Fast Edge Detection Using Structured Forests

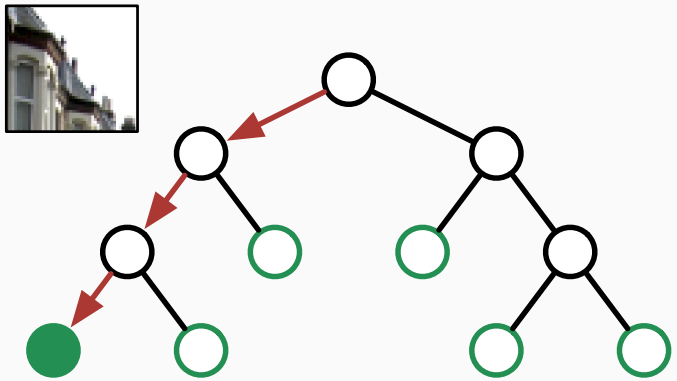
Piotr Dollár, C. Lawrence Zitnick [1]

Zhihao Li (zhihaol@andrew.cmu.edu)

Computer Science Department
Carnegie Mellon University

Table of contents

Introduction



Decision Tree

A **decision tree** $f_t(x)$ classifies a sample $x \in \mathcal{X}$ by recursively branching left or right down the tree until a leaf node is reached. Specifically, each node j in the tree is associated with a binary *split function*:

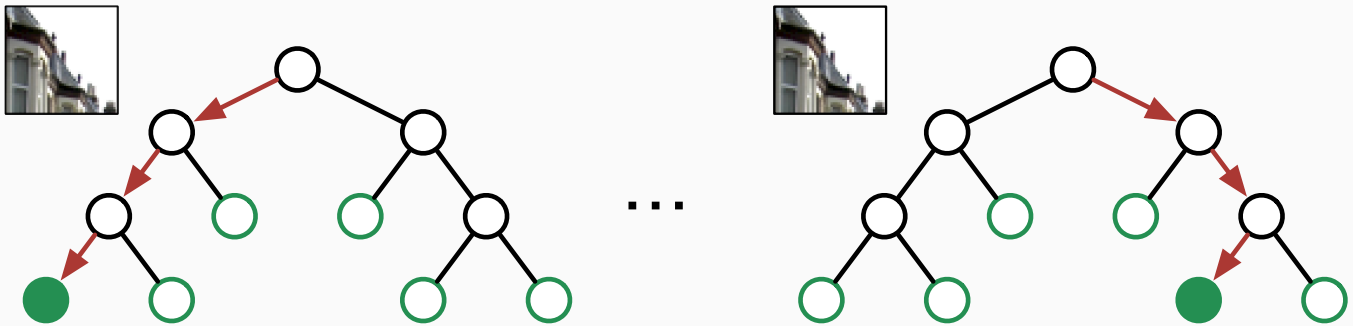
$$h(x, \theta_j) \in \{0, 1\}$$

with parameters θ_j. If $h(x, \theta_j) = 0$ node j sends x left, otherwise right.

Training Decision (Classification) Trees

Each tree is trained independently in a recursive manner. For a given node j and training set $\mathcal{S}_j \subset \mathcal{X} \times \mathcal{Y}$, the goal is to find parameters θ_j of the split function $h(x, \theta_j)$ that maximizes *Information Gain*, or equivalently, minimizing *Entropy*.

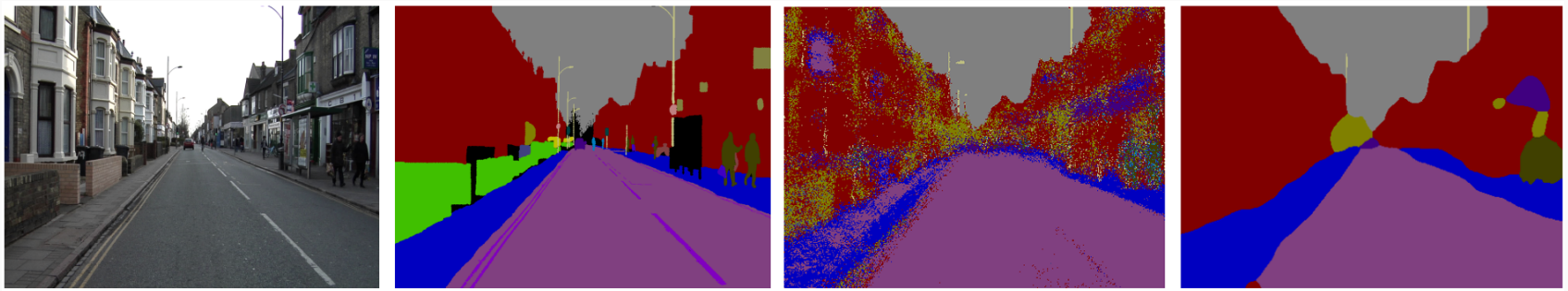
Random Forests



Randomness and Optimality

Individual decision trees exhibit high variance and tend to overfit. Decision forests ameliorate this by training multiple de-correlated trees and combining their output. **In effect, accuracy of individual trees is sacrificed in favor of high diversity ensemble.**

Structured Learning



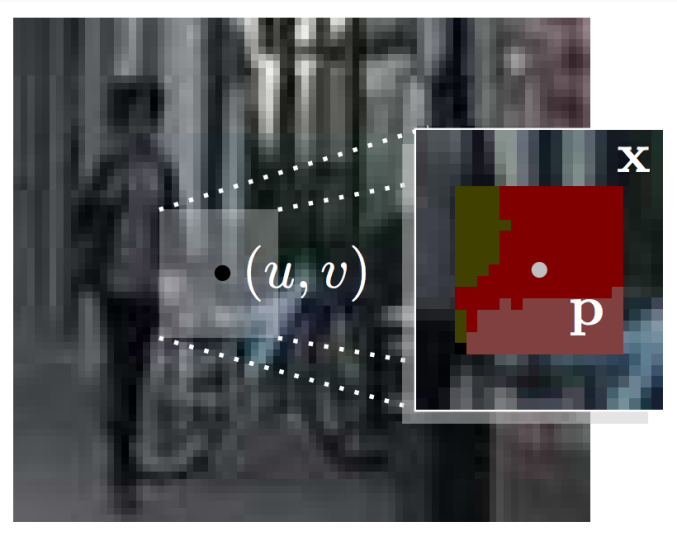
In traditional classification approaches, input data samples are assigned to single, *atomic* class labels, acting as arbitrary identifiers without any dependencies among them. For many computer vision problems however, this model is limited because the label space of a classification task exhibits an inherently topological structure. Therefore, **we try to address the problems by making the classifier aware of the local topological structure of the output label space.**

Kontschieder, Peter, et al. ICCV 11' [2]

Structured Random Forests

Overview

We extend random forests to general structured output spaces $\mathcal{Y}$. Of particular interest for computer vision is the case where $x \in \mathcal{X}$ represents an image patch and $y \in \mathcal{Y}$ encodes the corresponding local image annotation (e.g., a segmentation mask or set of semantic image labels).



Overview

Training random forests with structured labels is very challenging. Therefore, we want to reduce this problem to a simpler one.

- We use the observation that **approximate measures of information gain suffice to train effective random forest classifiers**. 'Optimal' splits are not necessary or even desired.
- Our core idea is to map all the structured labels $y \in \mathcal{Y}$ at a given node into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, ..., k\}$, such that *similar* structured labels y are assigned to the same discrete label c.
- Given $\mathcal{C}$, information gain calculated directly from $\mathcal{C}$ can serve as a proxy for the information gain over the structured labels $\mathcal{Y}$. As a result, at each node we can leverage existing random forest training procedures to learn structured random forests effectively.

Overview

Training random forests with structured labels is very challenging. Therefore, we want to reduce this problem to a simpler one.

- We use the observation that **approximate measures of information gain suffice to train effective random forest classifiers**. 'Optimal' splits are not necessary or even desired.
- Our core idea is to map all the structured labels $y \in \mathcal{Y}$ at a given node into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, ..., k\}$, such that *similar* structured labels y are assigned to the same discrete label c.
- Given $\mathcal{C}$, information gain calculated directly from $\mathcal{C}$ can serve as a proxy for the information gain over the structured labels $\mathcal{Y}$. As a result, at each node we can leverage existing random forest training procedures to learn structured random forests effectively.

Overview

Training random forests with structured labels is very challenging. Therefore, we want to reduce this problem to a simpler one.

- We use the observation that **approximate measures of information gain suffice to train effective random forest classifiers**. 'Optimal' splits are not necessary or even desired.
- Our core idea is to map all the structured labels $y \in \mathcal{Y}$ at a given node into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, ..., k\}$, such that *similar* structured labels y are assigned to the same discrete label c.
- Given $\mathcal{C}$, information gain calculated directly from $\mathcal{C}$ can serve as a proxy for the information gain over the structured labels $\mathcal{Y}$. As a result, at each node we can leverage existing random forest training procedures to learn structured random forests effectively.

Intermediate Mapping Π

For edge detection, the labels $y \in \mathcal{Y}$ are 16×16 segmentation masks. We first transform the output label patch to another space:

$$\Pi : \mathcal{Y} \to \mathcal{Z}$$

We define $z = \Pi(y)$ to be a long binary vector that encodes whether every pair of pixels in y belong to the same or different segments.

We therefore utilize a broadly applicable two-stage approach of first mapping $\mathcal{Y} \to \mathcal{Z}$ followed by a straightforward mapping of $\mathcal{Z} \to \mathcal{C}$.

Information Gain Criterion

We map a set of structured labels $y \in \mathcal{Y}$ into a discrete set of labels $c \in \mathcal{C}$, where $\mathcal{C} = \{1, ..., k\}$, such that labels with similar z are assigned tothe same discrete label c.

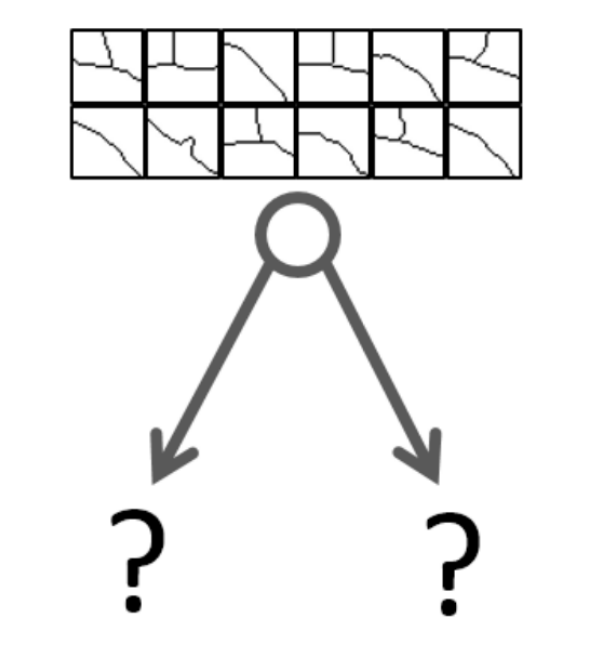
Get $\mathcal{C}$ from $\mathcal{Z}$

1. Cluster z into k clusters using K-means
2. Quantize z based on the top $\log_2(k)$ PCA dimensions

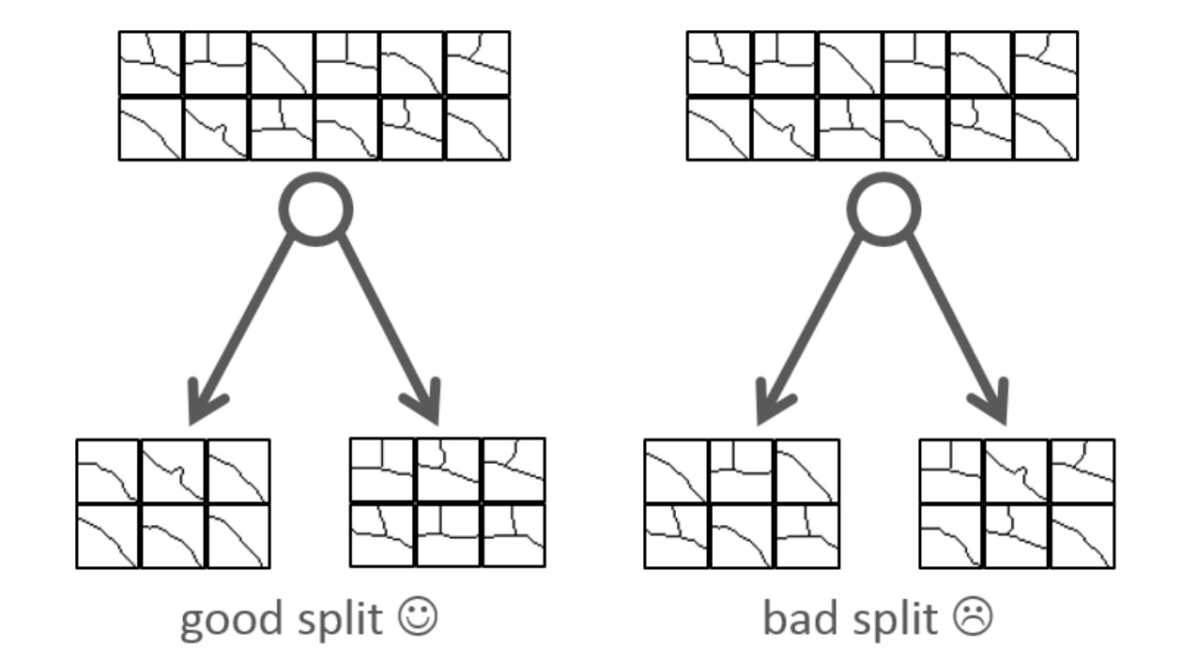
Both approaches perform similarly but the latter is slightly faster.

Now, the Structured Random Forest training problem is reduced to a ordinary random forest training problem.

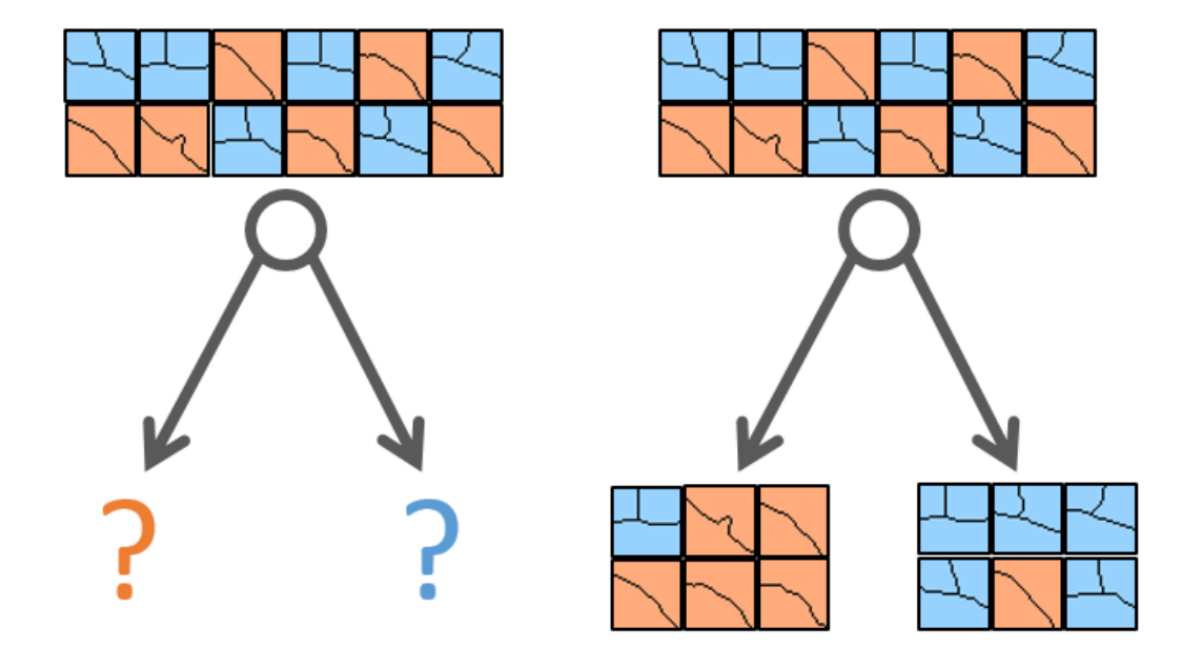
Training a Node in Action



Training a Node in Action



Training a Node in Action



Ensemble Model

To combine a set of n labels $y_1, y_2, \ldots, y_n$, we select the label y_k whose z_k is the medoid, i.e. the z_k that minimizes the distances to all other z_j.

Edge Detection

DEMO

Experiment Results

Overview

The experiments are performed on Berkeley Segmentation Dataset and Benchmark (BSDS500) and NYU Depth (NYUD) dataset.

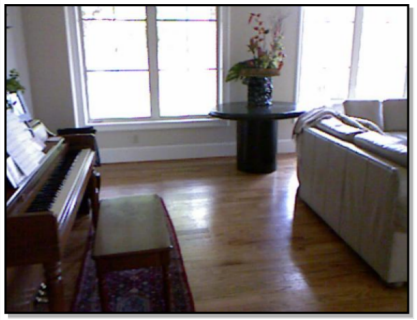
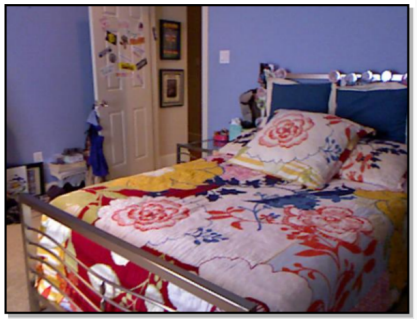
ODS Fixed contour threshold

OIS Per-image best threshold

AP Average Precision

R50 Recall at 50% precision


Examples from BSDS

Examples from NYUD

BSDS

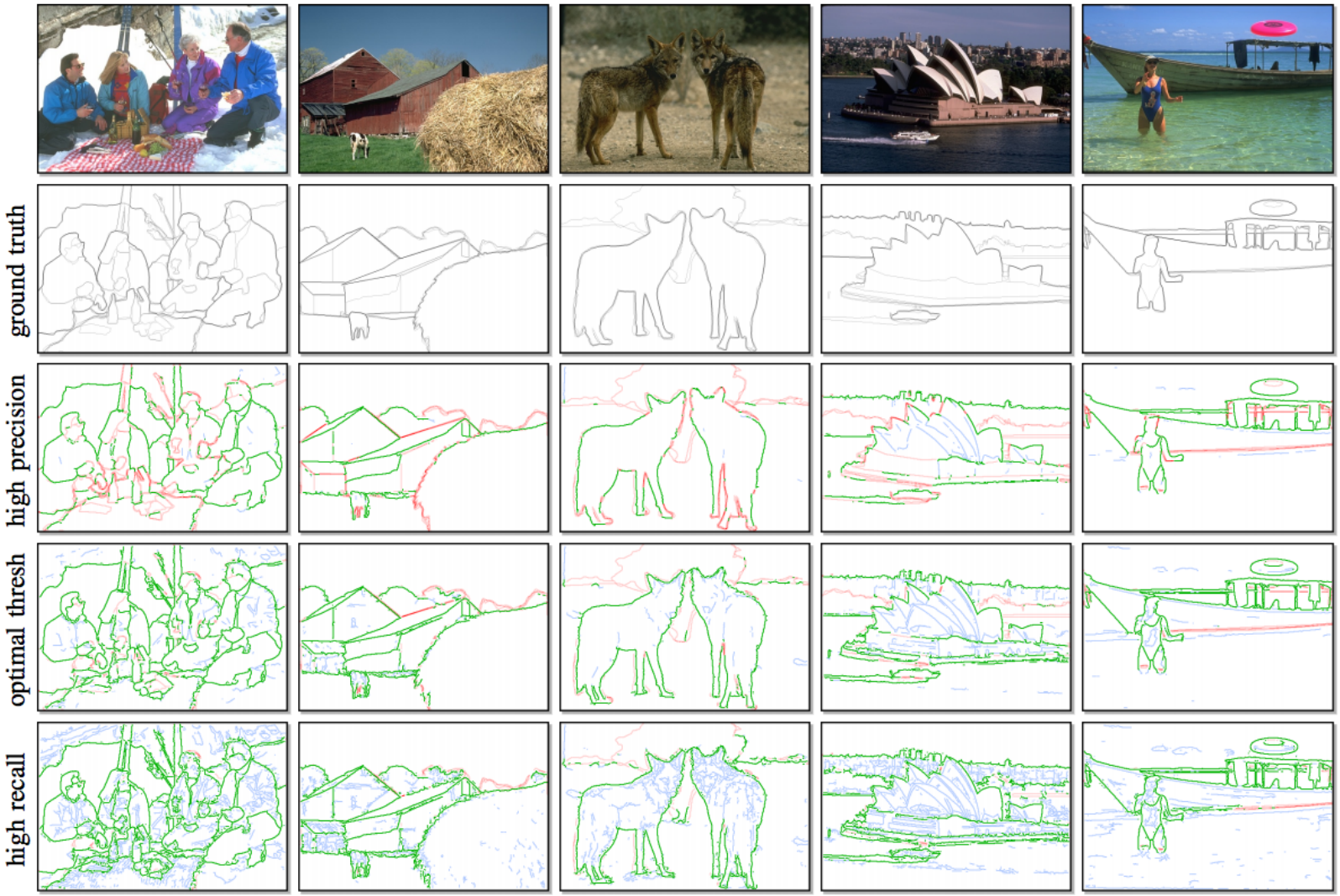
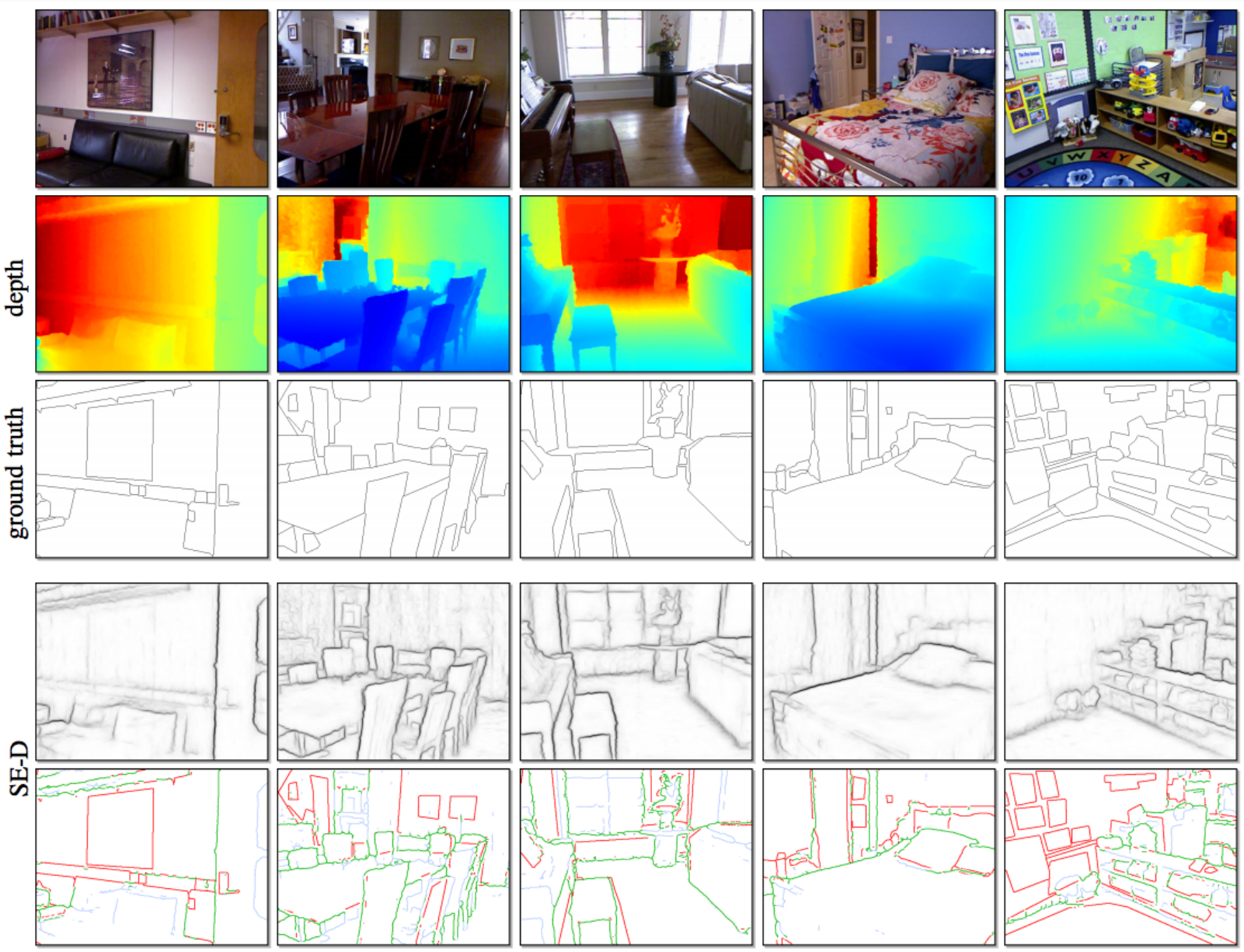


Fig. 4. Visualizations of matches and errors of SE+MS+SH compared to BSDS ground truth edges. Edges are thickened to two pixels for better visibility; the color coding is green=true positive, blue=false positive, red=false negative. Results are shown at three thresholds: high precision (T≈.26, P≈0.88, R=.50), ODS threshold (T≈.14, P=R≈.75), and high recall (T≈.05, P=.50, R≈0.93).

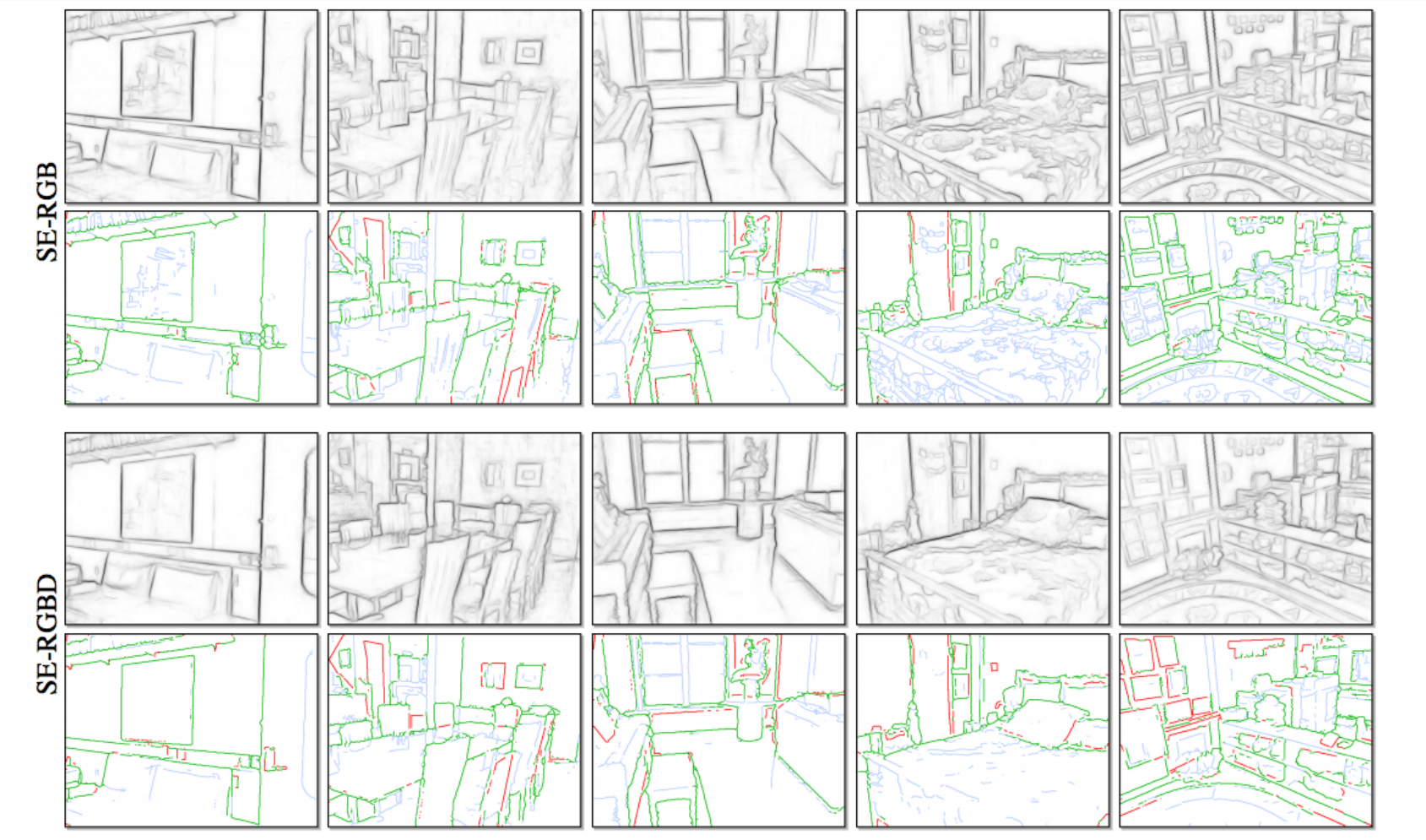
BSDS

	ODS	OIS	AP	R50	FPS
Human	.80	.80	-	-	-
Canny	.60	.63	.58	.75	15
Felz-Hutt [16]	.61	.64	.56	.78	10
Normalized Cuts [10]	.64	.68	.45	.81	-
Mean Shift [9]	.64	.68	.56	.79	-
Hidayat-Green [23]	.62†	-	-	-	20
BEL [13]	.66†	-	-	-	1/10
Gb [30]	.69	.72	.72	.85	1/6
gPb + GPU [8]	.70†	-	-	-	1/2‡
ISCRA [42]	.72	.75	.46	.89	1/30‡
gPb-owt-ucm [1]	.73	.76	.73	.89	1/240
Sketch Tokens [31]	.73	.75	.78	.91	1
DeepNet [27]	.74	.76	.76	-	1/5‡
SCG [41]	.74	.76	.77	.91	1/280
SE+multi-ucm [2]	**.75**	**.78**	.76	.91	1/15
SE	.73	.75	.77	.90	**30**
SE+SH	.74	.76	.79	**.93**	12.5
SE+MS	.74	.76	.78	.90	6
SE+MS+SH	**.75**	.77	**.80**	**.93**	2.5

NYUD



NYUD



NYUD

	ODS	OIS	AP	R50	FPS
gPb-owt-ucm [1]	.63	.66	.56	.79	1/360
Silberman [44]	.65	.66	.29	.84	1/360+
gPb+NG [21]	.68	.71	.63	.86	1/375
SE+NG+ [22]	**.71**	**.72**	**.74**	**.90**	1/15
SE-D	.64	.65	.66	.80	**7.5**
SE-RGB	.65	.67	.65	.84	**7.5**
SE-RGBD	.69	.71	.72	.89	5

Cross Dataset Generalization

	ODS	OIS	AP	R50	FPS
NYUD / NYUD	.65	.67	.65	.84	7.5
BSDS / NYUD	.64	.66	.63	.83	7.5
BSDS / BSDS	.75	.77	.80	.93	2.5
NYUD / BSDS	.73	.74	.77	.91	2.5

Train/Test

Across all performance measure, scores degrade by about 1 point when using the BSDS dataset. These experiments provide strong evidence that our approach could serve as a general purpose edge detector without the necessity of retraining.

Conclusion

Conclusion

1. Use structured learning to predict the labels for a patch a time, taking into consideration the spatial layout of the output label space
2. Generalized random forest training method using approximation

Questions?

References I


P. Dollár and C. L. Zitnick.
Fast edge detection using structured forests.
Pattern Analysis and Machine Intelligence, IEEE Transactions on, 37(8):1558–1570, 2015.


P. Kontschieder, S. Rota Bulò, H. Bischof, and M. Pelillo.
Structured class-labels in random forests for semantic image labelling.
In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2190–2197. IEEE, 2011.

Supplementary

Intermediate Mapping Π

$\mathcal{Z}$ may be high dimensional. For example, for edge detection there are $\binom{16\times16}{2} = 32640$ unique pixel pairs in a 16×16 segmentation mask, so computing z for every y would be expensive.

- We sample m dimensions of $\mathcal{Z}$, resulting in a reduced mapping $\Pi_\phi : \mathcal{Y} \to \mathcal{Z}$ parametrized by ϕ. During training, a distinct mapping Π_ϕ is randomly generated and applied to training labels $\mathcal{Y}_j$ at each node j.
- PCA to further reduce the dimensionality of $\mathcal{Z}$.

In practice, we use Π_ϕ with $m = 256$ dimensions followed by PCA projection to at most 5 dimensions.

Edge Detection Overview

Our learning approach predicts a structured 16×16 segmentation mask from a larger 32×32 image patch. Given an image, we predict a *segmentation mask* indicating segment membership for each pixel and a binary *edge map*.

Input Feature We construct a 7228 dimensional feature vector by considering color, scale, gradient and etc.

Mapping Function Let $y \in \mathcal{Y}$ be a 256 dimensional vector and z be a $\binom{256}{2}$ vector of the pairwise difference between every dimension of y. We reduce dimension of z to 256 and cluster to 2 clusters.

Ensemble Model The predictions are merged by simply averaging.

Efficiency Structured output is computed densely with a stride of 2 pixels, and we use a forest consists of 4 trees. Thus $16^2 \times 4/4 = 256$ votes per pixel.

Multiscale Detection (SE+MS) & Edge Sharpening (SE+SH)

Multiscale Detection (SE+MS)

Given an input image, we run our edge detection algorithm on original, half and double resolution version of the image and average the results.

Edge Sharpening (SE+SH)

We observed that predicted edge maps from our structured edge detector are somewhat diffuse. Therefore, we introduce a new *sharpening* procedure.

1. For each segment s, we compute its mean color μ_s
2. Iteratively update the assigned segment for each pixel by assigning it to the segment which minimizes $\|\mu_s - x(j)\|_2$