Structured Prediction Models in Computer Vision

Sebastian Nowozin

Machine Learning and Perception Group
Microsoft Research Cambridge

Guanajuato, 6th/7th September 2012
Computer Vision: Unique ML opportunities

1. *model uncertainty*: we do not know the “correct” model,
2. *ground truth available*: humans can solve high-level vision tasks easily,
3. *data availability*: enormous amounts of image data available.
Computer Vision: Unique ML opportunities

1. *model uncertainty*: we do not know the “correct” model,
2. *ground truth available*: humans can solve high-level vision tasks easily,
3. *data availability*: enormous amounts of image data available.

What kind of models to build?
Defense of simple models

_All models are wrong, but some are useful._  – George Box

_All models are wrong, and increasingly you can succeed without them._  – Peter Norvig
Simple Tasks, Shallow Models

- Image classification models: combine multiple features
- Caltech101/256, PASCAL VOC classification task, PASCAL ImageNet task
- (Gehler and Nowozin, ICCV 2009), LP-β

For simple computer vision tasks, simple shallow models are state-of-the-art.
Simple Tasks

Have

- well understood machine learning methods available (classification),
- data annotation is inexpensive,
- \(\rightarrow\) many benchmark data sets available,
- structured/hierarchical/deep/latent models unsuccessful (despite numerous attempts).

Why do we need structured models?
Simple Tasks

Have

- well understood machine learning methods available (classification),
- data annotation is inexpensive,
- many benchmark data sets available,
- structured/hierarchical/deep/latent models unsuccessful (despite numerous attempts).

Why do we need structured models?
Simple Tasks

Have

- well understood machine learning methods available (classification),
- data annotation is inexpensive,
- many benchmark data sets available,
- structured/hierarchical/deep/latent models unsuccessful (despite numerous attempts).

Why do we need structured models?

For simple tasks, we don’t need them.
And for more complex tasks?

- Human pose estimation (Ferrari et al., CVPR 2008)
- Scene understanding (Gould et al., ICCV 2009)
- Object detection (Everingham et al., VOC 2010 report)

Have in common,

- structured prediction of multiple dependent variables
- ground truth data expensive to obtain
- performance improvements by structured models
- learning is challenging
And for more complex tasks?

- Human pose estimation (Ferrari et al., CVPR 2008)
- Scene understanding (Gould et al., ICCV 2009)
- Object detection (Everingham et al., VOC 2010 report)

Have in common,

- structured prediction of multiple dependent variables
- ground truth data expensive to obtain
- performance improvements by structured models
- learning is challenging
Success cases: Richer models

- Deformable part models, Pictorial structure models
  - Object detection (Felzenszwalb et al., PAMI 2009), including “latent SVM”
  - Human pose estimation, (Felzenszwalb and Huttenlocher, IJCV 2005), (Andriluka et al., CVPR 2009)

- Conditional Random Fields for image labeling
- Tractable higher-order interactions
  - Enforcing consistency over large number of variables
    - (Kohli et al., CVPR 2007), (Kohli et al., CVPR 2008), (Rother et al., CVPR 2009), (Gonfaus et al., CVPR 2010), (Delong et al., CVPR 2010), (Ladicky et al., ECCV 2010)
Parameter Estimation in CRFs

- Conditional Random Field
  \[ p(y|x, w) = \frac{1}{Z(x, w)} \exp(-E(x, y, w)), \]

- Adding factors, latent variables, and features: enhance model capacity

- Model error = misspecification = approximation error

- Estimation error

- Optimization error, (Bottou and Bousquet, NIPS 2008)
Parameter Estimation in CRFs

- Conditional Random Field

\[
p(y|x, w) = \frac{1}{Z(x, w)} \exp(-E(x, y, w)),
\]

- Adding factors, latent variables, and features: enhance model capacity

- Model error = misspecification = approximation error

- Estimation error

- Optimization error, (Bottou and Bousquet, NIPS 2008)
Estimation and Prediction in CRFs

1. Estimation: KL-divergence minimization ⇔ MLE
2. Prediction function: predict y by minimizing the expected loss
   \[ f(x; w^*) = \arg\min_y \mathbb{E}_{z \sim p(z|x, w^*)} [\Delta(z, y)] \]
3. Test-time inference: evaluate \( f(x; w^*) \)
Misspecification (with Patrick Pletscher, Carsten Rother, and Pushmeet Kohli)

- Input: noisy image
- Output: denoised image
- A good model for denoising must respect the statistics of natural images.
Misspecification (cont)

- In computer vision, *model error* typically dominates everything else
- Model is *misspecified*
  - 1. MLE, 2. Minimize expected loss: still optimal?
Misspecification (cont)

- In computer vision, *model error* typically dominates everything else
- Model is *misspecified*
- 1. MLE, 2. Minimize expected loss: still optimal?
Risk Minimization

- Minimize the expected loss + regularizer
- Directly obtain a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- Consistency: asymptotically obtaining the predictor of smallest expected loss
- Methods: structured SVM (Tsochantaridis et al., JMLR 2006), inconsistent (McAllester, 2007)
Risk Minimization

- Minimize the expected loss + regularizer
- Directly obtain a prediction function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- **Consistency**: asymptotically obtaining the predictor of smallest expected loss
- **Methods**: structured SVM (Tsochantaridis et al., JMLR 2006), inconsistent (McAllester, 2007)
Denoising model

- Simple CRF model, discrete states (64)
- Unary interactions $E_u(x_i, y_i) = w_u(|x_i - y_i|)$
- Pairwise interactions $E_p(y_i, y_j) = w_p(|y_i - y_j|)$
- Loss function $\Delta$: mean squared error
No misspecification: setup

\[ x^{(n)} \xrightarrow{\text{add iid noise}} x^{(0)} \]

\[ p(y|x^{(n)}, w^{\text{true}}) \xrightarrow{\text{sample } y^{(n)}} \]

\[ \{(x^{(n)}, y^{(n)})\}_{n=1,...,N} \]

\[ p(\cdot|\cdot, w^{\text{mple}}) \]

\[ f(\cdot, w^{\text{mm}}) \]

\[ \Delta \]

\[ f'(\cdot, w^{\text{mple}}) \]
No misspecification: setup (cont)
No misspecification: sanity check
No misspecification: sanity check

- Top row, unary weights: true, mple, mm
- Bottom row, pairwise weights: true, mple, mm
Misspecification: setup

\[ x^{(n)} \xrightarrow{\text{add iid noise}} x^{(0)} \]

\[ q(y|x^{(n)}, w^{\text{true}}, w^{\text{miss}}) \xrightarrow{\text{sample } y^{(n)}} w^{\text{true}} w^{\text{miss}}(\epsilon) \]

\{ (x^{(n)}, y^{(n)}) \}_{n=1,...,N} \xrightarrow{\text{estimation}} p(\cdot|\cdot, w^{\text{mple}}) \xrightarrow{\text{risk minimization, loss } \Delta} f(\cdot, w^{\text{mm}}) \]

\[ f'(\cdot, w^{\text{mple}}) \]

▶ \[ q(y|x, w^{\text{true}}, w^{\text{miss}}) \]: use additional pairwise interactions between non-neighboring pixels

▶ As \( \epsilon \) increases model error increases
Misspecification: model

- For $\epsilon = 0$ we recover the original model
- For $\epsilon > 0$ we simulate Box’s “wrong model”
Misspecification: results

![Graph showing MSE vs ε for MM/MAP and MPLE/MMSE]
Misspecification: Conclusion

▶ “All models are wrong” $\rightarrow$ misspecification
▶ For the same model class, results suggest that risk minimization can be more robust to misspecification than MLE

▶ (MLE/MPLE estimation is agnostic about the loss function at the time when model error hits hardest: finding $w$)
Improving the overall accuracy

Reduce the

1. **model error**: add factors, features, latent variables, non-parametric,

2. **estimation error**: use more data and statistically efficient and consistent estimators, (Bayesian inference),

3. **optimization error**: better optimization methods for learning objectives (recent progress, e.g. COMID (Duchi et al., COLT 2010))
Discriminative Non-parametric Random Fields

Based on the following works:

- Nowozin, Rother, Bagon, Sharp, Yao, Kohli, ICCV 2011
  *Decision Tree Fields*

- Jancsary, Nowozin, Sharp, Rother, CVPR 2012
  *Regression Tree Fields – An Efficient, Non-parametric Approach to Image Labeling Problems*

- Jancsary, Nowozin, Rother, ECCV 2012
  *Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art*
CRFs: How do we use them?

- Factor graph notation (Kschischang, Frey, Loeliger, 1998)
- $x$: observed image
- $y_i, y_j$: dependent variables at pixel $i$ and $j$
CRFs: How do we use them?

- Unary energy $E_A(y_i, x)$
- Machine learning (SVM, Boosting, Random Forests, etc.)
CRFs: How do we use them?

- Pairwise energy $E_C(y_i, y_j, x)$
- Generalized Potts, image independent
- Contrast-sensitive smoothing (e.g. GrabCut, TextonBoost)

$$E_C(y_i, y_j, x) = I(y_i \neq y_j) \cdot \exp(-\alpha \|x_i - x_j\|^2)$$
CRFs: How do we use them?

$E_A(y_i, x) \quad A \quad E_C(y_i, y_k, x) \quad C \quad y_i \quad y_j \quad y_k

???
CRFs: How do we use them?

$$E_A(y_i, x)$$

$$E_C(y_i, y_j, y_k, x)$$
## Decision Trees in Computer Vision

- Random Forests (Breiman, MLJ 2000)
- Non-parametric, infinite model capacity
- Fast inference and training, parallelizable
- (Shotton et al., 2008, 2011), (Saffari et al., 2009), (Gall and Lempitsky, 2009), etc.
- **No structured prediction**
Decision Tree Classifiers
### Decision Tree Classifiers

![Decision Tree Classifier Diagram]

1. **Input**: $x$
2. **Decision Nodes**: Split the input $x$ into subsets based on certain criteria.
3. **Leaf Nodes**: Assign a class label or value to each subset.

---

**Sebastian Nowozin**

Microsoft Research Cambridge

**Structured Prediction Models in Computer Vision**
Decision Trees for Image Labeling
Decision Trees for Image Labeling (cont)

- Apply decision tree, to each pixel \textit{independently}
Decision Trees for Image Labeling (cont)

- Apply decision tree, to each pixel *independently*
Decision Tree Field (DTF) Example

![Decision Tree Field Example](image_url)
Unary Factor Example

- $x$: entire observed image
- $y_i$: prediction at pixel $i$, $y_i \in \{1, 2, 3, 4\}$
- $E_A(y_i, x)$: energy function
Unary Factor Example

\[ E_A(y_i, x) \]

\[ x \]

\[ y_i \]
Unary Factor Example

\[ E_A(y_i, x) \]

Sebastian Nowozin
Microsoft Research Cambridge
Structured Prediction Models in Computer Vision
Unary Factor Example

\[ E_A(y_i, x) \]

\[ \mathbf{y}_i \]
Unary Factor Example

\[ E_A(y_i, x) \]
Unary Factor Example

\[ E_A(y_i, x) = \sum_{q \in \text{Path}(x)} w_A(q, y_i) \]
**Pairwise Factor Example**

\[ E_A(y_i, x) \]

\[ E_C(y_i, y_j, x) \]
Pairwise Factor Example

\[ E_A(y_i, x) \]

\[ E_C(y_i, y_j, x) \]
Pairwise Factor Example

$$E_A(y_i, x)$$

$$E_C(y_i, y_j, x)$$
Pairwise Factor Example

\[ E_A(y_i, x) \]

\[ E_C(y_i, y_j, x) \]
Pairwise Factor Example

\[ E_A(y_i, x) \]
\[ E_C(y_i, y_j, x) \]
Pairwise Factor Example

\[ E_A(y_i, x) \]

\[ E_C(y_i, y_j, x) = \sum_{q \in \text{Path}(x)} w_A(q, y_i, y_j) \]
Full DTF Model

\[
E(y, x, w) = \sum_{F \in \mathcal{F}} E_{t_F}(y_F, x_F, w_{t_F}).
\]

\[
p(y|x, w) = \frac{1}{Z(x, w)} \exp(-E(y, x, w)),
\]

\[
Z(x, w) = \sum_{y \in \mathcal{Y}} \exp(-E(y, x, w))
\]

- \(x\): image, \(y\): predicted labels, one for each pixel
- \(w\): weights/energies, to be learned from data
- How is this different from other models?
- What about learning and inference?
Full DTF Model

\[
E(y, x, w) = \sum_{F \in \mathcal{F}} E_{t_F}(y_F, x_F, w_{t_F}).
\]

\[
p(y|x, w) = \frac{1}{Z(x, w)} \exp(-E(y, x, w)),
\]

\[
Z(x, w) = \sum_{y \in \mathcal{Y}} \exp(-E(y, x, w))
\]

- \(x\): image, \(y\): predicted labels, one for each pixel
- \(w\): weights/energies, to be learned from data
- How is this different from other models?
- What about learning and inference?
Relationship to Other Models

- Generalizes random forests (learned weights)
- Markov random fields

Here 2 trees
Relationship to Other Models

- Generalizes random forests (learned weights)
- Markov random fields
Learning DTFs

Given iid data \( \{(x, y)_i\}_{i=1,...,N} \), need to learn

- Structure of the factor graph,
- Tree structure defined by split functions,
- Weight parameters in decision nodes.

Let us assume structure and trees are given...
Learning DTFs

Given iid data $\{(x, y)_i\}_{i=1,\ldots,N}$, need to learn

- Structure of the factor graph,
- Tree structure defined by split functions,
- Weight parameters in decision nodes.

Let us assume structure and trees are given.
## Training

Maximum Likelihood Estimation, given ground truth $y^*$

$$w^* = \arg\max_w \log p(y^* | x, w)$$
Training

Maximum Likelihood Estimation, given ground truth $y^*$

$$w^* = \arg\max_w \log p(y^* | x, w)$$

Intractable!
Training
Training
**Training**

Maximum Pseudo-Likelihood Estimation (Besag, 1974)

\[
    \mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \log p(y_i | y_{\mathcal{V} \setminus \{i\}}, \mathbf{x}, \mathbf{w})
\]

with ground truth \( y^* \) and

\( \mathcal{V} : \) set of pixels in all images.

(details in paper)
Efficient Training by Subsampling

1M pixels

...
Efficient Training by Subsampling

1M pixels
Efficient Training by Subsampling

We subsample variables within each instance to train a structured model.
Training Algorithm

1. Fix factor graph structure
2. For each factor: learn classification tree
3. Jointly optimize convex pseudo-likelihood objective in $\mathbf{w}$
Test-time Inference in DTFs

1. Energy minimization (MAP)
   E.g. TRW-S

2. Maximum Posterior Marginal (MPM)
   E.g. Gibbs sampling
Experiment: Inpainting
Experiment: Inpainting
Experiment: Inpainting
Experiment: Inpainting
Experiment: Inpainting

Training set (300 images)
# Experiment: Inpainting

<table>
<thead>
<tr>
<th>Input</th>
<th>Truth</th>
<th>RF posterior</th>
<th>MAP</th>
<th>MRF posterior</th>
<th>MAP</th>
<th>DTF posterior</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Test set (100 images, disjoint characters)
Instances

- Densely-connected, 64 neighbors
- Each instance: 10k variables, 300k factors
- → hard to minimize energy

www.nowozin.net/sebastian/papers/DTF_CIP_instances.zip
Instances

- Densely-connected, 64 neighbors
- Each instance: 10k variables, 300k factors
- → hard to minimize energy

www.nowozin.net/sebastian/papers/DTF_CIP_instances.zip
Experiment: Body-part Recognition

- Body part recognition (Shotton et al., CVPR 2011)
- 1500 training images, 150 test images
Experiment: Body-part Recognition (cont)

Training set
Experiment: Body-part Recognition, Results

Input / Truth  unary  +1  +1,5,20

MRF  DTF
## Experiment: Body-part Recognition, Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Measure</th>
<th>Shotton et al. avg-acc</th>
<th>unary avg-acc</th>
<th>+1 avg-acc</th>
<th>+1,20 avg-acc</th>
<th>+1,5,20 avg-acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRF</td>
<td>avg-acc</td>
<td>34.4</td>
<td>36.15</td>
<td>37.82</td>
<td>38.00</td>
<td>39.30</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>6h34</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>(30h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>6.3M</td>
<td>6.2M</td>
<td>6.2M</td>
<td>6.3M</td>
</tr>
<tr>
<td>DTF</td>
<td>avg-acc</td>
<td>-</td>
<td>-</td>
<td>39.59</td>
<td>40.26</td>
<td>41.42</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>(40h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>-</td>
<td>6.8M</td>
<td>7.8M</td>
<td>8.8M</td>
</tr>
</tbody>
</table>
## Experiment: Body-part Recognition, Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Measure</th>
<th>Shotton et al.</th>
<th>unary</th>
<th>+1</th>
<th>+1,20</th>
<th>+1,5,20</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRF</td>
<td>avg-acc</td>
<td>34.4</td>
<td>36.15</td>
<td>37.82</td>
<td>38.00</td>
<td>39.30</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>6h34</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>(30h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>6.3M</td>
<td>6.2M</td>
<td>6.2M</td>
<td>6.3M</td>
</tr>
<tr>
<td>DTF</td>
<td>avg-acc</td>
<td>-</td>
<td>-</td>
<td>39.59</td>
<td>40.26</td>
<td>41.42</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>(40h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>-</td>
<td>6.8M</td>
<td>7.8M</td>
<td>8.8M</td>
</tr>
</tbody>
</table>
## Experiment: Body-part Recognition, Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Measure</th>
<th>Shotton et al.</th>
<th>unary</th>
<th>+1</th>
<th>+1,20</th>
<th>+1,5,20</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRF</td>
<td>avg-acc</td>
<td>34.4</td>
<td>36.15</td>
<td>37.82</td>
<td>38.00</td>
<td>39.30</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>6h34</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>(30h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>6.3M</td>
<td>6.2M</td>
<td>6.2M</td>
<td>6.3M</td>
</tr>
<tr>
<td>DTF</td>
<td>avg-acc</td>
<td>-</td>
<td>-</td>
<td>39.59</td>
<td>40.26</td>
<td><strong>41.42</strong></td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>(40h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>-</td>
<td>6.8M</td>
<td>7.8M</td>
<td>8.8M</td>
</tr>
</tbody>
</table>
Experiment: Body-part Recognition, Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Measure</th>
<th>Shotton et al.</th>
<th>unary</th>
<th>+1</th>
<th>+1,20</th>
<th>+1,5,20</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRF</td>
<td>avg-acc</td>
<td>34.4</td>
<td>36.15</td>
<td>37.82</td>
<td>38.00</td>
<td>39.30</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>6h34</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>(30h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>6.3M</td>
<td>6.2M</td>
<td>6.2M</td>
<td>6.3M</td>
</tr>
<tr>
<td>DTF</td>
<td>avg-acc</td>
<td>-</td>
<td>-</td>
<td>39.59</td>
<td>40.26</td>
<td>41.42</td>
</tr>
<tr>
<td></td>
<td>runtime</td>
<td>-</td>
<td>-</td>
<td>*</td>
<td>*</td>
<td>(40h)*</td>
</tr>
<tr>
<td></td>
<td>weights</td>
<td>-</td>
<td>-</td>
<td>6.8M</td>
<td>7.8M</td>
<td>8.8M</td>
</tr>
</tbody>
</table>
DTF Summary

- Decision Tree Fields: non-parametric CRF model for discrete image labeling tasks
  - Non-parametric: model class can scale with training set size
  - Scalable: make use of large training sets,
  - Conditional interactions: richer models without latent variables

But:

- independent learning of trees and weights,
- learning uses pseudolikelihood approximation,
- test-time inference may be slow or hard.
DTF Summary

- Decision Tree Fields: non-parametric CRF model for discrete image labeling tasks

- Non-parametric: model class can scale with training set size
- Scalable: make use of large training sets,
- Conditional interactions: richer models without latent variables

But:

- independent learning of trees and weights,
- learning uses pseudolikelihood approximation,
- test-time inference may be slow or hard.
Improving DTFs

What make DTFs work,

1. Rich non-parametric interaction functions,
2. A way to aggregate the outputs of these functions (random field).

Three improvements:

- use another way to aggregate (Gaussian CRFs, efficient and tractable),
- train interactions and field model jointly,
- optimize task-specific loss functions (empirical risk minimization).
Improving DTFs

What make DTFs work,

1. Rich non-parametric interaction functions,
2. A way to aggregate the outputs of these functions (random field).

Three improvements:

- use another way to aggregate (Gaussian CRFs, efficient and tractable),
- train interactions and field model jointly,
- optimize task-specific loss functions (empirical risk minimization).
Gaussian CRFs

\[ E(y|x; w) := \frac{1}{2} y^\top Q(x, w)y - y^\top l(x, w) \]

- Gaussian CRF, (Tappen et al, CVPR 2007)
- \( y_i \in \mathbb{R}^d \), one vector per pixel
- \( Q(x, w) \), data-dependent sparse psd matrix
- \( \hat{y}(x; w) := \text{argmin}_y E(y|x; w) \), prediction
Regression Tree Fields

\[ E_F(y_F | x_F; w_F) := \frac{1}{2} y_F^T Q(x_F) y_F - y_F^T L(x_F) b(x_F) \]

- \( y_F = (y_i^T, y_j^T)^T \in \mathbb{R}^{2d} \) stacked vector
- \( Q(x_F) \) psd quadratic parameters
- \( L(x_F) \) linear parameters
- \( b(x_F) \) basis function (e.g. filter responses)
Regression Tree Fields

\[
E_F(y_F|x_F; w_F) := \frac{1}{2} y_F^\top Q(x_F)y_F - y_F^\top L(x_F)b(x_F)
\]

\[
E(y|x; w) := \sum_t \sum_{F \in \mathcal{F}_t} E_F(y_F|x_F; w_F)
\]
Test-time Evaluation

1. Evaluate all regression trees to build quadratic form
2. Minimize quadratic form (conjugate gradients)
3. Output one vector in $\mathbb{R}^d$ for each pixel
Training: Empirical Risk Minimization

\[ \hat{R}_\ell(D, w) := \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{y}(x^{(i)}, w), y^{(i)}) \]

- Training set \( D = \{(x^{(i)}, y^{(i)})\}_{i=1,...,N} \)
- Parameters \( w \)
- Current model prediction \( \hat{y}(x^{(i)}, w) \)
- Task-specific loss function \( \ell(\hat{y}, y) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \)
Joint Training
Joint Training
Joint Training

- Functional gradient method
- Select split to maximize resulting learning objective gradient norm
- Intuition: “large gradient” → “reduction in training objective”
Joint Training (cont)

![Graph showing MSE vs. tree depth for different criteria.](image)

- **Squared residuals criterion**
- **MSE gradient norm criterion**

Sebastian Nowozin
Microsoft Research Cambridge
Structured Prediction Models in Computer Vision
Joint Training (cont)
Image Denoising (again)

\[ x = f(y) \]
\[ \hat{y} = f^{-1}(x) \]
Structured Noise
Denoising Loss Functions

- Pred 1: optimized for ISSIM, MSE 313, ISSM 0.932
- Pred 2: optimized for MSE, MSE 273, ISSIM 0.893
- Common losses: MSE, PSNR, MAD, SSIM, ISSIM
Image Denoising

- **FoE**, (Schmidt et al., CVPR 2010), (Roth and Black, IJCV 2009)
- **BM3D**, (Dabov et al., 2007, TIP)
- **LSSC**, (Mairal et al., ICCV 2009)
- **EPLL**, (Zoran and Weiss, ICCV 2011)
Experiment: Setup

- BSDS500 database: 200 train, 100 val, 200 test images
- Gaussian noise, $\sigma \in \{20, 30, 40, 50\}$
- Features: filterbank, outputs of four denoising methods
- Strict setup: one test set evaluation, statistical hypothesis tests
### Experiment: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (↑ better)</th>
<th>MAE (↓ better)</th>
<th>SSIM (↑ better)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>σ 20 30 40 50</td>
<td>20 30 40 50</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>22.11 18.59 16.09 14.15</td>
<td>15.96 23.93 31.91 39.89</td>
<td>0.541 0.401 0.307 0.242</td>
</tr>
<tr>
<td>FoE [11]</td>
<td>28.87 26.81 25.45 24.47</td>
<td>6.79 8.56 10.03 11.24</td>
<td>0.848 0.776 0.712 0.660</td>
</tr>
<tr>
<td>BM3D [9]</td>
<td>29.25 27.32 25.98 25.09</td>
<td>6.40 7.95 9.25 10.22</td>
<td>0.855 0.793 0.741 0.699</td>
</tr>
<tr>
<td>LSSC [5]</td>
<td>29.40 27.39 26.08 25.09</td>
<td>6.39 7.96 9.23 10.33</td>
<td>0.861 0.799 0.745 0.700</td>
</tr>
<tr>
<td>EPPL [4]</td>
<td>29.38 27.44 26.17 25.22</td>
<td>6.37 7.90 9.12 10.17</td>
<td>0.864 0.800 0.747 0.703</td>
</tr>
<tr>
<td>UNIFORM AVG</td>
<td>29.47 27.50 26.21 25.25</td>
<td>6.30 7.84 9.08 10.12</td>
<td>0.863 0.802 0.749 0.705</td>
</tr>
<tr>
<td>PSNRRTF Plain</td>
<td>28.95 26.97 25.71 24.76</td>
<td>6.78 8.44 9.72 10.85</td>
<td>0.840 0.771 0.716 0.666</td>
</tr>
<tr>
<td>PSNRRTF Bm3D</td>
<td>29.52 27.58 26.24 25.38</td>
<td>6.23 7.73 8.99 9.92</td>
<td>0.863 0.803 0.750 0.711</td>
</tr>
<tr>
<td>PSNRRTF All</td>
<td>29.67 27.72 26.43 25.51</td>
<td>6.14 7.62 8.80 9.78</td>
<td>0.868 0.809 0.758 0.717</td>
</tr>
<tr>
<td>MAE RTF Plain</td>
<td>28.92 26.94 25.69 24.75</td>
<td>6.78 8.43 9.71 10.81</td>
<td>0.840 0.771 0.715 0.669</td>
</tr>
<tr>
<td>MAE RTF Bm3D</td>
<td>29.53 27.58 26.22 25.36</td>
<td>6.21 7.71 8.96 9.88</td>
<td>0.863 0.803 0.750 0.711</td>
</tr>
<tr>
<td>MAE RTF All</td>
<td>29.67 27.72 26.43 25.50</td>
<td>6.12 7.59 8.77 9.74</td>
<td>0.867 0.808 0.758 0.717</td>
</tr>
<tr>
<td>SSMRTF Plain</td>
<td>28.49 26.55 25.31 24.41</td>
<td>7.17 8.92 10.23 11.34</td>
<td>0.844 0.778 0.721 0.676</td>
</tr>
<tr>
<td>SSMRTF Bm3D</td>
<td>29.17 27.13 25.69 24.85</td>
<td>6.60 8.31 9.80 10.79</td>
<td>0.868 0.809 0.757 0.719</td>
</tr>
<tr>
<td>SSMRTF All</td>
<td>29.23 27.14 25.67 24.75</td>
<td>6.60 8.39 9.96 11.06</td>
<td>0.872 0.815 0.766 0.726</td>
</tr>
<tr>
<td>NLPLRTF Plain</td>
<td>28.61 26.66 25.32 24.42</td>
<td>7.09 8.80 10.28 11.37</td>
<td>0.828 0.758 0.694 0.653</td>
</tr>
<tr>
<td>NLPLRTF Bm3D</td>
<td>29.43 27.44 26.10 25.21</td>
<td>6.32 7.88 9.16 10.13</td>
<td>0.861 0.799 0.747 0.708</td>
</tr>
<tr>
<td>NLPLRTF All</td>
<td>29.60 27.64 26.34 25.40</td>
<td>6.20 7.71 8.92 9.93</td>
<td>0.866 0.806 0.755 0.714</td>
</tr>
</tbody>
</table>
## Experiment: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (↑ better)</th>
<th>( \sigma ) = 20</th>
<th>( \sigma ) = 30</th>
<th>( \sigma ) = 40</th>
<th>( \sigma ) = 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td>22.11</td>
<td>18.59</td>
<td>16.09</td>
<td>14.15</td>
</tr>
<tr>
<td>BM3D [9]</td>
<td></td>
<td>29.25</td>
<td>27.32</td>
<td>25.98</td>
<td>25.09</td>
</tr>
<tr>
<td>UNIFORM AVG</td>
<td></td>
<td>29.47</td>
<td>27.50</td>
<td>26.21</td>
<td>25.25</td>
</tr>
<tr>
<td>( PSNRRTF_{PLAIN} )</td>
<td></td>
<td>28.95</td>
<td>26.97</td>
<td>25.71</td>
<td>24.76</td>
</tr>
<tr>
<td>( PSNRRTF_{BM3D} )</td>
<td></td>
<td>29.52</td>
<td>27.58</td>
<td>26.24</td>
<td>25.38</td>
</tr>
<tr>
<td>( PSNRRTF_{ALL} )</td>
<td></td>
<td>29.67</td>
<td>27.72</td>
<td>26.43</td>
<td>25.51</td>
</tr>
</tbody>
</table>
Experiment: Results (cont)
Experiment: Results (cont)

- RTF significantly outperforms all competitors
- On every performance measure (PSNR, MAE, SSIM)
- On every test image

- Reason 1: Loss function matters
- Reason 2: Image-dependent combination of baseline methods
Generative Models

(Joint work with John Winn)
Simulation: Physically Accurate

Scene “Sala”, 488k triangles
LuxMark photorealistic output (2 minutes)
Monte Carlo simulation of light transport
Simulation: Perceptually Accurate

In-game screenshot, “Battlefield 3”
50 frames per second
Dedicated Simulation Hardware

AMD Radeon 7970 GCN
2048 cores at 925MHz
3.8 TFLOP/s single precision
**Generative Computer Vision**

- Ulf Grenander, 1970’s-now: *Pattern Theory*

Broad scope, key points:

1. Stochastic generative models for real world signals
2. Meaningful hidden variables representing physical realities
3. Prior distributions for hidden variables
4. Simulation
Generative Computer Vision

- Horn, 1977: *Understanding Image Intensities*

- Image formation process
- Effect of materials, shape, reflectance on observation
Generative Computer Vision

- Horn, 1977: *Understanding Image Intensities*

  “... we must still understand how the world forms an image if we are to make machines see. Yet I do not mean to suggest analysis by synthesis. Nothing of the sort! I propose only that if we are to solve the problem of creating computer performance in this domain, we must first thoroughly understand that domain.”
Generative Computer Vision (cont)


Assume known scene geometry
- Recover lighting and reflectance from an observed image
- Method: render basis images, then use least-squares reconstruction
Generative Computer Vision (cont)

- Han and Zhu, 2003
  *Bayesian Reconstruction of 3D Shapes and Scenes From A Single Image*

- Define generative models for polyhedra, tree and plants
- Intermediate “sketch” graph representation
- Bayesian inference given observed image (MCMC)
Generative Computer Vision (cont)

- Wingate, Goodman, Stuhlmüller, Siskind, 2003
  *Nonstandard Interpretations of Probabilistic Programs for Efficient Inference*

- Define renderer as probabilistic program
- Condition renderer on observed image
- Bayesian inference (intelligent MCMC)
Generative Computer Vision (cont)

- Wingate, Goodman, Stuhlmüller, Siskind, 2003
  *Nonstandard Interpretations of Probabilistic Programs for Efficient Inference*

- Define *renderer* as probabilistic program
- Condition renderer on observed image
- Bayesian inference (intelligent MCMC)
Probabilistic Programming

- Program = generative model
- Running the program produces a sample \((x, y) \sim p(x, y)\)
- Conditioning on observation: \(p(y|x)\)
- *How to invert the program?*

Example: Localization

- Latent variables: \((x, y, z)\) position, \((\theta_x, \theta_y, \theta_z)\) orientation
- Observation: depth image
Example: Localization (cont)

```cpp
// Complete rendering function of house interior
void renderer_ins(RFPP::Context& ctx) {
    ctx.push_function("render");

    // Create z-buffer
    std::vector<std::vector<double>> zbuffer(N);
    for (size_t yi = 0; yi < zbuffer.size(); ++yi) {
        zbuffer[yi].resize(N);
        std::fill(zbuffer[yi].begin(), zbuffer[yi].end(), 100.0);
    }

    // List of polygons
    std::vector<std::vector<std::vector<double>>> polys = {
        {0, 0, 54, 1}, {0, 10, 54, 1}, {8, 16, 54, 1}, {16, 10, 54, 1}, {16, 0, 54, 1} }, // Backplane
        {0, 0, 4, 1}, {0, 10, 4, 1}, {8, 16, 4, 1}, {16, 10, 4, 1}, {16, 0, 4, 1} }, // Frontplane
        {0, 0, 54, 1}, {16, 0, 54, 1}, {16, 0, 4, 1}, {0, 0, 4, 1} }, // Ground plane
        {0, 0, 4, 1}, {0, 10, 4, 1}, {0, 10, 54, 1}, {0, 0, 54, 1} }, // Left side-plane
        {16, 0, 54, 1}, {16, 10, 54, 1}, {16, 10, 4, 1}, {16, 0, 4, 1} }, // Right side-plane
        {8, 16, 54, 1}, {0, 10, 54, 1}, {0, 10, 4, 1}, {8, 16, 4, 1} }, // Left roof-plane
        {16, 10, 54, 1}, {8, 16, 54, 1}, {8, 16, 4, 1}, {16, 10, 4, 1} }, // Right roof-plane
    }
```
Example: Localization (cont)

```c
// Setup camera
camera_t cam;
cam.alpha = 50.0;  // viewing angle in degrees
double stay_away = 6.0;
cam.c_x = ctx.rand_uniform("camera-pos-x", stay_away, 16.0-stay_away, 0);
cam.c_y = ctx.rand_uniform("camera-pos-y", stay_away, 16.0-stay_away, 0);
cam.c_z = ctx.rand_uniform("camera-pos-z", 20.0, 38.0, 0.1);
cam.theta_z = ctx.rand_uniform("camera-rot-z", -0.2, 0.2, 0.01);  //
cam.theta_y = ctx.rand_uniform("camera-rot-y", 0, 6.2831853, 0.05);  //
cam.theta_x = ctx.rand_uniform("camera-rot-x", 0, 6.2831853, 0.05);  //
cam.screen_width = N;
cam.screen_height = N;
camera_compute_N(cam);  // compute planar projection operator

// Render polygons into zbuffer
for (size_t pi(0); pi < polys.size(); ++pi)
    render_3d_polygon(zbuffer, polys[pi], cam);
```
Example: Localization (cont)

```c
// Output noisy depth observations from zbuffer
ctx.push_loop("output-y");
for (size_t y = 0; y < N; ++y) {
    ctx.push_loop("output-x");
    for (size_t x = 0; x < N; ++x) {
        // Additive Gaussian noise
        zbuffer[y][x] = ctx.rand_normal("noise", zbuffer[y][x], 1);
        ctx.increment_loop(); // output-x
    }
    ctx.pop_loop();
    ctx.increment_loop(); // output-y
}
ctx.pop_loop();
ctx.pop_function();
```
Example: Localization (cont)
Synthetic Experiment

1. Simulate from the model: camera variables and image
2. Forget camera variables
3. Infer posterior distribution over camera variables, given the image
Inference Results

Ground truth rendering

Camera rotation angles
Inference Results (another example)
Inference Results (cont)
Example: Dead Leaves

- Dead leaves, falling on top of each other
- Simple model for occlusion
Example: Dead Leaves (cont)

```c
// Draw a random number of boxes to layer
int number_of_boxes = ctx.rand_uniform_int("number of boxes", 4, 12);
ctx.push_loop("draw box");
for (int bi = 0; bi < number_of_boxes; ++bi) {
    int w(1);
    int x1 = ctx.rand_uniform_int("box x1", 0, N-2);
    int y1 = ctx.rand_uniform_int("box y1", 0, N-2);
    int w1, h1;
    if (ctx.rand_bernoulli("order", 0.5)) {
        w1 = ctx.rand_uniform_int("box w1", 1, N-x1-1);
        h1 = ctx.rand_uniform_int("box h1", 1, std::min(N-y1-1, w));
    } else {
        w1 = ctx.rand_uniform_int("box w2", 1, std::min(N-x1-1, w));
        h1 = ctx.rand_uniform_int("box h2", 1, N-y1-1);
    }
    int top_left_x = x1;
    int top_left_y = y1;
    int bottom_right_x = x1 + w1;
    int bottom_right_y = y1 + h1;
    int box_color = ctx.rand_uniform_int("box color", 1, 10);
    for (int yi = top_left_y; yi <= bottom_right_y; ++yi)
        for (int xi = top_left_x; xi <= bottom_right_x; ++xi)
            image[yi][xi] = box_color;
}
ctx.increment_loop();
ctx.pop_loop();
```
Example: Dead Leaves (cont)

```c
// Add noise (observation model)
ctx.push_loop("add noise y");
for (int yi = 0; yi < N; ++yi) {
    ctx.push_loop("add noise x");
    for (int xi = 0; xi < N; ++xi) {
        // Additive Skellam noise
        image[yi][xi] = ctx.rand_int_skellam("noise", image[yi][xi]);
    }
    ctx.increment_loop(); // add noise x
}
ctx.pop_loop();

ctx.increment_loop(); // add noise y
}
ctx.pop_loop();
```
Example: Dead Leaves (cont)
Inference Challenges

Challenges

- Opaque dependency structure
- Point-wise likelihood evaluations, no gradients
- Multimodal posteriors

What I used:

- Learned approximate block Gibbs sampler ("tonic sampling")
- Parallel tempering
Conclusion: Rich Generative Models

Being fully Bayesian pays off iff,

▶ you are in the right model class, and
▶ you can solve the inference problem (well-enough).

Future impact on computer vision

▶ Probabilistic programming: rich and physically plausible models
▶ Rapid progress in inference methods for probabilistic programs
▶ Rapid progress in dedicated “inference hardware” (GPUs)
Model Building
Model Building (cont)

Factors that influence the type of computer vision models we will build

▶ Data availability
  + non-parametric
  + discriminative
  + scalable
  + empirical risk minimization: end-to-end training with task-specific loss
    - semi-intractable models (most deep architectures)
    - Bayesian inference

▶ Progress in hardware
  + simulation for training data generation
  + simulation for inference
  + generative models trained from unsupervised data
Thank you!