**Markov Random Fields for Multi-Resolution Sensor Fusion**

Mac Mason & Ron Parr

Existing _n_-camera sensor fusion techniques make strong assumptions about sensor geometry and placement.

Relaxing these assumptions opens up a variety of new applications, including teams of sensor-equipped mobile robots and mixed-resolution sensor arrays.

In this work, we present a general framework for fusing data from _n_ heterogeneous cameras in an arbitrary configuration.

---

**Markov Random Fields**
- A Markov Random Field is an undirected graph _G = (V,E)_.
- Each node (or site) _s_ takes on a label _L_s_ in the range [0,1].
- The potential of a node _v_ is determined by a potential function of the labels of _v_ and the neighbors of _v_.
- The potential of _G_ is the product of the site potentials.
- The Maximum a Posteriori (MAP) assignment assigns a label to each node such that the total potential of _G_ is minimized.

**A Simple Example: Smoothing**

**Occupancy Grid**
We represent the scene with a two-dimensional occupancy grid, where each cell stores two values:
- _Y_, the brightness of that cell
- _α_, the opacity of that cell

Each grid cell gets two MRF sites: one for _Y_ and one for _α_. Each pixel in each camera also gets a site.

- A camera observing a grid with no objects.
- This labeling has a better MAP score.

**Potential Function**
Brightness Smoothing: we want the brightness of the grid to change smoothly; so we introduce a quadratic penalty for changes in _Y_.

Camera Potential: the value _P_Y_ of _Y_ is connected to its four _X_-neighbors, and each _α_-node is connected to its four neighbors. These connections allow us to enforce smoothing constraints on our inferred labeling. (red lines)

Object Penalty: we prefer solutions with few objects to solutions with many, so we introduce a direct penalty for opacity.

Small-object Penalty: we prefer small numbers of large objects to large numbers of small objects, so we penalize boundaries between opaque and transparent cells.

\[
P(Y) = \exp \left( C_1 \Pi_1 + C_2 \Pi_2 + C_3 \Pi_3 + C_4 \Pi_4 \right)
\]

(\_\_\_\_\_ denotes a constant weighting parameter)

**Method**
Each of these synthetic examples was solved using gradient descent from a random starting state. Convergence times varied from less than a second to about ten minutes.

**Notation**
In these figures, each grid cell shows the product _αY_, where 0 is black and 1 is white.

Each pair of red lines denotes the field of view of a specific single-pixel camera. The color immediately in front of each focal point denotes the observation X(i) taken by that camera.

Note that a physical camera is many single-pixel cameras, and therefore appears here as an array.

**Examples**
- Two eight-pixel cameras observing a 15x13 grid. Note that the right camera measured a white region in the center of its field, while the left camera observed a white region in the right side of its field. In this case, we have inferred two distinct objects to explain these readings.
- Two single-pixel cameras observing a 5x5 grid. Note that "off the edge of the grid" is implicitly black, so the left camera calculates a value very near zero.

---

**Future Work**
- Real-World Camera Data Coming Soon!
- 3D
  This framework should extend to 3D scenes without any significant architectural changes. However, the computational cost of inference in a 3D scene may be prohibitive.
- Other Sensor Types
  We plan to extend this framework to include range sensors like LIDAR and sonar.

In principle, adding a new type of sensor is straightforward: just describe the potential function that relates the value of that sensor to the nodes it observed.