Parsing Image Facades with Reinforcement Learning

Symmetry Detection from Real Word Images Workshop, CVPR 2011

Olivier Teboul, Iasonas Kokkinos, Panagiotis Katsourakis, Loic Simon and Nikos Paragios
Semantic Segmentation of Urban Scenes

- Image Parsing

- Cropped and Rectified Building Images: ‘Facade Parsing’
Image-based Procedural 3D Models

- Based on 2D parsing + simple extrude and insertion rules turn 2D to 3D...
Problem Statement

- Input: image
- Output: labelling
- Pixel-level classification function:
  \[
  m(c, x, y) = p(c | I(x, y))
  \]
- Objective:
  \[
  C(l) = \sum_{x,y} m(l(x, y), x, y)
  \]
- Wanted:
  \[
  l^* = \arg \max_{l:\text{building}} C(l)
  \]
Shape Grammars: Recursive Derivation of Labelling

- Top level: axiom
- Recursive application of shape operators
  - Partition domain and assign label to each part
- Terminals: semantic labels (e.g. window, door etc)
Binary Split Grammars

– Binary:

- N1
- N2

– Split: one dimension at a time
Challenges

- Joint optimization: topology + geometry
- Enforce the result to be in the language of the grammar: $C \in L(G)$
- High and unknown dimensionality: $\text{card}(L(G))$ up to 1 gogol! ($10^{100}$)
The 1D case

Task: horizontal split(s) of image slice

Binary Split Grammar

- 2 rules

- Recursive segmentation

\[ R_0 = \sum r_k \]
**Markov Decision Process (MDP) formulation**

- Agent iteratively interacting with environment
  - Agent takes action, lands in new state
    \[
    S_t \xrightarrow{a_t} S_{t+1}
    \]
  - Environment yields reward
    \[
    r_t = r(s_t, a_t)
    \]
  - Potentially stochastic state transition and reward functions
- Goal: maximize cumulative reward
  \[
  (N, a_1, \ldots, a_N) = \arg \max \left( \sum_{t=0}^{N} r_t \right)
  \]
- Markov assumption
  \[
  P(s_{t+1}, r_t|s_t, a_t, \ldots, s_0, a_0) = P(s_{t+1}, r_t|s_t, a_t)
  \]
MDP & Policy functions

- Policy function adopted by agent:  \( \pi(s, \alpha) = p(\alpha_t = \alpha|s_t = s) \)
- Merit function
  \[
  Q(s, \alpha) = E_\pi \left[ \sum_{t' > t} r_{t'} | s_t = s, \alpha_t = \alpha \right]
  \]
  - Expected reward-to-go if at \( s \) we perform \( \alpha \), and then follow \( \pi \)
- \( \varepsilon \)-greedy policy:  \( \pi(s, \alpha) = (1 - \varepsilon) \delta(\alpha, \alpha^*) + \varepsilon u(\alpha) \)

Reinforcement learning (Q-learning)

\[
Q(s_t, \alpha_t) \leftarrow Q(s_t, \alpha_t) + \alpha \left[ r_t + \max_{\alpha_{t+1}} Q(s_{t+1}, \alpha_{t+1}) - Q(s_t, \alpha_t) \right]
\]

Policy Evaluation

\[ \pi \]

Policy Improvement

\[ Q \]
Reinforcement Learning Algorithm

\[
\forall s, a \ Q(s,a) = 0 \\
\forall s, \pi(s,.) \leftarrow \text{Uniform}
\]

Loop

\[
s \leftarrow (s, 0)
\]

repeat

\[
a \leftarrow \text{choose an applicable rule according to } \pi(s,a)
\]

\[
\text{Apply rule } a, \text{ observe } s', r
\]

\[
Q(s,a) \leftarrow Q(s,a) + \alpha [r + \max_{a'} Q(s',a') - Q(s,a)]
\]

\[
\pi(s,.) \leftarrow \varepsilon\text{-greedy w.r.t } Q(s,.)
\]

\[
s \leftarrow s'
\]

update \(\alpha\)

update \(\varepsilon\)

until end of the episode

\[
\pi(s,.) \\
\]

\[
Q(s,.)
\]

6/29/2011
Enforcing Symmetry

- Straighforward extension: 2D state, 2D action
  - Large state: Slow convergence
  - Impossible to enforce floor symmetry

- Can we use single policy for all floors?
  - DP: ?
  - RL: Yes, with state aggregation

\[
\begin{align*}
  s &= (x, y, y + h) \rightarrow \tilde{s} = (x) \\
  s' &= (x, y', h') \rightarrow \tilde{s} = (x)
\end{align*}
\]
Data-Driven Exploration

• Bottom-up cues:
  – Line detection, window detection,...

• How can we exploit them in model fitting?
  – Modify ε-greedy exploration strategy
    \[ \pi(s, \alpha) = (1 - \epsilon)\delta(\alpha, \alpha^*) + \epsilon u(\alpha) \]

  – Accumulate gradients:
    \[ h(x) = \sum_y |\nabla_{\pi/2} I(x, y)| \]

  – Use to `propose' actions:
    \[ \pi(s, \alpha) = (1 - \epsilon)\delta(\alpha, \alpha^*) + \epsilon \frac{\exp(h(s + \alpha))}{\sum_{\alpha'} \exp(h(s + \alpha'))} \]
EXPERIMENTAL VALIDATION
Randomized Forest

• Multivariate Classifier based on decision trees

• For each class $c$ and each pixel $x$ in image $I$, it provides $p(c|x,I)$
  $$m(x,c) = p(c|x,I)$$

• Feature vectors = 13x13 RGB patches $\in \mathbb{R}^{507}$

• Well suited to very repetitive architectural styles
Quantitative Validation: Benchmark 2010

- 20 images for training
- 10 images for testing

### Original

<table>
<thead>
<tr>
<th>Potts</th>
<th>RL Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Original Image" /></td>
<td><img src="image2.png" alt="RL Parsing Image" /></td>
</tr>
</tbody>
</table>

### MAP

<table>
<thead>
<tr>
<th>Potts, $\lambda = 1$</th>
<th><img src="image3.png" alt="Potts Image" /></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4.png" alt="RL Parsing Image" /></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Results

<table>
<thead>
<tr>
<th>Method</th>
<th>#generated buildings</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Simon 2011]</td>
<td>$10^6$</td>
<td>~600</td>
</tr>
<tr>
<td>RLParsing</td>
<td>$3.10^3$</td>
<td>~30</td>
</tr>
</tbody>
</table>
Quantitative Validation: Benchmark 2011

- Complete Benchmark:
  - 104 annotated images
  - Manual parsing

<table>
<thead>
<tr>
<th></th>
<th>window</th>
<th>wall</th>
<th>balcony</th>
<th>door</th>
<th>roof</th>
<th>sky</th>
<th>shop</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>19</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>63</td>
<td>11</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>34</td>
<td>13</td>
<td>12</td>
<td>2</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>81</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>11</td>
<td>4</td>
<td>54</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>14</td>
<td>75</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>10</td>
<td>42</td>
<td>7</td>
<td>0</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MAP</th>
<th>RL Parsing</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Mean Std

<table>
<thead>
<tr>
<th></th>
<th>Topology</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.93</td>
<td>0.81</td>
</tr>
<tr>
<td>Std</td>
<td>0.09</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Qualitative Validation
Robustness to Artificial Noise and Occlusions

- Salt-and-pepper noise from 0 to 100% (GMM learnt on noise-free image)

- Artificial occlusions added on images
Robustness to Real Occlusions and Illuminations

- Natural Occlusions
- Cast Shadows
- Night Lights
CONCLUSION

Theoretical contributions

Binary Split Grammars: natural fit for façade modeling
Reinforcement learning: flexible techniques for shape parsing
  Enforcing symmetry via state aggregation
  Data-driven exploration
  Efficient exploration of state-action space
State-of-the-art results on many grammars

Practical contributions

Annotated benchmark for façade parsing
Rflib: Open Source Libraries for Randomized Forests
grapes: software for Facade Parsing with Shape Grammar
Q&A

- Thank you!
Parsing Algorithm Convergence

Artificial Data

ε-greedy

ε-greedy / data driven

Real Data

(a) Binary - Hue

(b) 4-color - GMM

(c) Hausmannian - RF
Contributions

Theory
- Binary Shape Grammar (BSG): generic mutually recursive grammars well-suited for façade modeling and optimization.
- Reformulation of the Parsing problem in the Reinforcement Learning framework
- Generic reinforcement learning algorithm for suitable for any BSG
- State aggregation for fast and consistent parsing
- Data-driven exploration to boost the convergence
- State-of-the-art quantitative and qualitative results on many grammars

Practice
- Annotated benchmark for façade parsing
- Rfllib: Open Source Libraries for Randomized Forests
- grapes: software for Facade Parsing with Shape Grammar
3 classes of solutions

- Dynamic Programming
  - At each state $s$, consider all actions $a$.
  - Obtain merit of state $s$, backpropagate.

  Needs to consider all state-action combinations

- Monte Carlo
  - Fix first action, $\alpha$
  - Probabilistically sample subsequent actions.

  Needs to consider full episode

- Reinforcement Learning
  - At each state pick single action
  - Back-propagate locally
Dynamic Programming vs. Reinforcement Learning

DP estimation of $Q^*(s,a)$

Q-Learning estimation of $Q^*(s,a)$

$x=45$

$Q(s,a)$

action $a$
User-defined Constraints

- The user selects a region \((x, y, w, h)\) and a semantic \(c\)

- **Idea:** Reward more the agent when he creates a labeled rectangle that coincides with the constraint.
- If \(m(x, c) \in [0,1]\), the reward obtained while the constraint is met is:
  \[
  r = 2wh
  \]

- **Constraint is not hard.** No guarantee.
MDP & Policy functions

- Policy function adopted by agent: \( \pi(s, \alpha) = p(\alpha_t = \alpha | s_t = s) \)

- Merit function

\[
Q(s, \alpha) = E_{\pi} \left[ \sum_{t' > t} r_{t'} | s_t = s, \alpha_t = \alpha \right]
\]

- Expected reward-to-go if at \( s \) we perform \( \alpha \), and then follow \( \pi \)

- Bellman’s recursion:

\[
Q^\pi(s_t, \alpha_t) = \sum_{s_{t+1}} P(s_{t+1} | s_t, \alpha_t) \left[ r(s_t, \alpha_t) + \sum_{\alpha_{t+1}} P(s_{t+1}, \alpha_{t+1}) Q^\pi(s_{t+1}, \alpha_{t+1}) \right]
\]

- Bellman’s recursion for optimal policy, \( \pi^*(s, a) \):

\[
Q^*(s_t, \alpha_t) = \sum_{s_{t+1}} P(s_{t+1} | s_t, \alpha_t) \left[ r(s_t, \alpha_t) + \max_{\alpha_{t+1}} Q^*(s_{t+1}, \alpha_{t+1}) \right]
\]

6/29/2011
Q-Learning Algorithm

- Watkins 1989

\[
\forall s, a \; Q(s, a) = 0 \\
\forall s, \pi(s, a) \leftarrow \text{Uniform}
\]

Loop

\[
s \leftarrow \text{first state of the episode}
\]

Repeat

\[
a \leftarrow \text{sample from } \pi(s, a)
\]

Take action \( a \), observe \( s', r \)

\[
Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]
\]

\[
\pi(s, a) \leftarrow \epsilon\text{-greedy w.r.t } Q(s, a)
\]

\[
s \leftarrow s'
\]

Until end of the episode

Learning rate \( \alpha \) decreases with the iterations (stochastic approximation)

Policy Evaluation

Policy Improvement

\( \epsilon \) decreases to 0 (Sutton)

\( \rightarrow \) Exploration/Exploitation trade-off

\( \rightarrow \) Converges towards \( Q^* \)
Semi Markov Decision Processes

- Some decisions may take more time than others
- Introduction of delayed rewards and a waiting-time $\tau$ (random variable)

\[ Q(s, a) \leftarrow Q(s, a) + \alpha \left[ \sum_{k=0}^{\tau-1} \gamma^k r_{k+1} + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \]

- Well-suited to model hierarchies
- Natural extension of Q-learning:
- Existence of specific algorithms (MAXQ)

6/29/2011
Other RL-friendly Techniques

- **Model selection:**
  - Design several compact grammars rather than a single very generic one
  - The choice of the grammar becomes the first decision of the process

- **Function Approximation:**
  - Idea: update several state-action pairs at a time ($Q$ is continuous)
  - Linear Approximation with basis $\phi_i \Rightarrow$ find the $M$ weights $w_i$
    \[
    Q(s, a) = \sum_{i=1}^{M} w_i \phi_i(s, a) = w^T \phi(s, a)
    \]
  - Stochastic Gradient descent to update the estimate of $w$ (therefore of $Q$)
    \[
    w_{t+1} = w_t + \alpha \left[ r_{t+1} + \max_a Q_t(s', a') - Q_t(s, a) \right] \phi
    \]
  - Consistent with tabular Q-learning
  - Choice of a basis of functions: failed with Radial-basis functions

![Graph](image_url)
Gaussian Mixture Models

- For each class $c$, a set of inputs $\{y_i = (r_i, g_i, b_i) \in \mathbb{R}^3\}_{i=1}^{N}$ (brush strokes)
- Observations are explained by a mixture of $K$ Gaussians
  \[ p(y|c) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\mu_k, \Sigma_k) \]
  where \[ \mathcal{N}(y|\mu, \Sigma) = \frac{1}{\sqrt{2\pi}^3 \sqrt{|\text{det}(\Sigma)|}} \exp \left( -\frac{1}{2} (y - \mu)^T \Sigma^{-1} (y - \mu) \right) \]
- Posterior probability comes from Bayes rules:
  \[ m(x, c) = p(c|x) = \frac{p(x|c)p(c)}{\sum_{c'} p(x|c')p(c')} \]
- Optimization using Expectation-Maximization (EM)
Hue

- RF and GMM are based on *supervised learning*
- The hue reward is based on *unsupervised learning* + heuristic
- Heuristic: the façade shows 2 kinds of elements with 2 different colors
- Idea: Cluster the two classes in the Hue space (K-Means or EM)
- Catch: the Hue is an angle → circular geometry → compute everything in $\mathbb{C}$

\[ m(x, c) = p(c|x, I) \]