RAPTOR Progress

Sam Kelly
Jeff Byers

Variance

- A statistical measure of how far a set of numbers is spread out
- A variance of zero indicates that all the values are identical
- Always non-negative
- A small variance indicates that the data tend to be very close to the mean (expected value)
- A high variance indicates that the data are very spread out around the mean and from each other

Covariance

- A statistical measure of the degree to which two random variables change together
- If x is large when y is large, and x is small when y is small, then the covariance is positive
- If x is large when y is small, and x is small when y is large, then the covariance is negative
- The sign of covariance thus illustrates the linear relationship between two random variables
- Covariance of x, y denoted $\sigma(x, y)$

Covariance Matrix

- Element at i, j is the covariance between the i^{th} and j^{th} elements of a random vector (that is, of a vector of random variables)
- Generalizes variance to multiple dimensions
- e.g. a collection of points in 2D or 3D space

$$\begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix} \Rightarrow \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ ... & \sigma(y,y) \end{bmatrix}$$

$$\begin{bmatrix} \sigma(x,x) & \sigma(x,y) & \sigma(x,z) \\ \sigma(y,x) & \sigma(y,y) & \sigma(y,z) \\ \sigma(z,x) & \sigma(z,y) & \sigma(z,z) \end{bmatrix} \Rightarrow \begin{bmatrix} \sigma(x,x) & \sigma(x,y) & \sigma(x,z) \\ ... & \sigma(y,y) & \sigma(y,z) \\ ... & ... & \sigma(z,z) \end{bmatrix}$$

Covariance Matrix

- Can be calculated very quickly for a 2D or 3D point cloud using basic linear algebra operations
- Variation between *xx*, *yy*, *zz*, *xy*, *xz*, *yz* implicitly encodes pose information of a collection of 3D points
- Formally, the eigenvalues of an n-dimensional covariance matrix define an n-ellipse capturing the size, shape, and orientation of the original data
- Sometimes used as a feature vector for classification and pose estimation of 2D (image) and 3D (i.e. LIDAR) data
- Connected with the idea of “image moments”

Covariance Matrix

The following will produce the d x d covariance matrix for a d-dimensional point cloud:

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \\ \dots & \dots & \dots & \dots \end{bmatrix} \qquad \vec{x}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \\ \dots \end{pmatrix}$$

$$\vec{m} = \frac{1}{m} \sum_{i=1}^{m} \vec{x}_i$$

$$\Sigma = \frac{1}{m-1} \sum_{i=1}^{m} (\vec{x}_i - \vec{m})(\vec{x}_i - \vec{m})^{\mathrm{T}}$$

Covariance Matrix Distance Metric

- “A Metric for Covariance Matrices” (Förstner, 1999). This metric is “...invariant under affine transformations and inversion”
- Commonly used today for 3D pose estimation and 3D reconstruction
- Typically used only in situations where LIDAR or depth-based data is available, but is applicable to 2D data as well
- For our purposes, measures the “distance” in n-dimensional pose space between two n-dimensional point clouds
- This is really the length of the geodesic between matrix A and matrix B along the manifold of all possible d x d covariance matrices
- Covariance matrices for two 3D point clouds representing the same 3D model with a slight difference in scale, rotation, or translation would yield a small distance
- Covariance matrices for two point clouds representing the same 3D model with a dramatic difference in scale, rotation, or translation would yield a large distance

$$d(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(\mathbf{A},\mathbf{B})}$$

RAPTOR: Plan A

Try to learn (via ANNs) 3D distance and orientation (pose) directly

- Will use 2 x 2 covariance matrix for thresholded version of the input image as an input feature vector
- Will possibly augment this with a scaled-down version of the input pixel grid
- Could try to learn to output 3 x 3 covariance matrix rather than pose parameters
- *Might* use stereo vision based on two sequential input frames rather than one to yield a richer 3 x 3 covariance matrix feature vector (though this might be dangerously similar to existing techniques)
- We expect that (at least without stereo vision) Plan A will be too difficult for shallow ANNs to learn, but it is very easy to test so we will try it anyway

RAPTOR: Plan B

Calculate the distances between the input pose and several reference poses and use this information as a feature vector or search heuristic

- Reference poses are chosen so that each degree of freedom (rot_x, rot_y, rot_z, dist) has a unique pose corresponding with its minimum and maximum value
- e.g. if rotation parameters vary from 0 to 2pi , and distance can vary from 0 to 100, then the min and max x rotation poses would be (0, pi, pi, 50) and (2pi, pi, pi, 50)
- The average of these distances should be an admissible heuristic for searching within this space so long as the 3D model is sufficiently complex (will prove this)
- Could use this information as a feature vector for an ANN
- The training data forms a 4-dimensional space where each point represents a 3D pose (orientation and distance from camera)
- Might use this heuristic with data clustering, locality-sensitive hashing, or space-partitioning data structures instead of an ANN if this proves feasible
- Would be extremely fast since covariance matrices for reference poses could be precomputed
- Might combine one of these techniques with stereo vision so we are working with 3 x 3 covariance matrices

RAPTOR: Plan C

Leverage RAPTOR's ability to produce on-the-fly renderings of arbitrary poses (and the corresponding 2D and 3D covariance matrices) to zero in on the correct 3D pose through an iterative guess-and-check procedure

Basic process would be as follows:

1. generate guess 3D pose
2. render pose and calculate 2 x 2 (2D) covariance matrix for that pose
3. measure distance between guess 2 x 2 covariance matrix and input image 2 x 2 covariance matrix
4. Come up with better guess, and repeat until correct pose is found

Would probably use something similar to nearest neighbor search