Probabilistic 3D Motion Estimation for Rolling Shutter Video Rectification from Visual and Inertial Measurements

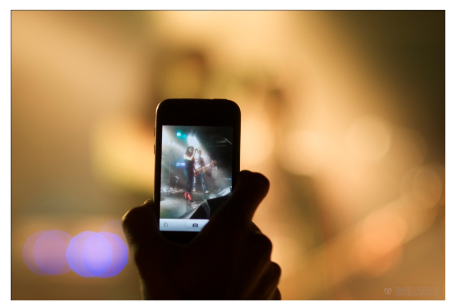
THE UNIVERSITY OF TEXAS AT AUSTIN

Wireless Networking & Communications Group

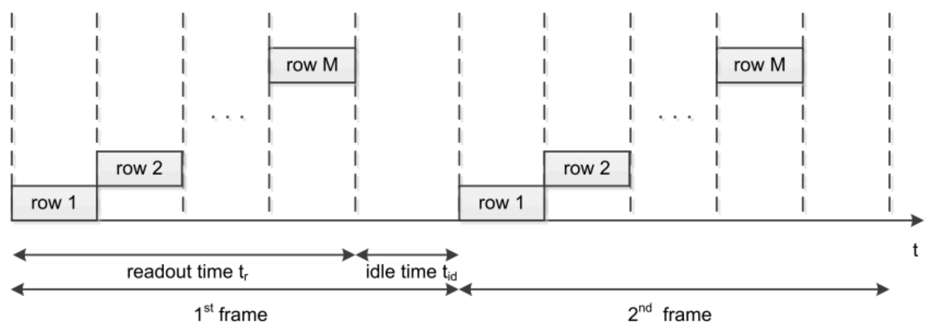
Chao Jia and Brian L. Evans

1. Problem

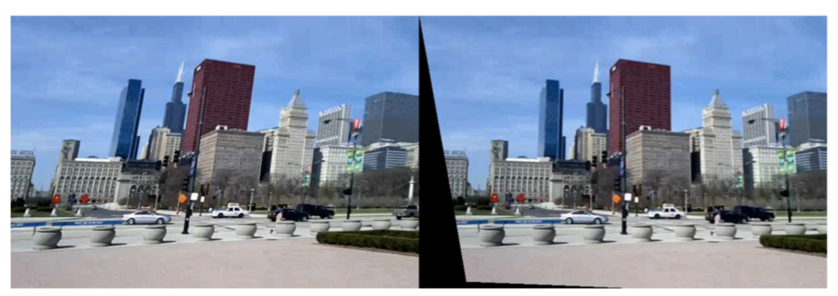
Handheld cameras
- Fast motion
- Camera shake



CMOS image sensors
- Rows in sensor array are exposed sequentially from top to bottom



Rolling shutter effects



2. Rolling shutter rectification

General steps:

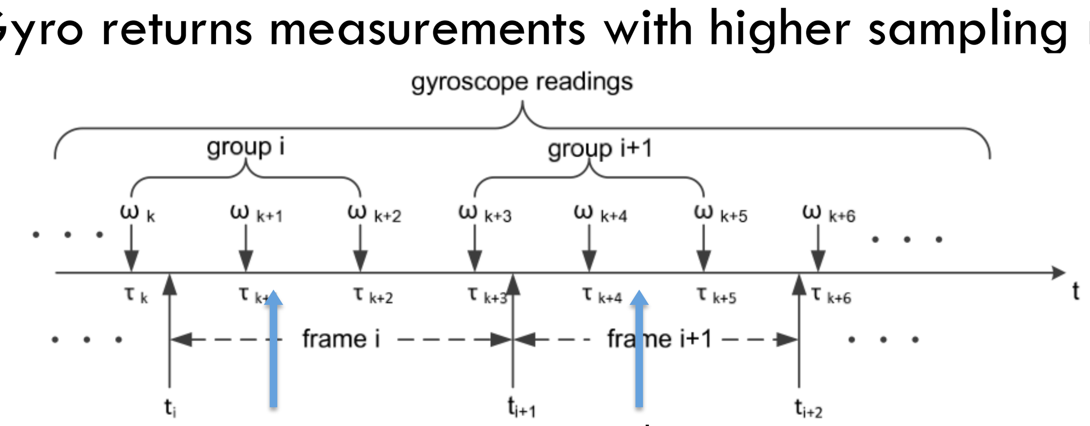
Camera motion estimation (for each row) → Keep unique pose for each frame → Warp each row

Challenges and solutions
- Image warping accuracy → Pure rotational model
- High camera pose resolution → Gyroscope

Gyroscopes in mobile phones *[Karpenko et al., 2011]*
- High sampling rate (> 100Hz)
- Samples have unknown bias and noise
- SLERP interpolation to align with row exposure time

Combine gyro with other information to improve accuracy
- Accelerometer *[Hanning et al., 2011]*
- **Our method: visual measurements (tracked features)**

Gyro and feature point correspondence
- Gyro returns measurements with higher sampling rate



- Compute the relative rotation between two exposure time

$$\mathbf{R}(t(\mathbf{u}', i+1))\mathbf{R}^T(t(\mathbf{u}, i)) = \prod_{n=k+1}^{k+4} \Delta\mathbf{R}(\omega_n \Delta t_n)$$

$$\Delta\mathbf{R}(\omega_n \Delta t_n) = \exp(\text{skew}(\omega_n)\Delta t_n)$$

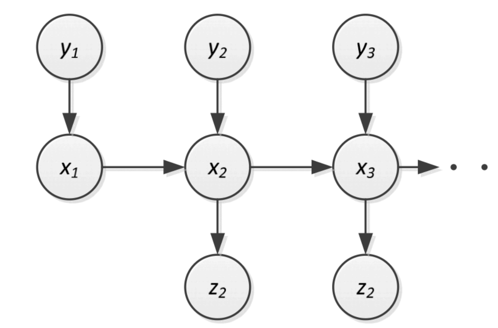
3. Combining Gyro and Visual Data

Angular velocity estimation based on extended Kalman filtering

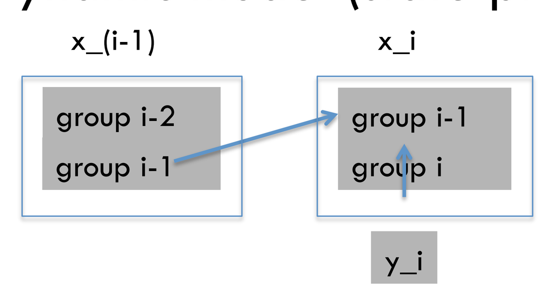
State vector: two groups of angular velocities

$$\mathbf{x}(i) = [\omega(i,1), \ldots, \omega(i,N_i), \omega(i+1,1), \ldots, \omega(i+1, N_{i+1})]^T$$

Probabilistic graphic model



Dynamic model (state prediction)



$$\mathbf{x}_i = \begin{bmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,2} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{i-1,2} \\ \mathbf{y}_i \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{w}_i \end{bmatrix}$$

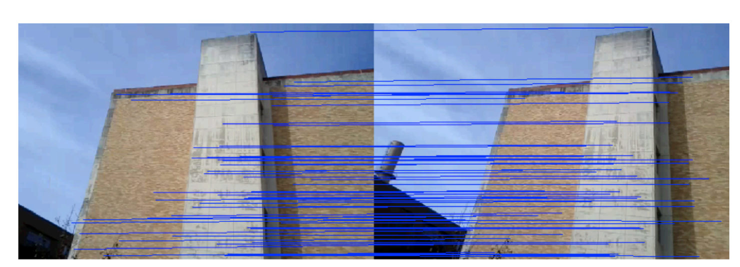
Measurement model (state update)
- Use only feature points in current frame as the measurements
- Matching points in previous frame used as known parameters

$$\mathbf{u}_{i,j} = g\left(K \Delta R K^{-1} \begin{bmatrix} \mathbf{u}_{i-1,j} + \mathbf{v}_{i,j,1} \\ 1 \end{bmatrix}\right) + \mathbf{v}_{i,j,2}$$

state vector determines the relative rotation

$$\prod_{k=1}^{N_{i-1}} \Delta R(\omega(i-1,k)\Delta t_{i-1,j,k}) \prod_{k=1}^{N_i} \Delta R(\omega(i,k)\Delta t_{i,j,k})$$

- Final measurement equation



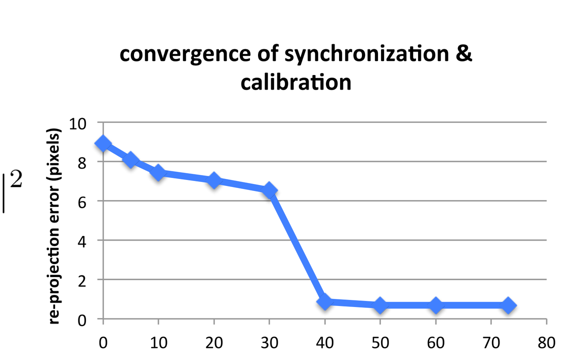
$$\mathbf{z}_i = \begin{bmatrix} \mathbf{u}_{i,1} \\ \mathbf{u}_{i,2} \\ \vdots \\ \mathbf{u}_{i,M} \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_i, \mathbf{u}_{i-1,1}, \mathbf{v}_{i,1}) \\ h_2(\mathbf{x}_i, \mathbf{u}_{i-1,2}, \mathbf{v}_{i,2}) \\ \vdots \\ h_M(\mathbf{x}_i, \mathbf{u}_{i-1,M}, \mathbf{v}_{i,M}) \end{bmatrix}$$

4. Sensor Synchronization & Calibration

Parameters
- Rolling shutter speed (actual exposure time)
- Camera intrinsic parameters
- Delay between timestamps of gyro and video

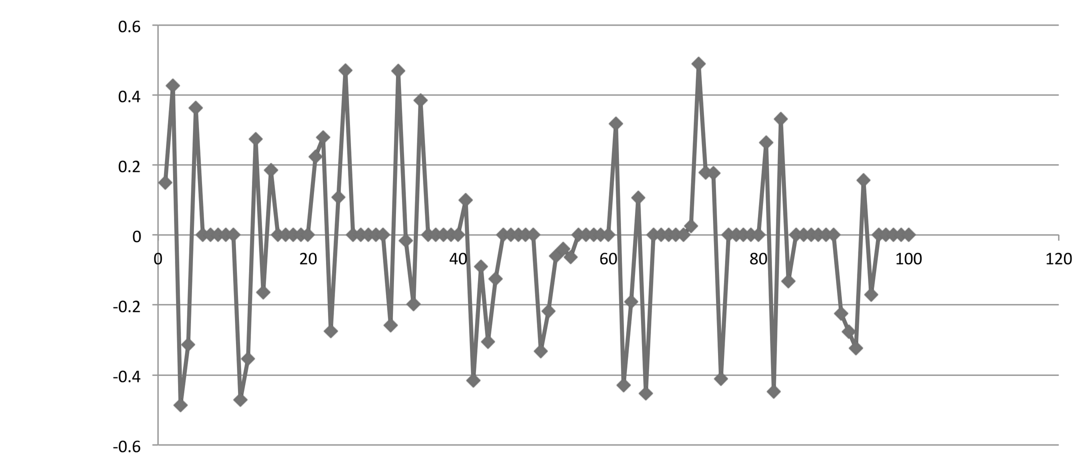
Batch optimization *[Karpenko et al., 2011]*
- Initialize camera intrinsic parameters by self-calibration
- Get relative rotation from gyro readings
- Minimize average re-projection error over all matching points
- Solve by Levenberg-Marquardt algorithm

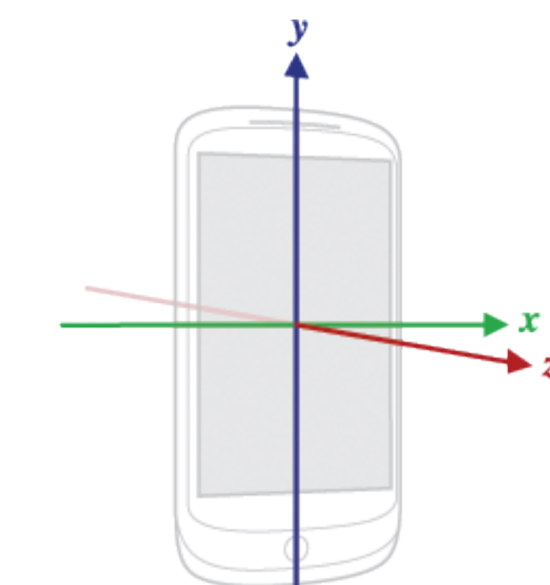
$$J = \sum_i \sum_{\mathbf{u},\mathbf{v}} \|\mathbf{u} - \mathbf{K}\mathbf{R}(t(\mathbf{u}, i+1))\mathbf{R}^T(t(\mathbf{v}, i))\mathbf{K}^{-1}\mathbf{v}\|^2$$



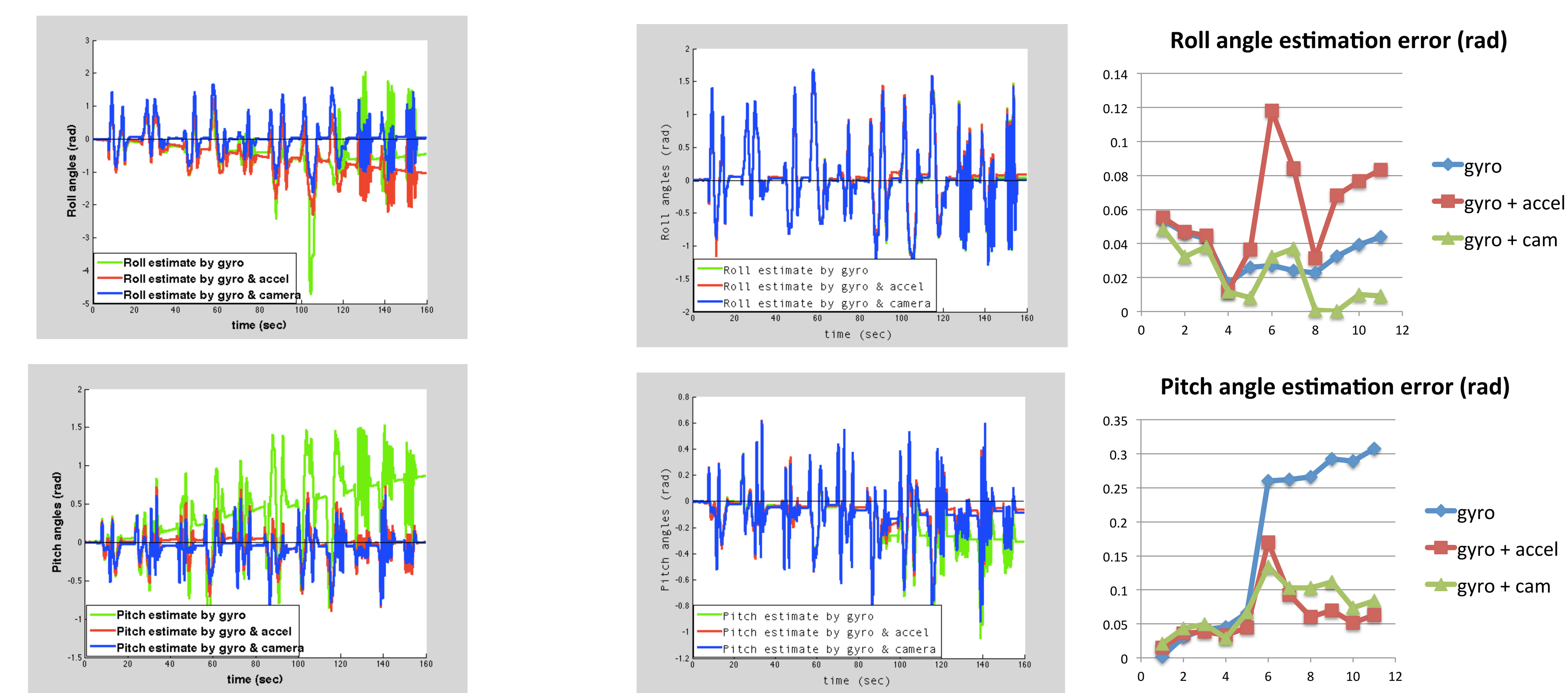
5. Experimental Results

Zero-angle test
- Start with cell phone on a flat surface
- Rotate cellphone at will, then put it back on the surface, stay still for several seconds
- Repeat ten times
- Ground truth available naturally for pitch and roll

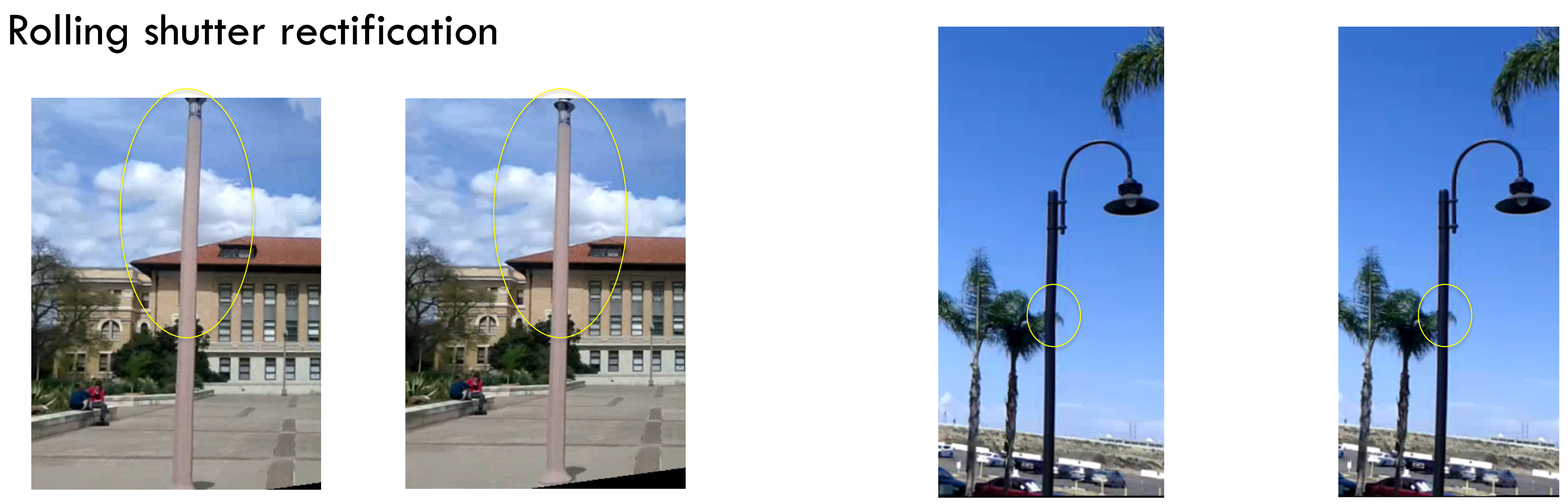




- Rotation estimation accuracy using raw gyro readings (with bias)
- Rotation estimation accuracy using unbiased gyro readings



Rolling shutter rectification



gyro | gyro + video data | gyro | gyro + video data

Numerical comparison
- No ground truth → no-reference method
- Vanishing point check
- Lines detected manually
- Find vanishing point by least-square