Lecture:

Non-Rigid Structure from Motion

3D Vision
Universitat Pompeu Fabra

Discussion

Non-Rigid Shapes?

- Can we obtain non-rigid 3D information from images?

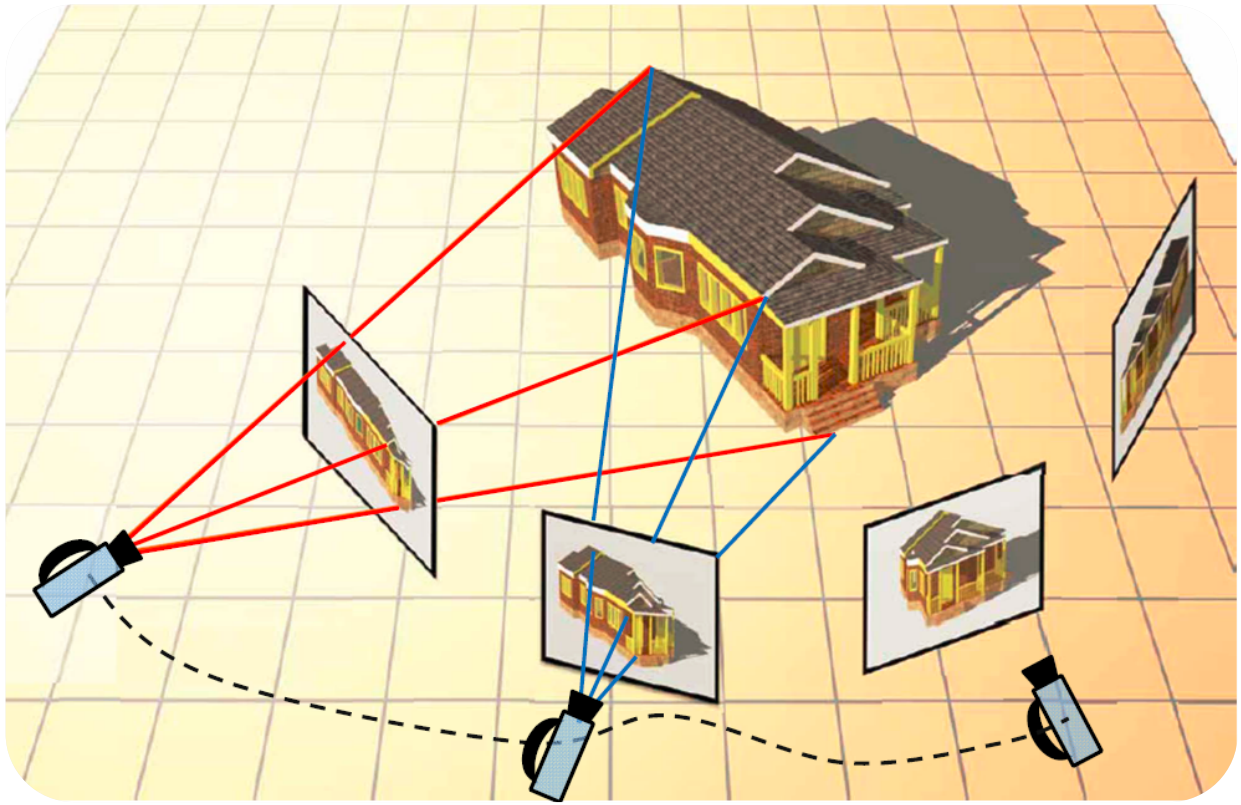
Structure from Motion

(Rigid) Structure from Motion

Given: a monocular video (or a collection of pictures)
We want: simultaneously recovering the 3D shape and the camera motion

Epipolar geometry can be used

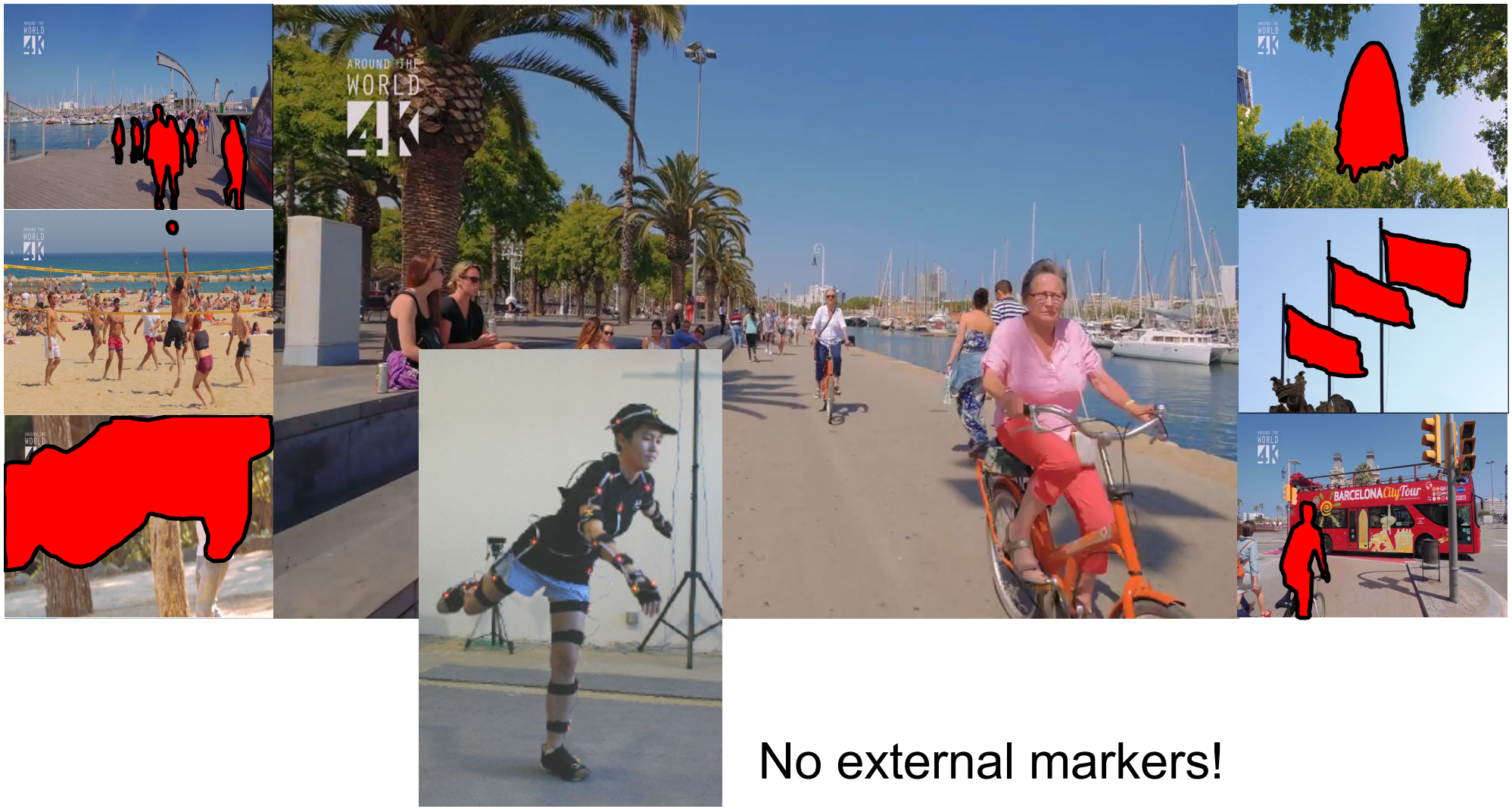
The assumption of rigidity is enough to make the problem well-posed



What about non-rigid motion?

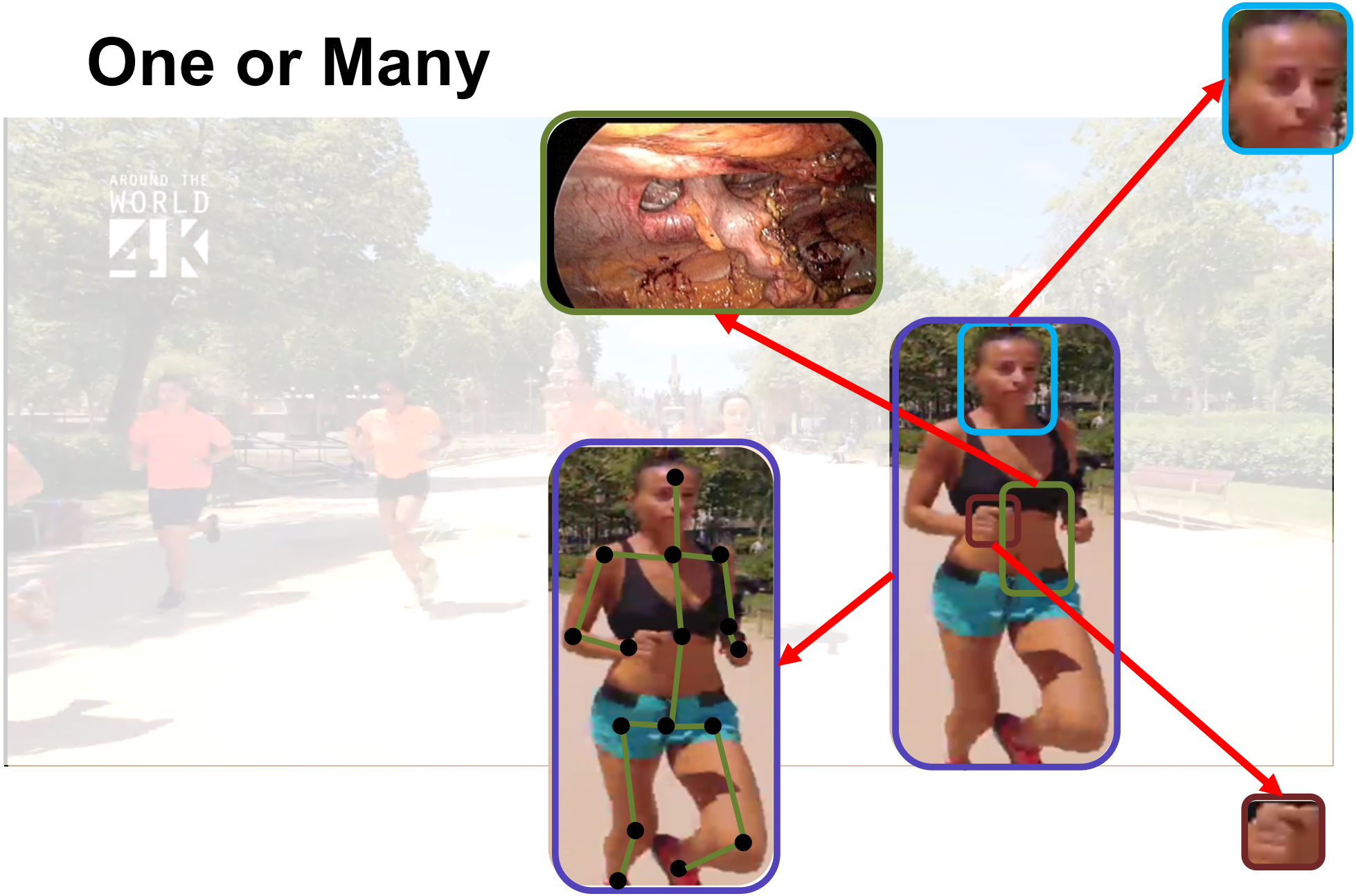
Input 2D tracks form an image sequence:

Our world is Non Rigid!



No external markers!

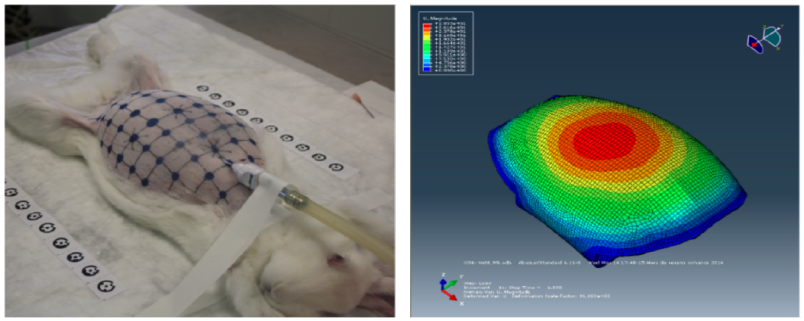
One or Many



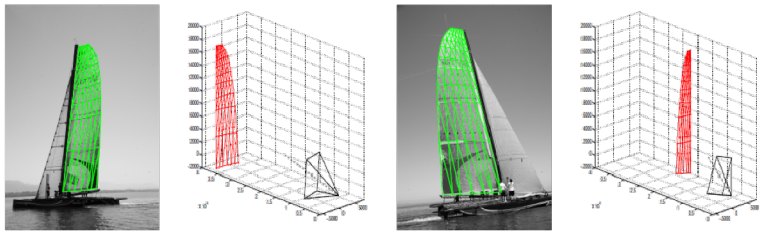
Why is this important?

The world is non-rigid! Too many everyday applications in many different domains

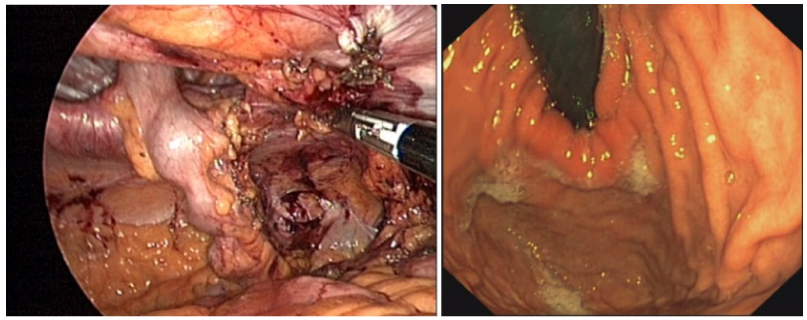
Movie industry, augmented reality



Experimental industry



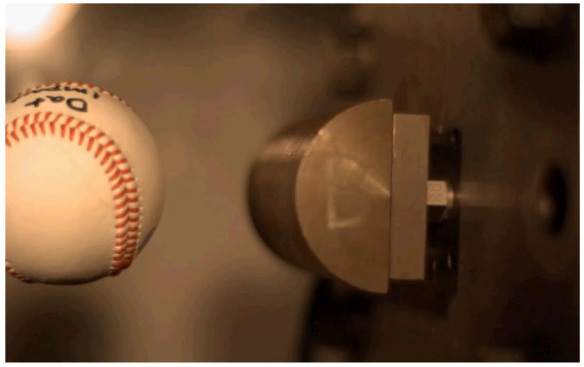
Sport industry: sailing

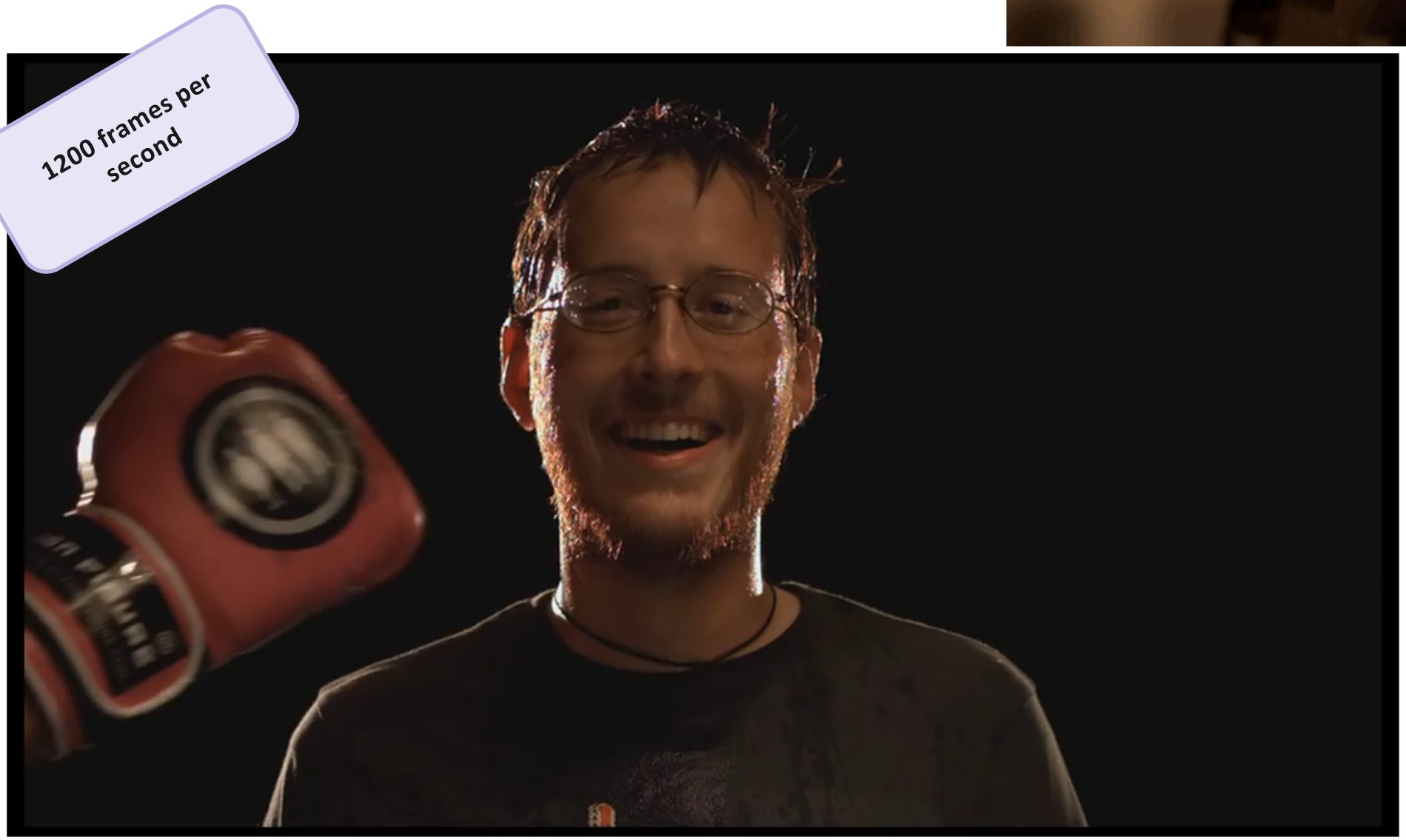


Endoscopy

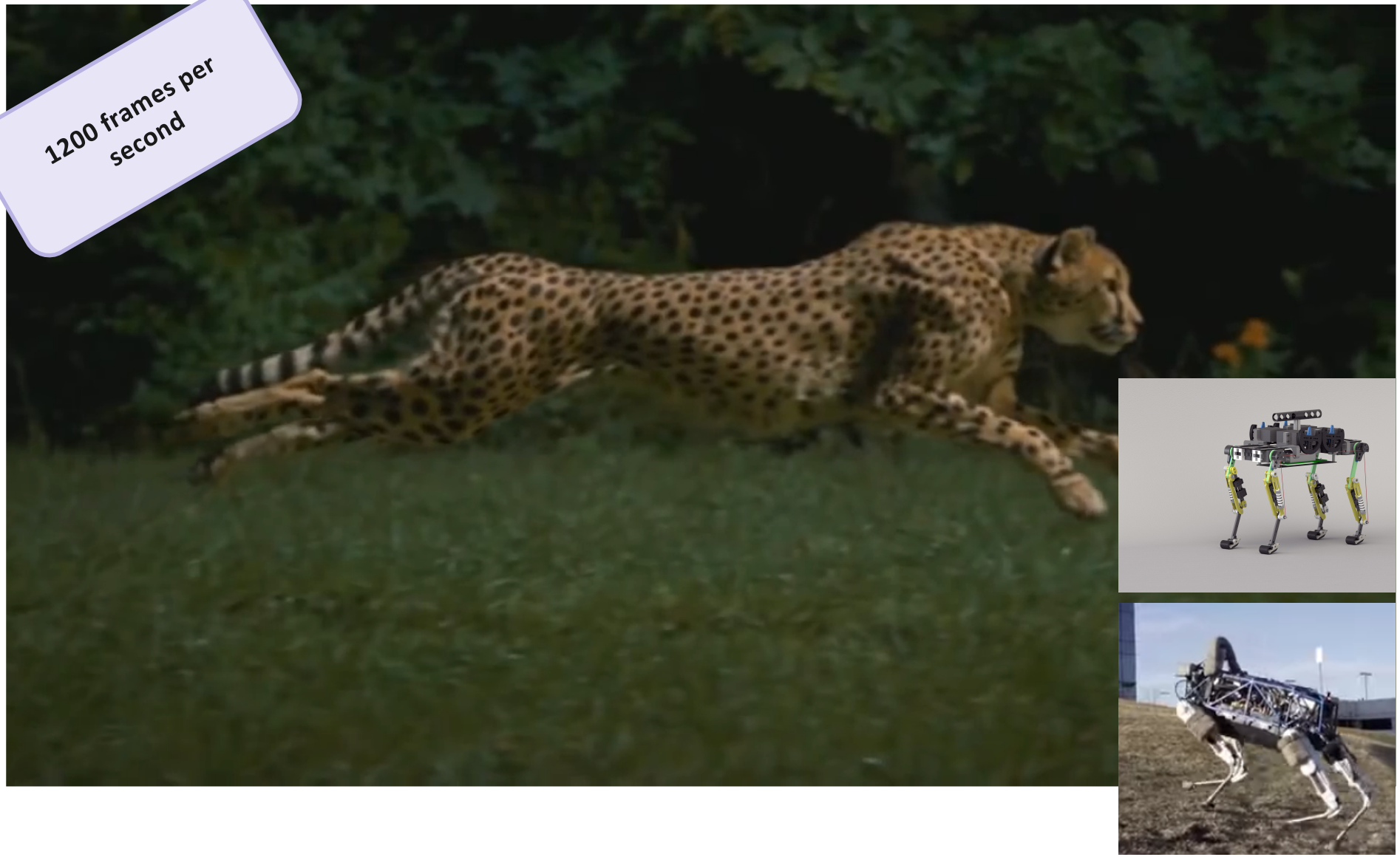
The movie industry

Even more details





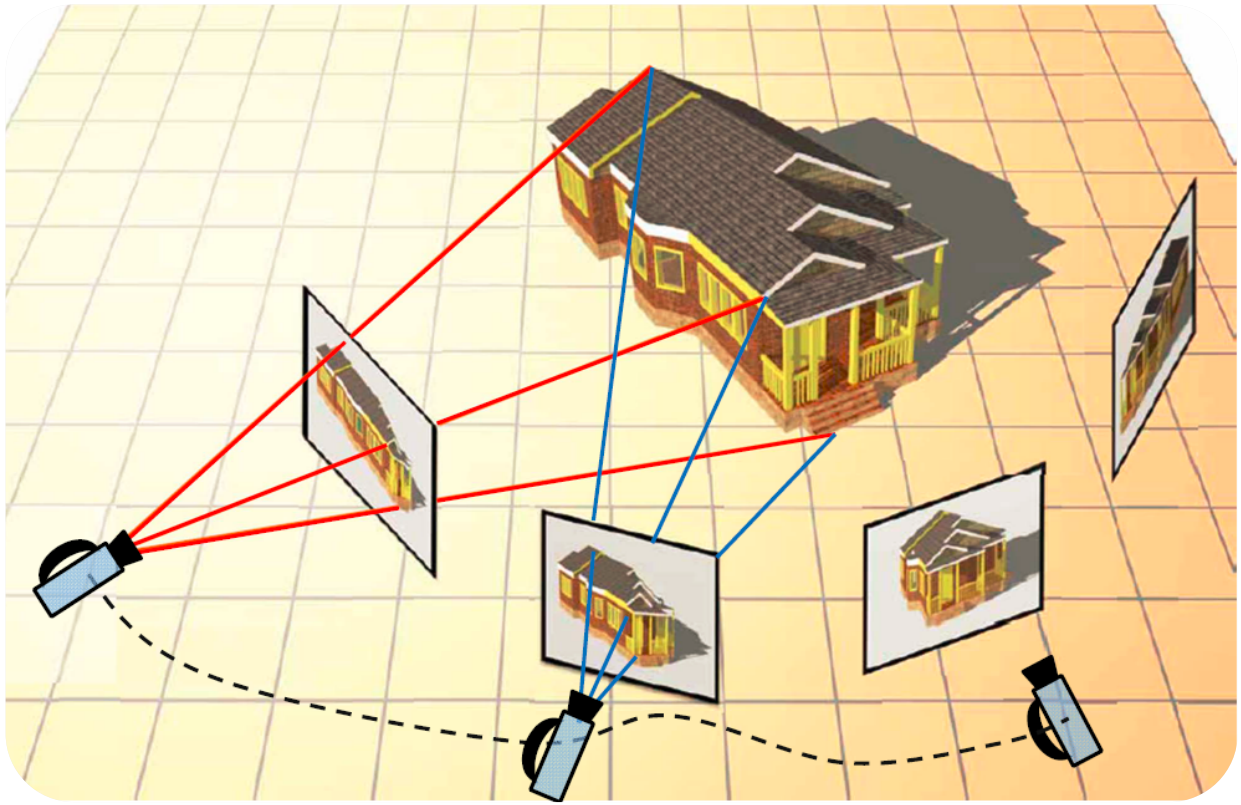
Animal Reconstruction



…produces better robots?

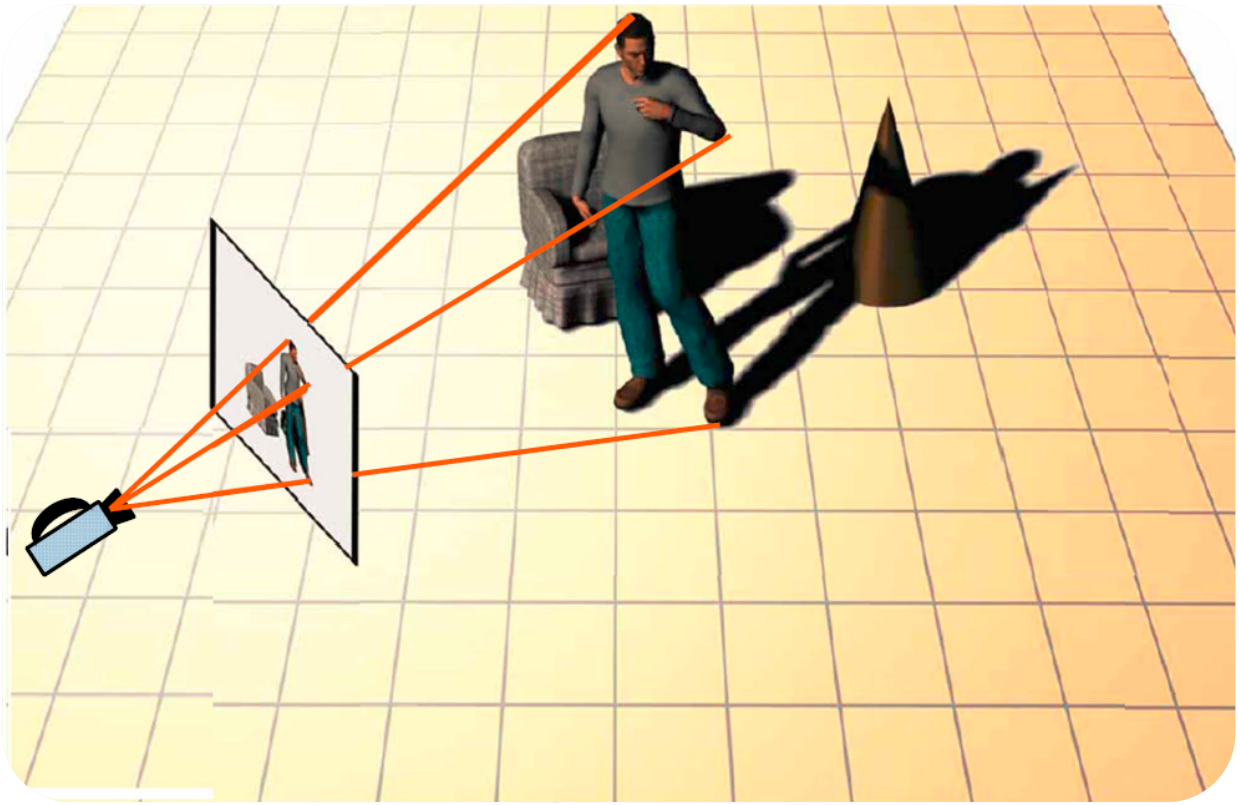
Epipolar geometry can be used

The assumption of rigidity is enough to make the problem well-posed



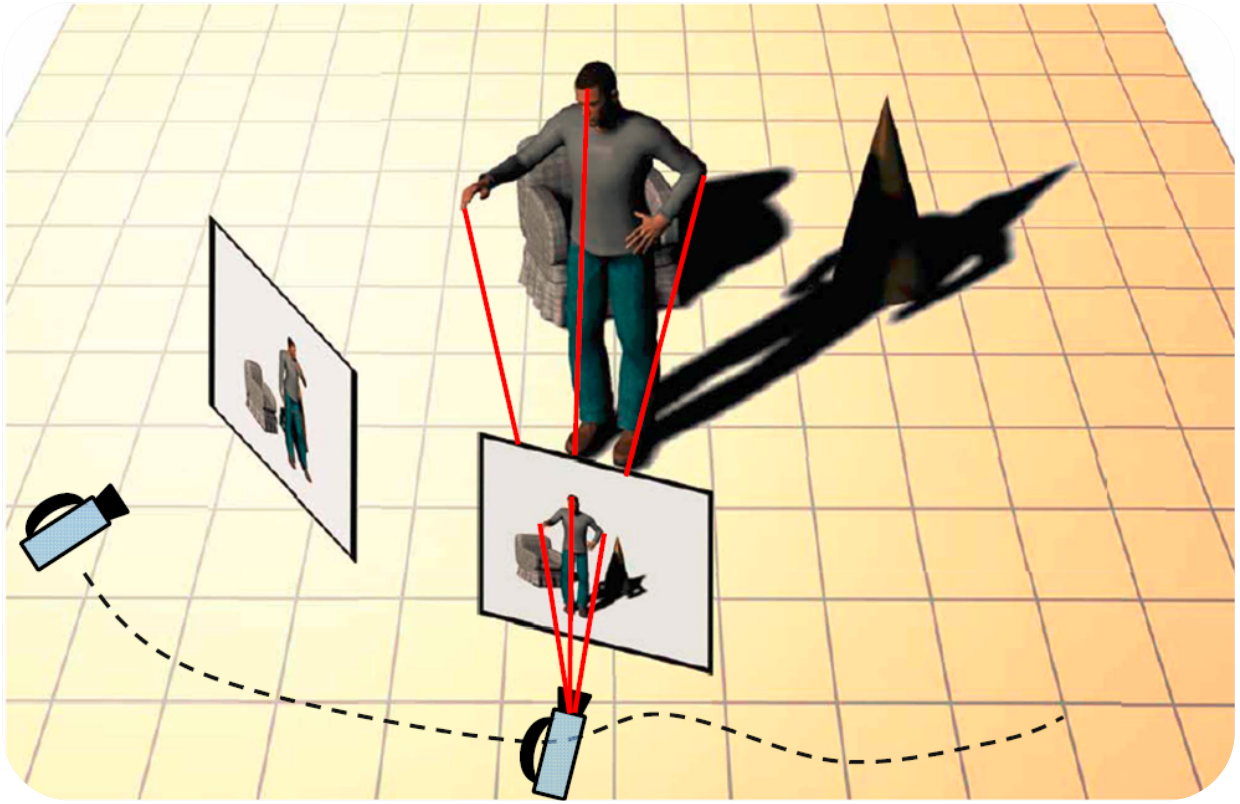
Can Epipolar geometry be used?

Considering only one image, we obtain the same 3D constraint



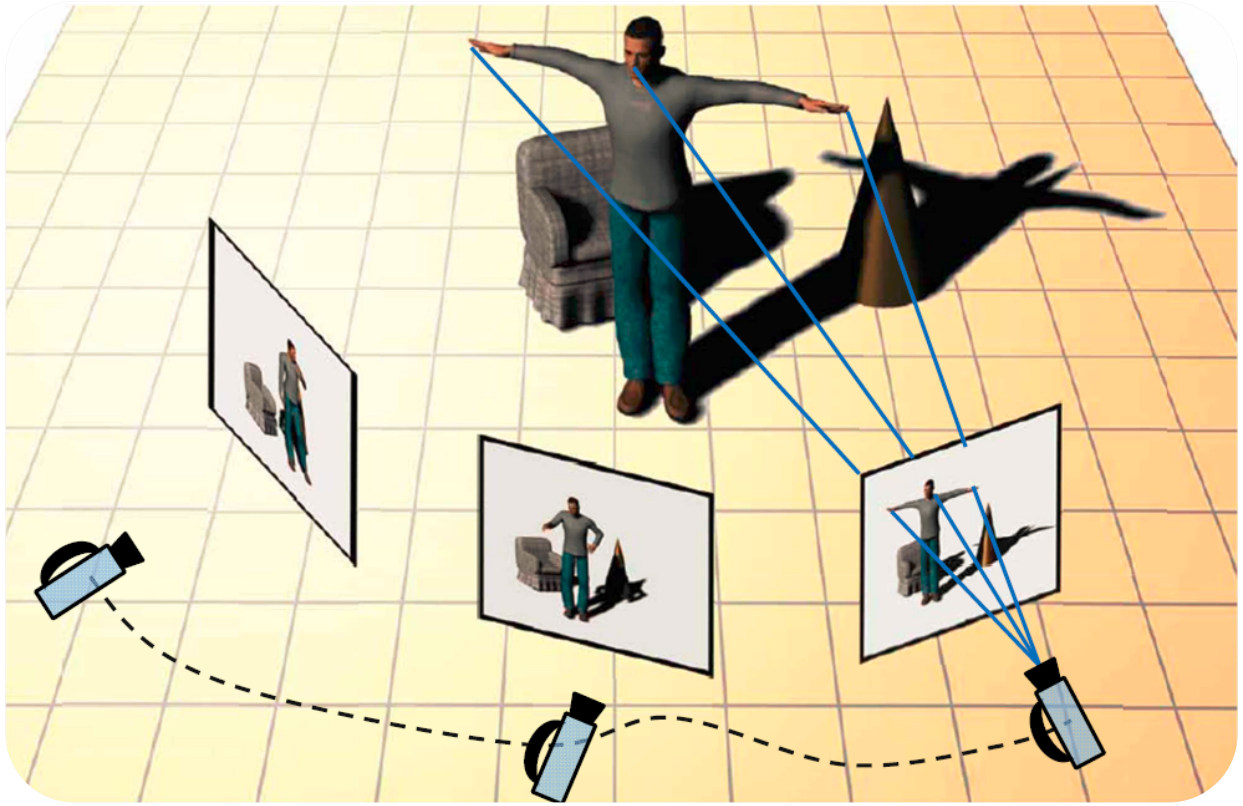
Epipolar geometry ~~can be used~~

After acquiring a new image, we obtain a similar constraint but now triangulation is not available since the shape is non rigid



Epipolar geometry ~~can be used~~

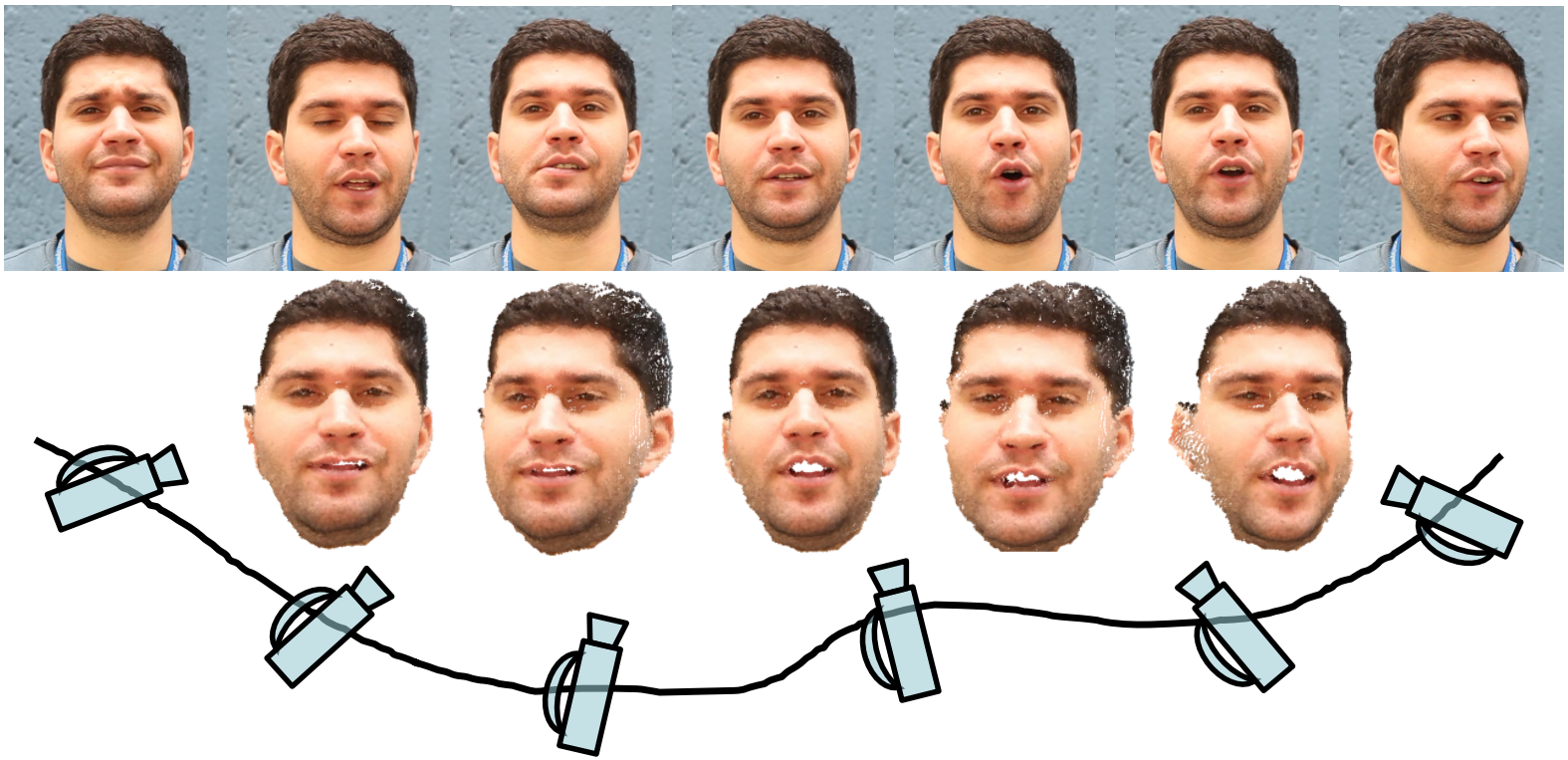
After acquiring a new image, we obtain a similar constraint but now triangulation is not available since the shape is non rigid



Non-Rigid
Structure from Motion

Non-Rigid Structure from Motion

Given: a monocular video (or a collection of pictures)
We want: simultaneously recovering the 3D shape of a time-varying object (4D estimation) and the camera motion



Some Results

Solving the problem

The problem can be solved by:

- **Factorization**: a closed-form solution can be achieved by using SVD factorization, enforcing a specific rank (this can change as a function of the type of camera model, or the type of scene). In theory, it is hard to accurately enforce constraints
- **Non-linear Optimization**: the solution is achieved iteratively, the computational cost can be bigger but additional priors can be enforced accurately

In terms of processing, the problem can be solved:

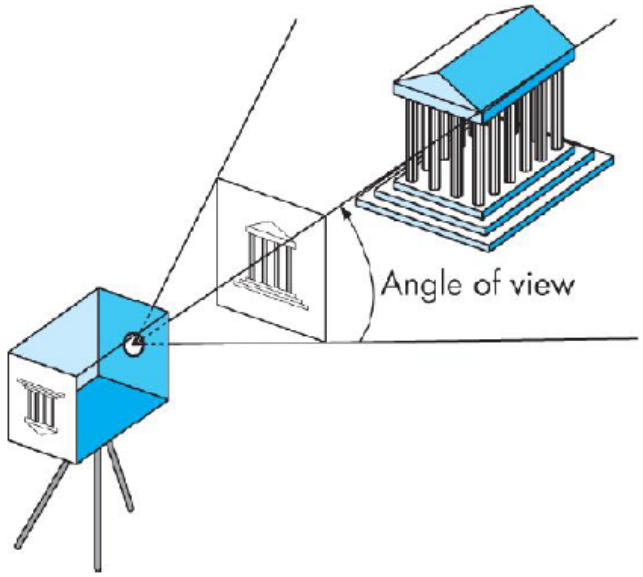
- **Offline:** all the frames are processed at once, after video capture
- **Online:** the frames are processed as the data arrive, frame by frame. More real applications, but can become less accurate

Non-Rigid Structure from Motion

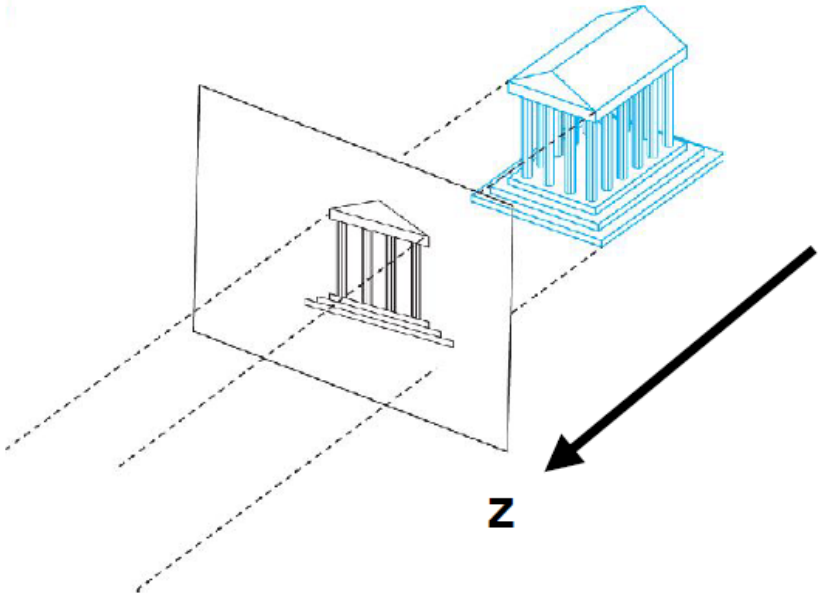
Problem Statement

A Reminder of Camera Models

- Perspective camera: All rays converge to the optical center
- Orthographic camera: All rays are parallel. Z-coordinate is irrelevant in the projection



Perspective camera



Orthographic camera

3D-to-2D: Perspective Model

A *p*-th 3D point $\mathbf{X}_p=[X_p, Y_p, Z_p]^T$ in homogeneous coordinates can be related with its 2D projection $\mathbf{x}_p=[x_p, y_p]^T$ by means of a matrix $\mathbf{P}^i$ for the *i*-th image, such as:

$$\begin{bmatrix} \mathbf{x}_p^i \\ 1 \end{bmatrix} = [\mathbf{P}^i] \begin{bmatrix} \mathbf{X}_p \\ 1 \end{bmatrix}$$

where $\mathbf{P}^i$ is 3x4 matrix as:

$$\mathbf{P}^i = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

3D-to-2D: Orthographic Model

A *p*-th 3D point $\mathbf{X}_p=[X_p, Y_p, Z_p]^T$ can be related with its 2D projection $\mathbf{x}_p=[x_p, y_p]^T$ by means of a matrix $\mathbf{R}^i$ for the *i*-th image, such as:

$$\left[\mathbf{x}_p^i\right] = \left[\mathbf{R}^i\right]\left[\mathbf{X}_p\right] + \mathbf{t}_p^i$$

where $\mathbf{R}^i$ is 2x3 matrix and $\mathbf{t}^i$ is a 2x1 translation vector as:

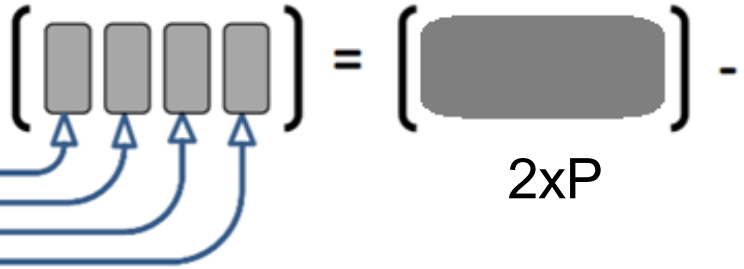
$$\mathbf{R}^i = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{bmatrix} \qquad \mathbf{t}^i = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

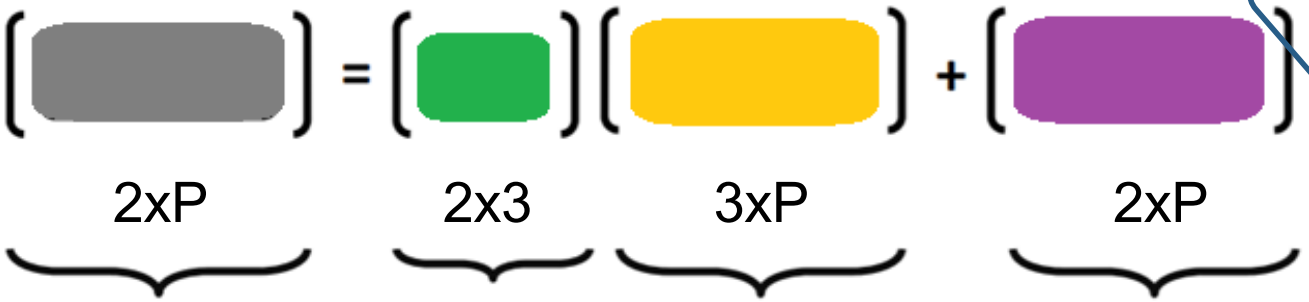
In practice, we subtract the translations by assuming centered observations (i.e., they are equivalent to the mean values of $\mathbf{x}_p$). For later computations, we will approximate $\mathbf{x}_p = \mathbf{x}_p - \mathbf{t}^i$

Problem Statement

Orthographic camera

A 2D image point is represented as a 2-vector containing the image coordinates at the given frame





Measurement Matrix

Camera Matrix

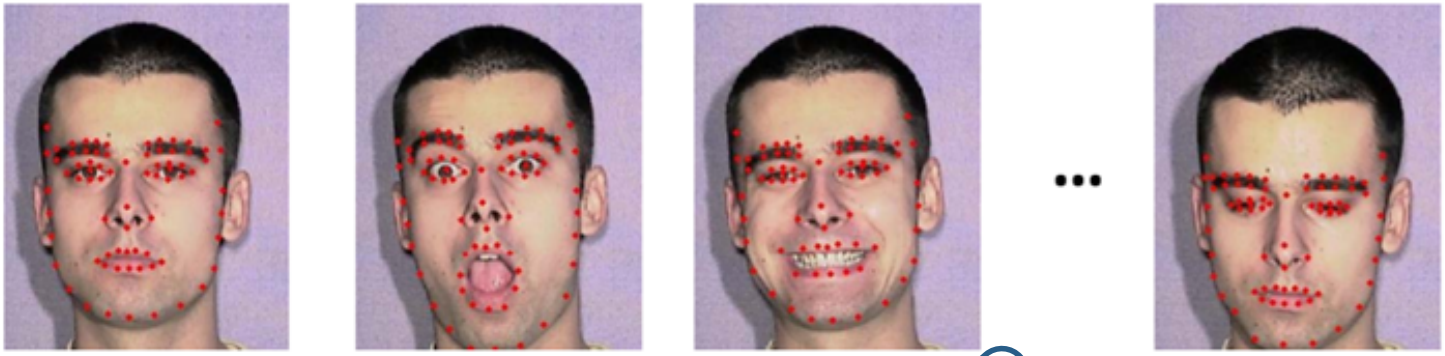
3D shape Matrix

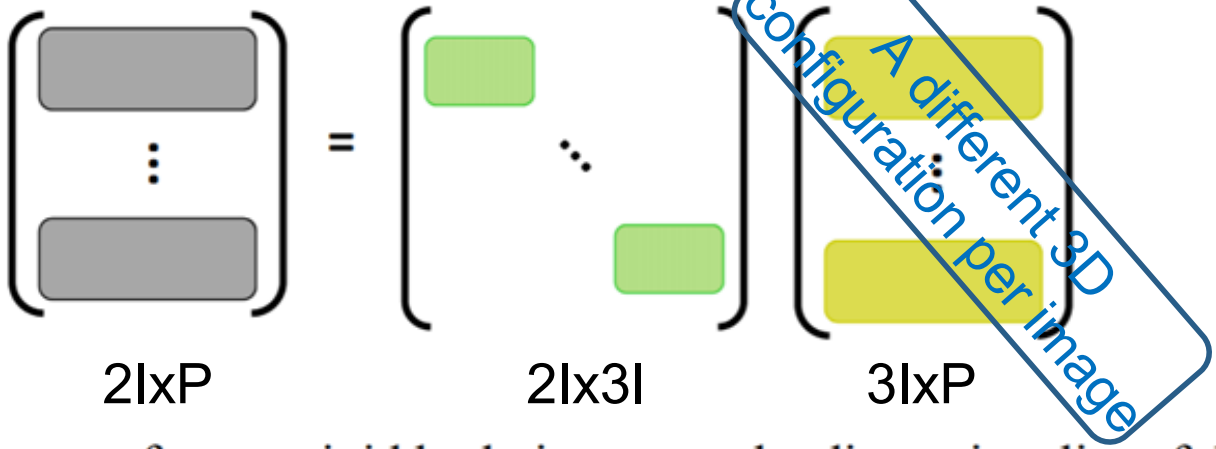
2D translation

In the rigid case, it is the same

Full Linear Relation

Orthographic camera

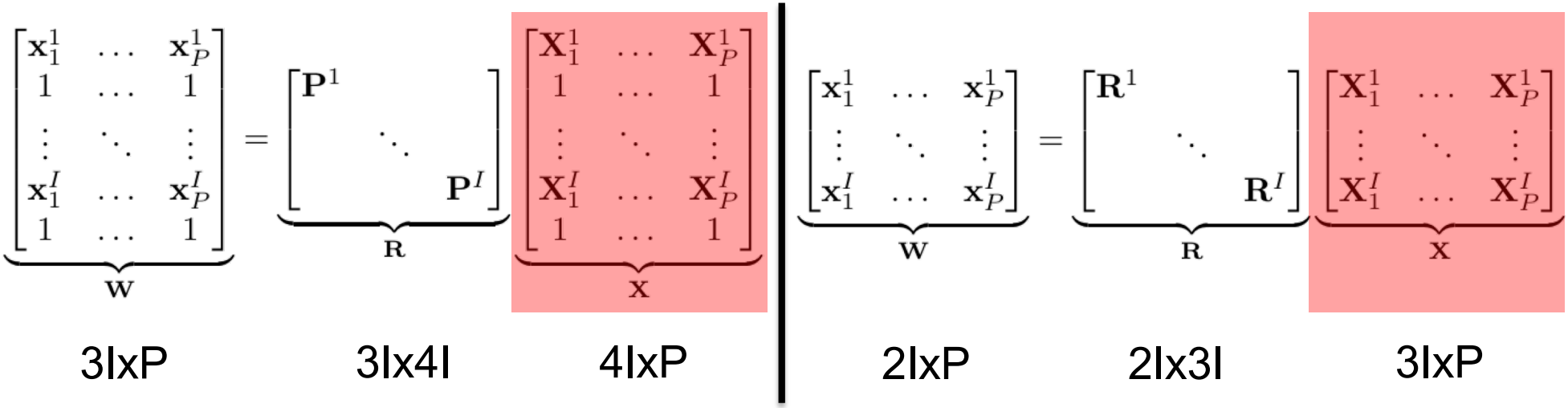




Each image of a non-rigid body increases the dimensionality of the problem and the number of unknowns.

Measurement Matrix

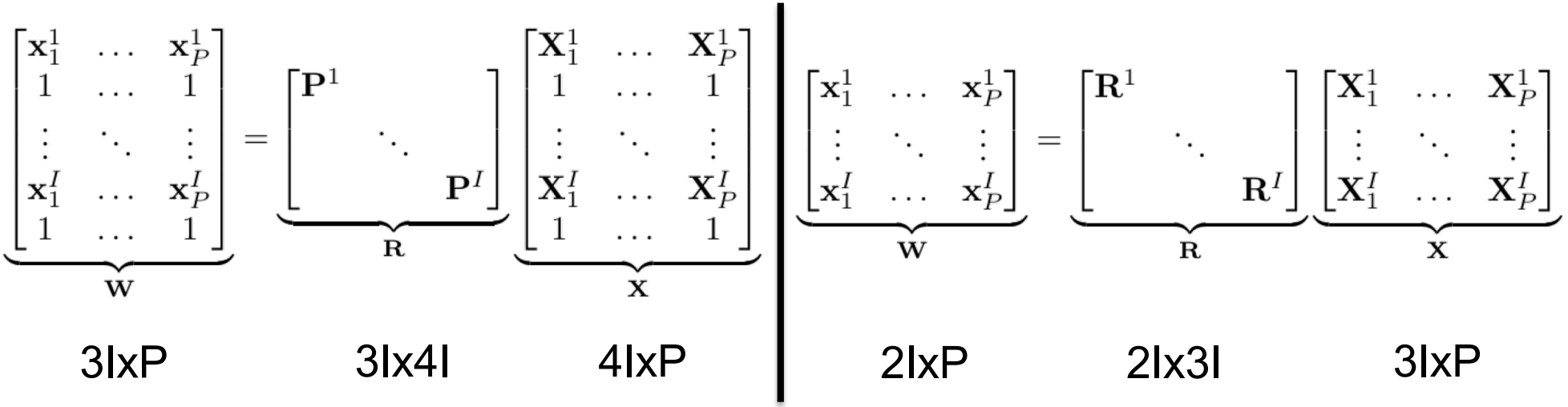
Considering P non-rigid 3D points observed in I RGB images, we can collect all observations to obtain a linear system such as:



where **W** is a 3IxP matrix, **P** is 3Ix4I, and **X** is 4IxP for the perspective case (relation on the left); and **W** is a 2IxP matrix, **R** is 2Ix3I, and **X** is 3IxP for the perspective case (relation on the right)

What about the rank of W?

Considering P non-rigid 3D points observed in I RGB images, we can collect all observations to obtain a linear system such as:



Perspective camera

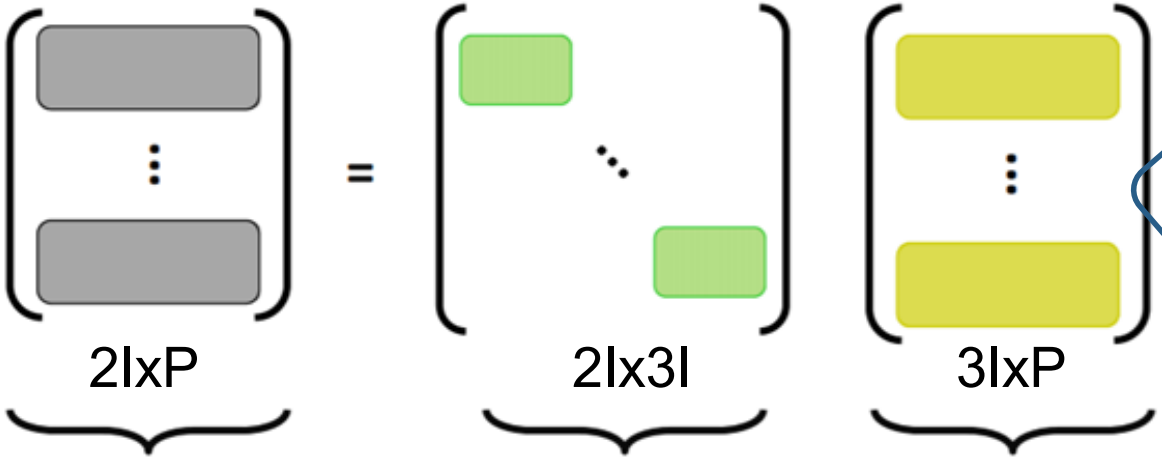
Orthographic camera

rank(**W**)≤ min(3I,P)

rank(**W**)≤ min(2I,P)

A severely ill-posed problem

Orthographic camera



This is an explosion of variables

2ip entries << 6i variables + 3ip variables

A Toy Comparison

Let us assume a 1 minute video with just 100 tracked points, and considering only the estimation of the 3D shape

Rigid Case

Input data:
100 points x 60 sec x 30 Hz x 2
= *360,000* measurements
Unknowns:
100 points x 3
= *300* unknowns

well-posed problem

Non-Rigid Case

Input data:
100 points x 60 sec x 30 Hz x 2
= *360,000* measurements

ill-posed problem

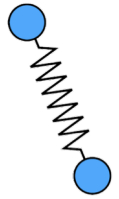
How can I solve the problem?

The art of priors

Including deformation priors is substantially more difficult than using simple rigidity

Many possibilities were presented

$$F_s(L) = k_s|L - L_0|$$



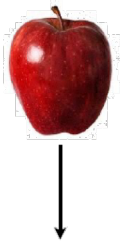
A wide variety of priors in literature:

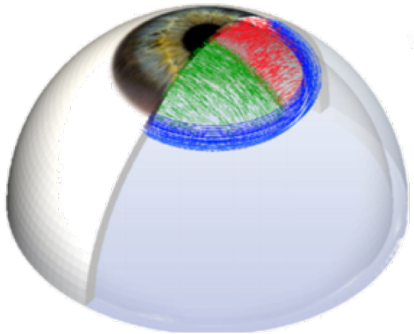
- **Physical priors**. Particle dynamics, elasticity, finite elements, and many others
- **Probabilistic priors**. Low-rank models on shape, trajectory, shape-trajectory or force domains. Union of subspaces, Gaussian priors
- **Geometric priors**: isometric, as rigid as possible, bone lengths, quadratic models
- **Temporal priors**: temporal-coherent deformations
- **Piecewise priors**
- **Many others**

$$\frac{\nu E}{1-\nu^2}\nabla(\nabla^\top \mathbf{u}) + \frac{E}{2(1+\nu)}\nabla^2\mathbf{u} = -\mathbf{f}_{xy}$$

$$F_g = -mg$$

$$g = (0, 0, -9.8)\ \mathrm{m/s^2}$$

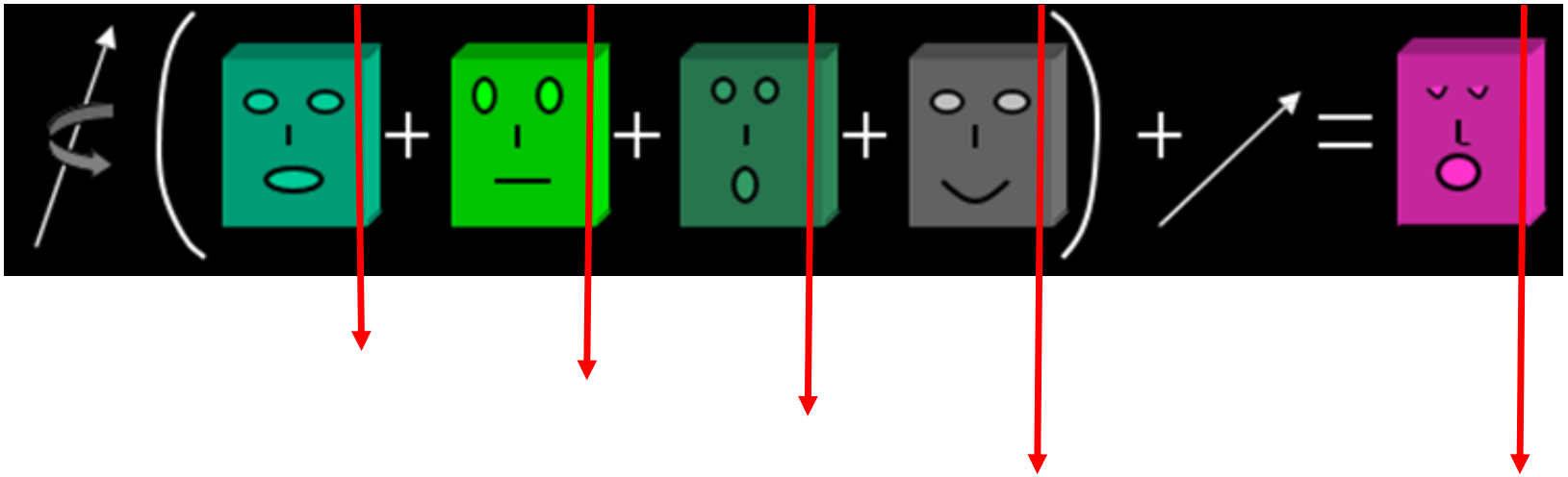




Shape Linear Subspace
(a probabilistic prior)

A Low-Rank Shape Model

Basically, the non-rigid 3D shape can be obtained as a *linear combination of fixed shape vectors*. For every combination of weight coefficients, a different solution can be achieved:



Rotation

Linear combination of some shapes

Translation

Your estimation

Including the low-rank shape model

We approximate the 3D shape by a linear combination of K shape vectors **b** (normally, K << P or I). For every k-th component, a weight coefficient l_k is needed. As the shape is non-rigid, by modifying the coefficients for every i-th image, we will change the 3D shape as:

$$\underbrace{\begin{bmatrix} \mathbf{X}_1^1 & \dots & \mathbf{X}_P^1 \\ \vdots & \ddots & \vdots \\ \mathbf{X}_1^I & \dots & \mathbf{X}_P^I \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} l_1^1\mathbf{I}_3 & \dots & l_K^1\mathbf{I}_3 \\ \vdots & \ddots & \vdots \\ l_1^I\mathbf{I}_3 & \dots & l_K^I\mathbf{I}_3 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} \overbrace{\mathbf{b}_{11} \quad \dots \quad \mathbf{b}_{1P}}^{\mathbf{B}_k} \\ \vdots \quad \ddots \quad \vdots \\ \mathbf{b}_{K1} \quad \dots \quad \mathbf{b}_{KP} \end{bmatrix}}_{\mathbf{B}}$$

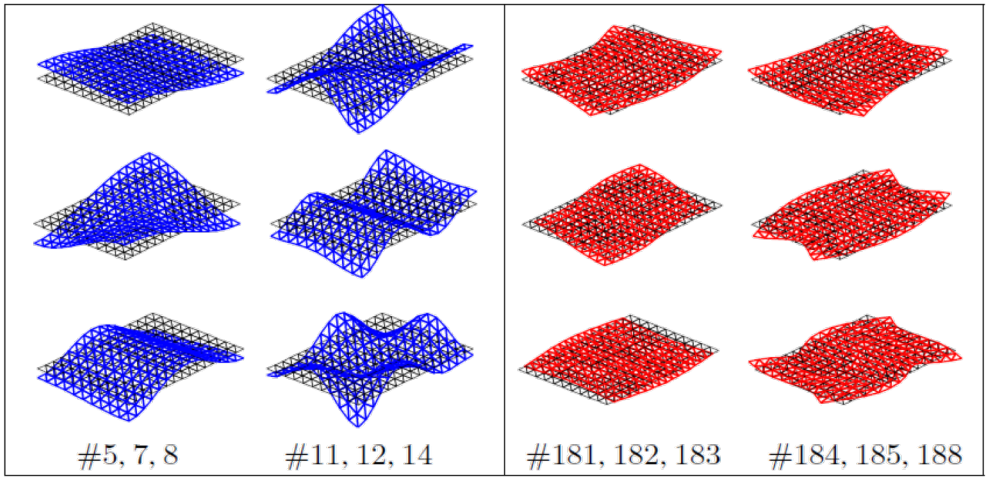
3IxP 3Ix3K 3KxP

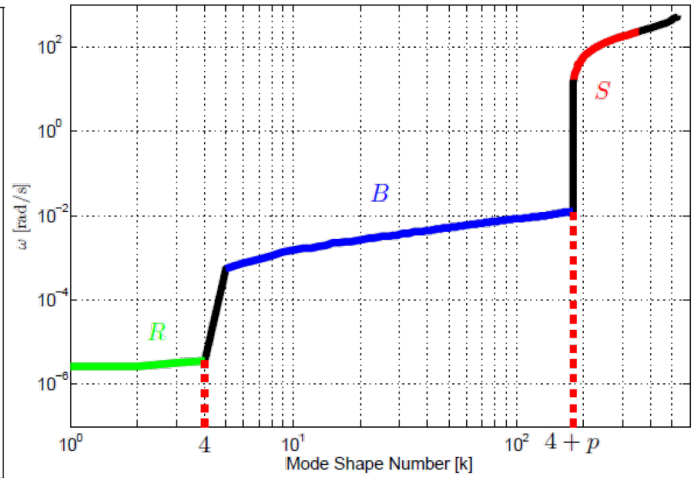
Another type of expression for the i-th image: $\mathbf{X}^i = \sum_{k=1}^{K} l_k^i \mathbf{B}_k$

Shape Basis Estimation

In non-rigid structure from motion, we have some alternatives to estimate the shape basis:

- The most natural is to learn it on the fly, using only the input data
- The input data can also be used to estimate a shape basis from a shape at rest (like a mean shape) by applying:
  - Modal analysis based on physical models
  - Spectral analysis based on a distance matrix
- If training data are assumed, we learn it by means of a learning approach (PCA, deep based, etc.). This approach is supervised





Non-Rigid
Structure from motion
by factorization

Including the low-rank shape model

Orthographic camera

Thanks to the relation between the 3D shape and the shape basis:

$$\underbrace{\begin{bmatrix} \mathbf{X}_1^1 & \dots & \mathbf{X}_P^1 \\ \vdots & \ddots & \vdots \\ \mathbf{X}_1^I & \dots & \mathbf{X}_P^I \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} l_1^1\mathbf{I}_3 & \dots & l_K^1\mathbf{I}_3 \\ \vdots & \ddots & \vdots \\ l_1^I\mathbf{I}_3 & \dots & l_K^I\mathbf{I}_3 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \end{bmatrix}}_{\mathbf{B}}$$

3IxP 3Ix3K 3KxP

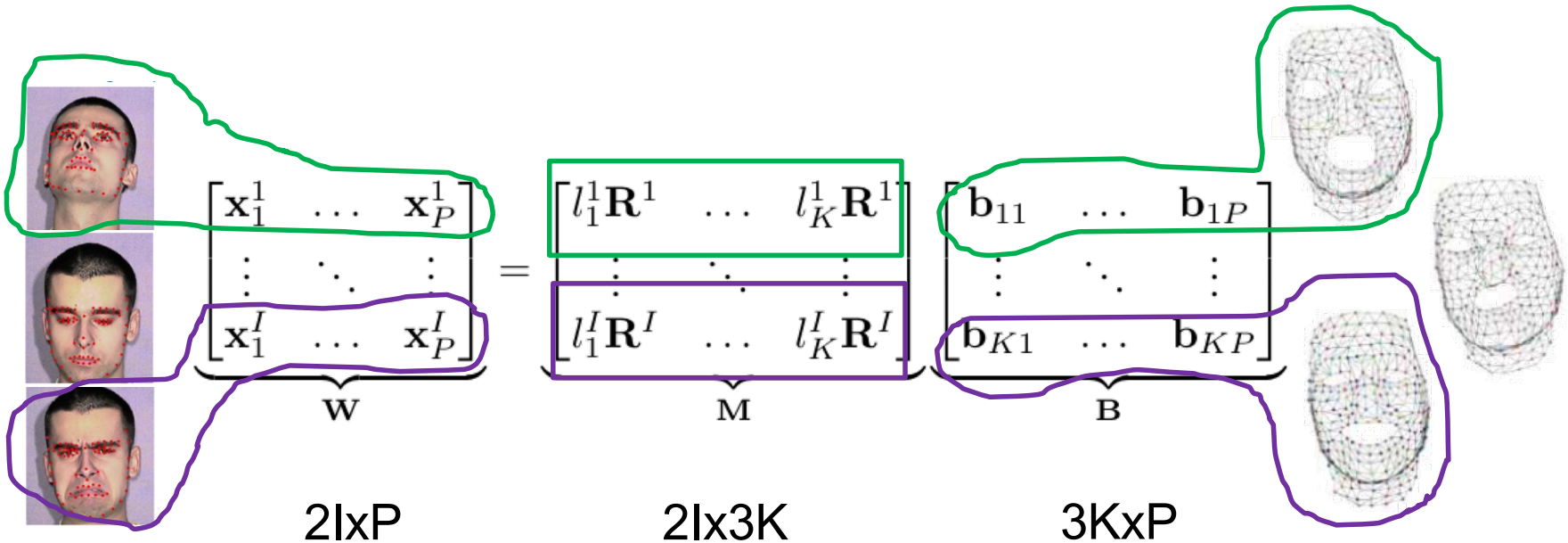
we obtain the projection equation by using the low-rank shape model as:

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} l_1^1\mathbf{R}^1 & \dots & l_K^1\mathbf{R}^1 \\ \vdots & \ddots & \vdots \\ l_1^I\mathbf{R}^I & \dots & l_K^I\mathbf{R}^I \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \end{bmatrix}}_{\mathbf{B}}$$

2IxP 2Ix3K 3KxP

Including the low-rank shape model

Orthographic camera



What about the perspective case?

A similar analysis can be followed, but now, considering homogeneous coordinates. We can obtain:

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} l_1^1\hat{\mathbf{P}}^1 & \dots & l_K^1\hat{\mathbf{P}}^1 & \bar{\mathbf{P}}^1 \\ \vdots & \ddots & \dots & \vdots \\ l_1^I\hat{\mathbf{P}}^I & \dots & l_K^I\hat{\mathbf{P}}^I & \bar{\mathbf{P}}^I \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{B}}$$

3IxP 3Ix3K+1 3K+1xP

$$\mathbf{P}^I = [\hat{\mathbf{P}}^I | \bar{\mathbf{P}}^I]$$

3x3 3x1

Factorization

In both cases, the goal is to infer the motion factor (**P** or **R**) and the 3D coordinates **X** of the observed non-rigid object from 2D point tracks in a monocular video **W**:

Orthographic camera

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} l_1^1\mathbf{R}^1 & \dots & l_K^1\mathbf{R}^1 \\ \vdots & \ddots & \vdots \\ l_1^I\mathbf{R}^I & \dots & l_K^I\mathbf{R}^I \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \end{bmatrix}}_{\mathbf{B}}$$

2IxP 2Ix3K 3KxP

Perspective camera

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} l_1^1\hat{\mathbf{P}}^1 & \dots & l_K^1\hat{\mathbf{P}}^1 & \bar{\mathbf{P}}^1 \\ \vdots & \ddots & \dots & \vdots \\ l_1^I\hat{\mathbf{P}}^I & \dots & l_K^I\hat{\mathbf{P}}^I & \bar{\mathbf{P}}^I \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{B}}$$

3IxP 3Ix3K+1 3K+1xP

The full linear system

$$W = MB$$

Two factors: motion factor **M** (camera rotation and weight coefficients) and shape one as a product of **B** and the coefficients

More on factorization

Orthographic camera

Because **M** is a 2Ix3K matrix and **B** is a 3KxP matrix, the rank of **W** is 3K. If we apply SVD to **W**, we will have only ***3K non-zero singular values***

However, measurements are normally noisy, and in practice the rank will not be 3K. We have to impose it

Applying SVD factorization, we have:

$$\mathbf{W} = \mathbf{U}\mathbf{A}\mathbf{V}^T = [\mathbf{U}\sqrt{\mathbf{A}}][\sqrt{\mathbf{A}}\mathbf{V}^T] = [\mathbf{U}\sqrt{\mathbf{A}}\mathbf{Q}][\mathbf{Q}^{-1}\sqrt{\mathbf{A}}\mathbf{V}^T]$$

We need to tune the rank K a priori

i.e., $\mathbf{M}=\mathbf{U}\sqrt{\mathbf{A}}\mathbf{Q}$ and $\mathbf{B}=\mathbf{Q}^{-1}\sqrt{\mathbf{A}}\mathbf{V}^T$ (the two factors we look for)

Many solutions can be achieved by modifying **Q**. Of course, for all invertible 3Kx3K **Q** matrices

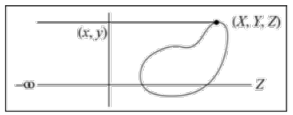
Metric Upgrade

How is **Q** computed?

Enforcing orthogonality constraints on the camera rotation. A rotation matrix always has some properties (it is not a random matrix), since lies in the SO(3) manifold

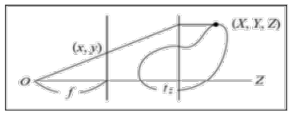
Be careful. Now, matrix **M** also includes the weight coefficients in addition to the camera rotations!

1.Orthographic:



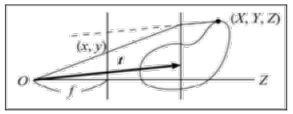
$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^\top$ (3 x 2) $=$ $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 2x2 identity matrix

2.Weak perspective:



$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^\top$ (3 x 2) $=$ $\begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ 2x2 diagonal matrix

3.Para perspective:



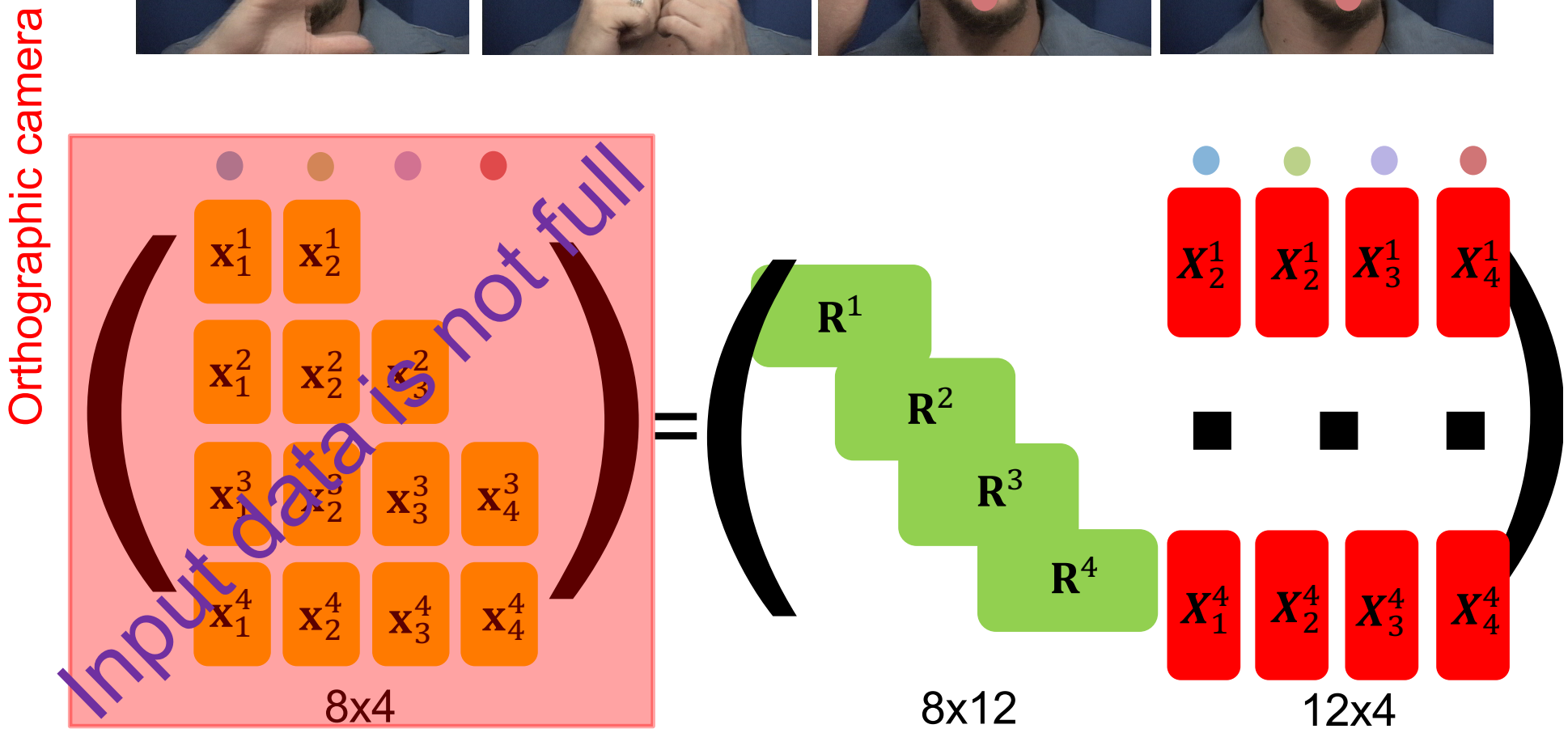
$\mathbf{R}_i$ (2 x 3) $\mathbf{R}_i^\top$ (3 x 2) $=$ $\begin{bmatrix} a & c \\ c & b \end{bmatrix}$ 2 x2 full matrix

But in many cases, we cannot observe all the points in all the images

==

Missing tracks

A toy example with missing tracks



Handling missing tracks

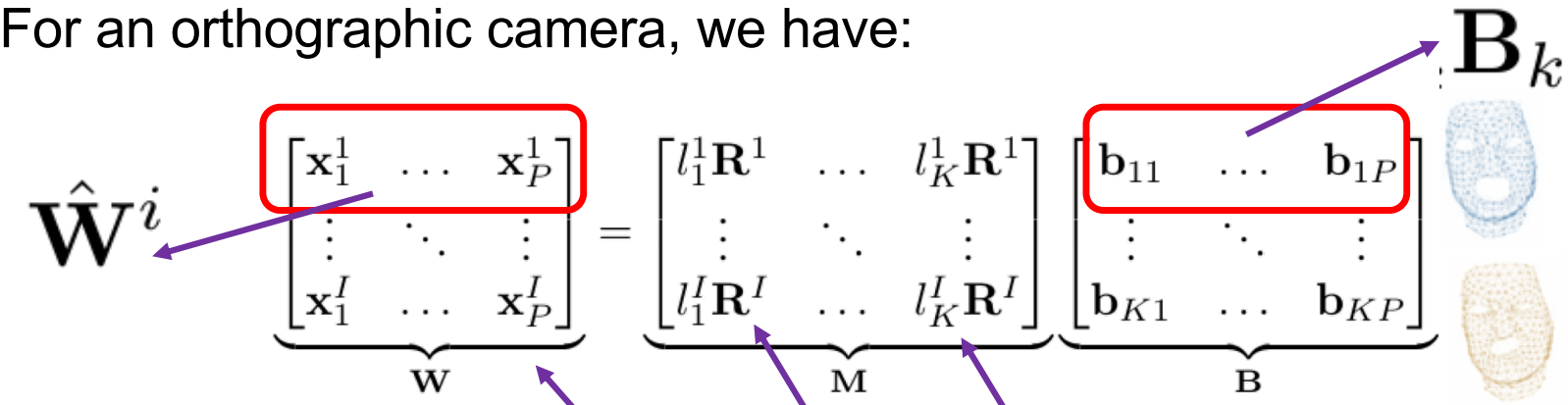
Two alternatives are possible:

- Applying a *matrix completion* algorithm to infer the missing entries, and then run factorization over the full measurement matrix
- No consider missing entries in the formulation by applying non-linear optimization. Once the 3D model and camera pose are computed, the 2D missing tracks can be inferred too

Non-Rigid
Structure from Motion
by Non-Linear Optimization

Problem Statement

For an orthographic camera, we have:



The problem (compacting over the points) can be formulated as:

$$\underset{\mathbf{R}^i,\mathbf{B}_k,l_k^i,\mathbf{t}^i}{\arg\min} \sum_{i=1}^{I} \|\hat{\mathbf{W}}^i - \mathbf{R}^i \sum_{k=1}^{K} l_k^i \mathbf{B}_k - \mathbf{t}^i\|^2$$

and we perform non-linear optimization by minimizing a geometric error cost function. Translation $\mathbf{t}^i$ is optional

Bundle Adjustment

Normally, the Levenberg-Marquardt method is used to minimize the problem. We need a Jacobian matrix **J** as the derivative of the function with respect to the unknowns (**R**, **B** and the set of weight l_k)

Again, there are many variants on how to proceed to reduce the computational complexity of the problem:

- Alternate minimization of motion and shape parameters
- *Sparse methods*. The computation of **J** is complex, but it can be approximated by considering a *binary pattern*

Initialization: The optimization can be initialized assuming a rigid shape, i.e., using rigid factorization or non-linear optimization for a rigid shape

Bundle Adjustment

The bundle adjustment method:

- Minimize the cost function with *Levenberg-Marquadt*
- Exploit the *sparseness* of the Jacobian function matrix to *decrease computation and memory requirements*

The Levenberg-Marquadt algorithm does:

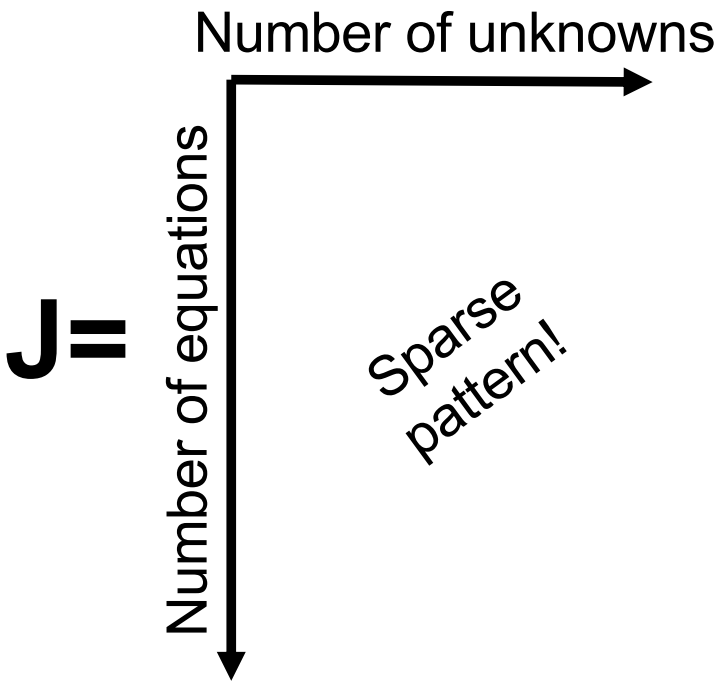
- Mixture of Gauss-Newton and Gradient descent
- Behaves like Gauss-Newton when close to the minimum (quadratic region)
- Gradient descent when the prediction is poor
- Depends on a parameter θ that controls the *mixture* of Gauss-Newton and Gradient descent as:

$$(\mathbf{J}\mathbf{J}^{T}+\theta\mathbf{I})\,\delta\mathbf{p} = -\mathbf{g}$$

Parameters we want to estimate

Exercise

Let us assume a monocular video of 3 images, where 6 points are observed. Considering the map is non-rigid and the visibility is full, define the corresponding Jacobian matrix. A low-rank shape model of rank 2 can be considered



Including priors

As in the rigid case, we can apply temporal smoothness priors, but now, in both camera motion and shape deformation (be careful when input data are a collection of pictures). To this end, we may consider the expression:

$$\underset{\mathbf{R}^i,\mathbf{B}_k,l_k^i,\mathbf{t}^i}{\arg\min} \sum_{i=1}^{I} \|\hat{\mathbf{W}}^i - \mathbf{R}^i \sum_{k=1}^{K} l_k^i \mathbf{B}_k - \mathbf{t}^i\|^2 + \gamma \sum_{i=1}^{I-1} \|\mathbf{L}^i - \mathbf{L}^{i+1}\|_{\mathcal{F}}^2 + \phi \sum_{i=1}^{I-1} \|\mathbf{R}^i - \mathbf{R}^{i+1}\|_{\mathcal{F}}^2$$

where $\mathbf{L}^i$ includes all K weight coefficients in the i-th image

How can we obtain a sequential solution?

We solve the optimization in a sequential manner, considering the information *as the data arrive*. Future frames are not available. Two options:

- Pure sequential (frame by frame)
- Sliding window (from 3 to 5 consecutive frames)

Initialization is performed by rigid estimation (assuming just the initial frames). The problem is actually challenging

Then, can we infer 3D information from a single RGB image?

Yes… but with some constraints

Maybe, the most used approach to handle this scenario is assuming training data. For instance, a shape basis can be computed from training data, and on testing, we only have to adjust the weight coefficients. Supervised approach.

Obtaining training data is not easy in some scenarios

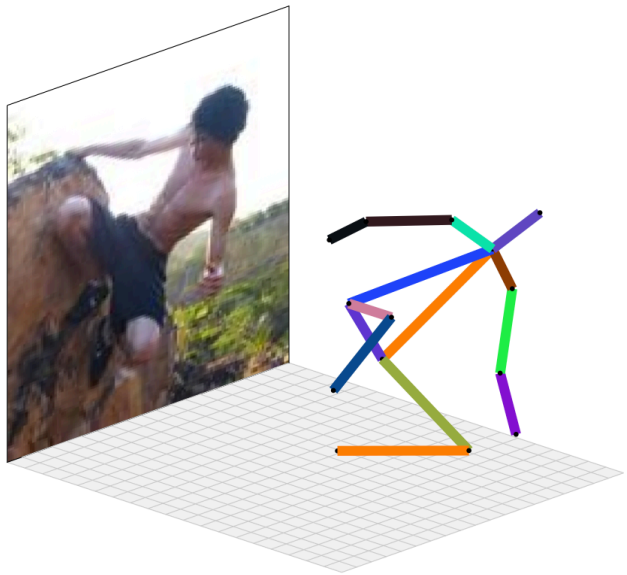
Training data for a type of object or deformation are not available to handle other cases

Alternative approaches can be also found in literature, by exploiting silhouettes or a mean shape in combination with geometrical priors to stablish correspondences

Exploiting Training Data

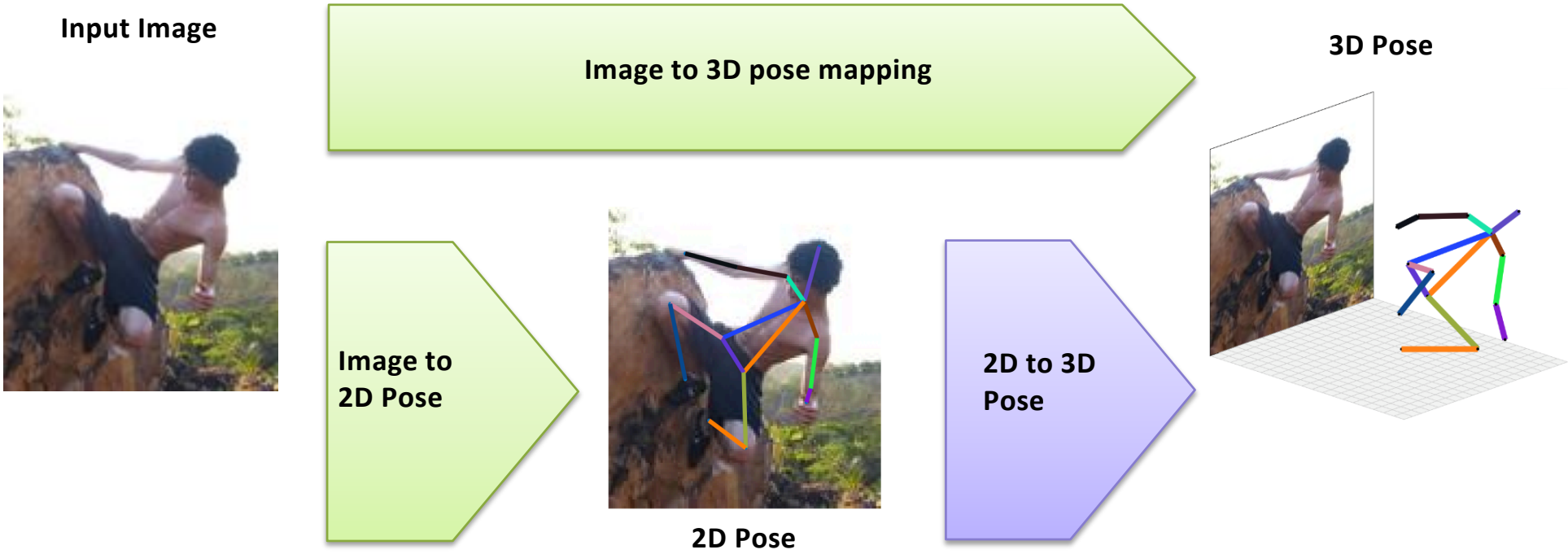
Problem Formulation

Objective: Given a **single image** of an object (a person in the example), estimate the 3D position of its body represented by a skeleton with P=14 joints:

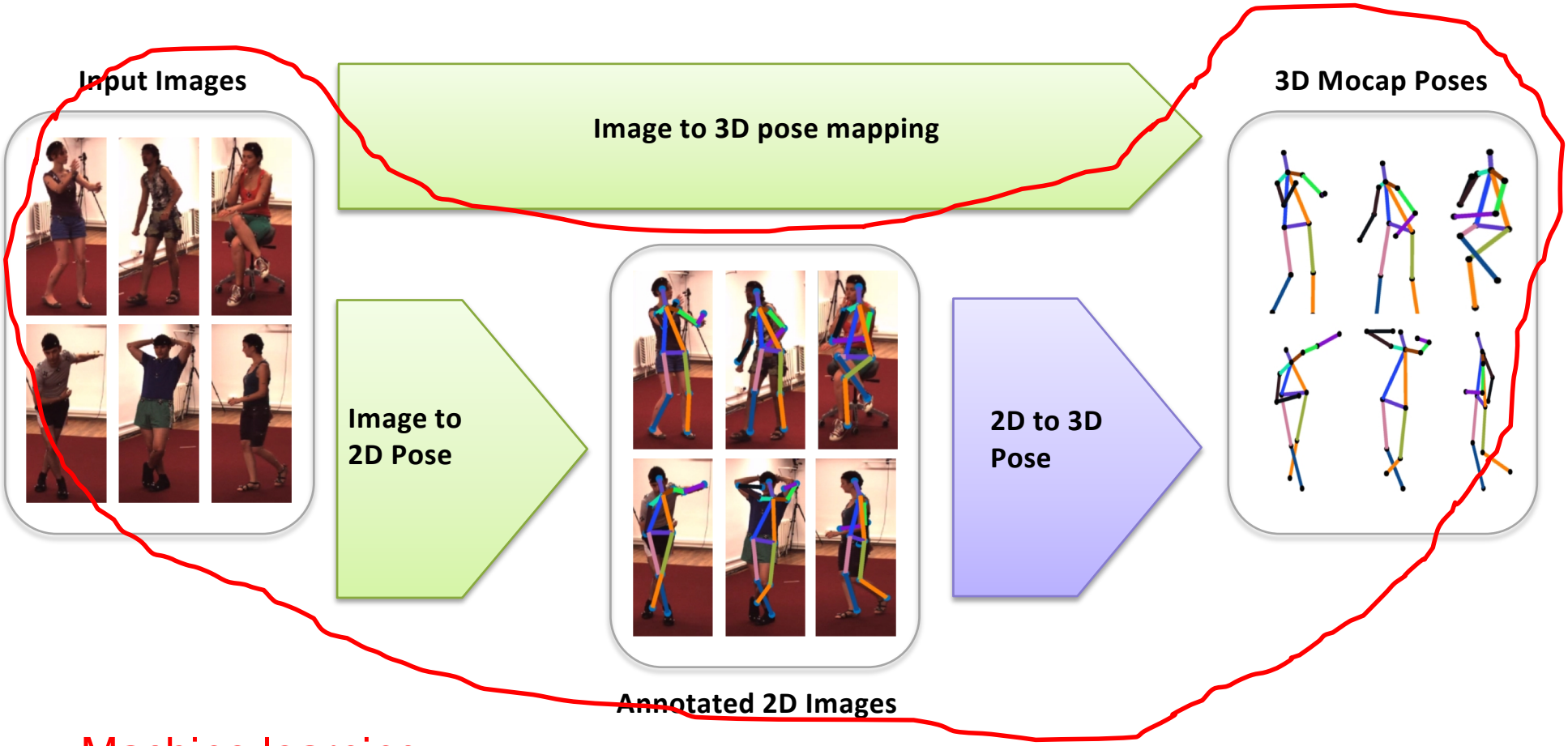


Challenge: This is an ill-posed problem; many different 3D poses have very similar 2D projections

From image to 3D

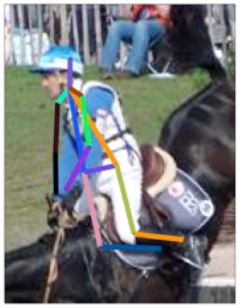


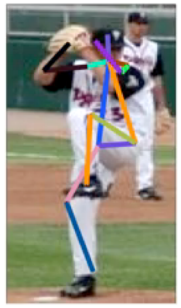
The Learning Process (Training)

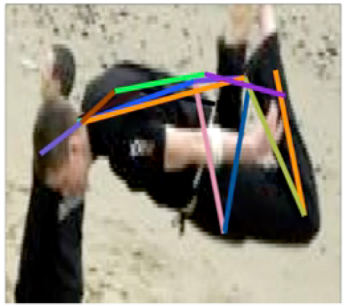
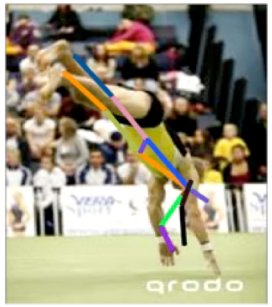


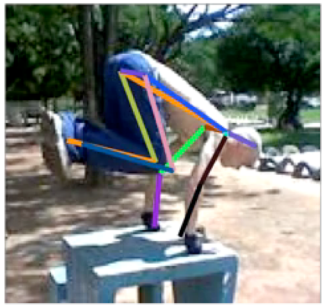
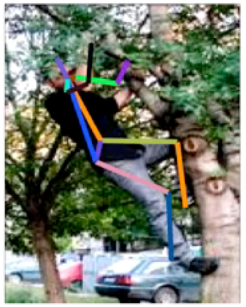
Machine learning algorithm

Some Results

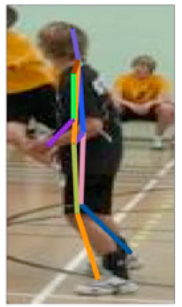


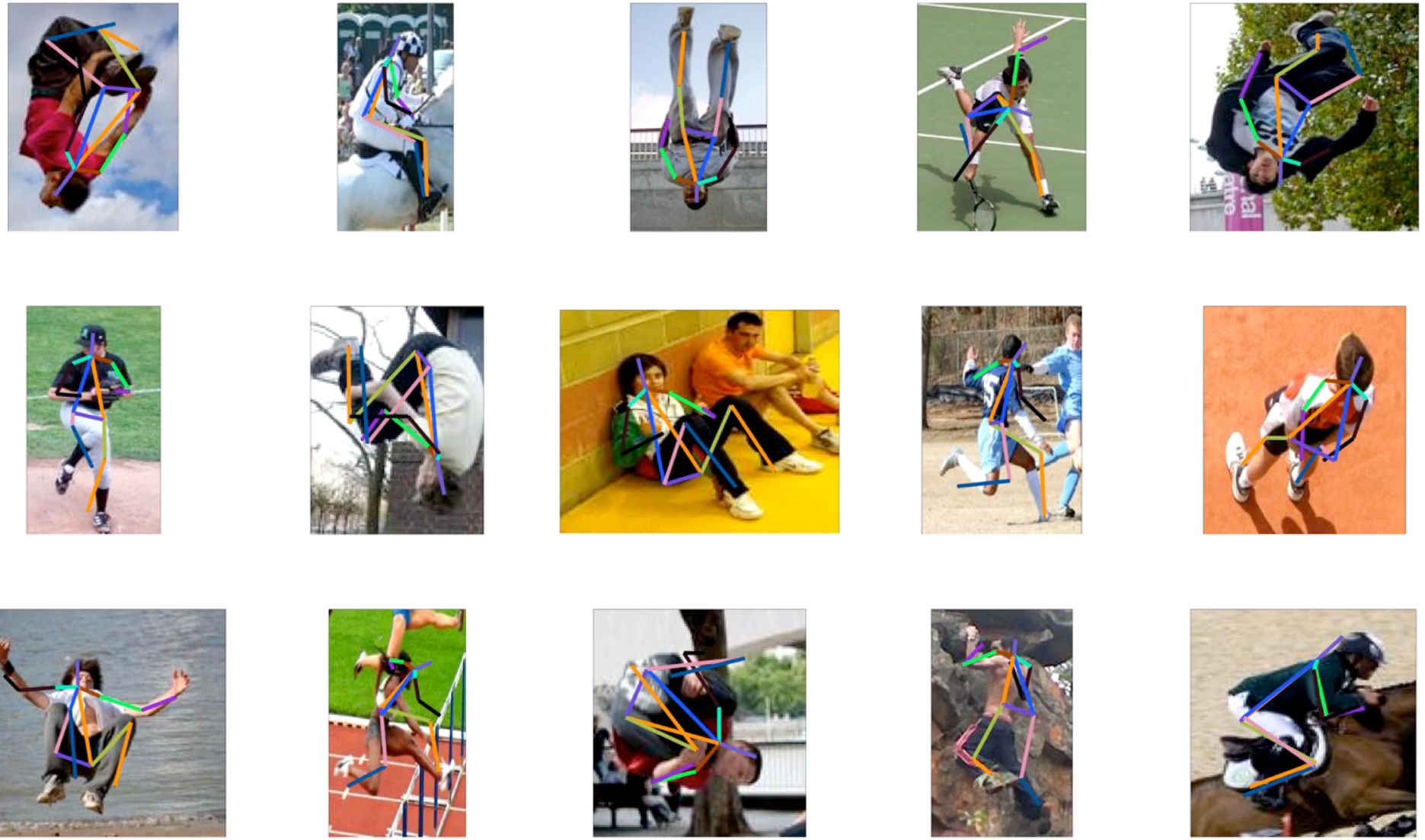






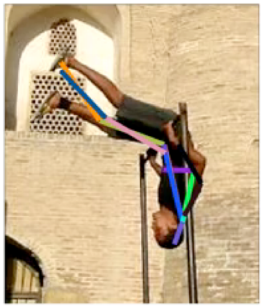
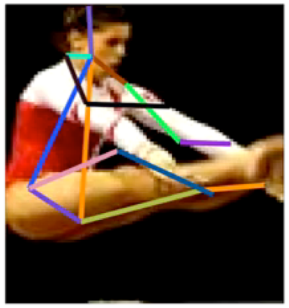


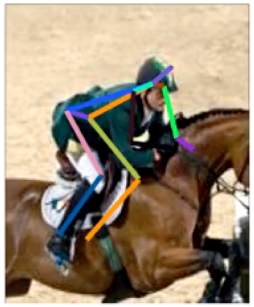
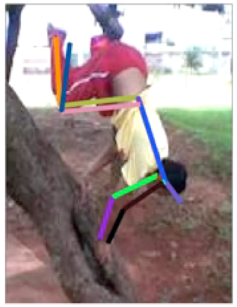

Some Results

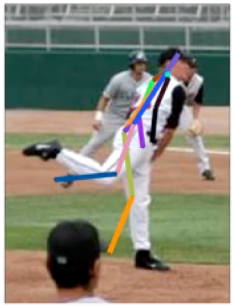
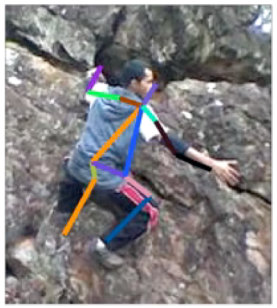
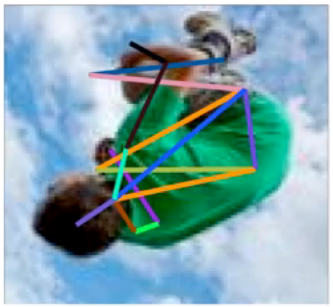
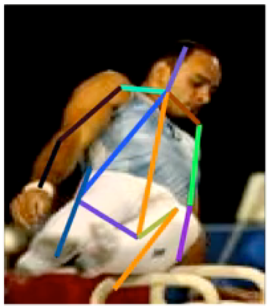
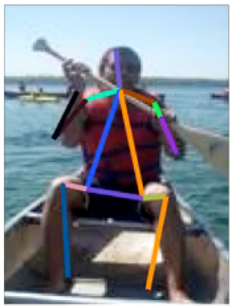
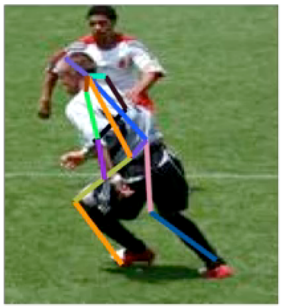


Some Results



Some Results: Failure Cases

Many 3D configurations can produce the same 2D projection

Things to remember

3D and 4D information can be obtained from a sequence of images

For rigid objects, the problem is well-posed. For non-rigid ones, it is inherently ill-posed (additional priors are necessary)

Model-based approaches can handle a wide variety of deformations. They are normally universal and generic. No supervision is needed

Data-based approaches require a lot of data to constrain the solution space. Obtaining *good* data can become hard. Only for a particular object or deformation (depending on the training data)

Future must be unsupervised, and probably, combining both model- and data-based approaches. With a hand-held camera, performing the estimation of multiple scenarios

Acknowledgments

Thanks to Kris Kitani, Yaser Sheikh, Alessio del Bue, Lourdes Agapito