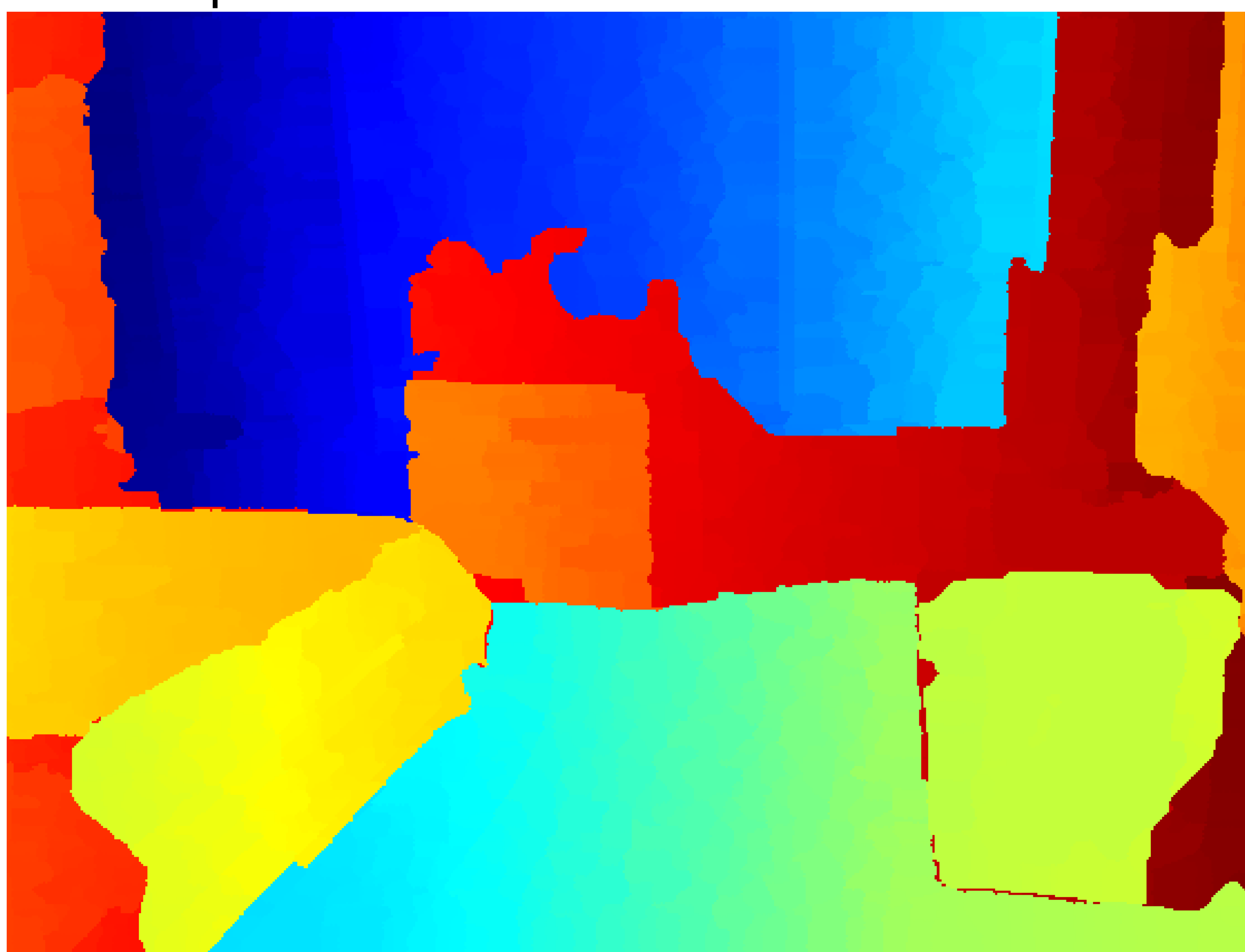
$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$





Watershed Segmentation

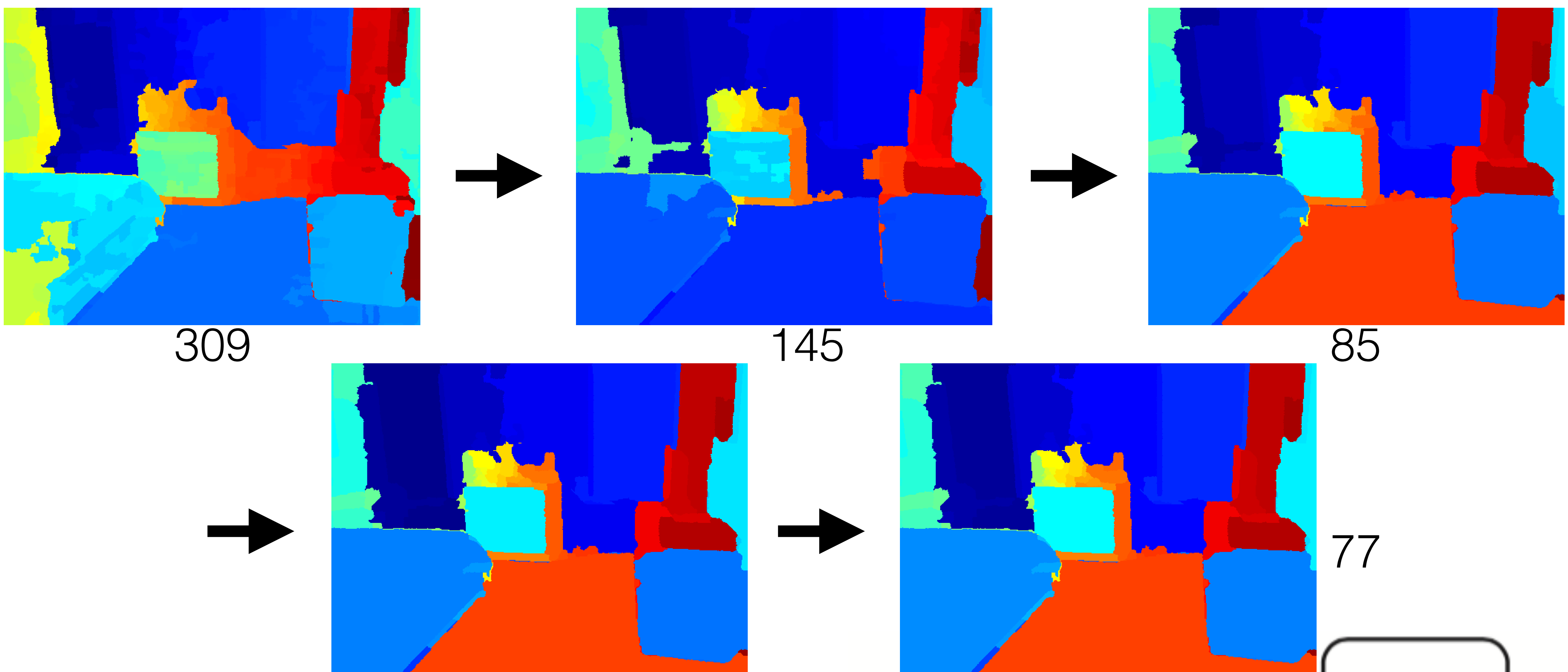
- Force the over-segmentation to be consistent with the previous planes

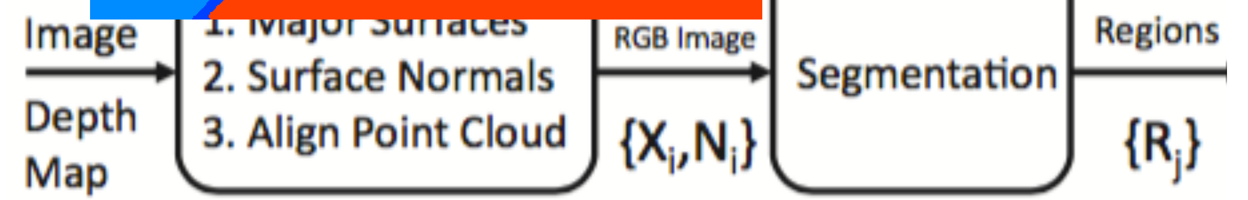


1614

Hierarchical Grouping

- Bottom-up grouping by boundary classifier (Logistic regression AdaBoost) $\mathrm{P}(y_i \neq y_j | \mathbf{x}^s_{ij})$

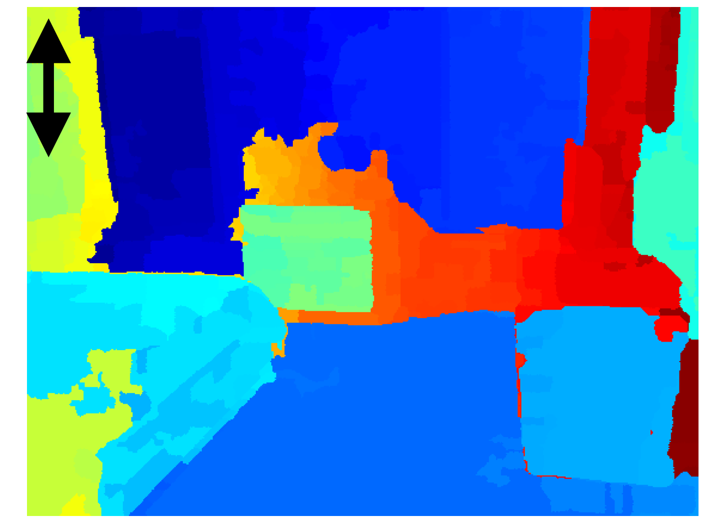




AdaBoost Decision Tree

$\mathrm{P}(y_i \neq y_j | \mathbf{x}_{ij}^s)$

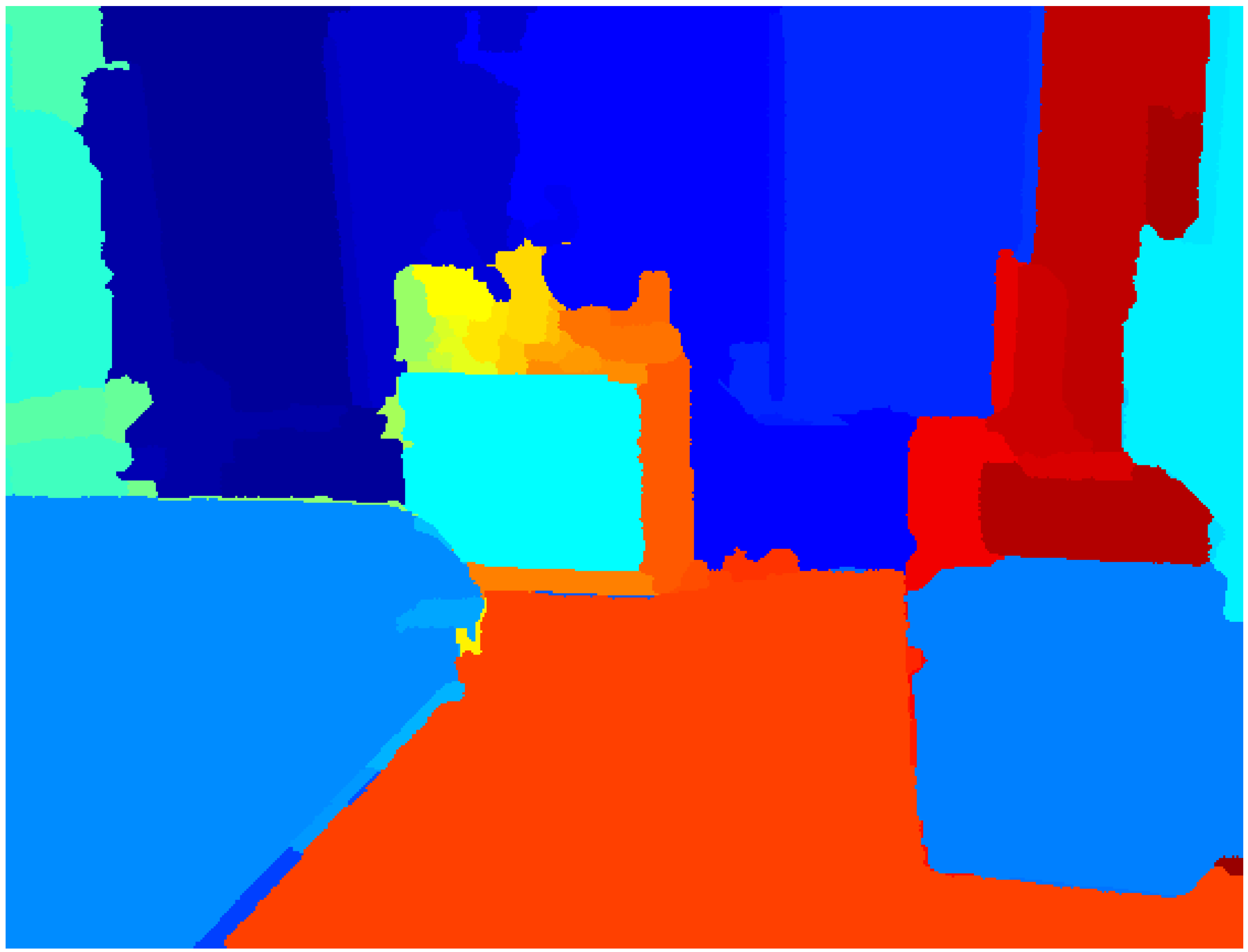
merge?



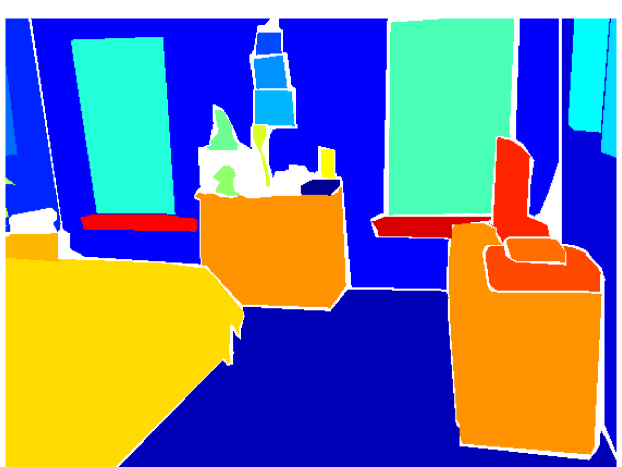
- Reweigh misclassified regions
- Optimize new tree with reweighed regions
- Score the tree
- Weighted sum over all trees optimized in each iteration

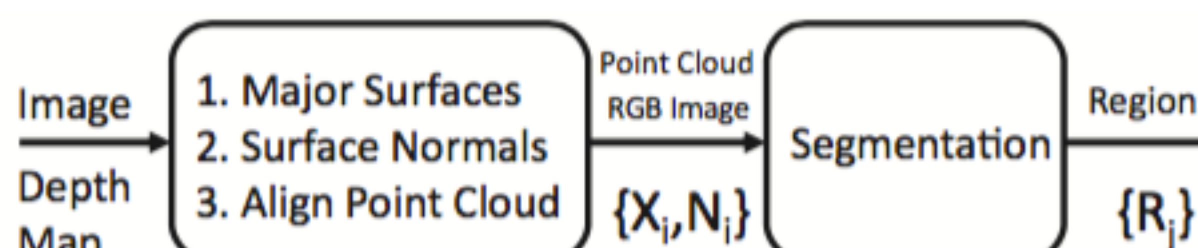
$$T(x_i) = \sum_m \alpha_m T_m(x_i)$$

Final Regions



Ground truth





Outline

- Run the segmentation pipeline
- Experiment on the segmentation pipeline
- Run the support inference pipeline
- Address strength and weakness

Experiment on Segmentation Pipeline

- NYU Depth Dataset V2
- Images 909~1200
- Assign pixels to major planes
- AdaBoost decision tree as boundary classifier

Hypothesis

- The trade-off between matching to 3D values, normals, and gradient smoothing

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

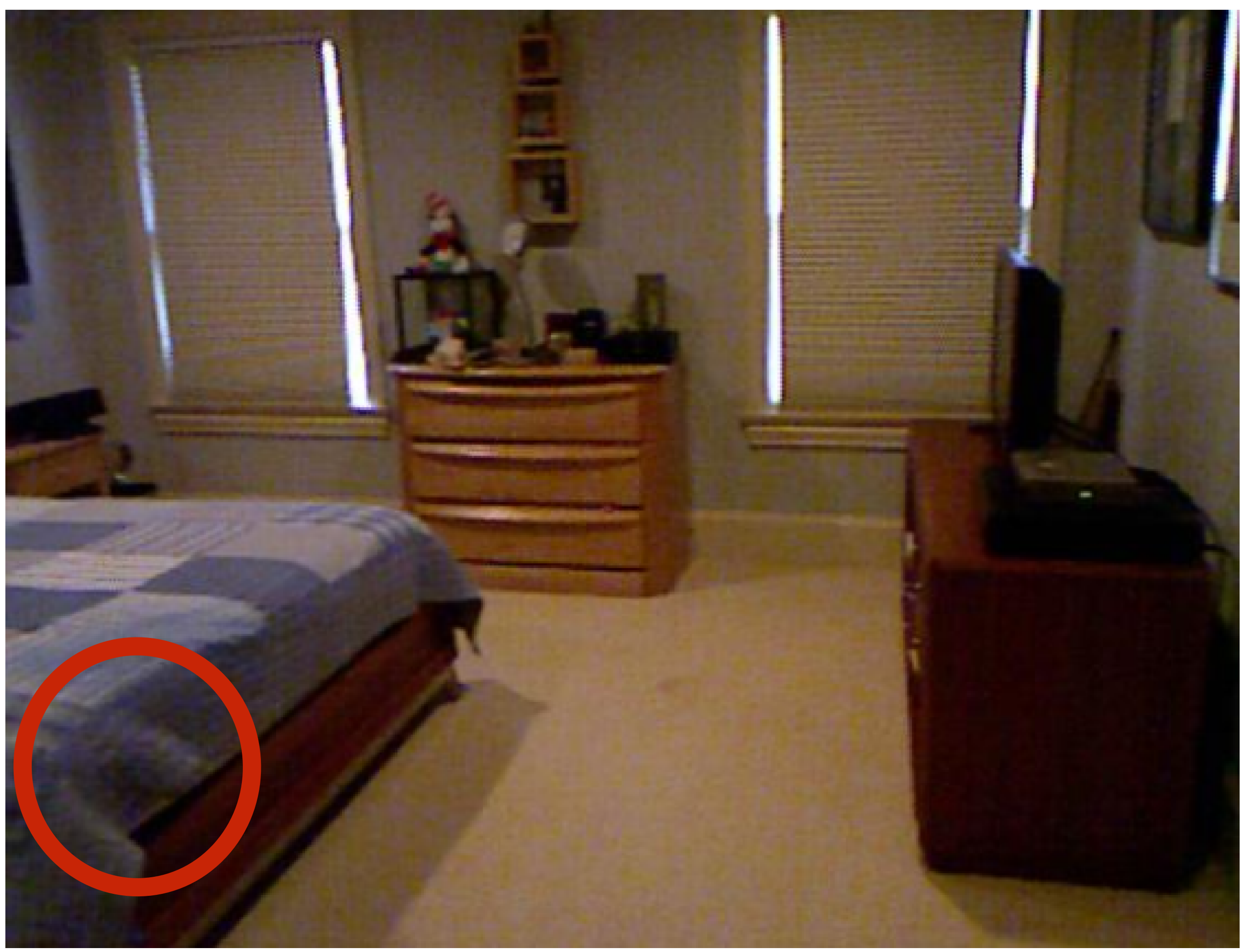
- If alpha is small, neighbor pixels with similar RGB tend to be assigned to a same plane

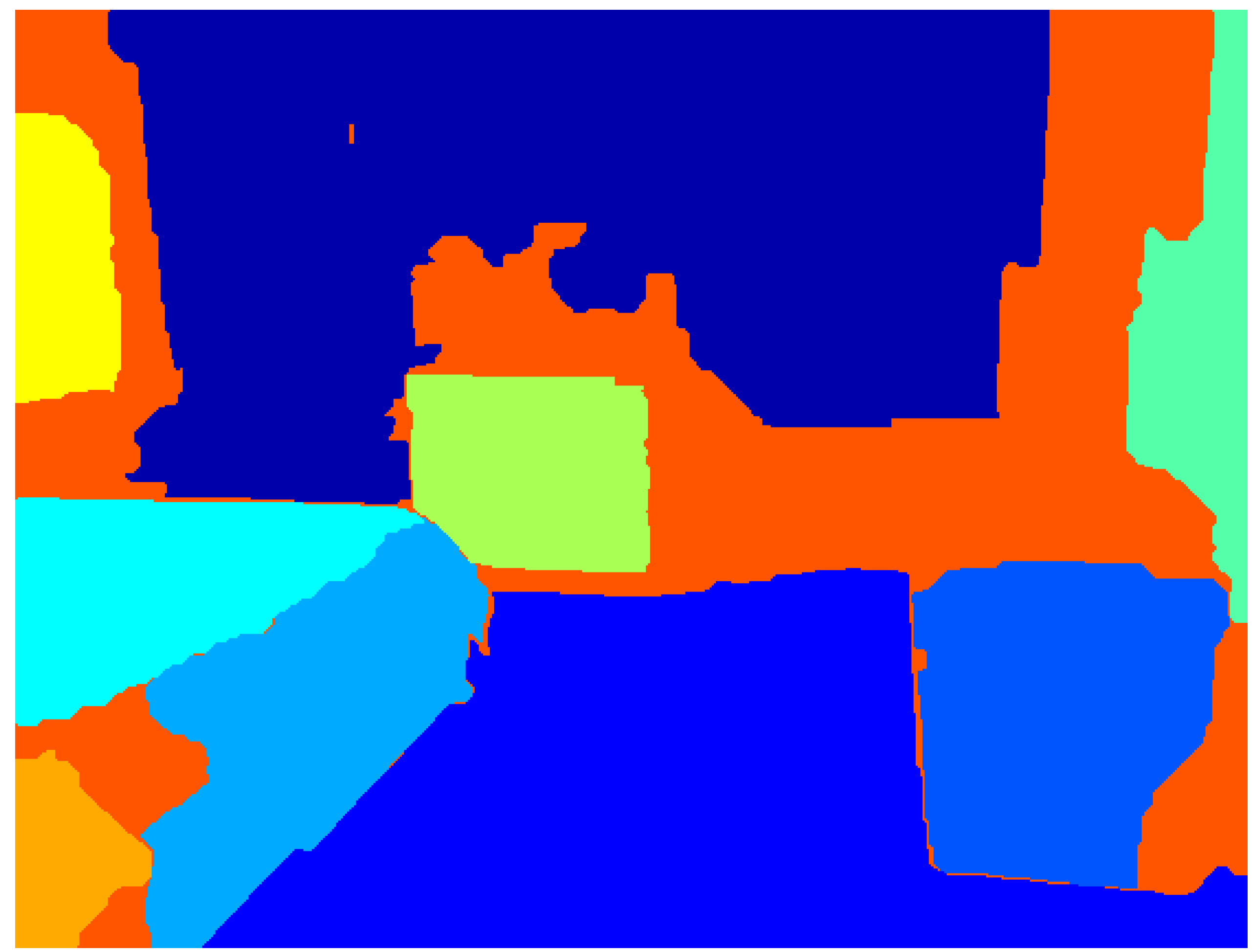
- If alpha is large, match pixels to planes based on 3D points and normals, regardless gradient smoothing

Result of Plane Labeling

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

alpha=0

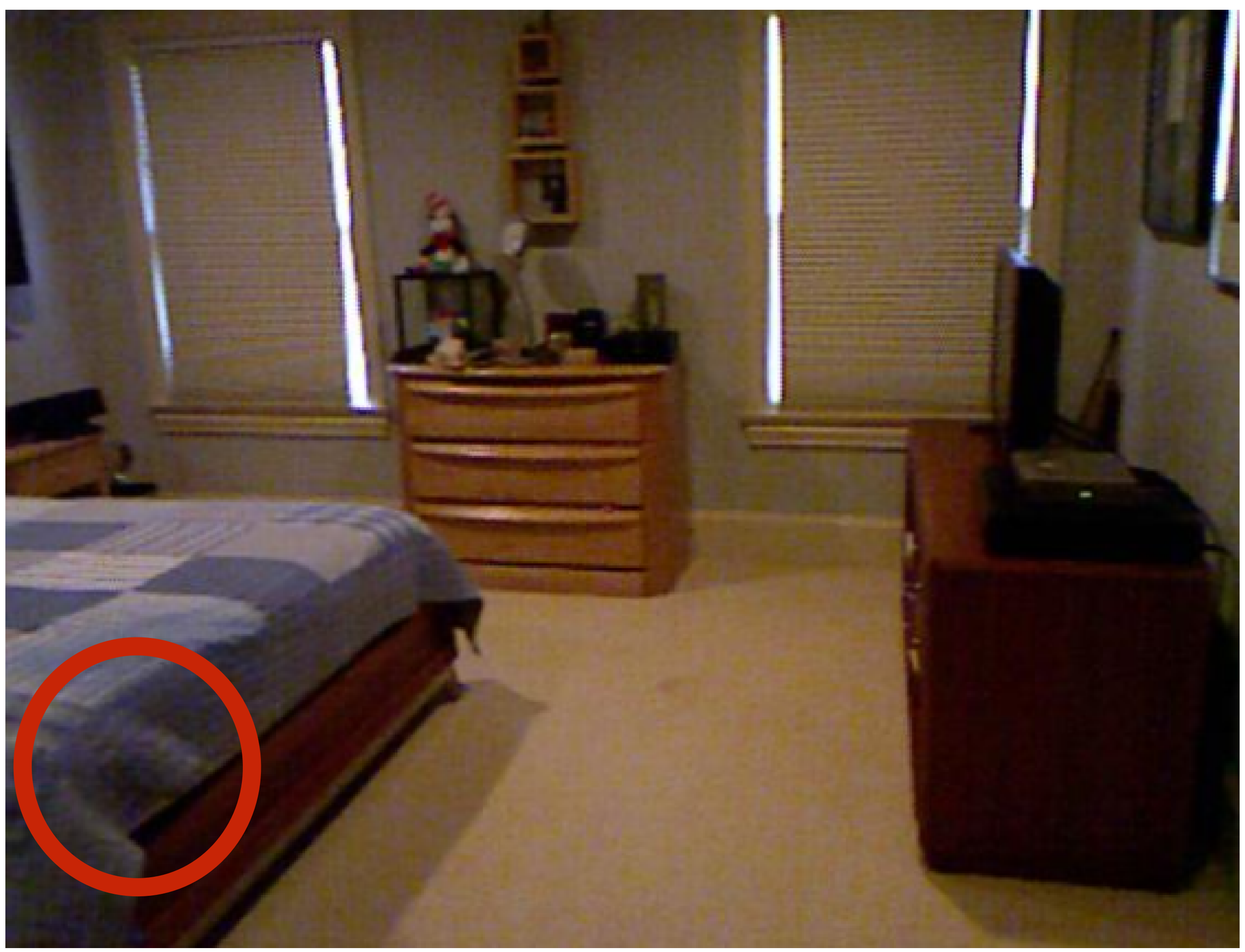


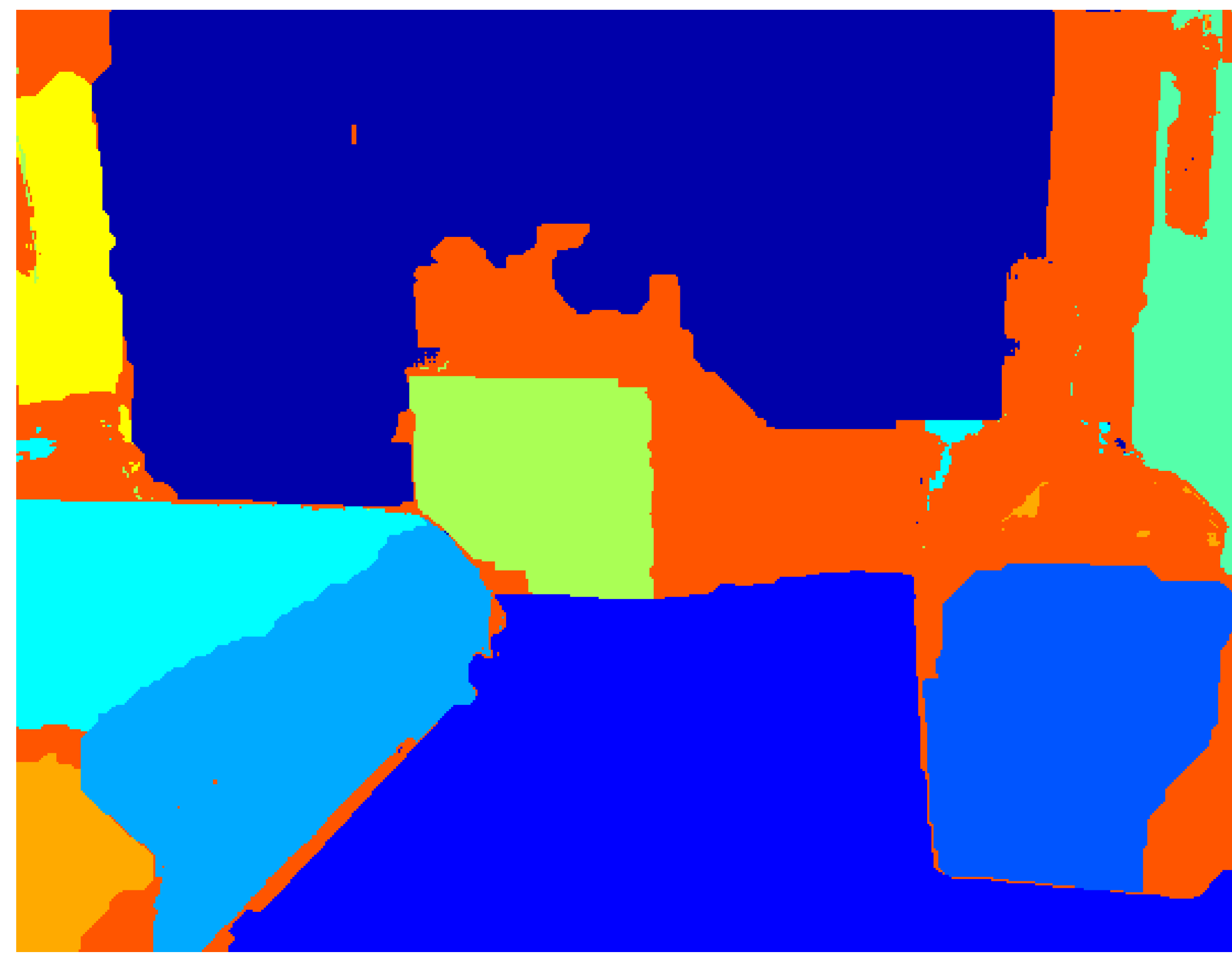


Result of Plane Labeling

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

alpha=2500

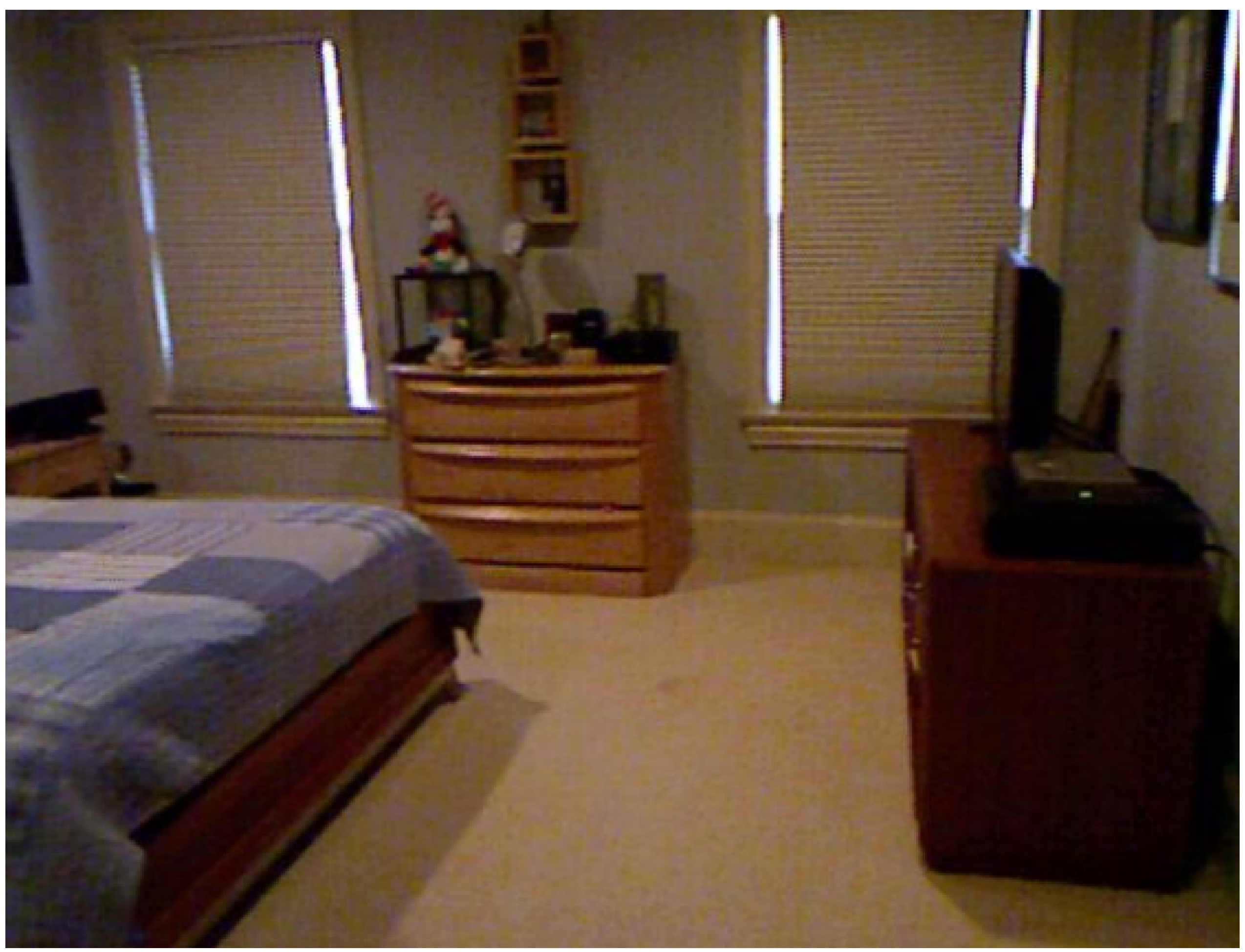


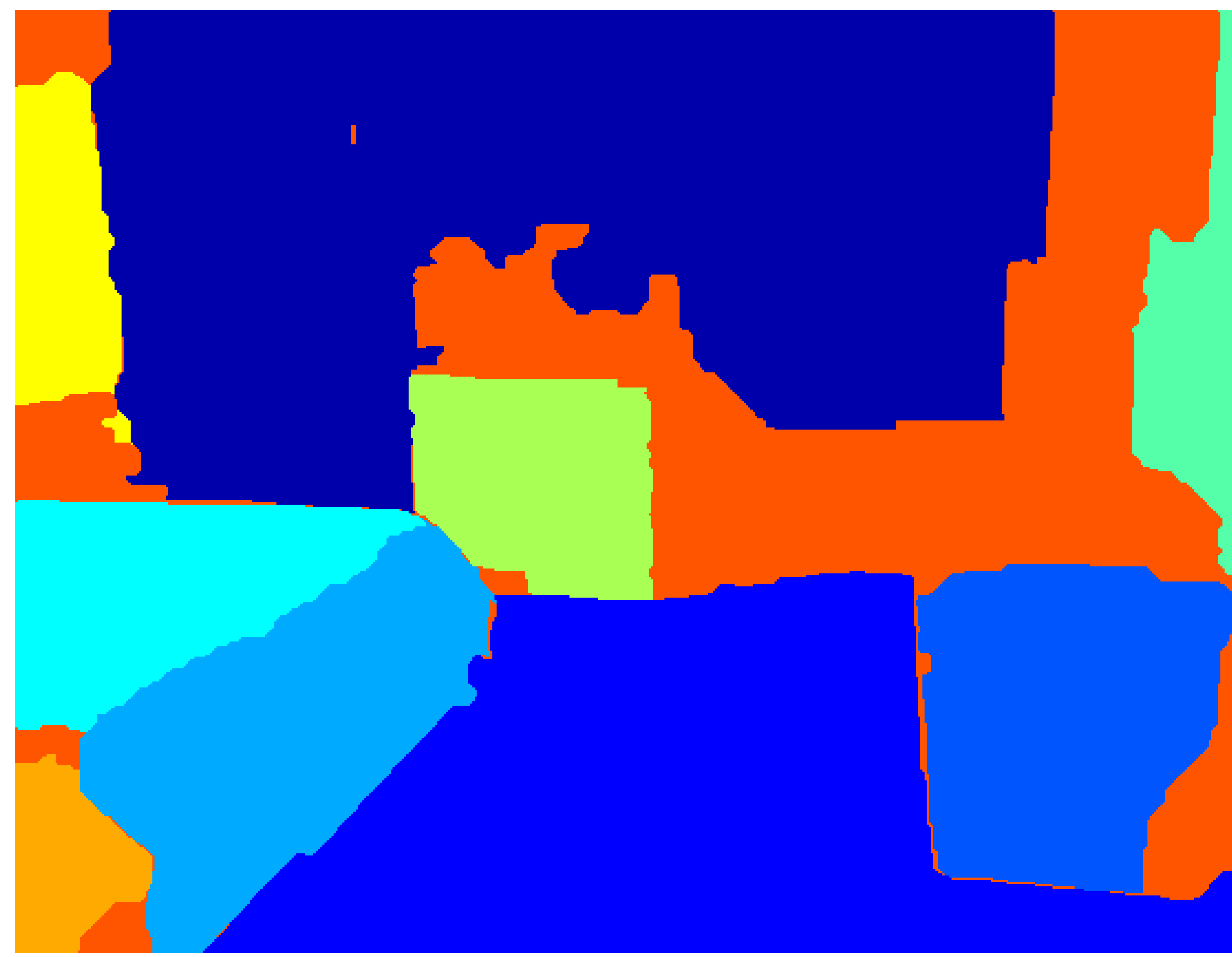


Result of Plane Labeling

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

alpha=0.25

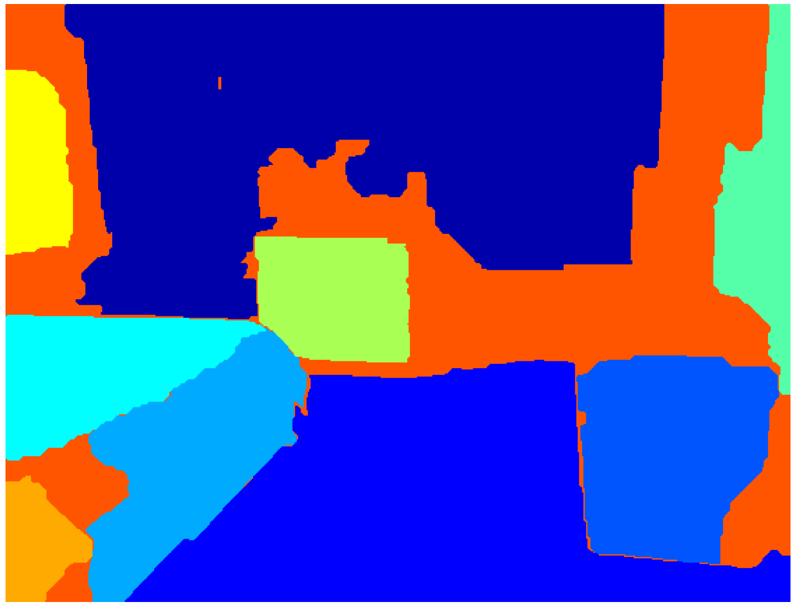




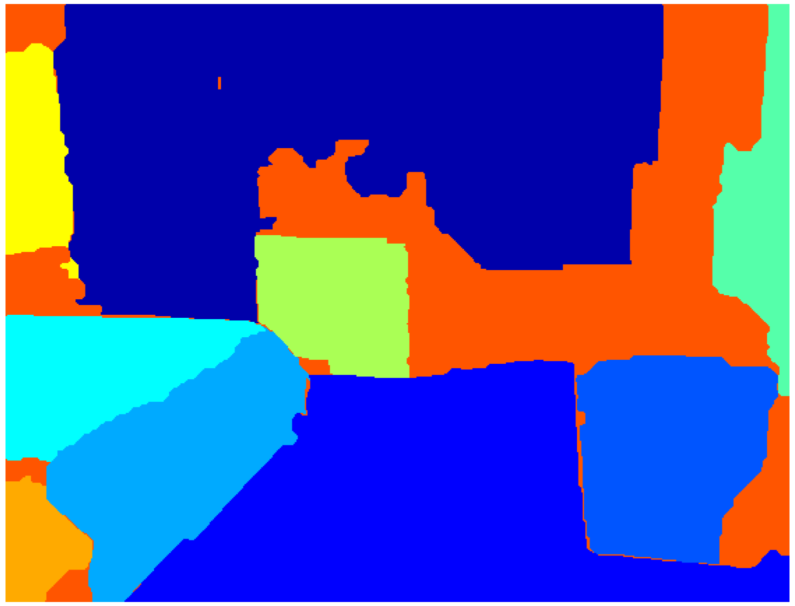
Result of Plane Labeling

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

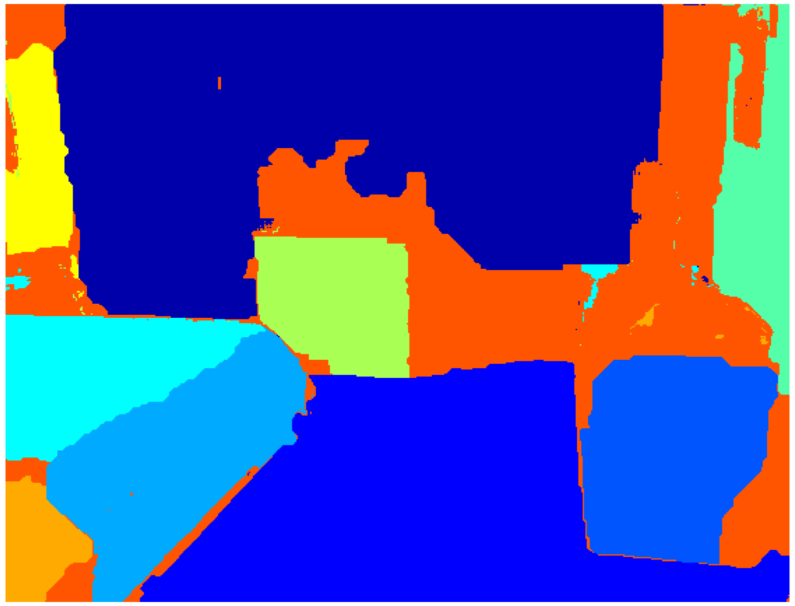
alpha=0



alpha=0.25



alpha=2500



Segmentation Score

alpha=0.25e-12

```
========================================
               Results:
========================================
Stage 1:
Weighted:    Train=41.8   Test=40.1
Unweighted: Train=38.9   Test=38.5

Stage 2:
Weighted:    Train=59.8   Test=56.4
Unweighted: Train=49.9   Test=48.0

Stage 3:
Weighted:    Train=66.7   Test=61.9
Unweighted: Train=53.5   Test=50.2

Stage 4:
Weighted:    Train=67.8   Test=62.7
Unweighted: Train=53.8   Test=50.1

Stage 5:
Weighted:    Train=68.4   Test=63.0
Unweighted: Train=54.0   Test=50.1
```

alpha=0.25

```
========================================
               Results:
========================================
Stage 1:
Weighted:    Train=41.3   Test=39.7
Unweighted: Train=38.9   Test=38.7

Stage 2:
Weighted:    Train=60.3   Test=57.4
Unweighted: Train=50.2   Test=49.1

Stage 3:
Weighted:    Train=67.4   Test=62.3
Unweighted: Train=53.5   Test=50.1

Stage 4:
Weighted:    Train=68.3   Test=62.8
Unweighted: Train=53.9   Test=50.0

Stage 5:
Weighted:    Train=68.8   Test=63.2
Unweighted: Train=54.1   Test=50.0
```

alpha=2.5

```
========================================
               Results:
========================================
Stage 1:
Weighted:    Train=41.3   Test=39.5
Unweighted: Train=37.9   Test=37.7

Stage 2:
Weighted:    Train=59.5   Test=57.0
Unweighted: Train=48.8   Test=47.9

Stage 3:
Weighted:    Train=66.3   Test=62.2
Unweighted: Train=52.7   Test=49.5

Stage 4:
Weighted:    Train=67.6   Test=63.2
Unweighted: Train=53.2   Test=49.4

Stage 5:
Weighted:    Train=68.0   Test=63.5
Unweighted: Train=53.1   Test=49.3
```

$$E(\mathbf{data}, \mathbf{y}) = \alpha_i \left[\sum_i f_{3d}(\mathbf{X}_i, y_i) + f_{norm}(\mathbf{N}_i, y_i) \right] + \sum_{i,j \in \mathcal{N}_8} f_{pair}(y_i, y_j, \mathbf{I})$$

Hypothesis

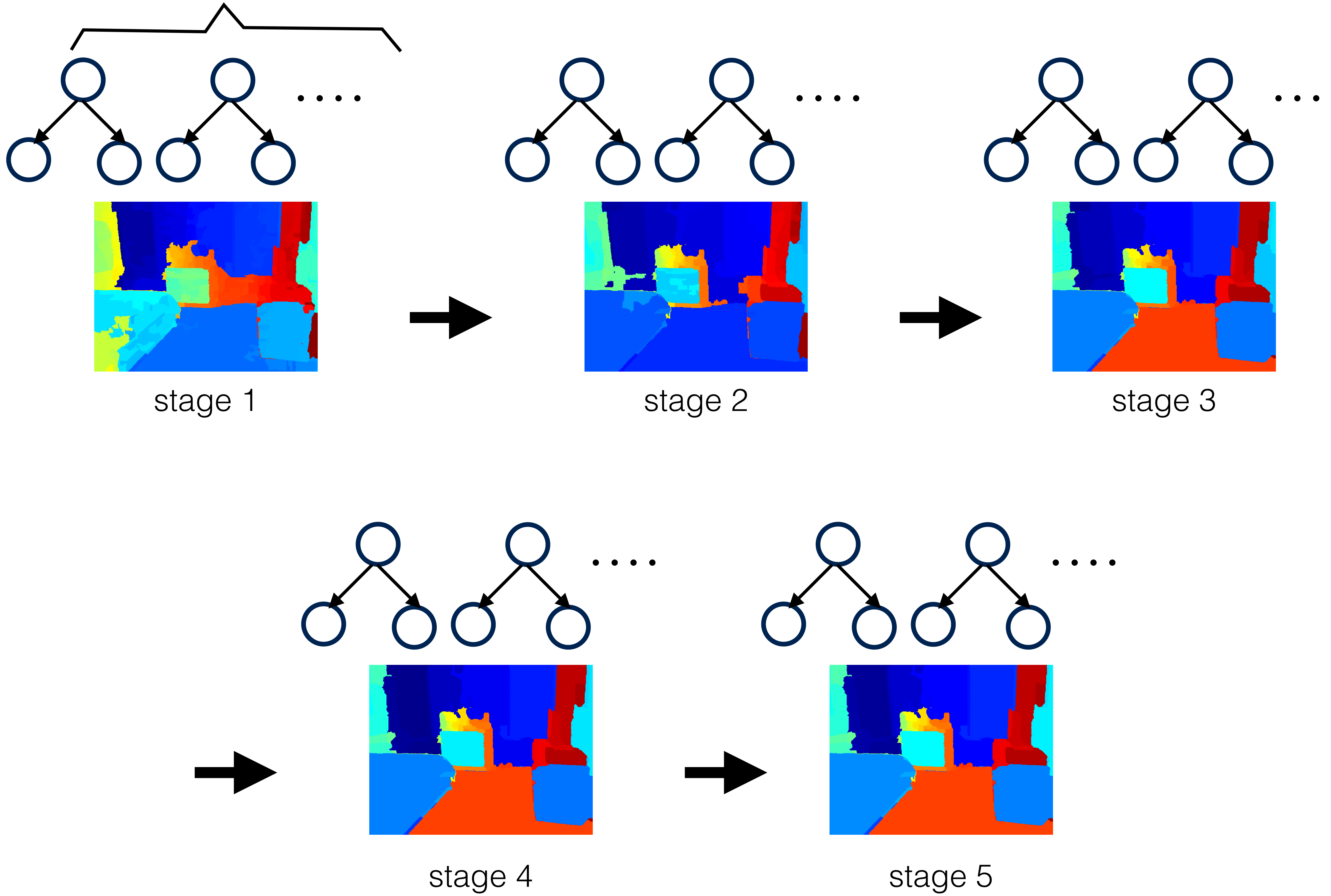
- Number of iteration of an AdaBoost decision forest boundary classifier (underfit vs. overfit)

$$T(x_i) = \sum_m \alpha_m T_m(x_i)$$

- At higher stage, the number of training example(boundary) decreases, causing lower accuracy and overfitting

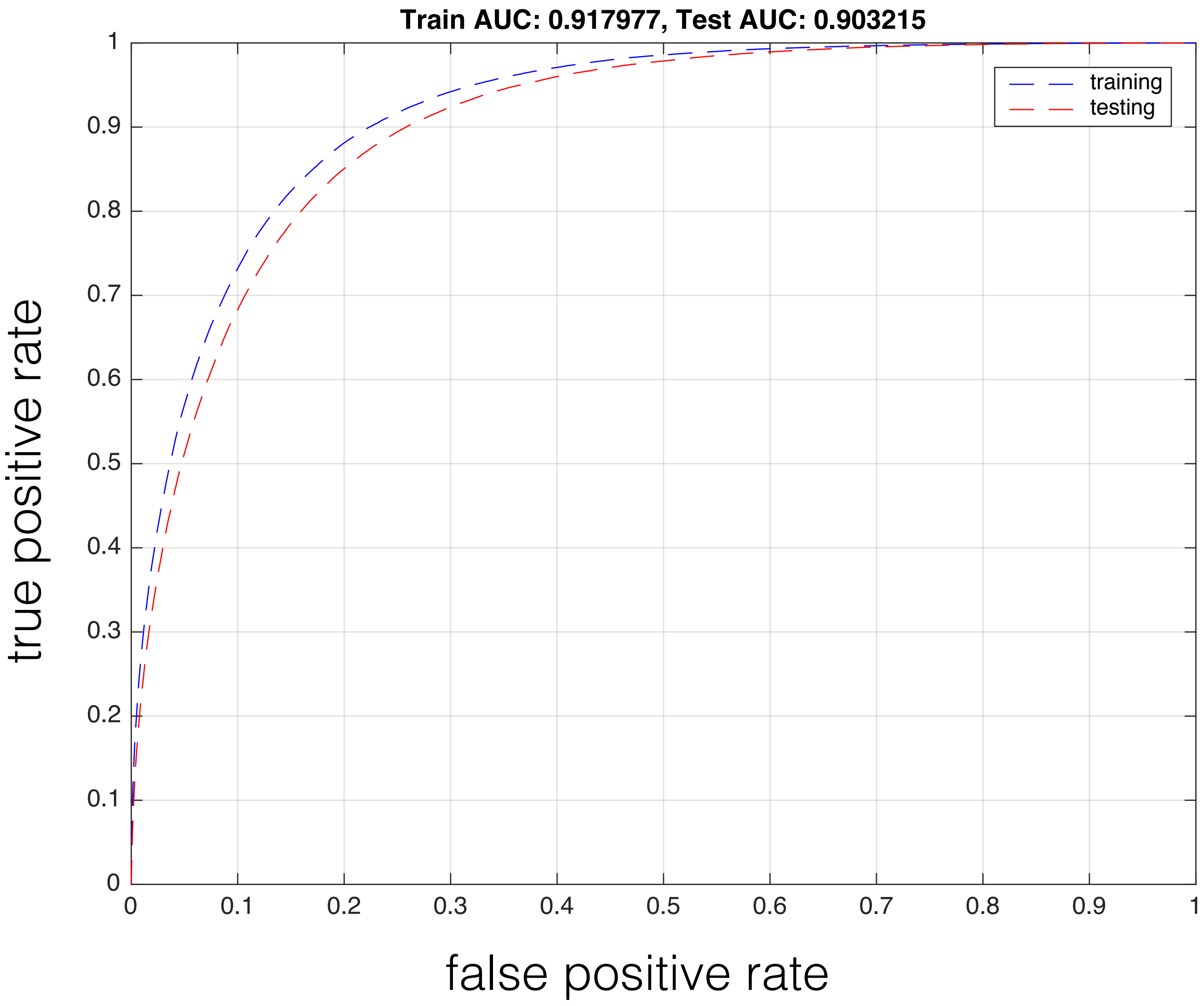
- Accuracy at lower stage is more important because of error propagation

$$T(x_i) = \sum_m \alpha_m T_m(x_i)$$

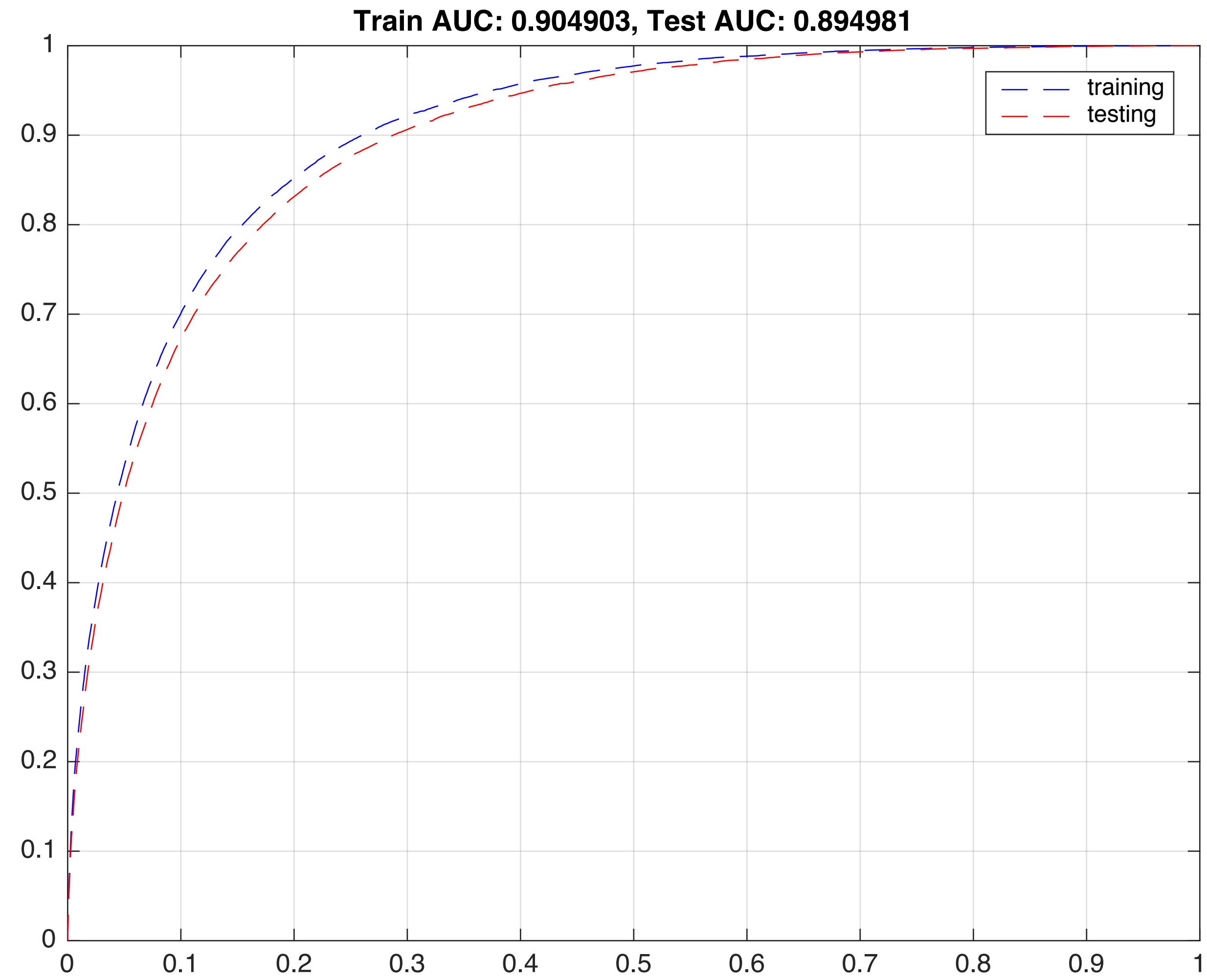


ROC Curve at Stage 1

- iteration = 30

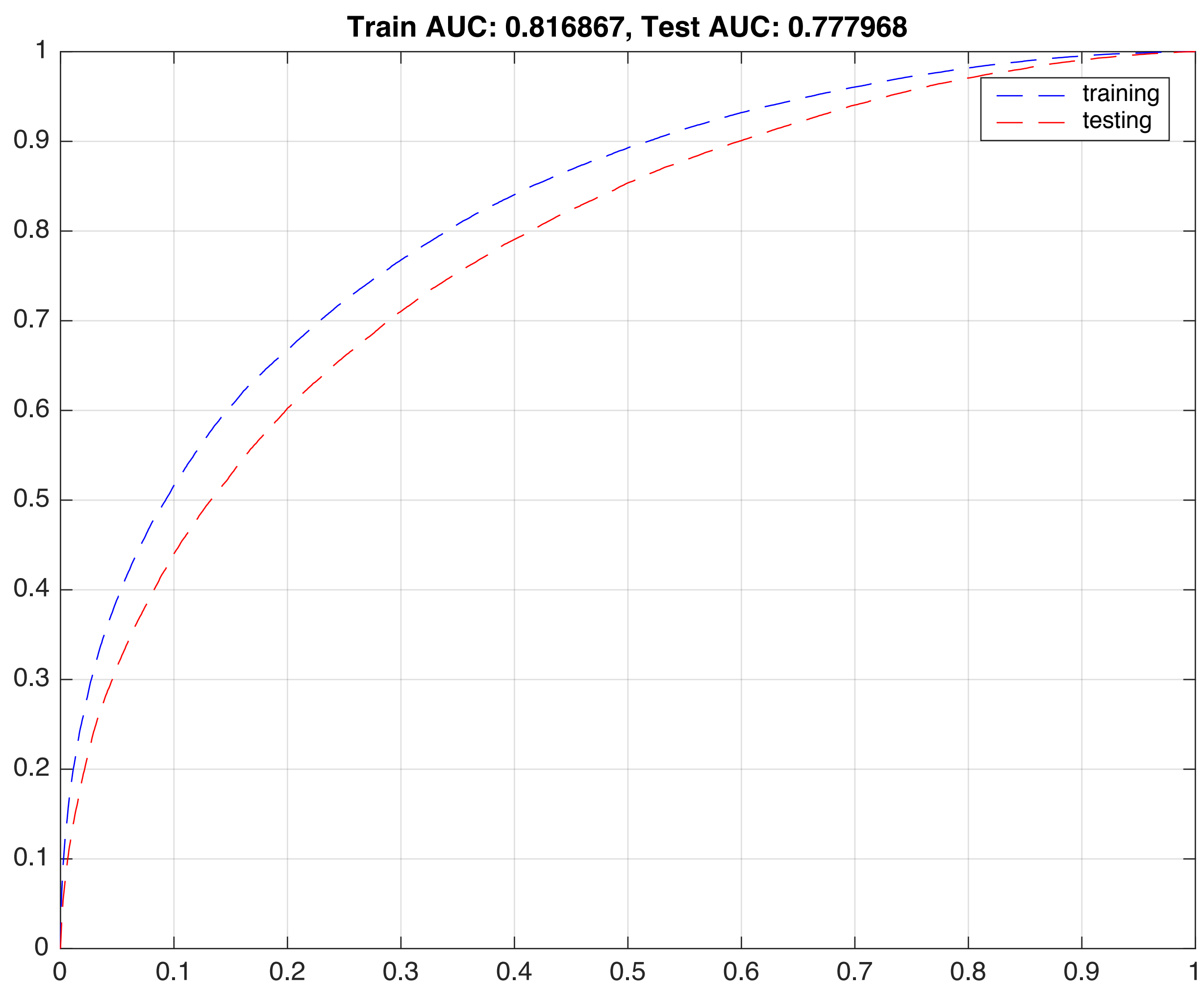


- iteration = 5

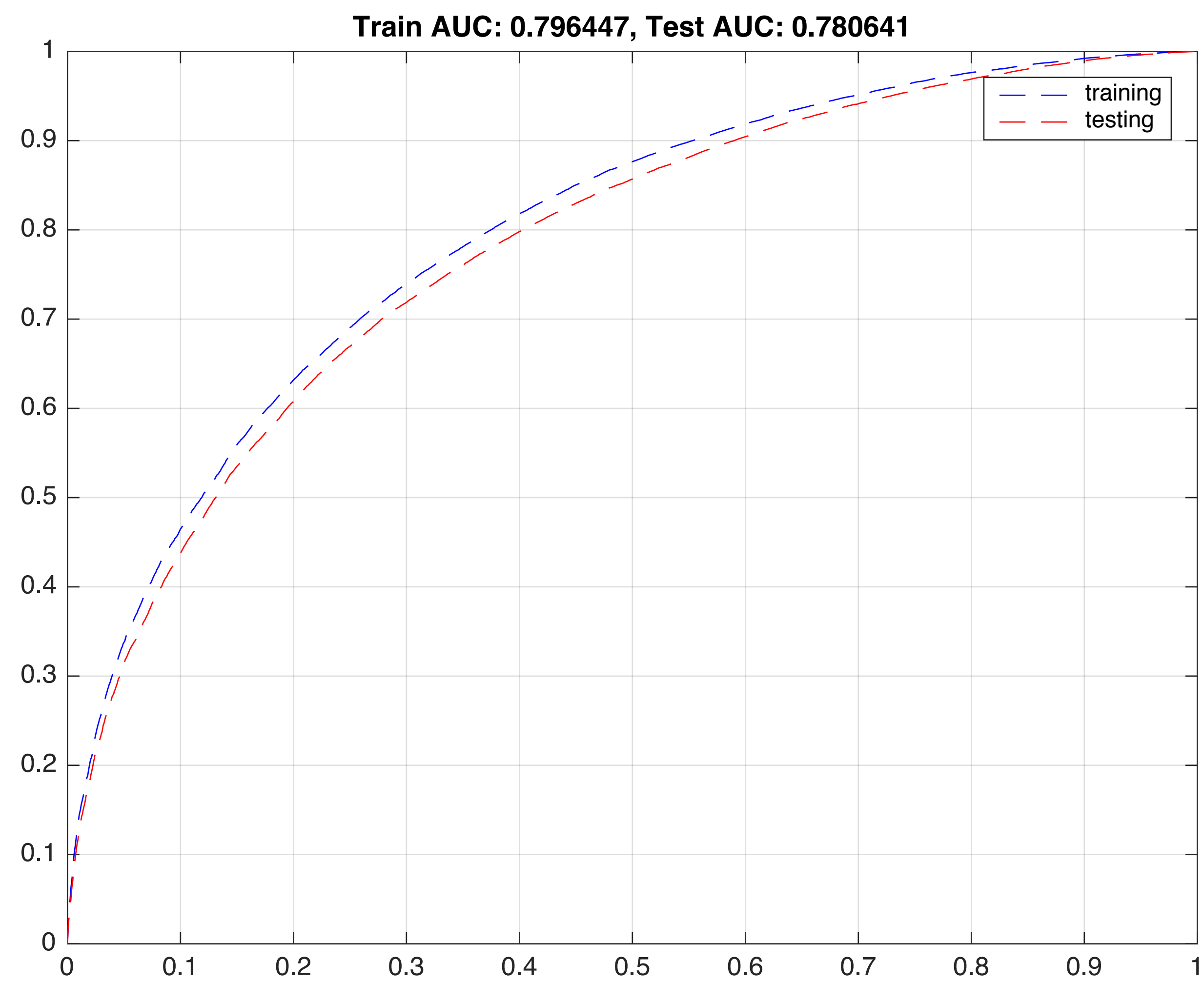


ROC Curve at Stage 2

- iteration = 30

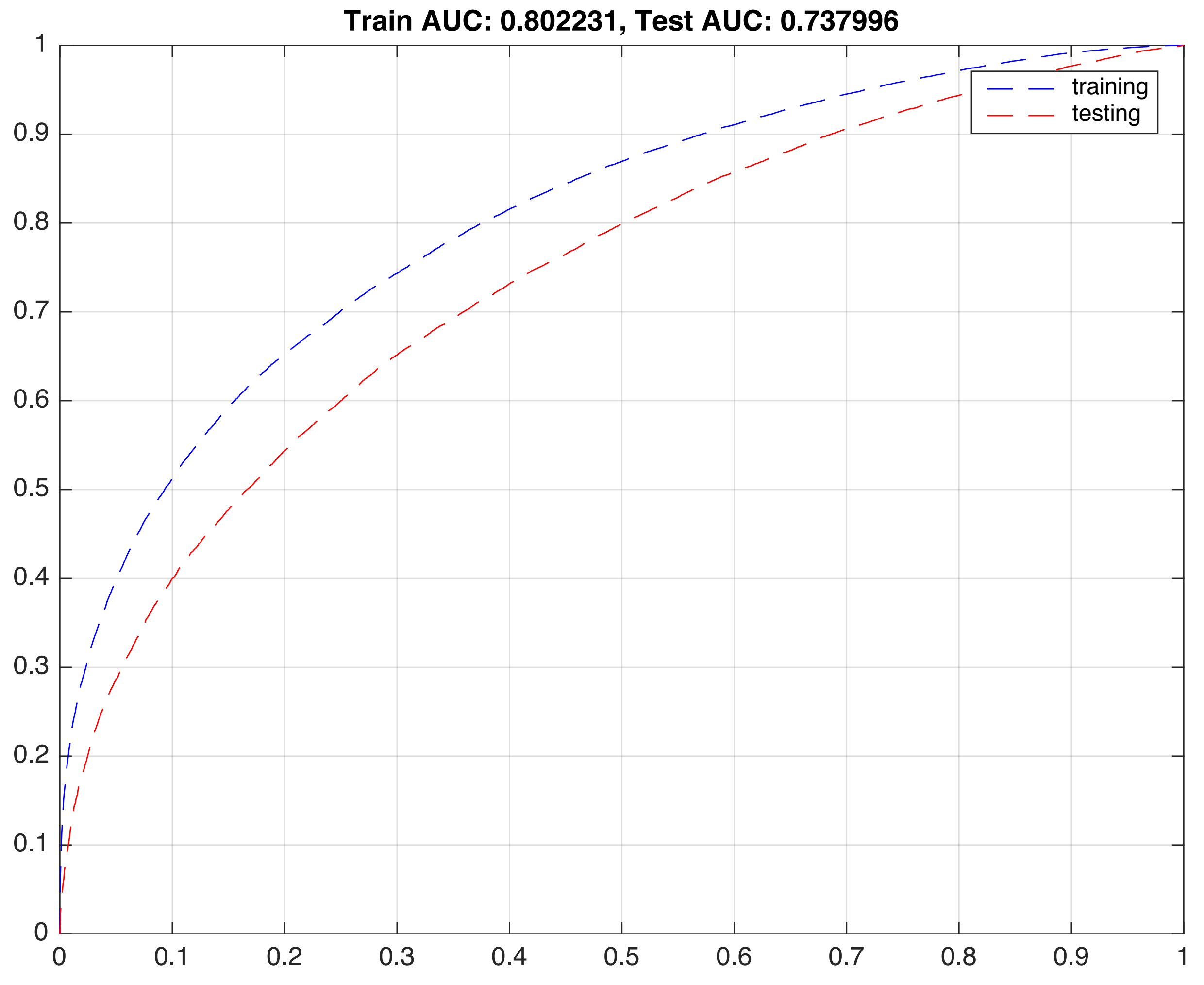


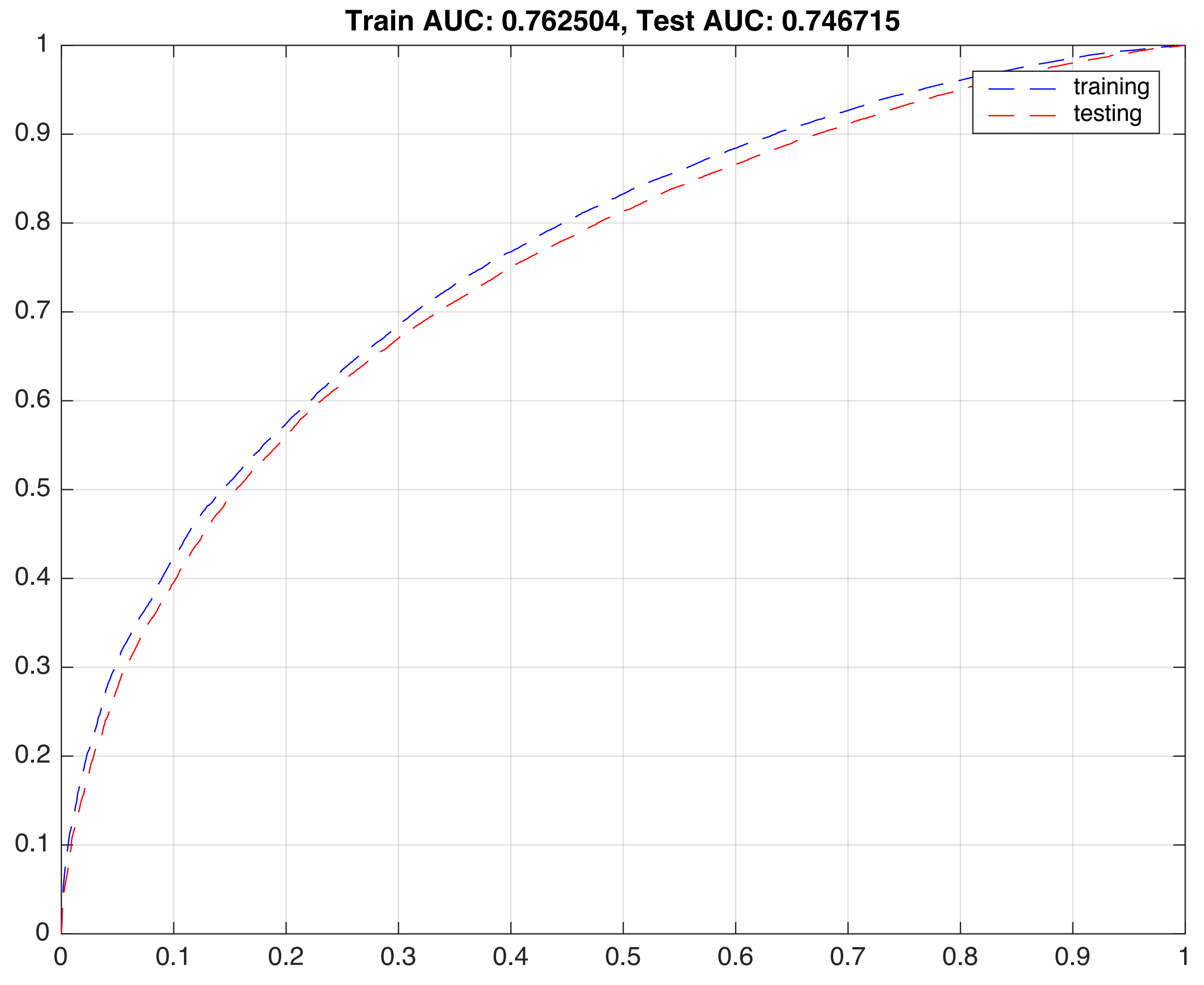
- iteration = 5



ROC Curve at Stage 3

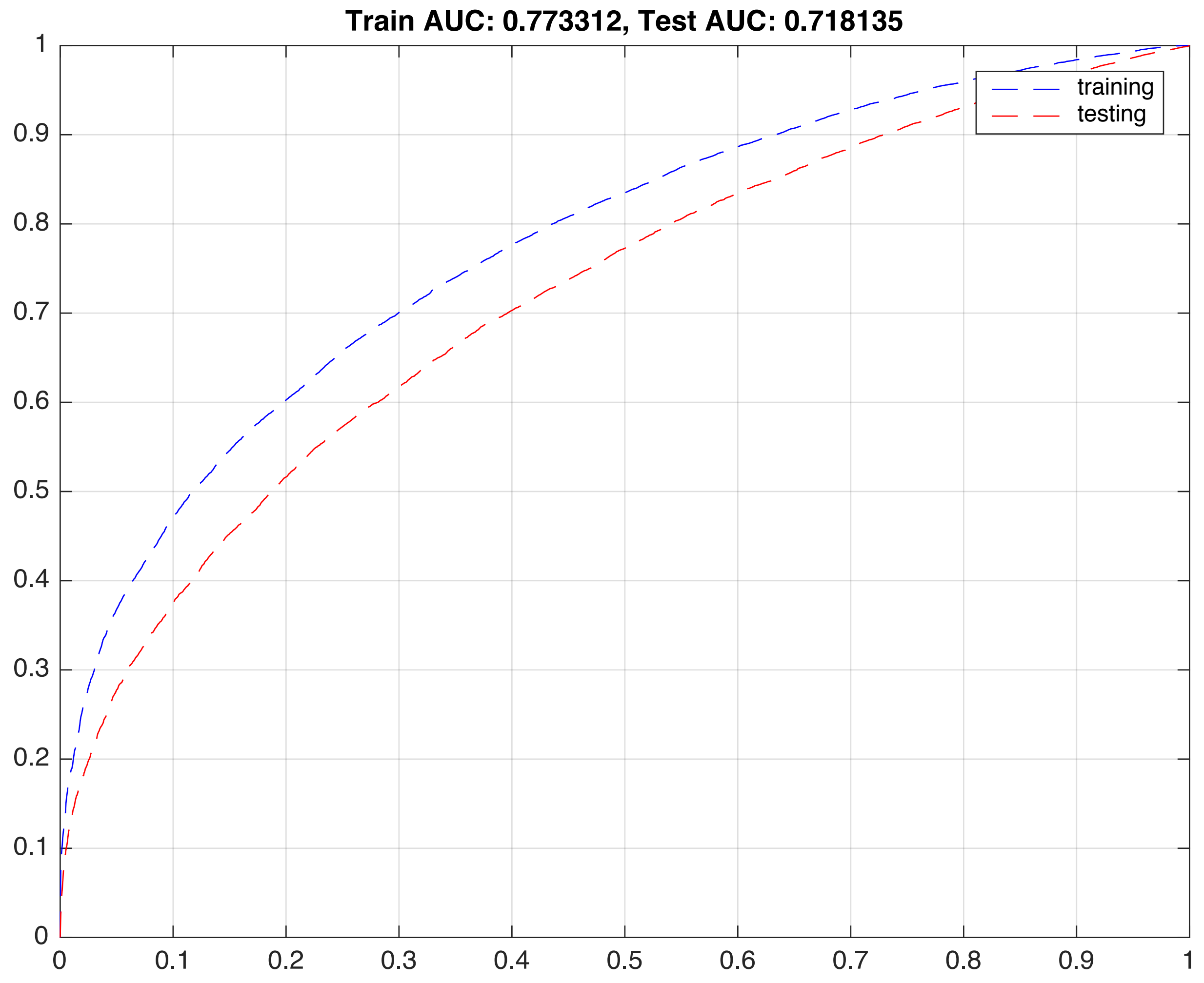
- iteration = 30
- iteration = 5



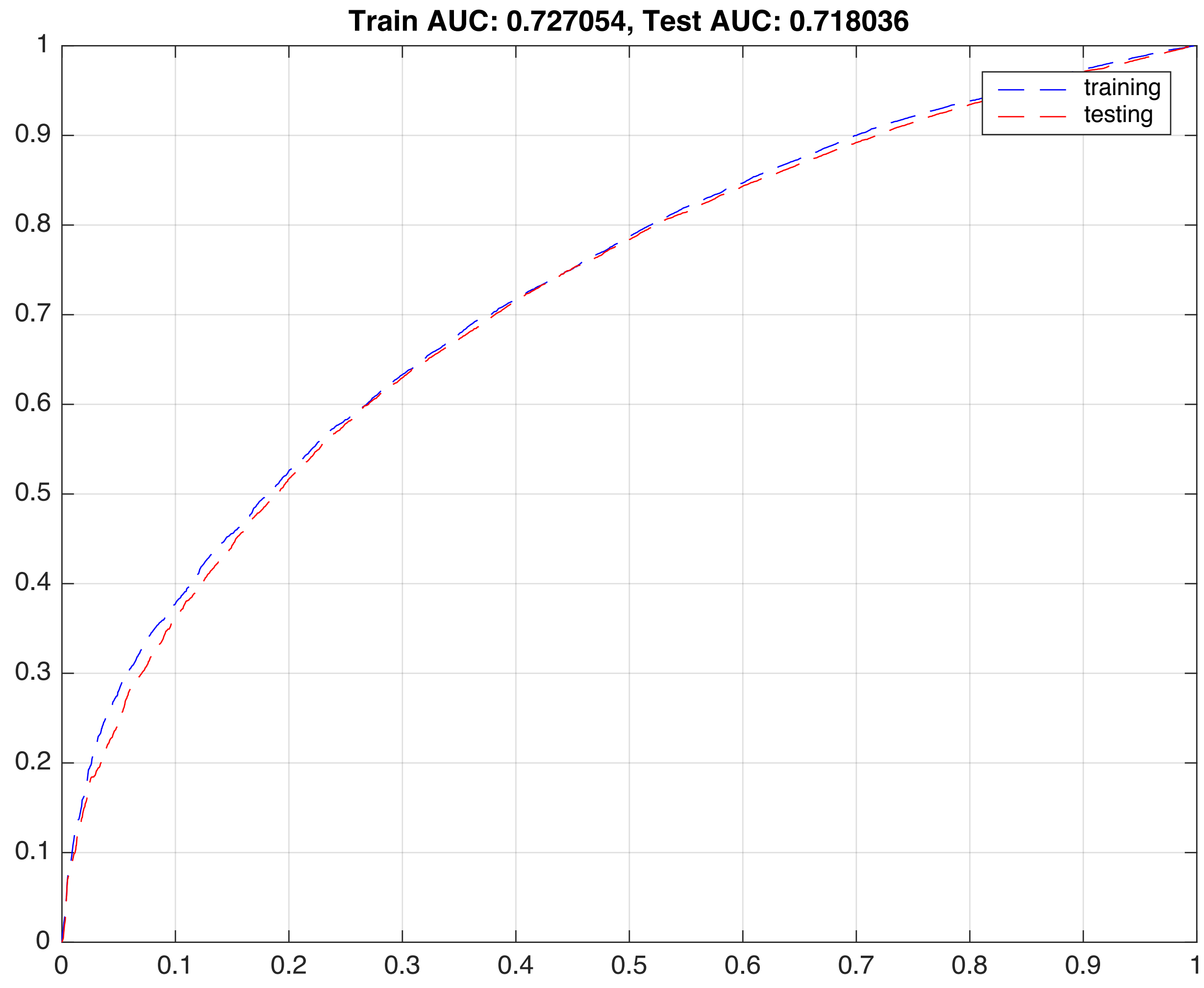


ROC Curve at Stage 4

- iteration = 30

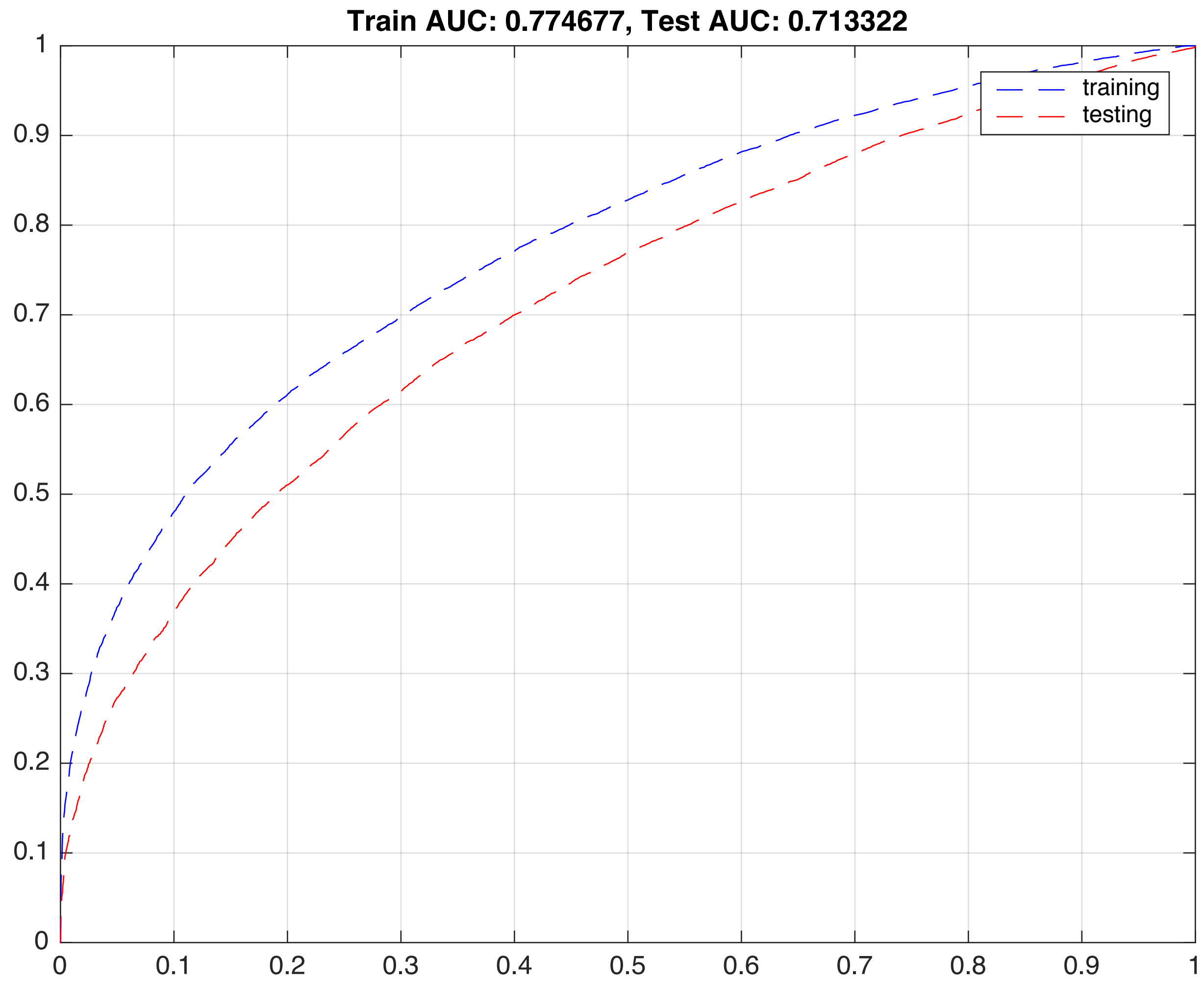


- iteration = 5

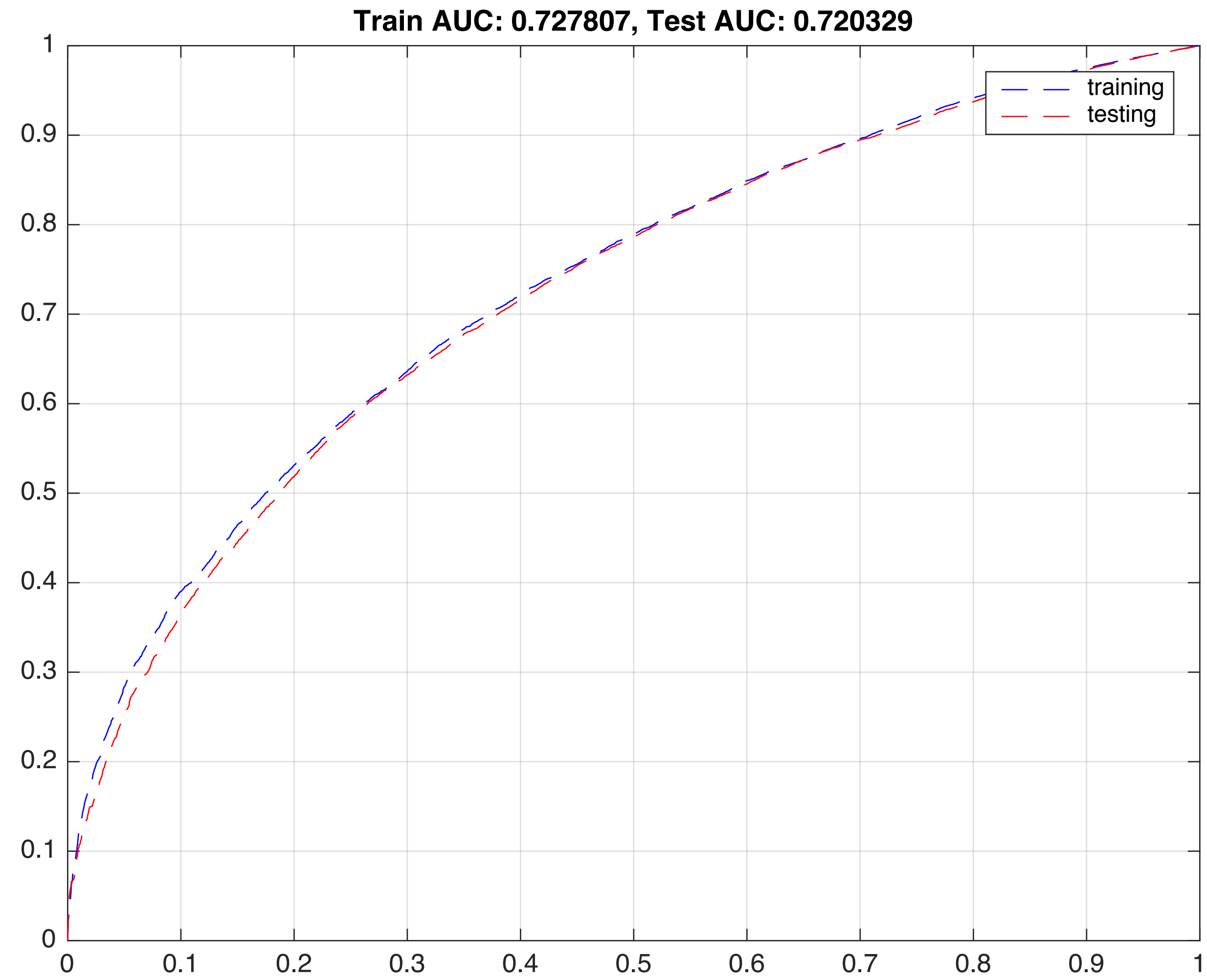


ROC Curve at Stage 5

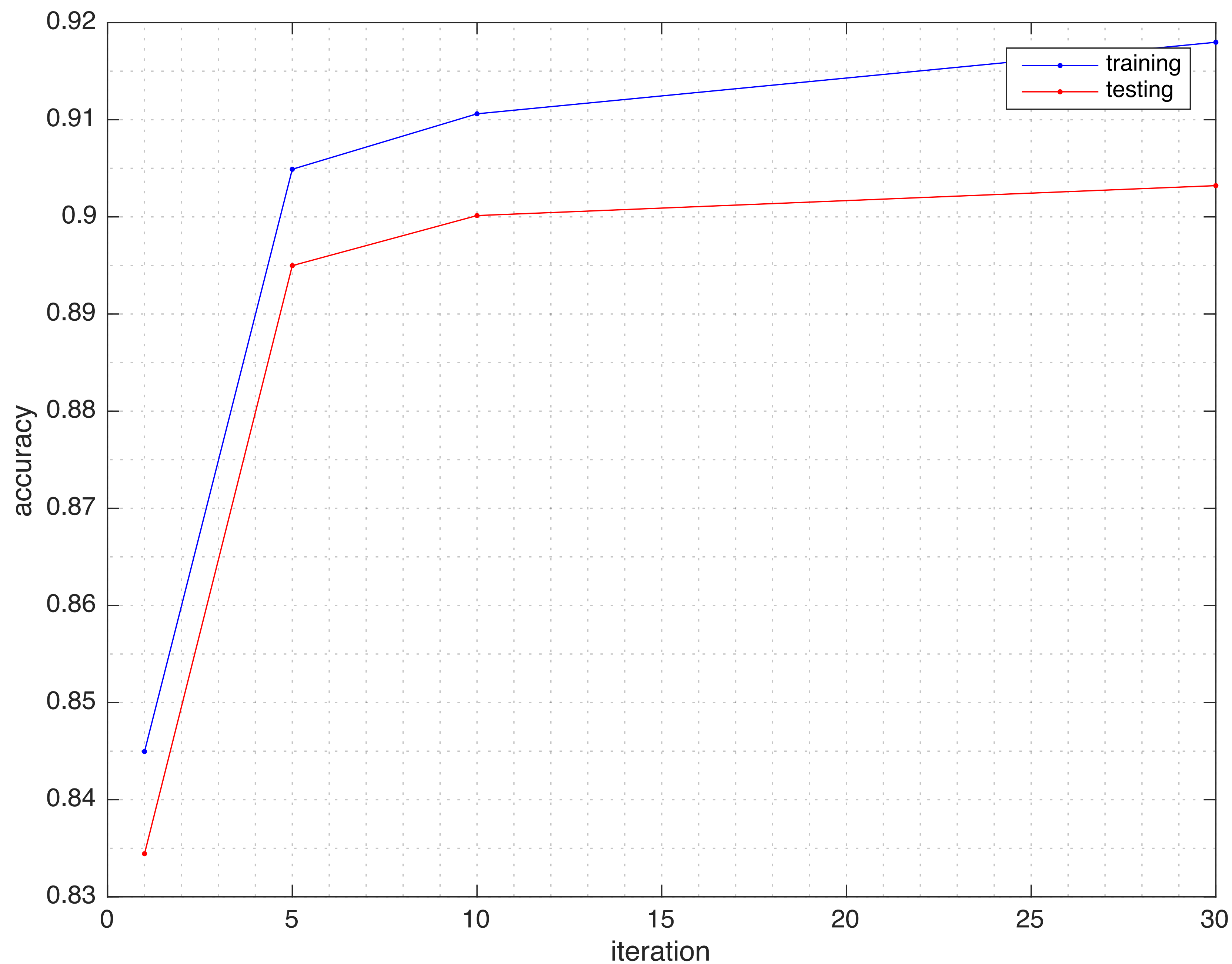
- iteration = 30



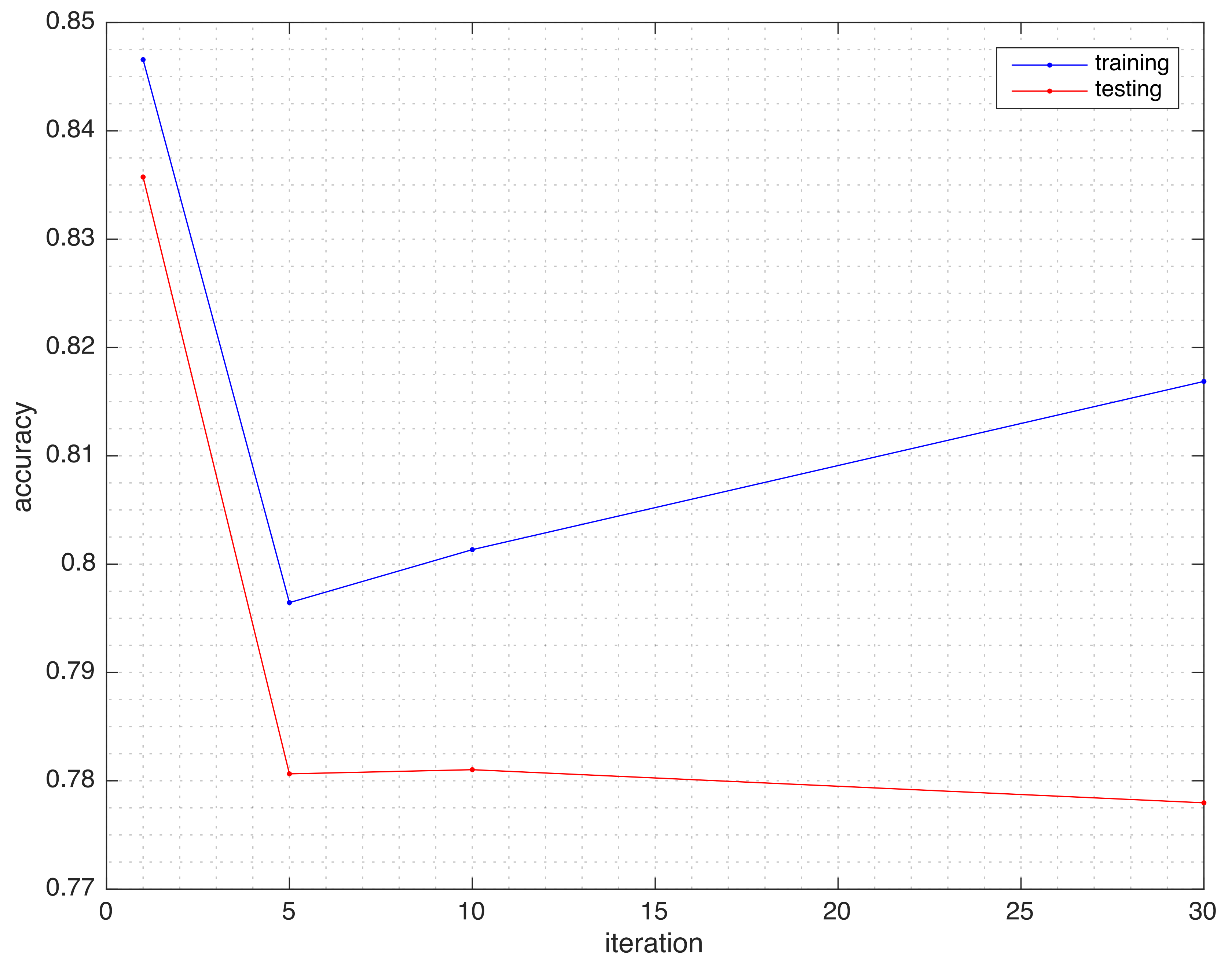
- iteration = 5



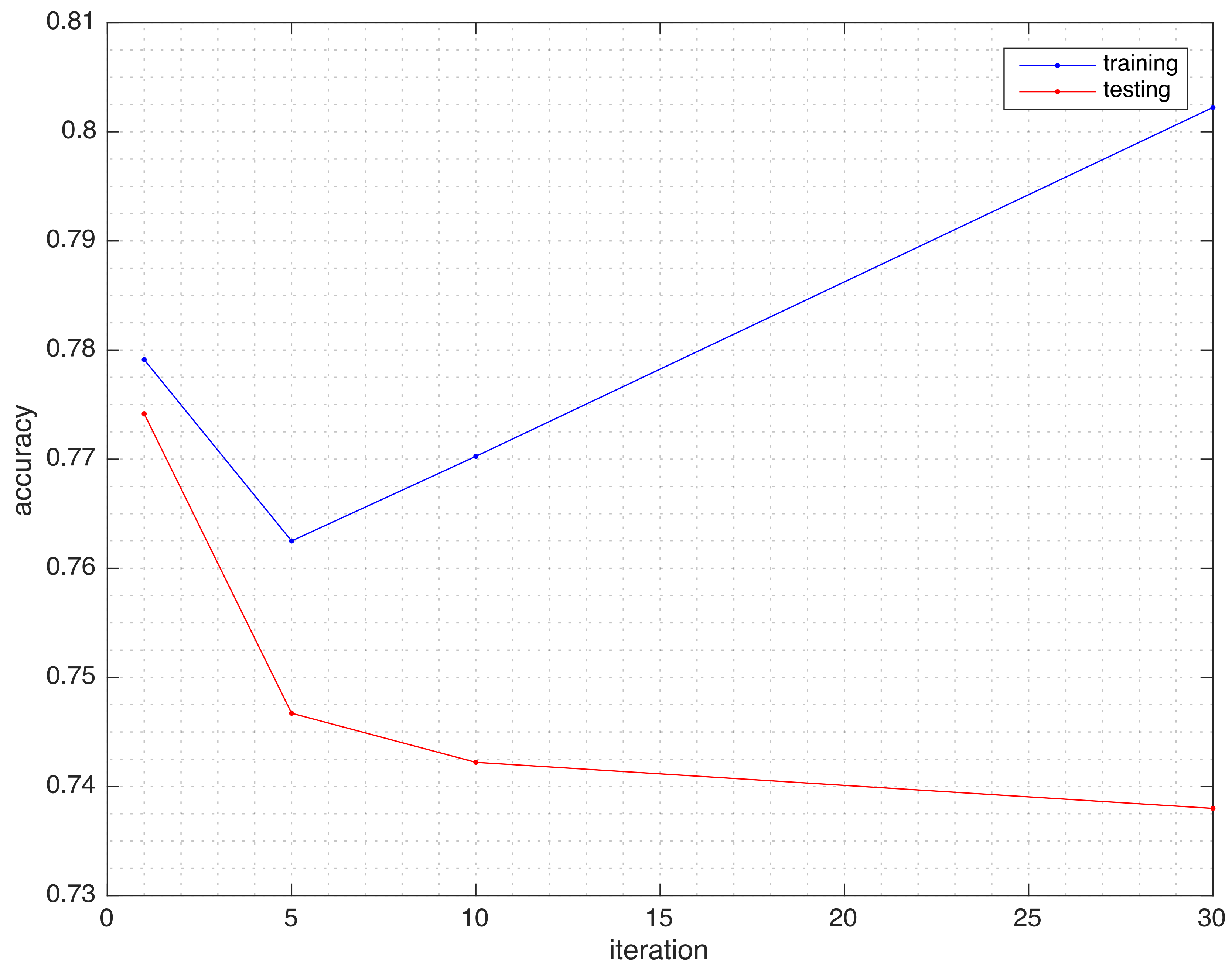
Accuracy versus Iteration at Stage 1



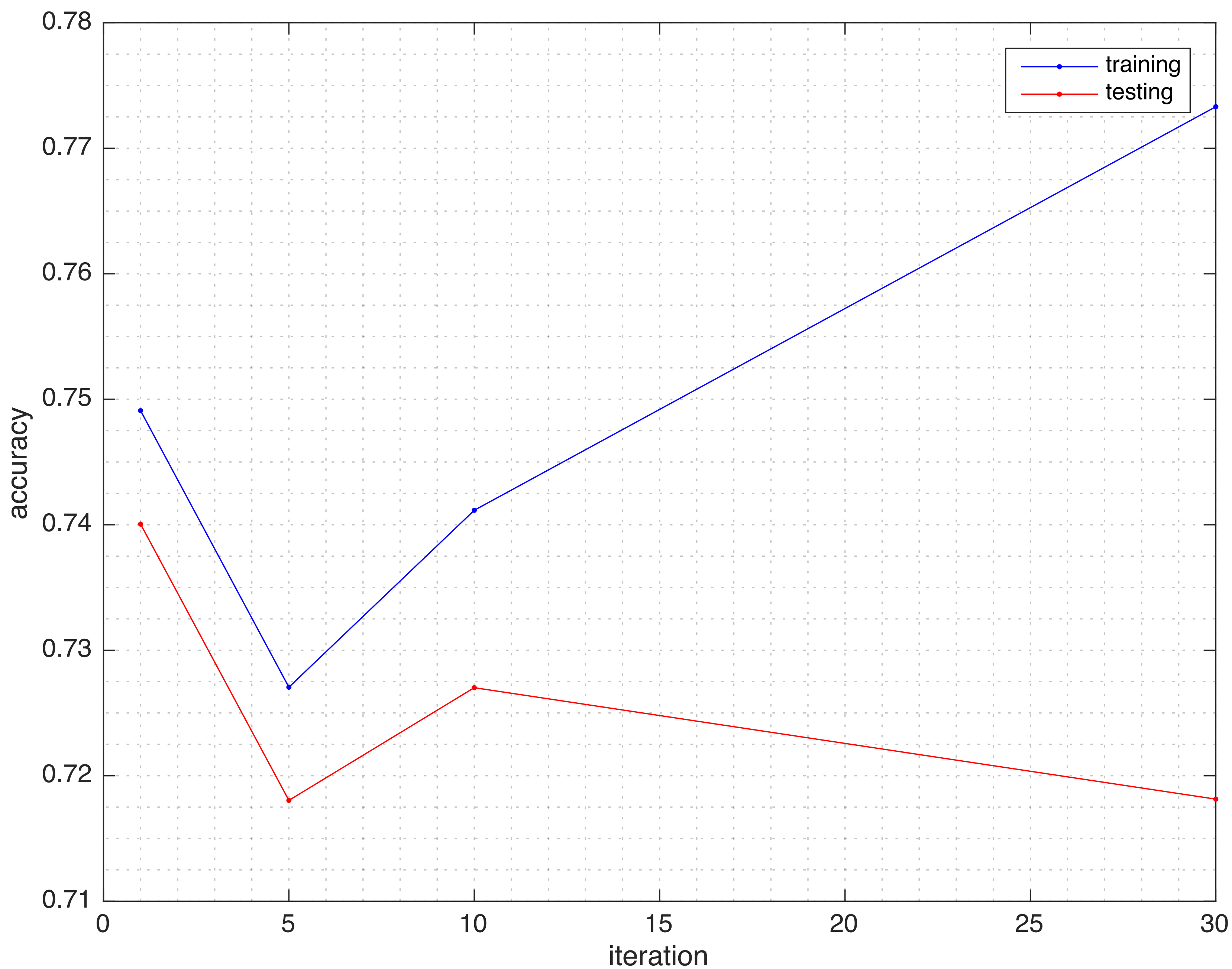
Accuracy versus Iteration at Stage 2



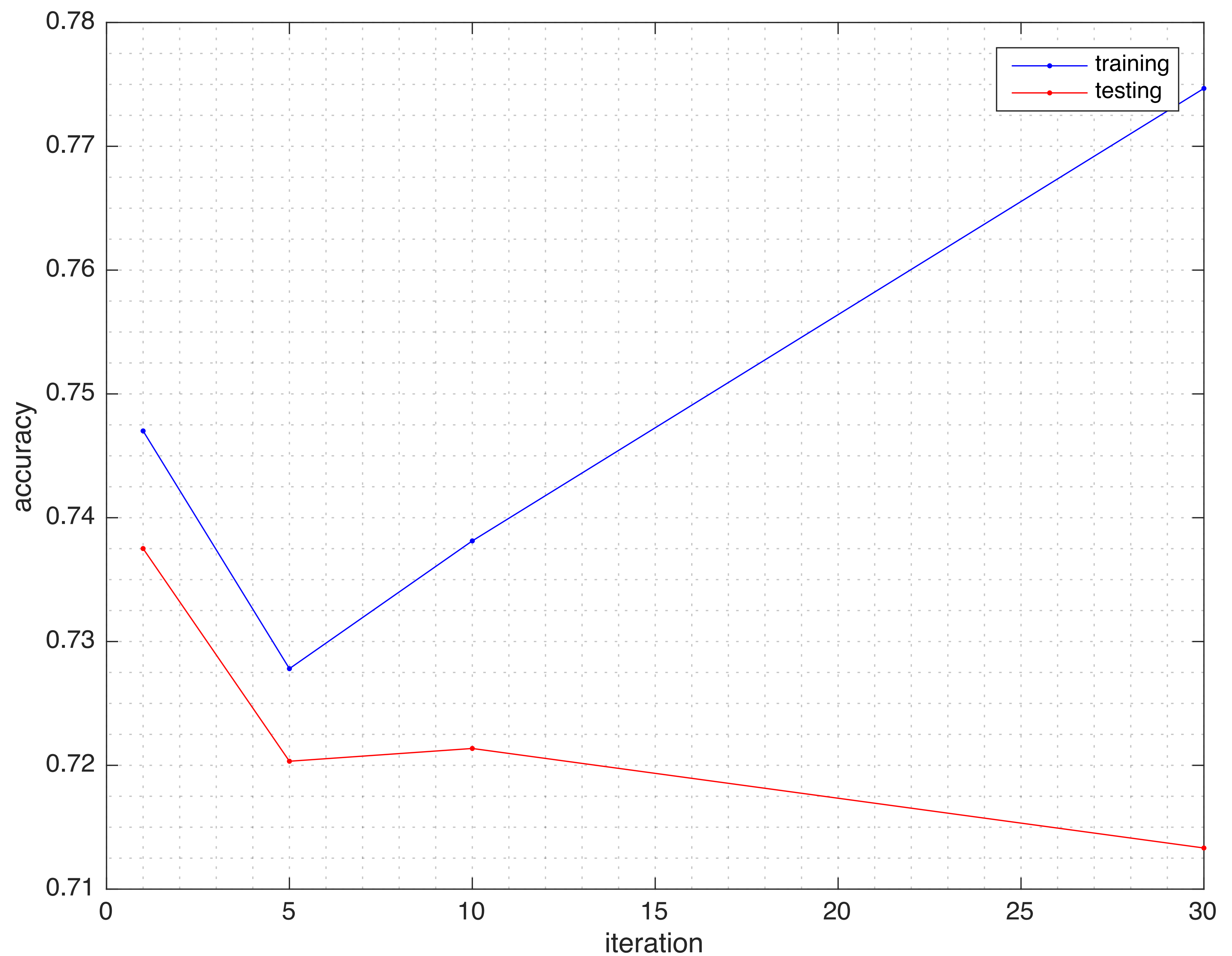
Accuracy versus Iteration at Stage 3



Accuracy versus Iteration at Stage 4



Accuracy versus Iteration at Stage 5



Segmentation Score

iteration = [30 30 30 30 30]

```
=========================================
               Results:
=========================================
Stage 1:
Weighted:   Train=42.3  Test=40.7
Unweighted: Train=39.0  Test=38.9

Stage 2:
Weighted:   Train=59.8  Test=56.8
Unweighted: Train=49.6  Test=48.8

Stage 3:
Weighted:   Train=67.1  Test=62.2
Unweighted: Train=53.6  Test=50.2

Stage 4:
Weighted:   Train=68.9  Test=63.5
Unweighted: Train=54.2  Test=50.4

Stage 5:
Weighted:   Train=69.6  Test=63.9
Unweighted: Train=54.3  Test=50.4
```

iteration = [10 10 10 10 10]

```
=========================================
               Results:
=========================================
Stage 1:
Weighted:   Train=39.4  Test=38.8
Unweighted: Train=37.9  Test=37.9

Stage 2:
Weighted:   Train=57.8  Test=56.2
Unweighted: Train=49.2  Test=48.5

Stage 3:
Weighted:   Train=64.2  Test=61.8
Unweighted: Train=51.7  Test=50.0

Stage 4:
Weighted:   Train=65.0  Test=62.3
Unweighted: Train=52.1  Test=50.2

Stage 5:
Weighted:   Train=65.4  Test=62.6
Unweighted: Train=52.2  Test=50.2
```

iteration = [5 5 5 5 5]

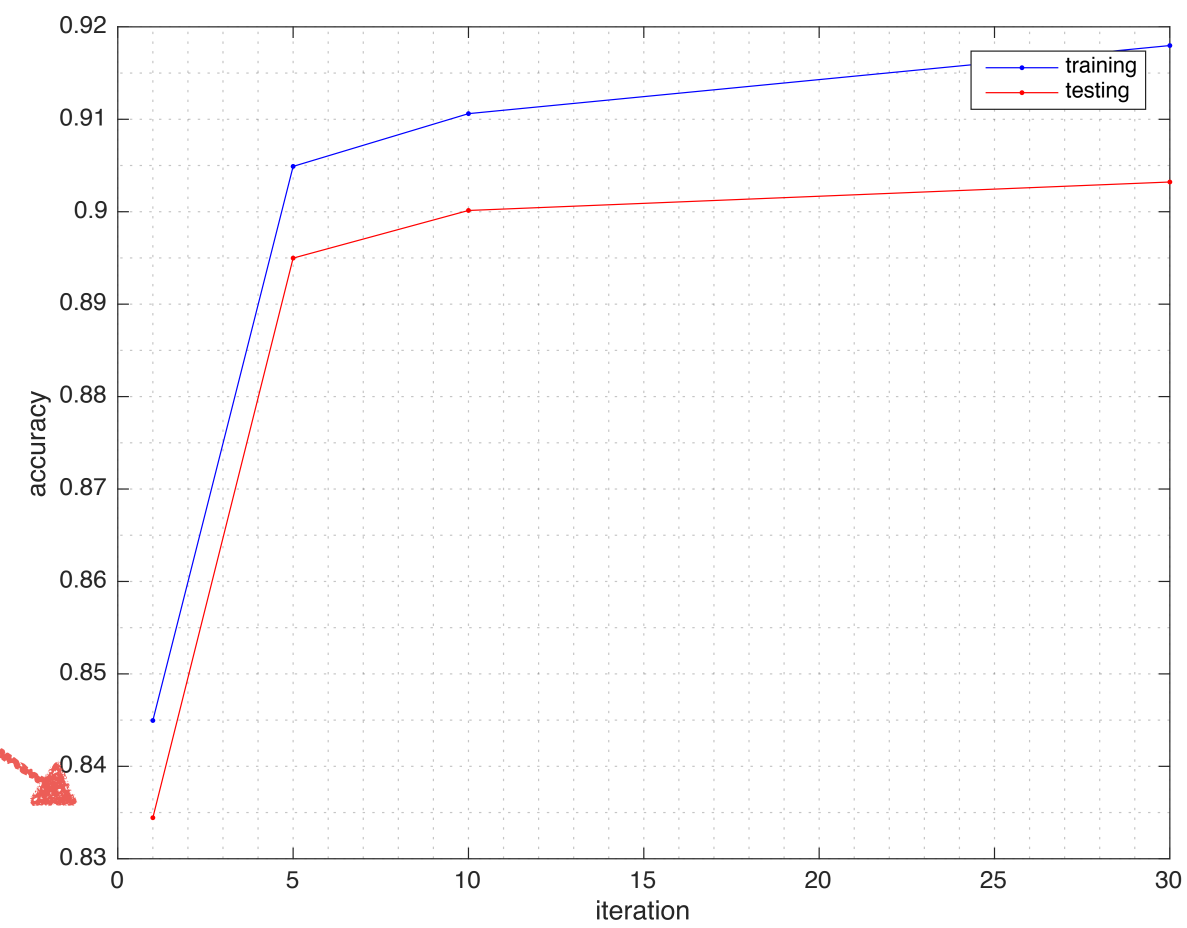
```
=========================================
               Results:
=========================================
Stage 1:
Weighted:   Train=38.6  Test=37.4
Unweighted: Train=37.5  Test=37.6

Stage 2:
Weighted:   Train=54.8  Test=52.9
Unweighted: Train=47.6  Test=47.5

Stage 3:
Weighted:   Train=61.6  Test=59.0
Unweighted: Train=51.4  Test=49.6

Stage 4:
Weighted:   Train=62.1  Test=59.7
Unweighted: Train=51.5  Test=49.3

Stage 5:
Weighted:   Train=62.4  Test=60.3
Unweighted: Train=51.3  Test=49.4
```

Accuracy at lower stage is more important!

Segmentation Score

iteration = [1 1 1 1 1]

```
========================================
               Results:
========================================
Stage 1:
Weighted:    Train=8.9   Test=8.6
Unweighted: Train=19.9   Test=19.3

Stage 2:
Weighted:    Train=36.5   Test=35.6
Unweighted: Train=37.2   Test=37.5

Stage 3:
Weighted:    Train=49.7   Test=48.6
Unweighted: Train=43.3   Test=43.0

Stage 4:
Weighted:    Train=49.7   Test=48.6
Unweighted: Train=43.3   Test=43.0

Stage 5:
Weighted:    Train=49.7   Test=48.6
Unweighted: Train=43.3   Test=43.0
```

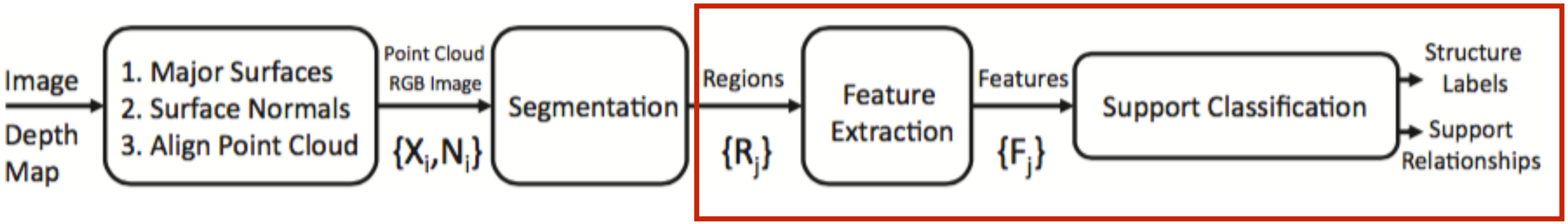


Accuracy at lower stage is more important!

Outline

- Run the segmentation pipeline
- Experiment on the segmentation pipeline
- Run the support inference pipeline
- Address strength and weakness

Support Inference Pipeline

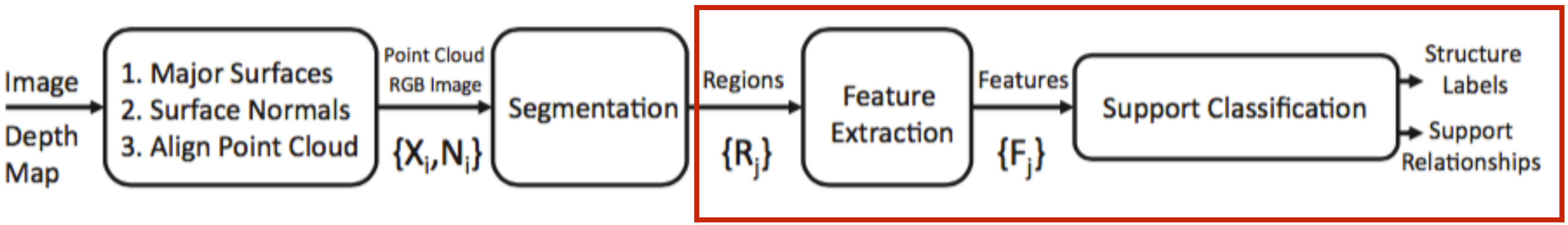


$$\{\mathbf{S}^*, \mathbf{T}^*, \mathbf{M}^*\} = \arg \max_{\mathbf{S},\mathbf{T},\mathbf{M}} P(\mathbf{S}, \mathbf{T}, \mathbf{M}|I) = \arg \min_{\mathbf{S},\mathbf{T},\mathbf{M}} E(\mathbf{S}, \mathbf{T}, \mathbf{M}|I)$$

$$P(\mathbf{S}, \mathbf{T}, \mathbf{M}|I) \propto \prod_{i=1}^{R} P(I|S_i, T_i) P(I|M_i)\, P(\mathbf{S}, \mathbf{T}, \mathbf{M})$$

$$E(\mathbf{S}, \mathbf{T}, \mathbf{M}) = -\sum_{i=1}^{R} \log(D_s(F^s_{i,S_i}|S_i, T_i) + \log(D_m(F^m_i|M_i)) + E_P(\mathbf{S}, \mathbf{T}, \mathbf{M})$$

Structure Class Classifier



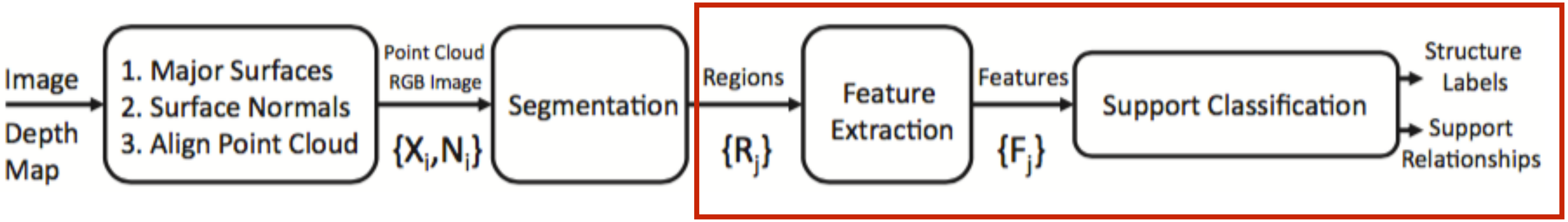
$$\{\mathbf{S}^*,\mathbf{T}^*,\mathbf{M}^*\} = \arg\max_{\mathbf{S},\mathbf{T},\mathbf{M}} P(\mathbf{S},\mathbf{T},\mathbf{M}|I) = \arg\min_{\mathbf{S},\mathbf{T},\mathbf{M}} E(\mathbf{S},\mathbf{T},\mathbf{M}|I)$$

$$P(\mathbf{S},\mathbf{T},\mathbf{M}|I) \propto \prod^{R} P(I|S_i,T_i)P(I|M_i)\, P(\mathbf{S},\mathbf{T},\mathbf{M})$$

$$E(\mathbf{S},\mathbf{T},\mathbf{M}) = -\sum_{i=1}^{R} \log(D_s(F^s_{i,S_i}|S_i,T_i)+\log(D_m(F^m_i|M_i))+E_P(\mathbf{S},\mathbf{T},\mathbf{M})$$

```
% Extract features and train a classifier for structure-class prediction.
run_extract_structure_class_features_seg;
run_create_dataset_structure_class_features_seg;
run_train_structure_class_classifier_seg;
```

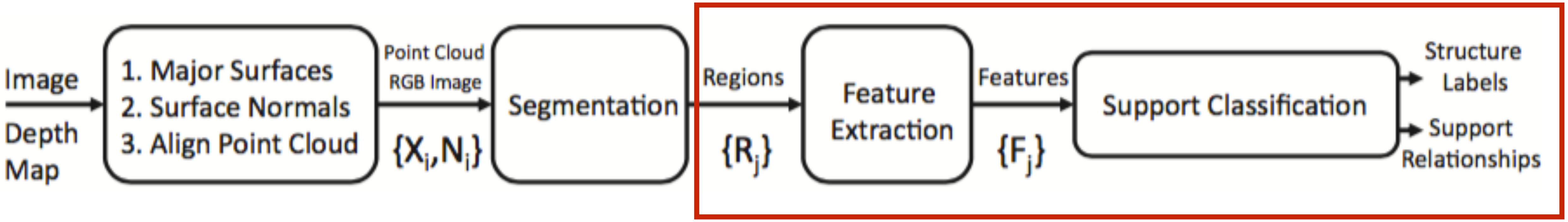
Structure Class Classifier



$$E(\mathbf{S},\mathbf{T},\mathbf{M}) = -\sum_{i=1}^{R} \log(D_s(F^s_{i,S_i}|S_i,T_i) + \log(D_m(F^m_i|M_i)) + E_P(\mathbf{S},\mathbf{T},\mathbf{M})$$

```
(region_classifier) Batch 50/101, avgLoss=0.552447
(region_classifier) Batch 51/101, avgLoss=0.552748
(region_classifier) Updates 5000/5000, SSE=2937.837935, MSE: 0.576047
Time elapsed:   0 hours,   0 minutes,  51 seconds.
Time remaining:   0 hours,   0 minutes,   0 seconds.
Acc Train: 0.774624
Acc Test: 0.615448
Mean diag (Train): 0.804457
Mean diag (Test): 0.640516
```

Support Classifier



$$\{\mathbf{S}^*, \mathbf{T}^*, \mathbf{M}^*\} = \arg\max_{\mathbf{S},\mathbf{T},\mathbf{M}} P(\mathbf{S}, \mathbf{T}, \mathbf{M}|I) = \arg\min_{\mathbf{S},\mathbf{T},\mathbf{M}} E(\mathbf{S}, \mathbf{T}, \mathbf{M}|I)$$

$$P(\mathbf{S}, \mathbf{T}, \mathbf{M}|I) \propto \prod^{R} P(I|S_i, T_i) P(I|M_i)\, P(\mathbf{S}, \mathbf{T}, \mathbf{M})$$

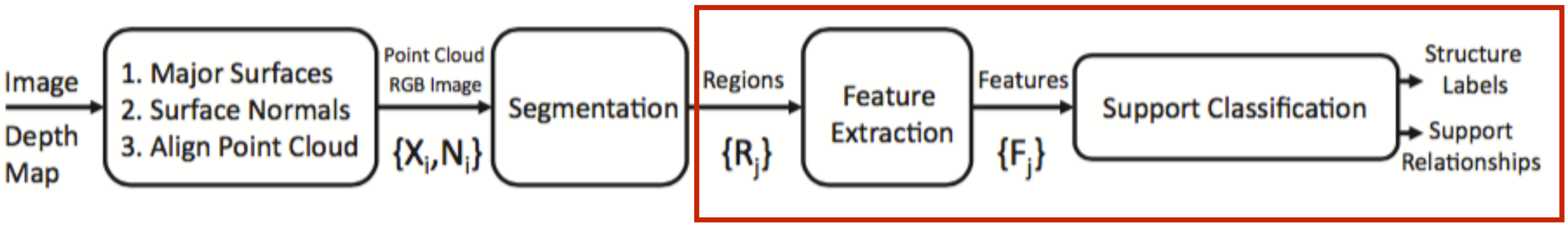
$$E(\mathbf{S}, \mathbf{T}, \mathbf{M}) = -\sum_{i=1}^{R} \log(D_s(F^s_{i,S_i}|S_i, T_i) + \log(D_m(F^m_i|M_i)) + E_P(\mathbf{S}, \mathbf{T}, \mathbf{M})$$

```
% Extract features and train a classifier for local support prediction.
run_extract_support_features_seg;
run_create_dataset_support_features_seg;
run_train_support_classifier_seg;
```

containment, geometry, and horz feature

take ~1 day to extract features for 292 images!

Support Classifier



$$E(\mathbf{S},\mathbf{T},\mathbf{M}) = -\sum_{i=1}^{R} \log(D_s(F^s_{i,S_i}|S_i,T_i) + \log(D_m(F^m_i|M_i)) + E_P(\mathbf{S},\mathbf{T},\mathbf{M})$$

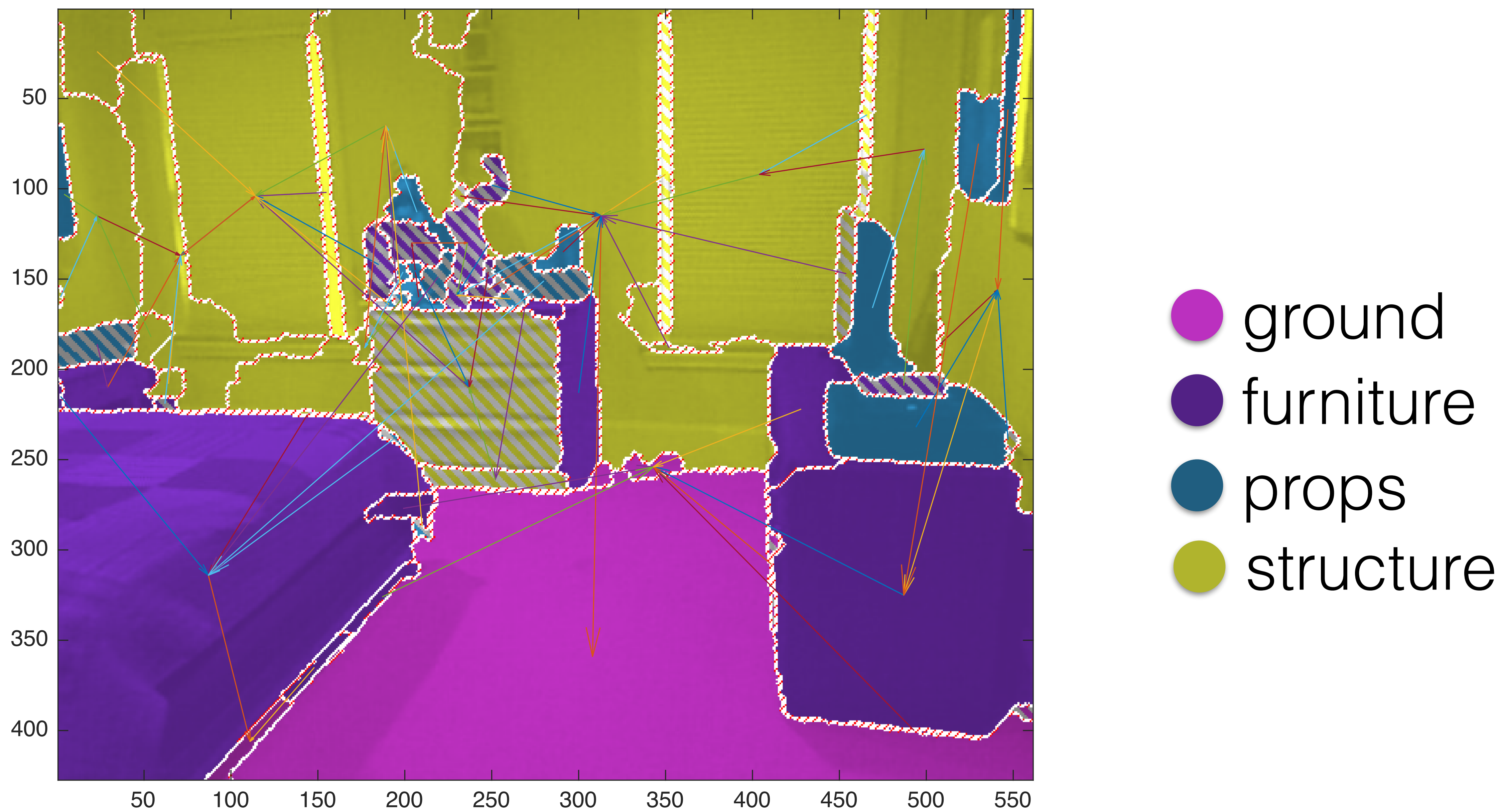
```
(support_classifier) Batch 844/1004, avgLoss=0.043889
(support_classifier) Updates 40000/40000, SSE=7157.250000, MSE: 0.084802
Time elapsed:   0 hours,   5 minutes,  58 seconds.
Time remaining:   0 hours,   0 minutes,   0 seconds.
AccTrain: 0.971561
AccTest: 0.964081

Mean Diag train: 0.414853
Mean Diag test: 0.412547

       93926          0          0        526
         339          3          0          1
         100          0          0          0
        1889          0          0       3606

       80496          3          0       1002
         277          2          0          1
          52          0          0          0
        1791          0          0       3406
```

Infer by Linear Program

$$
\begin{aligned}
&\arg\min_{\mathbf{s},\mathbf{m},\mathbf{w}} \textstyle\sum_{i,j} \theta^s_{i,j} s_{i,j} + \sum_{i,u} \theta^m_{i,u} m_{i,u} + \displaystyle\sum_{i,j,u,v} \theta^w_{i,j,u,v} w^{u,v}_{i,j} && (10)\\
&\text{s.t. } \textstyle\sum_j s_{i,j} = 1, \quad \sum_u m_{i,u} = 1 \ \forall i && (11)\\
&\textstyle\sum_{j,u,v} w^{u,v}_{i,j} = 1, \qquad \forall i && (12)\\
&s_{i,2R'+1} = m_{i,1}, \qquad \forall i && (13)\\
&\textstyle\sum_{u,v} w^{u,v}_{i,j} = s_{i,j}, \qquad \forall u, v && (14)\\
&\textstyle\sum_{j,v} w^{u,v}_{i,j} \le m_{i,u}, \qquad \forall i, u && (15)\\
&\cancel{s_{i,j},\ m_{i,u},\ w^{u,v}_{i,j} \in \{0,1\}, \quad \forall i, j, u, v} && \cancel{(16)}\\
&s_{i,j},\ m_{i,u},\ w^{u,v}_{i,j} \in [0,1], \quad \forall i, j, u, v. && (17)
\end{aligned}
$$

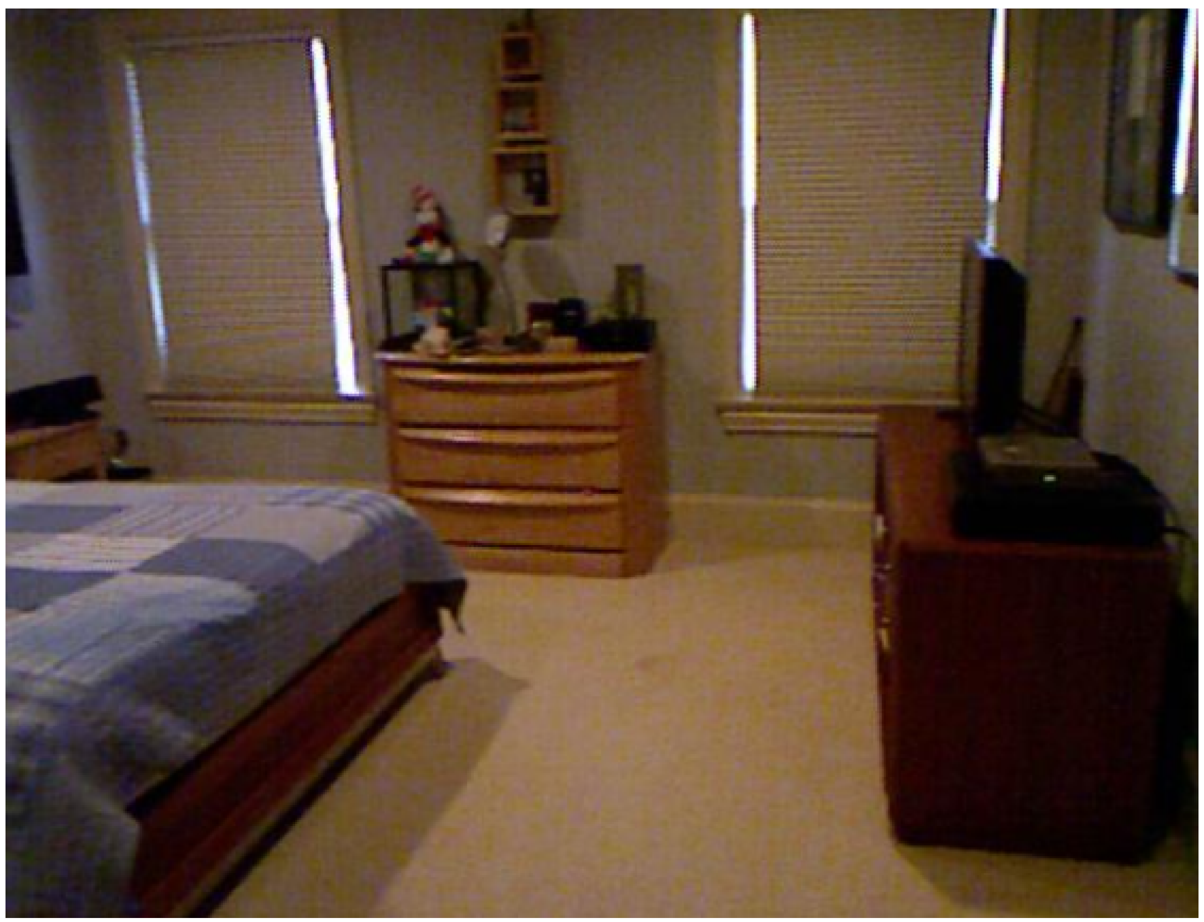
6 minutes for an image!

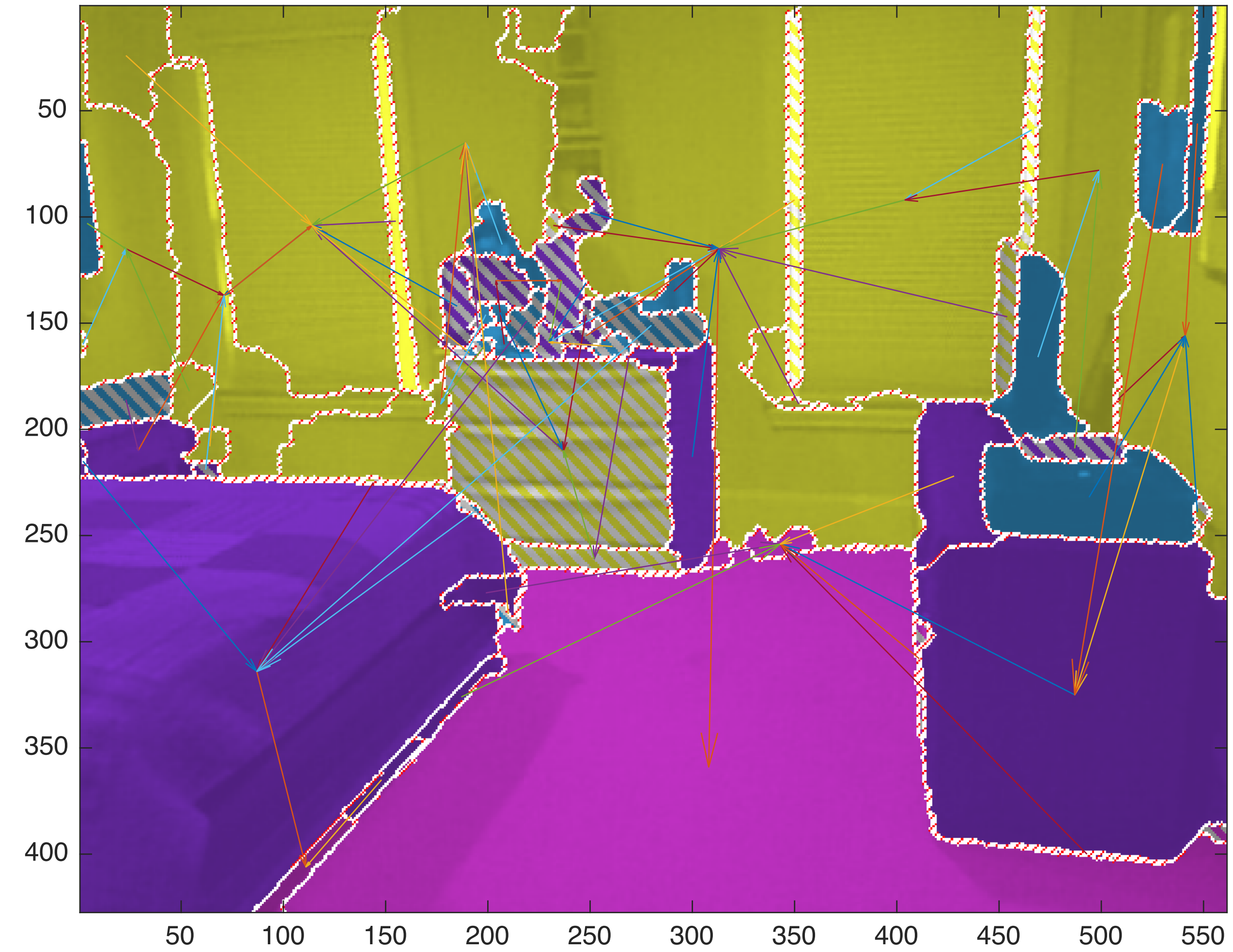
Structure and Support Inference



stripe: incorrect structure prediction

Structure and Support Inference

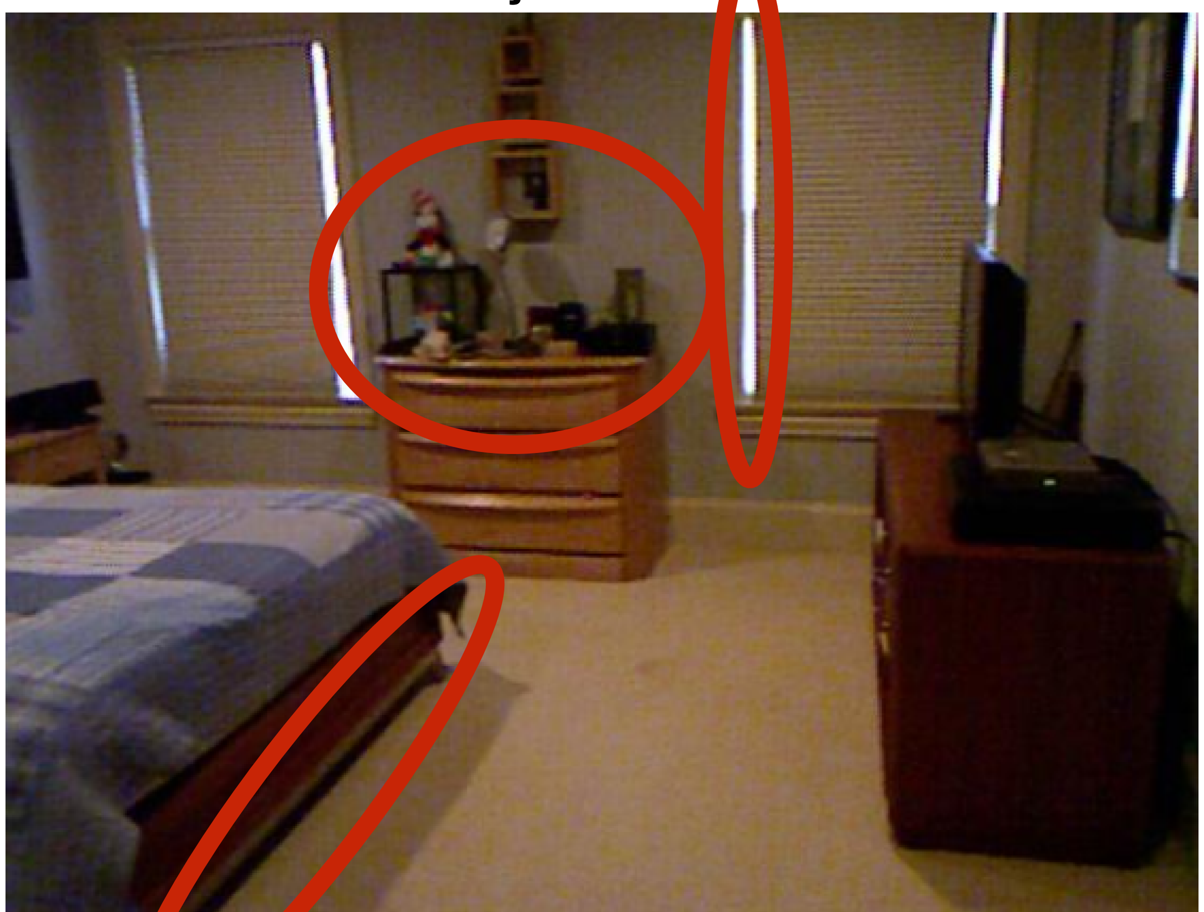


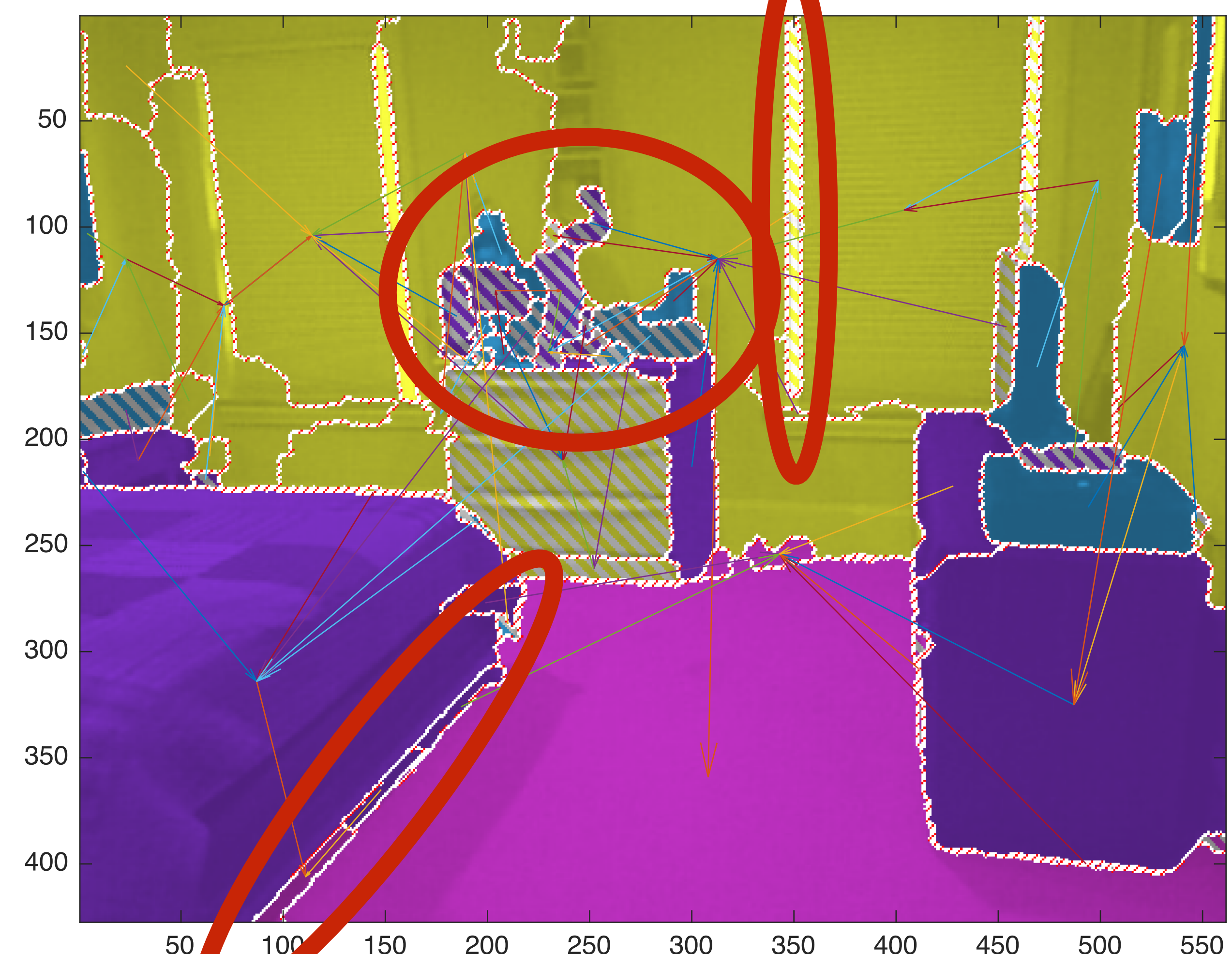


Structure and Support Inference

out of 4 classes

clutter, small objects





over-segmentation(color variance in an object)

Outline

- Run the segmentation pipeline
- Experiment on the segmentation pipeline
- Run the support inference pipeline
- Address strength and weakness

Strength

- Reason joint assignment for structure and support
- ~73% accuracy if ground truth segmentation is given

Weakness

- Slow in testing time
  -5 minutes for feature extraction
  -6 minutes for inference by linear programming

- Clutters, small(thin) objects, color variance in objects

- Only 4 structure classes(no human, pet,…etc)

- ~55% accuracy if bottom up segmentation followed by support inference

Reference

- Code: http://cs.nyu.edu/~silberman/projects/indoor_scene_seg_sup.html