Real-Time 3D Rotation Smoothing for Video Stabilization

Chao Jia, Zeina Sinno, and Brian L. Evans

Department of Electrical and Computer Engineering

The University of Texas at Austin

Asilomar Conference on Signals, Systems, and Computers

2014-11-03

Introduction

- Video recording by handheld cameras is growing exponentially due to:
  - *Compactness*
  - *Everywhere & Anytime*
  - *Easy Sharing*
  - *Good User Experience (touchscreen)*

⋮

Common Problem: Unwanted inter-frame jitter …

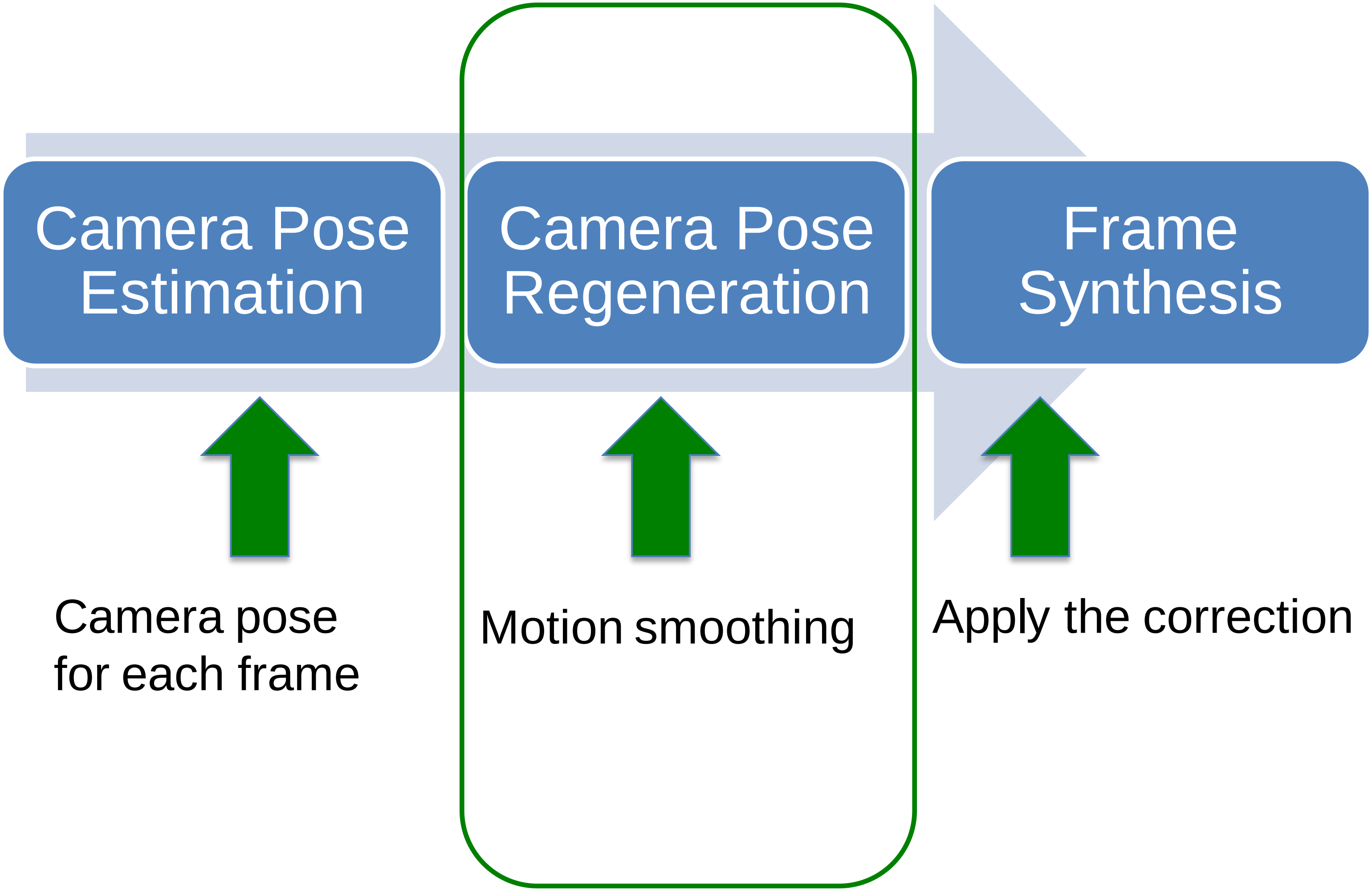
Introduction

Why online video stabilization

- Real-time delivery: video conferencing, broadcasting, etc.
- Improved user experience: What You See is What You Get.
- More efficient compression

Online Video Stabilization

- Removing unwanted jitter (**inter**-frame correction)



Motion Model Selection

- 2D motion: apparent pixel displacements

  Translation Similarity Euclidean Affine Projective

  Degrees of Freedom

- 3D real camera motion

  Rotation Full (Rotation + Translation)

Motion Model	**Estimation Complexity**	**Smoothing Effectiveness**	**Correction Complexity**
2D	high	low	low
3D Full	high	high	high
3D Rotation	low (**using gyro**)	high	low (**projective transform**)

- No approximation in 3D rotational stabilization (proposed method)
  - We are not assuming pure camera rotation
  - Translation is kept as is, and not smoothed

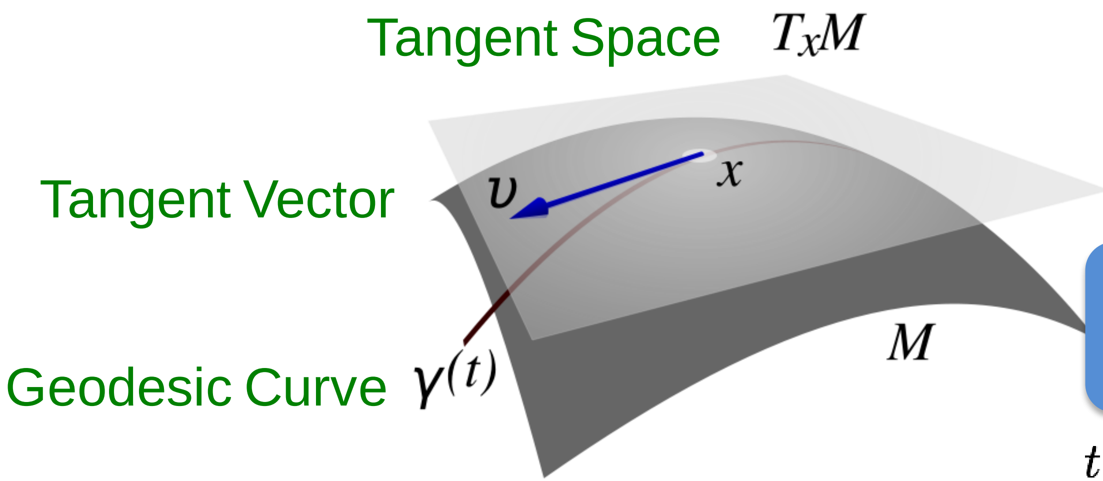
Online 3D Rotation Smoothing

- Classical approaches for 2D motion models
  - (1st order low-pass) IIR filtering
  - Kalman filtering with constant-velocity (CV) model

- Extension to 3D rotation smoothing
  - Euclidean space → SO(3) manifold
  - Ad-hoc projection for black-border constraint

3D Rotation Matrix

- Manifold of 3D Rotation Matrices
  - Special Orthogonal Group $\mathbf{SO(3)}$ $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$
  - An embedded submanifold in $\mathbb{R}^9$(dimension = 3)



Generalization of Straight Line in Euclidean Space

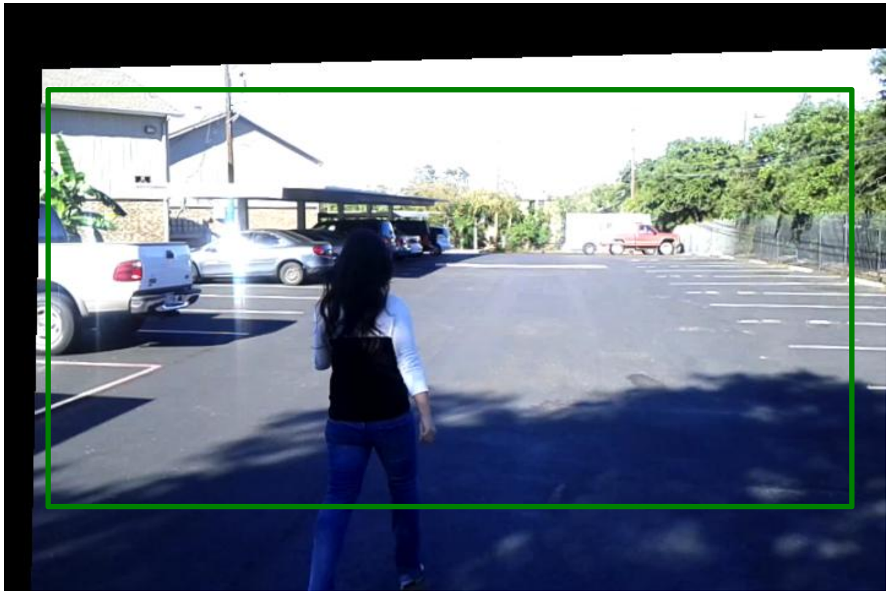
$$t \in [0, 1] \mapsto \gamma(t) = \exp_x(t \cdot v)$$

- Minimizing Geodesic & Geodesic Distance

$$d_g(\mathbf{R}, \mathbf{R}') = ||\mathrm{logm}(\mathbf{R}^{-1}\mathbf{R}')||_F$$

Constrained Motion Smoothing

- Inevitably some pixels are not visible after view change



$$\begin{bmatrix} \tilde{u}_{ij} \\ \tilde{v}_{ij} \end{bmatrix} = g\left(\mathbf{K}\tilde{\mathbf{R}}_i {\mathbf{R}_i}^{\mathrm{T}} \mathbf{K}^{-1} \begin{bmatrix} u_{ij} \\ v_{ij} \\ 1 \end{bmatrix}\right)$$

Correction by image warping

$$\begin{cases} 0 \le \tilde{u}_{ij} \le w \\ 0 \le \tilde{v}_{ij} \le h \end{cases}, \forall \begin{bmatrix} u_{ij} \\ v_{ij} \end{bmatrix} s.t. \begin{cases} c_1 \le u_{ij} \le c_2 \\ d_1 \le v_{ij} \le d_2 \end{cases}$$

Hard Constraint: All of the pixels in the cropped new frame should be visible in the original frame.

IIR-like 3D Rotation Smoothing

- First-Order IIR filtering

$$\hat{\boldsymbol{\theta}}_k = \alpha \hat{\boldsymbol{\theta}}_{k-1} + (1-\alpha)\boldsymbol{\theta}_k$$

SO(3) ✗

$$\hat{\boldsymbol{\theta}}_k = \operatorname*{argmin}_{\boldsymbol{\theta}} \alpha ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_{k-1}||^2 + (1-\alpha)||\boldsymbol{\theta} - \boldsymbol{\theta}_k||^2$$

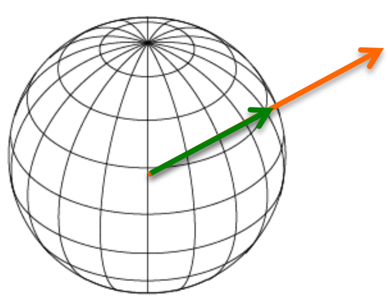
SO(3) ✓

Euclidean distance → Geodesic distance

$$\hat{\mathbf{R}}_k = \operatorname*{argmin}_{\mathbf{R}} \alpha d_g(\mathbf{R}, \hat{\mathbf{R}}_{k-1})^2 + (1-\alpha) d_g(\mathbf{R}, \mathbf{R}_k)^2$$

spherical linear interpolation (SLERP)

- Ad-hoc projection $\hat{\mathbf{R}} = \mathbb{P}(\hat{\mathbf{R}}^*) = \mathbf{R}\mathrm{expm}(\beta^* \mathrm{logm}(\mathbf{R}^{-1}\hat{\mathbf{R}}^*))$



Move closer to the original rotation if necessary for black-border constraint

UKF-based 3D Rotation Smoothing

- Constant-Velocity Model (widely used in target tracking)

smoothed rotation

angular velocity

$$\begin{bmatrix} \mathbf{q}_k \\ \boldsymbol{\omega}_k \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{k-1} \bigotimes q(\boldsymbol{\omega}_{k-1}) \\ \boldsymbol{\omega}_{k-1} + \mathbf{w}_k \end{bmatrix}$$

Dynamic Model

angular acceleration (process noise)

original rotation

$$\tilde{\mathbf{q}}_k = \mathbf{q}_k \bigotimes q(\mathbf{v}_k)$$

Measurement Model

jitter (measurement noise)

- Hard to solve on SO(3)
- Nonlinear on Euclidean space
- Solved approximately by unscented Kalman filter (UKF)

Proposed Algorithms

Algorithm IIR-like 3D Rotation Smoothing

Input: $\mathbf{q}_1, \cdots, \mathbf{q}_K$ (original rotations)
Output: $\hat{\mathbf{q}}_1, \cdots, \hat{\mathbf{q}}_K$ (smoothed rotations)
$\hat{\mathbf{q}}_1 = \mathbf{q}_1$
for $k = 2$ **to** K **do**
    $\hat{\mathbf{q}}_k = \text{slerp}(\mathbf{q}_k, \hat{\mathbf{q}}_{k-1}, \alpha)$
    $\hat{\mathbf{q}}_k \leftarrow \mathbb{P}(\hat{\mathbf{q}}_k)$
end for

1.54ms/frame

Algorithm UKF-based 3D Rotation Smoothing

Input: $\mathbf{q}_1, \cdots, \mathbf{q}_K$ (original rotations)
Output: $\hat{\mathbf{q}}_1, \cdots, \hat{\mathbf{q}}_K$ (smoothed rotations)
Parameters: $\mathbf{Q}, \mathbf{R}$ (process and measurement noise variance)
for $k = 1$ **to** K **do**
    Obtain unconstrained UKF estimate $\hat{\mathbf{q}}_k^*$, $\hat{\boldsymbol{\omega}}_k^*$, $\mathbf{P}_k$
    $\hat{\mathbf{q}}_k^* = \hat{\mathbf{q}}_k^* / ||\hat{\mathbf{q}}_k^*||_2$ (normalization)
    $\hat{\mathbf{q}}_k \leftarrow \mathbb{P}(\hat{\mathbf{q}}_k)$
    (Mean and covariance estimate to pass to the next stage are $\hat{\mathbf{q}}_k$, $\hat{\boldsymbol{\omega}}_k$, $\mathbf{P}_k$)
end for

6.97ms/frame

Experimental Results – 2D vs. 3D KF

Experimental Results– 2D vs. 3D KF

Experimental Results– 2D vs. 3D IIR

Experimental Results – 2D vs. 3D IIR

Thanks!

Backup – Pure Rotation

Camera Projection

pixel location

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \sim \mathbf{KR}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{t}\right)$$

Camera Location

Camera Orientation

3D point coordinates

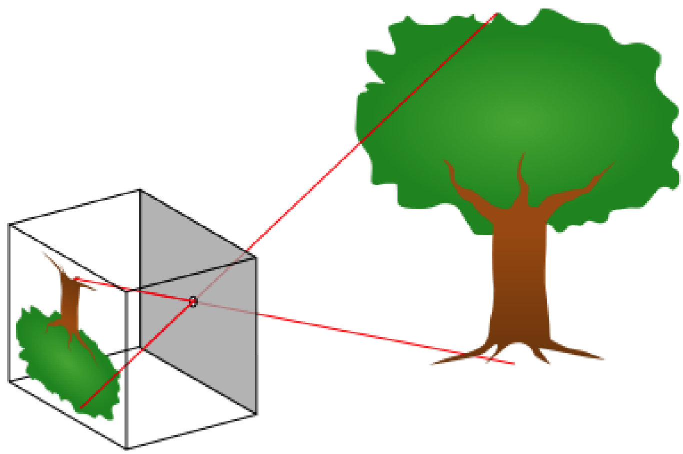
Relative Camera Orientation

pixel location in
(i+1) th frame

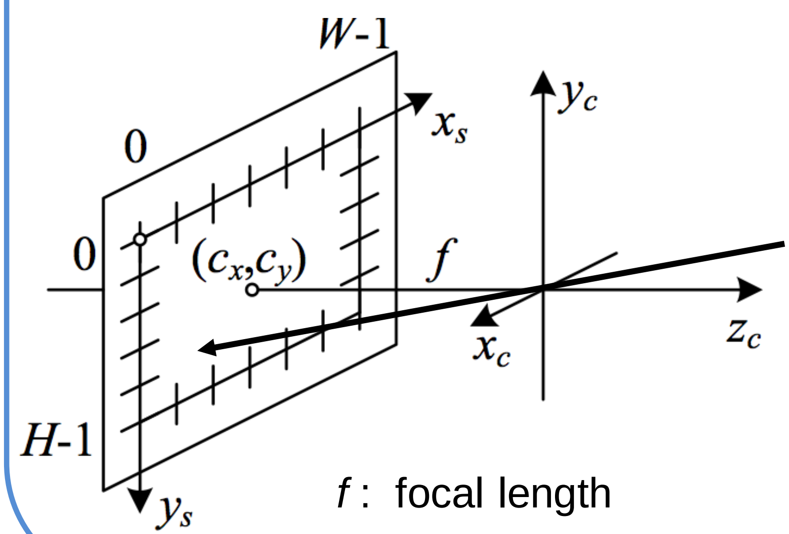
$$\begin{bmatrix} x_s^{i+1} \\ y_s^{i+1} \\ 1 \end{bmatrix} \sim \underbrace{\mathbf{K}\Delta\mathbf{R}\mathbf{K}^{-1}}_{\text{Projective Transformation}} \begin{bmatrix} x_s^{i} \\ y_s^{i} \\ 1 \end{bmatrix}$$

pixel location in
i th frame

Backup - Pinhole Camera Model



Camera Coordinate System



f : focal length

Camera Intrinsic Matrix K

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

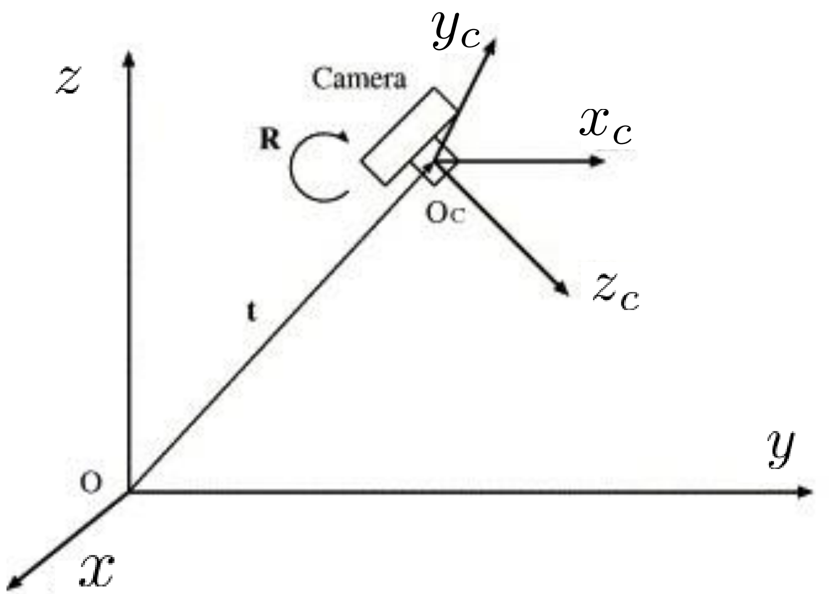
$$x_s = x_c \frac{f}{z_c} + c_x$$

$$y_s = y_c \frac{f}{z_c} + c_y$$

2D Coordinates in Image Plane

3D Coordinates in Camera Coordinate System

Backup - Camera Model & Camera Motion



World Coordinates → Camera Coordinates

$$\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \boxed{\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}} \Rightarrow \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \mathbf{R} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right)$$