Factorization and Optical Flow

Raquel Urtasun

TTI Chicago

March 14, 2013

Let's talk about Factorization

Factorization from Video Sequences

- When we have video sequences, we can get **feature tracks**
- Often, we can reconstruct structure and motion from those tracks using **factorization**

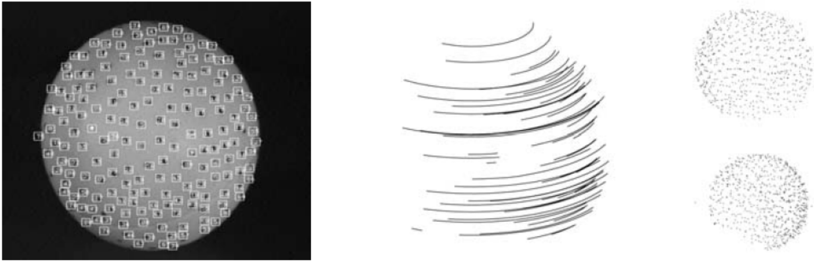


Figure: 3D reconstruction of a rotating ping pong ball using factorization [Tomasi and Kanade, 92]

Orthographic and Weak perspective

- In orthographic and weak perspective, the last row is always $[0, 0, 0, 1]$, there is no division by the last row and thus we can write

$$\mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \bar{\mathbf{p}}_i$$

  with $\mathbf{x}_{ji}$ the location of the projection of the i-th point in the j-th frame, and $\tilde{\mathbf{P}}_j$ a 2×4 projection matrix, and $\bar{\mathbf{p}}_i = (X_i, Y_i, Z_i, 1)$.

- We can compute the centroid of the points

$$\bar{\mathbf{x}}_j = \frac{1}{N} \sum_i \mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \frac{1}{N} \sum_i \bar{\mathbf{p}}_i = \tilde{\mathbf{P}}_j \bar{\mathbf{c}}$$

  with $\bar{\mathbf{c}} = (\bar{X}, \bar{Y}, \bar{Z}, 1)$ the augmented 3D centroid of the point cloud

Orthographic and Weak perspective

- In orthographic and weak perspective, the last row is always $[0, 0, 0, 1]$, there is no division by the last row and thus we can write

$$\mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \bar{\mathbf{p}}_i$$

with $\mathbf{x}_{ji}$ the location of the projection of the i-th point in the j-th frame, and $\tilde{\mathbf{P}}_j$ a 2×4 projection matrix, and $\bar{\mathbf{p}}_i = (X_i, Y_i, Z_i, 1)$.

- We can compute the centroid of the points

$$\bar{\mathbf{x}}_j = \frac{1}{N} \sum_i \mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \frac{1}{N} \sum_i \bar{\mathbf{p}}_i = \tilde{\mathbf{P}}_j \bar{\mathbf{c}}$$

with $\bar{\mathbf{c}} = (\bar{X}, \bar{Y}, \bar{Z}, 1)$ the augmented 3D centroid of the point cloud

- We place the origin of the coordinates at $(\bar{X}, \bar{Y}, \bar{Z}) = (0, 0, 0)$

Orthographic and Weak perspective

- In orthographic and weak perspective, the last row is always $[0, 0, 0, 1]$, there is no division by the last row and thus we can write

$$\mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \bar{\mathbf{p}}_i$$

with $\mathbf{x}_{ji}$ the location of the projection of the i-th point in the j-th frame, and $\tilde{\mathbf{P}}_j$ a 2×4 projection matrix, and $\bar{\mathbf{p}}_i = (X_i, Y_i, Z_i, 1)$.

- We can compute the centroid of the points

$$\bar{\mathbf{x}}_j = \frac{1}{N} \sum_i \mathbf{x}_{ji} = \tilde{\mathbf{P}}_j \frac{1}{N} \sum_i \bar{\mathbf{p}}_i = \tilde{\mathbf{P}}_j \bar{\mathbf{c}}$$

with $\bar{\mathbf{c}} = (\bar{X}, \bar{Y}, \bar{Z}, 1)$ the augmented 3D centroid of the point cloud

- We place the origin of the coordinates at $(\bar{X}, \bar{Y}, \bar{Z}) = (0, 0, 0)$

Factorization

- Let $\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_j$ be the 2D point locations after their image centroid has been substracted. We have

$$\tilde{\mathbf{x}}_{ji} = \mathbf{M}_j \mathbf{p}_i$$

  where $\mathbf{M}_j$ is the upper 2×3 portion of the projection matrix $\mathbf{P}_j$, and $\mathbf{p}_i = (X_i, Y_i, Z_i)$.

- Concatenating all measurements we have

$$\hat{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_{11} & \cdots & \tilde{\mathbf{x}}_{1i} & \cdots & \tilde{\mathbf{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{j1} & \cdots & \tilde{\mathbf{x}}_{ji} & \cdots & \tilde{\mathbf{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{M1} & \cdots & \tilde{\mathbf{x}}_{Mi} & \cdots & \tilde{\mathbf{x}}_{MN} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_j \\ \vdots \\ \mathbf{M}_N \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_i & \cdots & \mathbf{p}_N \end{bmatrix} = \hat{\mathbf{M}}\hat{\mathbf{S}}$$

Factorization

- Let $\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_j$ be the 2D point locations after their image centroid has been substracted. We have
$$\tilde{\mathbf{x}}_{ji} = \mathbf{M}_j \mathbf{p}_i$$
where $\mathbf{M}_j$ is the upper 2×3 portion of the projection matrix $\mathbf{P}_j$, and $\mathbf{p}_i = (X_i, Y_i, Z_i)$.
- Concatenating all measurements we have

$$\hat{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_{11} & \cdots & \tilde{\mathbf{x}}_{1i} & \cdots & \tilde{\mathbf{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{j1} & \cdots & \tilde{\mathbf{x}}_{ji} & \cdots & \tilde{\mathbf{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{M1} & \cdots & \tilde{\mathbf{x}}_{Mi} & \cdots & \tilde{\mathbf{x}}_{MN} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_j \\ \vdots \\ \mathbf{M}_N \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_i & \cdots & \mathbf{p}_N \end{bmatrix} = \hat{\mathbf{M}}\hat{\mathbf{S}}$$

- $\hat{\mathbf{X}}$ is called the measurement matrix, and $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ are the motion and structure respectively.

Factorization

- Let $\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_j$ be the 2D point locations after their image centroid has been substracted. We have
$$\tilde{\mathbf{x}}_{ji} = \mathbf{M}_j \mathbf{p}_i$$
where $\mathbf{M}_j$ is the upper 2×3 portion of the projection matrix $\mathbf{P}_j$, and $\mathbf{p}_i = (X_i, Y_i, Z_i)$.

- Concatenating all measurements we have

$$\hat{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_{11} & \cdots & \tilde{\mathbf{x}}_{1i} & \cdots & \tilde{\mathbf{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{j1} & \cdots & \tilde{\mathbf{x}}_{ji} & \cdots & \tilde{\mathbf{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{M1} & \cdots & \tilde{\mathbf{x}}_{Mi} & \cdots & \tilde{\mathbf{x}}_{MN} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_j \\ \vdots \\ \mathbf{M}_N \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_i & \cdots & \mathbf{p}_N \end{bmatrix} = \hat{\mathbf{M}}\hat{\mathbf{S}}$$

- $\hat{\mathbf{X}}$ is called the measurement matrix, and $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ are the motion and structure respectively.

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?
- This depends on the motion model used (weak-perspective, orthographic)

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?
- This depends on the motion model used (weak-perspective, orthographic)
- See Szelisky 7.3 for an explanation

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?
- This depends on the motion model used (weak-perspective, orthographic)
- See Szelisky 7.3 for an explanation
- Rotation left to right or right to left of the depth reverse version, Necker cube illusion

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?
- This depends on the motion model used (weak-perspective, orthographic)
- See Szelisky 7.3 for an explanation
- Rotation left to right or right to left of the depth reverse version, Necker cube illusion
- What if perspective? Assume orthographic, solve for it and iterate to be perspective.

More on factorization

- Because $\hat{\mathbf{M}}$ is a $2M \times 3$ matrix and $\hat{\mathbf{S}}$ a $3 \times N$ matrix, if we apply SVD to $\hat{\mathbf{X}}$, we will have only 3 non-zero singular values.
- Measurements are typically noisy, so return only the rank-3 factorization
- We still have to obtain $\hat{\mathbf{M}}$ and $\hat{\mathbf{S}}$ as the SVD does not return this directly

$$\hat{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T = [\mathbf{U}\mathbf{Q}][\mathbf{Q}^{-1}\Sigma\mathbf{V}^T]$$

- We can thus set $\hat{\mathbf{M}} = \mathbf{U}\mathbf{Q}$, and $\mathbf{Q}^{-1}\Sigma\mathbf{V}^T$
- How can we recover the 3×3 matrix $\mathbf{Q}$?
- This depends on the motion model used (weak-perspective, orthographic)
- See Szelisky 7.3 for an explanation
- Rotation left to right or right to left of the depth reverse version, Necker cube illusion
- What if perspective? Assume orthographic, solve for it and iterate to be perspective.

What if the motion is non-rigid?

Non-rigid Structure from Motion

[C. Bregler, A. Hertzmann and H. Biermann, CVPR00]

- Observed shapes can be represented as a linear combination of a compact set of basis shapes
- Each instantaneous structure is expressed as a point in the linear space of shapes spanned by the shape basis

$$\mathbf{S} = \sum_{l=1}^{K} l_i \mathbf{S}_i$$

with $\mathbf{S}, \mathbf{S}_i \in \Re^{3 \times P}$, $l_i \in \Re$

- Since the space of spatial deformations is highly object specic, the shape basis need to be estimated anew for each video sequence
- The shape basis of a mouth smiling, for instance, cannot be recycled to compactly represent a person walking

More details

- Under the scale orthographic projection

$$\begin{bmatrix} u_1 & u_2 & \cdots & u_P \\ v_1 & v_2 & \cdots & v_P \end{bmatrix} = \mathbf{R} \left(\sum_{i=1}^{K} l_i \mathbf{S}_i \right) + \mathbf{T}$$

with

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \end{bmatrix}$$

containing the first 2 rows of the full 3d camera rotation matrix, and $\mathbf{T}$ is the camera translation

- As in Tomasi-Kanade, we eliminate $\mathbf{T}$ by substracting the mean of all 2D points, and assuming that $\mathbf{S}$ is centered at the origin
- Thus

$$\begin{bmatrix} u_1 & \cdots & u_P \\ v_1 & \cdots & v_P \end{bmatrix} = \begin{bmatrix} l_1 \mathbf{R} & \cdots l_K \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_K \end{bmatrix}$$

More details

$$\begin{bmatrix} u_1 & \cdots & u_P \\ v_1 & \cdots & v_P \end{bmatrix} = \begin{bmatrix} l_1\mathbf{R} & \cdots l_K\mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_K \end{bmatrix}$$

- Adding a temporal index to each 2D point, and denoting the tracked points in frame t as (u_i^t, v_i^t), we have

$$\mathbf{W} = \begin{bmatrix} u_1^1 & \cdots & u_P^1 \\ v_1^1 & \cdots & v_P^1 \\ \vdots & & \vdots \\ u_1^N & \cdots & u_P^N \\ v_1^N & \cdots & u_P^N \end{bmatrix} = \begin{bmatrix} l_1^1\mathbf{R}^1 & \cdots & l_K^1\mathbf{R}^1 \\ \vdots & & \vdots \\ l_1^N\mathbf{R}^N & \cdots & l_K^N\mathbf{R}^N \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_K \end{bmatrix} = \mathbf{QB}$$

- Performing SVD, and taking the first $3K$ singular vectors / values

$$\mathbf{W}^{2N\times P} = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{Q}^{2N\times 3K}\mathbf{B}^{3K\times P}$$

Factorizing Pose from Configuration

- Performing SVD, and taking the first $3K$ singular vectors / values

$$\mathbf{W}^{2N\times P} = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{Q}^{2N\times 3K}\mathbf{B}^{3K\times P}$$

- In the second step, we extract the camera rotations $\mathbf{R}_t$ and shape basis weights l_t from the matrix $\hat{\mathbf{Q}}$
- $\hat{\mathbf{Q}}$ only contains $N(K+6)$ free variables
- Consider two rows of $\hat{\mathbf{Q}}$ that correspond to one single time frame t, and drop the index on t

$$\hat{\mathbf{q}}^t = \begin{bmatrix} l_1^t\mathbf{R}^t & \cdots & l_K^t\mathbf{R}^t \end{bmatrix} = \begin{bmatrix} l_1 r_1 & l_1 r_2 & l_1 r_3 & \cdots l_K r_1 & l_K r_2 & l_K r_3 \\ l_1 r_4 & l_1 r_5 & l_1 r_6 & \cdots l_K r_4 & l_K r_5 & l_K r_6 \end{bmatrix}$$

- Reordering,

$$\hat{\mathbf{q}} = \begin{bmatrix} l_1 r_1 & l_1 r_2 & l_1 r_3 & l_1 r_4 & l_1 r_5 & l_1 r_6 \\ \vdots & & & & & \vdots \\ l_K r_1 & l_K r_2 & l_K r_3 & l_K r_4 & l_K r_5 & l_K r_6 \end{bmatrix} = \begin{bmatrix} l_1 \\ \vdots \\ l_K \end{bmatrix} \begin{bmatrix} r_1 & r_2 & \cdots & r_6 \end{bmatrix}$$

Factorizing Pose from Configuration

- Reordering,

$$\hat{\mathbf{q}} = \begin{bmatrix} l_1 r_1 & l_1 r_2 & l_1 r_3 & l_1 r_4 & l_1 r_5 & l_1 r_6 \\ \vdots & & & & & \vdots \\ l_K r_1 & l_K r_2 & l_K r_3 & l_K r_4 & l_K r_5 & l_K r_6 \end{bmatrix} = \begin{bmatrix} l_1 \\ \vdots \\ l_K \end{bmatrix} \begin{bmatrix} r_1 & r_2 & \cdots & r_6 \end{bmatrix}$$

- This has rank 1, and can be obtained from SVD by applying successively reordering and factorization to all time blocks of $\hat{\mathbf{Q}}$
- In the final step, we need to enforce the orthonormality of the rotation matrices
- A linear transformation $\mathbf{G}$ is found by solving a least squares problem, where $\mathbf{G}$ maps $\hat{\mathbf{R}}^t$ into a rotation matrix $\mathbf{R}^t = \hat{\mathbf{R}}^t \mathbf{G}$
- The least squares problem imposes orthogonality by

$$\begin{aligned} \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} \mathbf{G}\mathbf{G}^T \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T &= 1 \\ \begin{bmatrix} r_3 & r_4 & r_5 \end{bmatrix} \mathbf{G}\mathbf{G}^T \begin{bmatrix} r_3 & r_4 & r_5 \end{bmatrix}^T &= 1 \\ \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} \mathbf{G}\mathbf{G}^T \begin{bmatrix} r_4 & r_5 & r_6 \end{bmatrix}^T &= 0 \end{aligned}$$

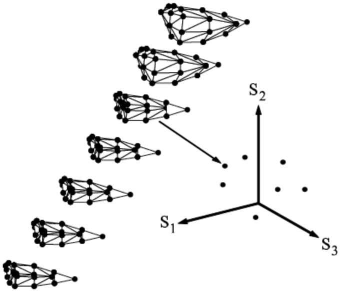
Trajectory basis

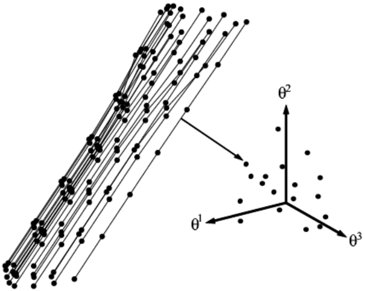
- Let's look again at the shape matrix

$$\mathbf{S}^* = \begin{bmatrix} X_{11} & \cdots & X_{1P} & Y_{11} & \cdots & Y_{1P} & Z_{11} & \cdots & Z_{1P} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ X_{F1} & \cdots & X_{FP} & Y_{F1} & \cdots & Y_{FP} & Z_{F1} & \cdots & Z_{FP} \end{bmatrix}$$

- In the previous case, we assume that this matrix has rank K, with K the number of shape basis. We took the row space
- Now let's take the column space, and we call this **trajectory space**
- If the time varying shape of an object can be expressed by a minimum of k shape basis, then there exist exactly k trajectory basis vectors that can represent the same time varying shape
- We consider the structure as a set of trajectories $T(i) = [T_x(i)^T, T_y(i)^T, T_z(i)^T]$, with $T_x(i)^T = [X_{1,i}, \cdots, X_{F,i}]$, etc.

Shape vs Trajectory basis





Trajectory basis

- We consider the structure as a set of trajectories $T(i) = [T_x(i)^T, T_y(i)^T, T_z(i)^T]$, with $T_x(i)^T = [X_{1,i}, \cdots, X_{F,i}]$, etc.
- We can then say

$$T_x(i) = \sum_{j=1}^{K} a_{xj}(i)\theta^j \qquad T_y(i) = \sum_{j=1}^{K} a_{yj}(i)\theta^j \qquad T_z(i) = \sum_{j=1}^{K} a_{zj}(i)\theta^j$$

  with θ^j the trajectory basis vector, and $a_{xj}(i), a_{yj}(i), a_{zj}(i)$ the coefficients corresponding to that basis vector.

- The time varying structure matrix can then be factorized into an inverse projection matrix and coefficient matrix

$$\mathbf{S}_{3F\times P} = \Theta_{3F\times 3k}\mathbf{A}_{3k\times P}$$

  with $\mathbf{A} = [\mathbf{A}_x^T, \mathbf{A}_y^T, \mathbf{A}_z^T]$

More on Trajectory Basis

- The time varying structure matrix can then be factorized into an inverse projection matrix and coefficient matrix

$$\mathbf{S}_{3F\times P} = \Theta_{3F\times 3k}\mathbf{A}_{3k\times P}$$

  with $\mathbf{A} = [\mathbf{A}_x^T, \mathbf{A}_y^T, \mathbf{A}_z^T]$

- With a particular form

$$\mathbf{A}_x = \begin{bmatrix} a_{x1}(1) & \cdots & a_{x1}(P) \\ \vdots & & \vdots \\ a_{xk}(1) & \cdots & a_{xk}(P) \end{bmatrix} \qquad \Theta = \begin{bmatrix} \theta_1^T & & \\ & \theta_1^T & \\ & & \theta_1^T \\ & \vdots & \\ \theta_F^T & & \\ & \theta_F^T & \\ & & \theta_F^T \end{bmatrix}$$

Benefits

- Benefit of the trajectory space representation is that a basis can be pre-defined that can compactly approximate most real trajectories
- Before, PCA based on the data, so it could not represent all possible shapes
- Which basis to use?
- Discrete Fourier Transform basis, Discrete Wavelet Transform, etc
- They employed DCT

Non-Rigid structure from motion with Trajectory basis

- Before we had

$$\mathbf{W} = \begin{bmatrix} u_1^1 & \cdots & u_P^1 \\ v_1^1 & \cdots & v_P^1 \\ \vdots & & \vdots \\ u_1^N & \cdots & u_P^N \\ v_1^N & \cdots & u_P^N \end{bmatrix} = \begin{bmatrix} l_1^1 \mathbf{R}^1 & \cdots & l_K^1 \mathbf{R}^1 \\ \vdots & & \vdots \\ l_1^N \mathbf{R}^N & \cdots & l_K^N \mathbf{R}^N \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_K \end{bmatrix} = \mathbf{QB}$$

- Performing SVD, and taking the first $3K$ singular vectors / values

$$\mathbf{W}^{2N \times P} = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{Q}^{2N \times 3K}\mathbf{B}^{3K \times P}$$

- Now

$$\mathbf{W} = \begin{bmatrix} u_1^1 & \cdots & u_P^1 \\ v_1^1 & \cdots & v_P^1 \\ \vdots & & \vdots \\ u_1^N & \cdots & u_P^N \\ v_1^N & \cdots & u_P^N \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 & & \\ & \ddots & \\ & & \mathbf{R}_F \end{bmatrix} \mathbf{S} = \mathbf{R}\Theta\mathbf{A} = \Lambda\mathbf{A}$$

with $\Lambda = \mathbf{R}\Theta$ a $3F \times 3K$ matrix.

Factorization

- We can use SVD to factorize into

$$\mathbf{W} = \hat{\Lambda}\hat{\mathbf{A}} \qquad \text{with} \qquad \Lambda = \hat{\Lambda}\mathbf{Q}, \qquad \mathbf{A} = \mathbf{Q}^{-1}\hat{\mathbf{A}}$$

- The problem of recovering the rotation and structure is reduced to estimating the rectification matrix $\mathbf{Q}$ from Λ

$$\Lambda = \begin{bmatrix} r_1^1\theta_1^T & r_2^1\theta_2^T & r_3^1\theta_1^T \\ r_4^1\theta_1^T & r_5^1\theta_2^T & r_6^1\theta_1^T \\ & \vdots & \\ r_1^F\theta_F^T & r_2^F\theta_F^T & r_3^F\theta_F^T \\ r_4^F\theta_F^T & r_5^F\theta_F^T & r_6^F\theta_F^T \end{bmatrix}$$

- One can estimate $\mathbf{Q}$ from $\hat{\Lambda}$ by imposing orthogonality conditions
- Estimate $\mathbf{R}$ from it using non-linear least squares
- Once $\mathbf{R}$ is known, we can estimate $\Lambda = \mathbf{R}\Theta$
- Then the coefficients can be solved via least squares $\Lambda\hat{A} = \mathbf{W}$

Let's talk about Optical Flow

Optical Flow

- We saw how to estimate 2D motion, in the sense of a parametric transformation from one image to another
- The most general (and challenging) version of motion estimation is to compute an independent estimate of motion at each pixel
- This is called **optical flow**
- This typically involves minimizing the brightness or color difference between corresponding pixels summed over the image

$$E_{SSD-OF}(\mathbf{u}) = \sum_i |I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x})|^2$$

- The assumption that corresponding pixel values remain the same in the two images is often called the brightness constancy constraint
- The displacement u can be fractional, so a suitable interpolation function must be applied to image
- We can make E_{SSD-OF} more robust by applying robust estimators

More on Optical Flow

- The energy

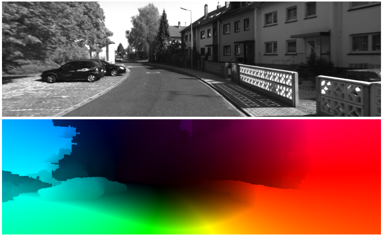
$$E_{SSD-OF}(\mathbf{u}) = \sum_i |I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x})|^2$$

- The number of variables u is twice the number of pixels, thus the problem is under-constraint
- What can we do?
- The two classic approaches to this problem are to perform the summation locally over overlapping regions
- Or to formulate a MRF and do energy minimization
- Think about how you will formulate this

Metrics and Benchmarks

- Same two datasets as for stereo: Middlebury and KITTI
- Have a look at their status
- What's a good metric?
- Mean-end point distance
- Percentage of pixels with distance bigger than some number of pixels
- What is the advantage of disadvantage of each?

Examples and Visualizations



Good luck with the exam!