Crowd Scene Analysis

• Using computer vision tools to look at people in public places

• Real-time monitoring
  – situation awareness
  – notifications/alarms

• After-action review
  – traffic analysis
Crowd Scene Analysis

Things we might want to know:

• How many people are there?
• How to track specific individuals?
• How to determine who is with whom?

Challenges:

Crowd scenes tend to have low resolution.
You rarely see individuals in isolation.
Indeed, there are frequent partial occlusions.
Crowd Counting

FAQ: How many people participated in ...

• Tahrir Square Protests
• Obama’s inauguration
• Occupy Wall Street
• Kumbh Mela
Jacob’s Method

• Herbert Jacobs, Berkeley, 1960s
• count = area * density
  – 10 sqft/person – loose crowd (arm’s length from each other)
  – 4.5 sqft/person – more dense
  – 2.5 sqft/person – very dense (shoulder-to-shoulder)

• Problem: Pedestrians do not uniformly distribute over a space, but clump together into groups or clusters.

• Refinement: break area into a grid of ground patches and estimate a different density in each small patch. Accumulate these counts over whole area.
Example of Jacob’s Method

Computer Vision Could do Better!

Cavaet: nobody really wants accurate counts

e.g. organizers of the “Million Man March” in Washington DC threatened to sue the National Park Service for estimating that only 400K people attended.
Vision-based Counting

- detection and tracking (light density)
- clustering feature trajectories that move coherently (moderate density)
- treat crowd as a dynamic texture and compute regression estimates based on measured properties (heavy density)
Detecting and Counting Individuals


Good for low-resolution / wide-angle views.
Relies on foreground/background segmentation.
Not appropriate for very high crowd density or stationary people.
GateA Path Counts

Maintain a running count of number of people whose trajectories cross a set of user-specified lines (color-coded).
Crowd Flow/Density

30 minute period
Crowd Flow/Density

movie

Time Lapse. Integrated over spatial/temporal windows.
Motion Segmentation

Idea: track many small features (e.g. corners) over time and cluster sets of features that have similar motion.

Motion Segmentation

Basic steps: Form a corner connectivity graph. Assign each edge a dissimilarity score based on distance and motion coherence of trajectories. Prune edges with high scores. The remaining connected components are the independent objects.
Motion Segmentation

Note: Sugimara et.al. add a feature based on gait periodicity to help disambiguate nearby people.
Texture-based Crowd Detection

- SIFT descriptors
- K-means clustering to form “SIFT-Words”
- Likelihood ratio of distributions of word counts over 10 patch sizes yields 10-D feature vector
- Radial basis SVM for classification into crowd / non-crowd
Texture-based Crowd Detection

Sparse classifications turned into dense segmentation using graph cuts. Unary costs based on SVM output and pairwise costs based on magnitude of patch likelihood scores (small magnitudes indicate interclass boundaries).
Texture-based Counting


Extract feature vector for each frame:

- **region features**
  - e.g. area, perimeter, num connected components...

- **internal edge features**
  - e.g. num edges, histogram of orientations

- **grey-level texture features**
  - e.g. homogeneity, energy, entropy
Texture-based Counting


short video clip → segmentation
→ internal edges → internal texture

Extract feature vector for each frame:

estimate counts using learned regression function

$fi(x_t)$
Texture-based Crowd Detection

green/red = crowd walking towards/away     blue = total
numeric results formatted as: estimated count (uncertainty) [true count]
Texture-based Crowd Detection

green/red = crowd walking towards/away
numeric results formatted as: estimated count (uncertainty)
Our Approach*

Bayesian Marked Point Process
- the prior models expected size/shape of people, indexed by location
- the likelihood measures how well a proposed configuration explains the data
- MAP estimate of number/configuration of people is found using RJ-MCMC

Foreground Object Detection

Bayesian Approaches

Pixel-wise MRF models [Sheikh05]

Object-level models (MPP)

Gibbs Point Process [Ortner08]

Conditional Bayesian Models [Rue99; Zhao03]

Non-Bayesian [Dalal05; Rabaud06]
Marked Point Process

• Spatial Point Process
  – Distribution of a set of points in a bounded space

• Marked Point Process (MPP)
  – A spatial point process + a “mark” process

• Examples
  – Spatial process could model tree locations in a forest and the mark could be tree height
  – Spatial process could model cell locations on a microscope slide and the mark could be a polygonal representation of the cell boundary [Rue and Hurn, 1999]
Simplified Problem Statement

Given a foreground image, find a configuration of bounding boxes* that cover a majority of foreground pixels while leaving a majority of background pixels uncovered.

As an MPP:
• Spatial process models number and (x,y) locations of bounding boxes
• Mark process models (height, width, orientation) of each bounding box.

*Footnote: We will add more realistic “shape” models in a moment.
To measure how “good” a proposed configuration is, we generate a foreground image from it and compare with the observed foreground image to get a likelihood score.

\[
\text{config} = \{\{x_1, y_1, w_1, h_1, \theta_1\}, \{x_2, y_2, w_2, h_2, \theta_2\}, \{x_3, y_3, w_3, h_3, \theta_3\}\}
\]
Likelihood Score

Bernoulli distribution model

\[ p_{00} = p(y_i = 0 | x_i = 0) = \text{prob of observing background given a label of background} \]
\[ p_{01} = p(y_i = 0 | x_i = 1) = \text{prob of observing background given a label of foreground} \]
\[ p_{10} = p(y_i = 1 | x_i = 0) = \text{prob of observing foreground given a label of background} \]
\[ p_{11} = p(y_i = 1 | x_i = 1) = \text{prob of observing foreground given a label of foreground} \]

\[ c_{00} = \text{count of pixels where observation is background and label is background} \]
\[ c_{01} = \text{count of pixels where observation is background and label is foreground} \]
\[ c_{10} = \text{count of pixels where observation is foreground and label is background} \]
\[ c_{11} = \text{count of pixels where observation is foreground and label is foreground} \]

Likelihood

\[ L(Y|X) = \prod_{i=1}^{N} p(y_i|x_i) = p_{00}^{c_{00}} p_{01}^{c_{01}} p_{10}^{c_{10}} p_{11}^{c_{11}} \]

simplify, by assuming

\[ p_{00} = p_{11} = \mu \quad \text{and} \quad p_{01} = p_{10} = 1 - \mu \]

Log likelihood

\[ \log L(Y|X) = (c_{00} + c_{11}) \log \mu + (c_{01} + c_{10}) \log (1 - \mu) \]
\[ = [N - (c_{01} + c_{10})] \log \mu + (c_{01} + c_{10}) \log (1 - \mu) \]
\[ = N \log \mu - c_{01} + c_{10} \log [\mu \cdot \log (1 - \mu)] \]
Prior Model

We use a prior to model our expectations about bounding box configurations in the image.

$$\pi(o_i) = \pi(p_i)\pi(w_i, h_i, \theta_i|p_i)$$

- Prior for bounding box $i$
- Prior on location (center point)
- Prior on bounding box height, width and orientation, conditioned on center location.
Estimating Priors

Example: learning height distribution as a function of image row

- Mean
- Standard deviation

Diagram showing half height of person in pixels vs. row number.
Estimating Priors

Example: learning orientation as a function of image location

dataset sample frame extracted blobs / axes

extracted from training sequence inliers for VP estimation
Estimated vanishing point

Scaled, oriented rectangles

Bottom line: it is not difficult to estimate priors on location, size and orientation of people as seen from a specific viewpoint.
Searching for the Max

The space of configurations is very large. We can’t exhaustively search for the max likelihood configuration. We can’t even really uniformly sample the space to a reasonable degree of accuracy.

\[ \text{config}_k = \{ \{x_1,y_1,w_1,h_1,\theta_1\}, \{x_2,y_2,w_2,h_2,\theta_2\}, \ldots, \{x_k,y_k,w_k,h_k,\theta_k\} \} \]

Let \( N \) = number of possible locations for \((x_i,y_i)\) in a \( k \)-person configuration.

Size of \( \text{config}_k \) = \( N^k \)

And we don’t even know how many people there are...

Size of config space = \( N^0 + N^1 + N^2 + N^3 + \ldots \)

If we also wanted to search for width, height and orientation, this space would be even more huge.
Searching for the Max

• Local Search Approach
  – Given a current configuration, propose a small change to it
  – Compare likelihood of proposed config with likelihood of the current config
  – Decide whether to accept the change
Proposals

• Add a rectangle (birth)
Proposals

- Remove a rectangle (death)
Proposals

• Move a rectangle
Searching for the Max

• Naïve Acceptance
  – Accept proposed configuration if it has a larger likelihood score, i.e.
    \[
    \text{Compute } a = \frac{L(\text{proposed})}{L(\text{current})}
    \]
    Accept if \( a > 1 \)
  – Problem: leads to hill-climbing behavior that gets stuck in local maxima

![Diagram showing the process of searching for the max with a hill-climbing behavior illustration. The 'naive acceptance' method is depicted as a process that starts at 'start', goes through several points ('brings us here'), and eventually gets stuck at a local maximum ('but we really want to be over here!').]
Searching for the Max

- **The MCMC approach**
  - Generate random configurations from a distribution proportional to the likelihood!
Searching for the Max

• The MCMC approach
  – Generate random configurations from a
distribution proportional to the likelihood!
  – This searches the space of configurations in an
efficient way.
  – Now just remember the generated configuration with
  the highest likelihood.
Sounds good, but how to do it?

- Think of configurations as nodes in a graph.
- Put a link between nodes if you can get from one config to the other in one step (birth, death, move)

Note links come in pairs: birth/death; move/move
Recall: Markov Chains

Markov Chain:

- A sequence of random variables $X_1, X_2, X_3, \ldots$
- Each variable is a distribution over a set of states (a, b, c...)
- Transition probability of going to next state only depends on the current state. e.g. $P(X_{n+1} = a \mid X_n = b)$

Transition probs can be arranged in an NxN table of elements:

$$k_{ij} = P(X_{n+1} = j \mid X_n = i)$$

where the rows sum to one.
A simple Markov chain

\[ K = \begin{bmatrix}
0.1 & 0.5 & 0.6 \\
0.6 & 0.2 & 0.3 \\
0.3 & 0.3 & 0.1
\end{bmatrix} \]

\( K = \) transpose of transition probability table \( \{k_{ij}\} \) (columns sum to one. We do this for computational convenience.
Question:

Assume you start in some state, and then run the simulation for a large number of time steps. What percentage of time do you spend at $X_1$, $X_2$ and $X_3$?
Experimental Approach

Start in some state, and then run the simulation for some number of time steps. After you have run it “long enough” start keeping track of the states you visit.

\{... X1 X2 X1 X3 X3 X2 X1 X2 X1 X1 X3 X3 X2 ...\}

These are samples from the distribution you want, so you can now compute any expected values with respect to that distribution empirically.
Analytic Approach

four possible initial distributions

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
[.33 \ .33 \ .33]
\]

initial distribution
distribution after one time step

\[
q_0 = \frac{1}{3}
\]
\[
q_1 = K q_0
\]
\[
q_2 = K q_1 = K^2 q_0
\]
\[
q_3 = K q_2 = K^2 q_1 = K^3 q_0
\]

all eventually end up with same distribution -- this is the stationary distribution!
**Eigen-analysis**

\[ K = \begin{bmatrix} 0.1000 & 0.5000 & 0.6000 \\ 0.6000 & 0.2000 & 0.3000 \\ 0.3000 & 0.3000 & 1.0000 \end{bmatrix} \]

\[ E = \begin{bmatrix} 0.6396 & 0.7071 & -0.2673 \\ 0.6396 & -0.7071 & 0.8018 \\ 0.4264 & 0.0000 & -0.5345 \end{bmatrix} \]

\[ D = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & -0.4000 & 0 \\ 0 & 0 & -0.2000 \end{bmatrix} \]

in matlab:

\[ [E,D] = \text{eigs}(K) \]

**Eigenvalue** \( \nu_1 \) always 1

**Stationary distribution**

\[ \pi = \frac{e_1}{\text{sum}(e_1)} \]

i.e. \( K \pi = \pi \)
The PageRank of a webpage as used by Google is defined by a Markov chain. It is the probability to be at page $i$ in the stationary distribution on the following Markov chain on all (known) webpages. If $N$ is the number of known webpages, and a page $i$ has $k_i$ links then it has transition probability \( \frac{(1-q)}{k_i} + \frac{q}{N} \) for all pages that are linked to and \( \frac{q}{N} \) for all pages that are not linked to. The parameter $q$ is taken to be about 0.15.
Google Pagerank

Pagerank == First Eigenvector of the Web Graph!

Computation assumes a 15% "random restart" probability

Sergey Brin and Lawrence Page, The anatomy of a large-scale hypertextual (Web) search engine, Computer Networks and ISDN Systems, 1998

ICCV05 Tutorial: MCMC for Vision. Zhu / Dellaert / Tu

October 2005
How to Design a Chain?

Assume you want to spend a particular percentage of time at X1, X2 and X3. What should the transition probabilities be?

\[ \begin{align*}
P(x1) &= 0.2 \\
P(x2) &= 0.3 \\
P(x3) &= 0.5
\end{align*} \]

\[ K = \begin{bmatrix}
? & ? & ? \\
? & ? & ? \\
? & ? & ?
\end{bmatrix} \]
Detailed Balance

- Consider a pair of configuration nodes r, s
- Want to generate them with frequency relative to their likelihoods L(r) and L(s)
- Let q(r,s) be relative frequency of proposing configuration s when the current state is r (and vice versa)

A sufficient condition to generate r,s with the desired frequency is

\[ L(r) \cdot q(r,s) = L(s) \cdot q(s,r) \]

“detailed balance”
Detailed Balance

- Typically, your proposal frequencies do NOT satisfy detailed balance (unless you are extremely lucky).
- To “fix this”, we introduce a computational fudge factor $a$

**Detailed balance:**

$$a \times L(r) \, q(r,s) = L(s) \, q(s,r)$$

Solve for $a$:

$$a = \frac{L(s) \, q(s,r)}{L(r) \, q(r,s)}$$
MCMC Sampling

- Metropolis Hastings algorithm
  Propose a new configuration

  \[
  \text{Compute } a = \frac{L(\text{proposed}) \cdot q(\text{proposed},\text{current})}{L(\text{current}) \cdot q(\text{current},\text{proposed})}
  \]

  Accept if \( a > 1 \)

  Else accept anyways with probability \( a \)

Difference from Naïve algorithm
Trans-dimensional MCMC

• Green’s reversible-jump approach (RJMCMC) gives a general template for exploring and comparing states of differing dimension (diff numbers of rectangles in our case).
• Proposals come in reversible pairs: birth/death and move/move.
• We should add another term to the acceptance ratio for pairs that jump across dimensions. However, that term is 1 for our simple proposals.
MCMC in Action

Sequence of proposed configurations

Sequence of “best” configurations

movies
MCMC in Action

MAP configuration

Looking good!

num objects: 4  etime: 1.7 sec

num objects: 4  penalty 0.10  itr: 1500
Examples

[Images of various scenarios with objects highlighted]

VLPF 2012
Example: Nov 22, Curtin Road

count 94
Example: Sep 6, Gate A

count  37
Adding Shape to the Estimation

video frame

estimated state

library of candidate shapes
Intrinsic vs Extrinsic Shape

**Intrinsic shapes:**
(silhouettes, aligned and normalized)

**Extrinsic shape**
(how bounding box of silhouette maps into the image)
Revised Prior Model

\[ \pi(o_i) = \pi(p_i) \pi(w_i, h_i, \theta_i | p_i) \pi(s_i) \]

- Prior for object \( i \)
- Prior on extrinsic shape (location + bounding box height, width and orientation)
- Prior on intrinsic shape selection
Learning Intrinsic Shapes

Silhouette shape represented as a Bernoulli mixture model

$$p(x|\mu, \pi) = \sum_{k=1}^{K} \pi_k p(x|\mu_k)$$

Training shapes (foreground masks)

Learned Bernoulli mixture model (“soft” silhouettes)
Learning Intrinsic Shapes

“standard” EM

Bayesian EM with Dirichlet prior

Provides automatic selection of number of mixture components
Adding Shape to the Estimation

MCMC iterations

MCMC proposes changes to current configuration
  • add/remove a person
  • shift location of person
  • change their shape

iteration 1: birth

movie
VSPETS Soccer Dataset

Evaluation run on every 10th frame
- green contours: true positives
- red boxes: false negatives
- pink contours: false positives

movie

detailed view
Caviar Dataset

Learned shapes: results from sample frames

evaluation on six sequences

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total # People</th>
<th>Detection Rate</th>
<th>False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAVIAR</td>
<td>1258</td>
<td>.84</td>
<td>.16</td>
</tr>
<tr>
<td>SOCCER</td>
<td>3728</td>
<td>.92</td>
<td>.06</td>
</tr>
</tbody>
</table>

average detection rate: 0.90
average false positive rate: 0.00

CAVIAR sequence #1
Contributions

• We introduce a Marked Point Process framework for detecting people in crowds
• Conditional mark process models known correlations between bounding box size/orientation and image location
• Extrinsic and intrinsic shape models are learned automatically from training data
• Bayesian EM is used to automatically determine the number of components in the mixture of Bernoulli model for intrinsic shape
Research Directions

Idea: might be good to try other human shapes (activities)
or other animals/objects.