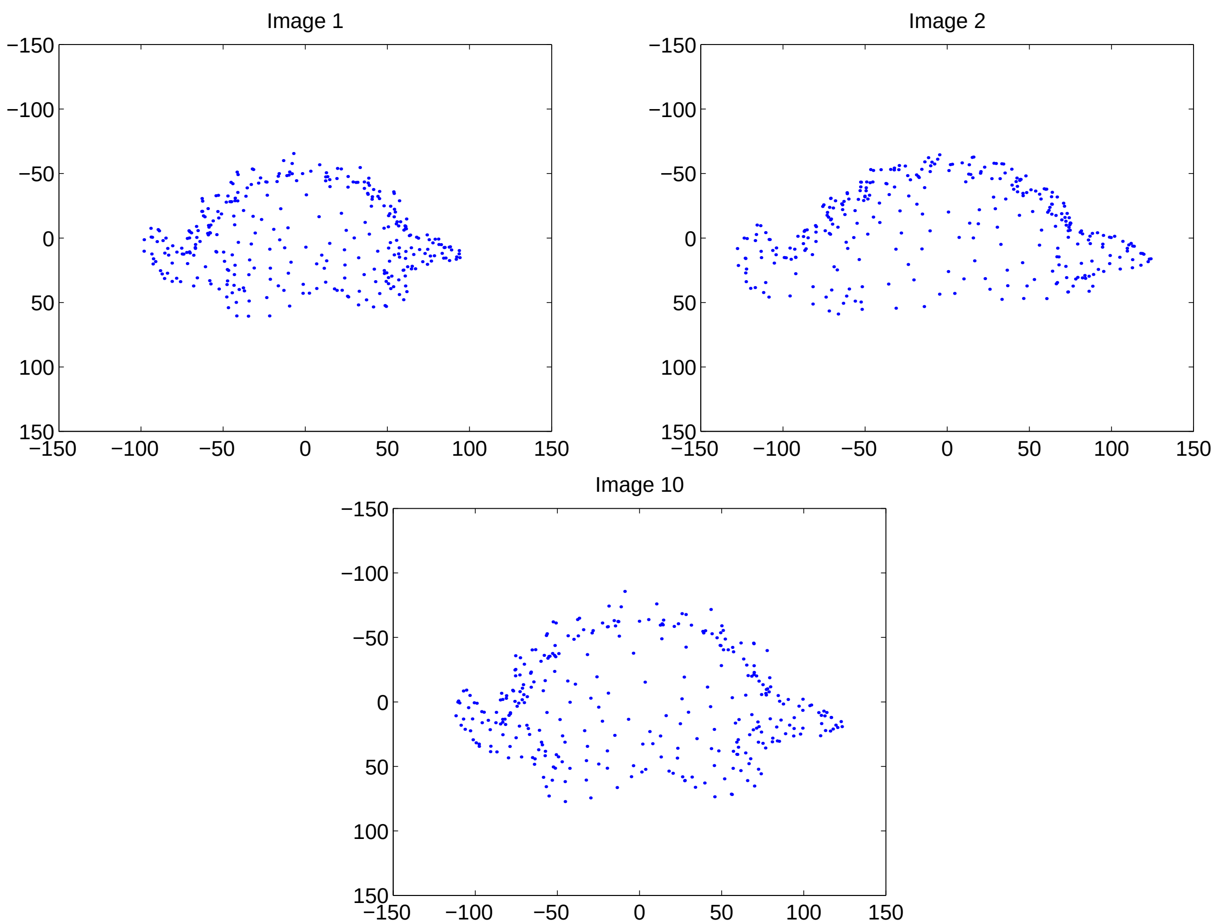
Multi-Frame Factorization Techniques

Suppose $\{\vec{x}_{j,n}\}_{j=1,n=1}^{J,N}$ is a set of *corresponding* image coordinates, where the index $n = 1, \ldots, N$ refers to the n^{th} scene point and $j = 1, \ldots, J$ refers to the j^{th} image.



Such corresponding points may be obtained from local feature points, for example.

Problem. Estimate the 3D point positions, $\{\vec{X}_n\}_{n=1}^{N}$, along with the placement and calibration parameters for the J cameras.

Perspective Projection

The image points and the scene points are related by perspective projection,

$$\vec{p}_{j,n} = \frac{1}{z_{j,n}} M_j \vec{P}_n. \tag{1}$$

Here $\vec{p}_{j,n} = (x_{j,n}, y_{j,n}, 1)^T$ is in homogeneous pixel coordinates, the scene point $\vec{P}_j = (P_{j,1}, P_{j,2}, P_{j,3}, 1)^T$ is in homogeneous 3D coordinates. Also $M_j = M_{in,j} M_{ex,j}$ is the 3×4 camera matrix formed from the product of the intrinsic and extrinsic calibration matrices. Finally, $z_{j,n}$ is the projective depth, $z_{j,n} = \vec{e}_3^T M_j \vec{P}_n$, where $\vec{e}_3^T = (0, 0, 1)$.

For convenience we assume the intrinsic matrices have the form

$$M_{in,j} = \begin{pmatrix} f_j & 0 & 0 \\ 0 & f_j & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

The extrinsic calibration matrices are in general given by

$$M_{ex,j} = \left(R_j, -R_j \vec{d}_j \right), \tag{3}$$

where R_j is the rotation from the world to the j^{th}-camera's coordinates, and $\vec{d}_j$ is the position, in world coordinates, of the nodal point for the j^{th} camera.

Bundle Adjustment

We wish to solve for the point positions $\vec{P}_n$ for $n = 1, \ldots, N$ and the camera matrices M_j for $j = 1, \ldots, J$ by minimizing

$$\mathcal{O}(\{M_j\}_{j=1}^{J}, \{\vec{P}_n\}_{n=1}^{N}) \equiv \sum_{j,n} \left\| \begin{pmatrix} x_{j,n} \\ y_{j,n} \end{pmatrix} - \frac{1}{\vec{e}_3^{\,T} M_j \vec{P}_n} \left(I_2,\ \vec{0}\right) M_j \vec{P}_n \right\|^2 . \tag{4}$$

Here the camera matrices M_j must be of the form $M_{in,j} M_{ex,j}$ where $M_{in,j}$ and $M_{ex,j}$ are as given in equations (2) and (3).

This nonlinear optimization problem is called **bundle adjustment**.

In these notes we discuss two approximations to bundle adjustment:

1. Approximate perspective projection by scaled orthographic projection.
2. Rescale each term in the bundle adjustment objective function (4) and solve a bilinear problem.

Scaled-Orthographic Projection

Scaled-orthographic projection provides an approximation of perspective projection (1) for the case of narrow fields of view,

$$\max\{|x_{j,n}|, |y_{j,n}|\} << f_j,$$

and relatively shallow depth variations,

$$z_{j,n} \approx 1/s.$$

For scaled-orthographic projection, the image points and the scene points are related by

$$\left(I_2,\ \vec{0}\right) \vec{p}_{j,n} = s \left(I_2,\ \vec{0}\right) M_j \vec{P}_n. \tag{5}$$

Here $\vec{p}_{j,n}$, $\vec{P}_n$ and M_j are as above, and s is a constant scale factor. This is **bilinear** in the scaled camera matrix sM_j and the 3D point $\vec{P}_n$.

Differences from Mean Image Points

Let $\vec{\bar{p}}_j = \frac{1}{N} \Sigma_{n=1}^{N} \vec{p}_{j,n}$ be the average image point, and $\vec{\bar{P}}_j = \frac{1}{N} \Sigma_{n=1}^{N} \vec{P}_{j,n}$ be the average scene point. Then, by equation (5), we can show

$$\vec{u}_{j,n} = \tilde{M}_j \vec{U}_n, \tag{6}$$

where

$$\begin{aligned}
\vec{u}_{j,n} &= \left(I_2,\ \vec{0}\right)(\vec{p}_{j,n} - \vec{\bar{p}}_j), \\
\vec{U}_{j,n} &= (I_3,\ \vec{0})(\vec{P}_{j,n} - \vec{\bar{P}}_j), \\
\tilde{M}_j &= s\left(I_2,\ \vec{0}\right) M_j \left(I_3,\ \vec{0}\right)^T .
\end{aligned}$$

Moreover, from equations (2) and (3) it follows that the scaled-orthographic projection matrix $\tilde{M}_j$ has the form

$$\tilde{M}_j = s \begin{pmatrix} f_j & 0 & 0 \\ 0 & f_j & 0 \end{pmatrix} R_j = s f_j \left(I_2,\ \vec{0}\right) R_j, \tag{7}$$

where R_j is the rotation matrix for the j^{th} camera, as above.

Derivation: Difference from Mean

Let $\bar{\vec{p}}_j = \frac{1}{N}\sum_{n=1}^{N}\vec{p}_{j,n}$ be the average image point, and $\bar{\vec{P}}_j = \frac{1}{N}\sum_{n=1}^{N}\vec{P}_{j,n}$ be the average scene point.

Then, by equation (5), we have

$$\left(I_2,\ \vec{0}\right)\bar{\vec{p}}_j = s\left(I_2,\ \vec{0}\right)M_j\bar{\vec{P}}$$

Subtracting this from (5) we find

$$\left(I_2,\ \vec{0}\right)(\vec{p}_{j,n} - \bar{\vec{p}}_j) = s\left(I_2,\ \vec{0}\right)M_j(\vec{P}_{j,n} - \bar{\vec{P}}).$$

Note the 4^{th} component of $\vec{P}_{j,n} - \bar{\vec{P}}$ is equal to $1 - 1 = 0$. Therefore we can drop this 4^{th} component, and obtain

$$\vec{u}_{j,n} = \tilde{M}_j\vec{U}_n,$$

where

$$\begin{aligned}
\vec{u}_{j,n} &= \left(I_2,\ \vec{0}\right)(\vec{p}_{j,n} - \bar{\vec{p}}_j),\\
\vec{U}_{j,n} &= (I_3,\ \vec{0})(\vec{P}_{j,n} - \bar{\vec{P}}_j),\\
\tilde{M}_j &= s\left(I_2,\ \vec{0}\right)M_j\left(I_3,\ \vec{0}\right)^T.
\end{aligned}$$

Which is what we set out to show.

Notice we can use the definitions of $M_{in,j}$ and $M_{ex,j}$ to simplify $\tilde{M}_j$ above. We find

$$\begin{aligned}
\tilde{M}_j &= s\left(I_2,\ \vec{0}\right)M_j\left(I_3,\ \vec{0}\right)^T,\\
&= s\left(I_2,\ \vec{0}\right)M_{in,j}M_{ex,j}\left(I_3,\ \vec{0}\right)^T,\\
&= s\begin{pmatrix} f_j & 0 & 0\\ 0 & f_j & 0\end{pmatrix}R_j
\end{aligned}$$

This gives equation (7) above.

Scaled-Orthographic Factorization

Let $D = (\vec{u}_{j,n})$ be the $2J \times N$ data matrix formed by stacking $\vec{u}_{j,n}$, for $j = 1, \ldots, J$ in columns, and combining these columns for $n = 1, \ldots, N$ (here j is the camera index and n the feature point index). From above, $\vec{u}_{j,n} = \vec{x}_{j,n} - \vec{\bar{x}}_j$, where $\vec{x}_{j,n}$ is the observed pixel position of the j^{th} point in the n^{th} frame, and $\vec{\bar{x}}_j$ is the average of these over all n. In particular, the data matrix can be built from the observed corresponding points.

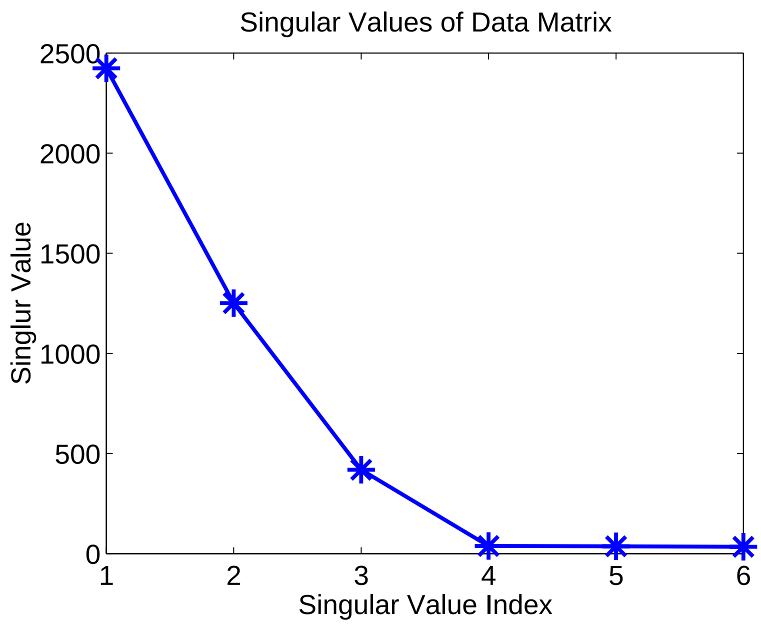
From equation (6) we then have

$$D = MU, \tag{8}$$

where M is the $2J \times 3$ matrix formed by stacking the $\tilde{M}_j$ matrices, and U is the $3 \times N$ matrix having columns given by $\vec{U}_n$. This equation states that **the data matrix has at most rank 3** (without considering noise).

Factorization via SVD

Performing an SVD on the data matrix D, for a case with $J = 3$ images, provides $D = W\Sigma V^T$ with the singular values shown below:



See the 3dRecon Matlab tutorial **`orthoMassageDino.m`** ($\sigma_n = 1$ pixel).

Affine Shape

What does the factorization $D = W\Sigma V^T$ tell us about the shape of the objects being imaged? For notational convenience, we assume that all but the first 3 singular values of Σ have been set to zero or, equivalently, Σ is 3×3, W is $2J \times 3$ and V^T is $3 \times N$.

We now have two rank 3 factorizations of D, namely MU and $W\Sigma V^T$. But this factorization is only unique up to a nonsingular matrix A, as follows (assuming $\text{rank}(D) = 3$):

$$D = MU = (WA)(A^{-1}\Sigma V^T) = W\Sigma V^T. \tag{9}$$

That is, for some 3×3 matrix A, the 3D point positions and the camera matrices must be given by

$$\begin{aligned} U &= A^{-1}\Sigma V^T \\ M &= WA. \end{aligned} \tag{10}$$

Equivalently, we could place AA^{-1} between the Σ and the V^T in equation (9).

Therefore we know the shape U up to the 9 parameters in A. This is known as an **affine reconstruction** of U.

Affine Shape (Cont.)

What can A (or, equivalenty, A^{-1}) do to a shape...?

For example, consider a configuration of 3D points as specified by the matrix U in the factorization above. Suppose we hava a nonsingular matrix A. What does the configuration $A^{-1}U$ look like?

Use SVD to decompose A into $U_a \Sigma_a V_a^T$. So $A \Sigma V^T$ is obtained by rotating/reflecting ΣV^T using V_a^T, then stretching/shrinking the result along the axes according to Σ_a, and finally rotating/reflecting this result using U_a. (Imagine applying such transforms to your lecturer's head.)

The equivalence class of all configurations that can be obtained with transformations of this form is called **affine shape**.

It can be shown that affine shape preserves parallel lines and intersecting lines, but not angles and lengths.

See Tomasi and Kanade, IJCV, Vol. 9, 1992, pp.137-154, for the original factorization method.

Euclidean Reconstructions

We can determine many of the parameters in A from knowledge about the cameras.

In particular, if we only know the pixels are square, then the projection matrix $\tilde{M}_j$ (for scaled-orthographic projection) is as described in equation (7), that is,

$$\tilde{M}_j \ = \ sf_j \left(I_2, \ \vec{0} \right) R_j.$$

From (10) we have $\tilde{M}_j = W_j A$ where W_j is the j^{th} 2×3 block in W corresponding to the same two rows as $\tilde{M}_j$ occupies in M. Since $R_j R_j^T = I_3$ it then follows that

$$\tilde{M}_j \tilde{M}_j^T = s^2 f_j^2 I_2 = W_j A A^T W_j^T. \tag{11}$$

Here the scale factor for the j^{th} image sf_j and the 3×3 symmetric matrix $Q = AA^T$ are the only unknowns.

For each j, equation (11) provides 2 linear homogeneous equations for the coefficients of Q. Then for $J \geq 3$ we have $2J \geq 6$ homogeneous linear equations which we can solve for Q, up to a scalar multiple r_q^2. Finally, given Q we can factor it (assuming the eigenvalues are all non-negative) by performing an SVD, $Q = U_q S_q V_q^T$, and then recognizing the factor A must be $A = \frac{1}{r_q} U_q S_q^{1/2} R_q^T$. Here r_q is the unknown scale factor in Q, and R_q is an arbitrary orthogonal 3×3 matrix.

Euclidean Reconstruction (Cont.)

Given this expression for A we can therefore recover $A^{-1} = r_q R_q K_q$ where $K_q = S_q^{-1/2} U_q^T$ is known. As a consequence we have recovered the shape matrix U_r and the camera matrix M_r where

$$\begin{aligned} U &= r_q R_q U_r, \quad \text{for } U_r = K_q \Sigma V^T, \\ M &= M_r R_q^T \frac{1}{r_q}, \quad \text{for } M_r = W K_q^{-1}. \end{aligned} \tag{12}$$

This is called a **Euclidean reconstruction**, since we have recoverd the shape up to a 3D scale r_q and rotation R_q (ignoring the reflection ambiguity). Equivalently, this is referred to as **metric shape** recovery.

The ambiguity of the overall rotation R_q reflects (no pun intended) the fact that we cannot recover the orientation of the original world coordinate frame. This unknown rotation R_q affects both the shape, via $U = R_q U_r$, and all of the camera matrices, via $M = M_r R_q^T$. That is, R_q rotates the both the scene and the cameras together.

Similarly, the ambiguity of the overall scale r_q reflects the fact that we do not know the scale of the world coordinate frame. We could be imaging a tiny scene with large scale factors sf_j, and we could not tell from the images alone. (Think about making the movie Titanic.) Here r_q rescales the shape via $U = r_q U_r$, and also rescales all the scale parameters f_j in the cameras, via $M = M_r \frac{1}{r_q}$.

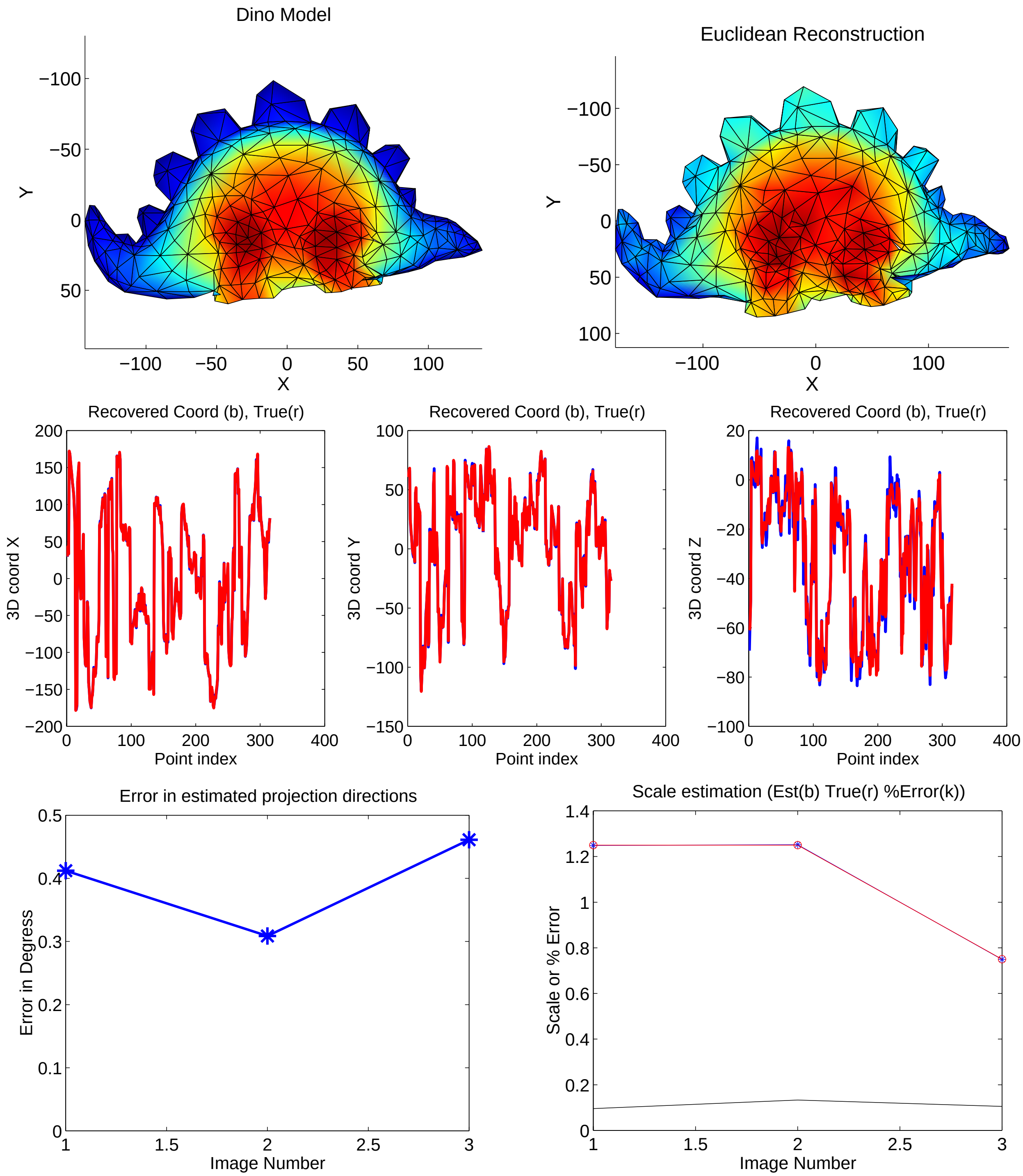
Remaining Ambiguities

The remaining ambiguity in R_q is the **Necker ambiguity**, that is, R_q could be a reflection (say $R_q = diag(1, 1, -1)$). Effectively, with orthographic projection we cannot tell the difference between a concave-in shape viewed from the left, and the reflected concave-out shape viewed from the right. Unlike the previous two ambiguities, this ambiguity does not persist (mathematically) when we switch to perspective projection.

For $J = 2$ orthographic views there is an additional ambiguity, known as the **bas-relief** ambiguity. For this ambiguity, there is an additional unknown parameter (in K_q above), which ties the overall depth variation of the shape to the amount of rotation between the two cameras. See `orthoMassageDino.m`.

Refs: See the classic paper by Koenderink and van Doorn, Affine structure from motion, Journal of the Optical Society of America, 8(2), 1991, pp. 377-385.

Dino Example, Orthographic Case



Introduction to Projective Reconstruction

Returning to perspective projection, it is tempting to modify the bundle adjustment objective function (4) by multiplying each term in the sum by the projective depths $z_{j,n} = \vec{e}_3^{\,T} M_j \vec{P}_n$, providing a reweighted version of (4)

$$\mathcal{O} \;=\; \sum_{j,n} \left\| z_{j,n} \begin{pmatrix} x_{j,n} \\ y_{j,n} \\ 1 \end{pmatrix} - M_j \vec{P}_n \right\|^2 \tag{13}$$

where $z_{j,n}$, M_j and $\vec{P}_n$ are all unknowns for $j = 1, \ldots, J$ and $n = 1, \ldots, N$.

The form of (13) suggests the following factorization approach.

Projective Factorization

Suppose we know the projective depths $z_{j,n}$, and form the data matrix $D = (z_{j,n}\vec{p}_{j,n})$. This is a $3J \times N$ matrix formed by stacking the 3-vectors $z_{j,n}\vec{p}_{j,n}$ in columns for the same point n, and then arranging these columns side by side for $n = 1, \ldots, N$. By equation (1) we have

$$z_{j,n}\vec{p}_{j,n} \;=\; M_j P_n.$$

Therefore D (for the correct $z_{j,n}$'s) must have the rank 4 factorization

$$D = MP, \tag{14}$$

where M is the $3J \times 4$ matrix formed by stacking up the camera matrices M_j, and $P = (\vec{P}_1, \ldots, \vec{P}_N)$ is the $4 \times N$ shape matrix.

Iterative Projective Factorization

Suppose we normalize $D_n = DL$ so that the columns have unit length (using a diagonal matrix L). Then we factor D_n using SVD to form

$$D_n = W \Sigma V^T, \tag{15}$$

where we set all but the first 4 singular values to zero. Equivalently, we have W is $3J \times 4$, Σ is 4×4, and V^T is $4 \times N$.

We can rewrite the n^{th} column of D_n as $C_n \vec{z}_n$, where C_n is a $3J \times N$ matrix obtained from the image points $\vec{p}_{j,n}$ and the n^{th} weight $L_{n,n}$. Here $\vec{z}_n = (z_{1,n}, \ldots, z_{J,n})^T$, which are the projective depths for the n^{th} point in each of the J frames. We then update $\vec{z}_n$ to better match the current factorization. That is, we wish to minimize

$$||C_n \vec{z}_n - W P_{*,n}|| \tag{16}$$

for $\vec{z}_n$ subject to the constraint that the updated column of D_n still has unit length, i.e., $||C_n \vec{z}_n|| = 1$. (In `projectiveMassageDino.m` this update of $\vec{z}_n$ is done with one step along the gradient direction for this constrained optimization problem.) Once all the projective depths have been updated, we reform the normalized data matrix D_n, and redo the factorization (15). This process is iterated until convergence.

Projective Reconstruction

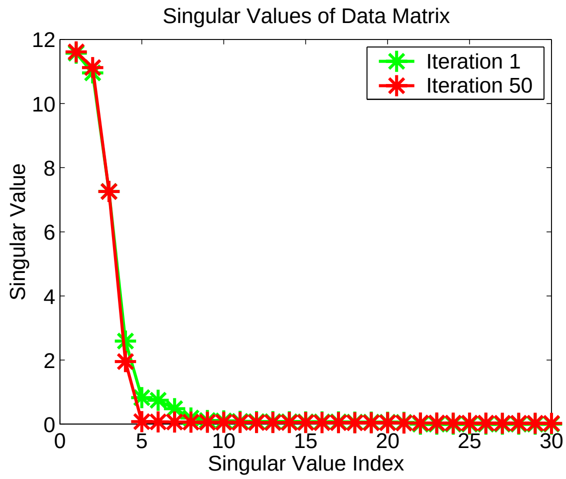
Upon convergence we have a projective factorization $D_n = W \Sigma V^T$. As in the orthographic case, this factorization is only unique up to a nonsingular matrix H. In this case, H is a 4×4, 3D homography matrix. In particular, we have the factorization, $D = D_n L^{-1} = MP$ with

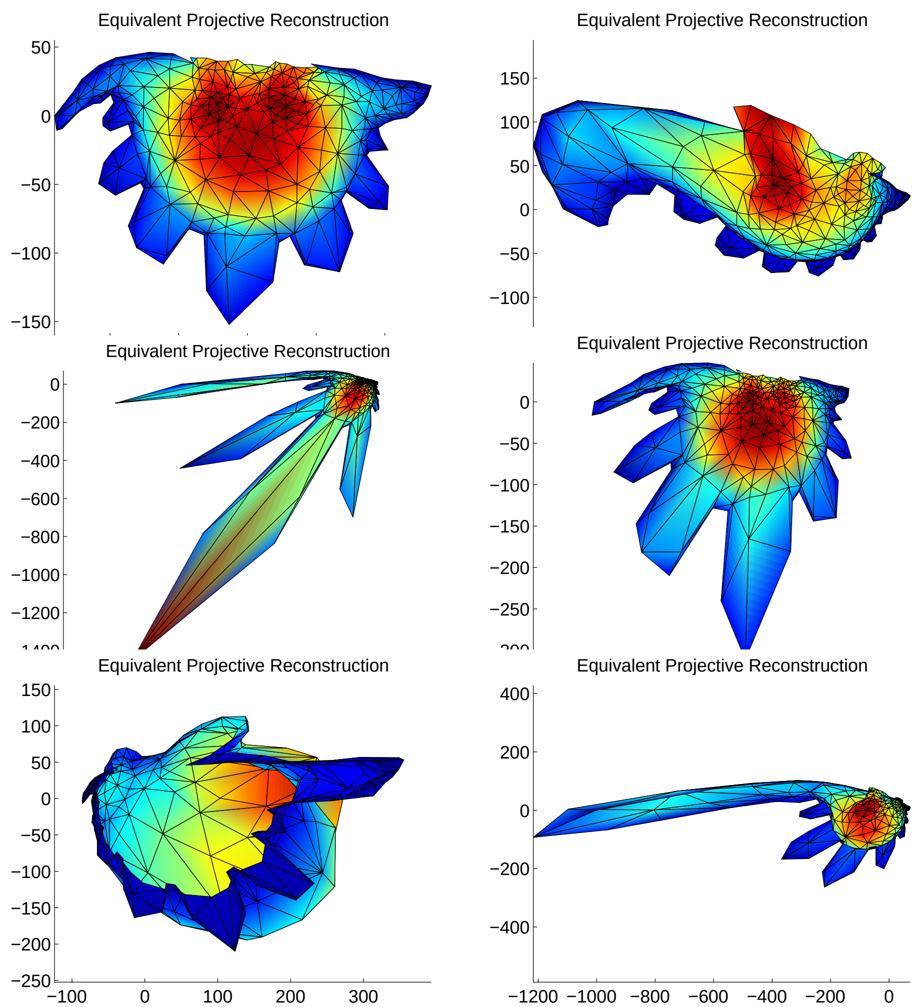
$$\begin{aligned} P &= H^{-1} \Sigma V^T L^{-1} \\ M &= WH. \end{aligned} \tag{17}$$

Since the shape matrix P is known up to a 3D homography H, this is called a **projective reconstruction**.

This projective reconstruction can be "upgraded" to an affine reconstruction or a metric reconstruction by using information about the camera matrices M_j to constrain the 3D homography matrix H. We omit these details (see the papers linked under further readings for this lecture on the course homepage).

Dino Example, Projective Case





Dino Example, Projective Case