# Light-Weight Novel View Synthesis for Casual Multiview Photography

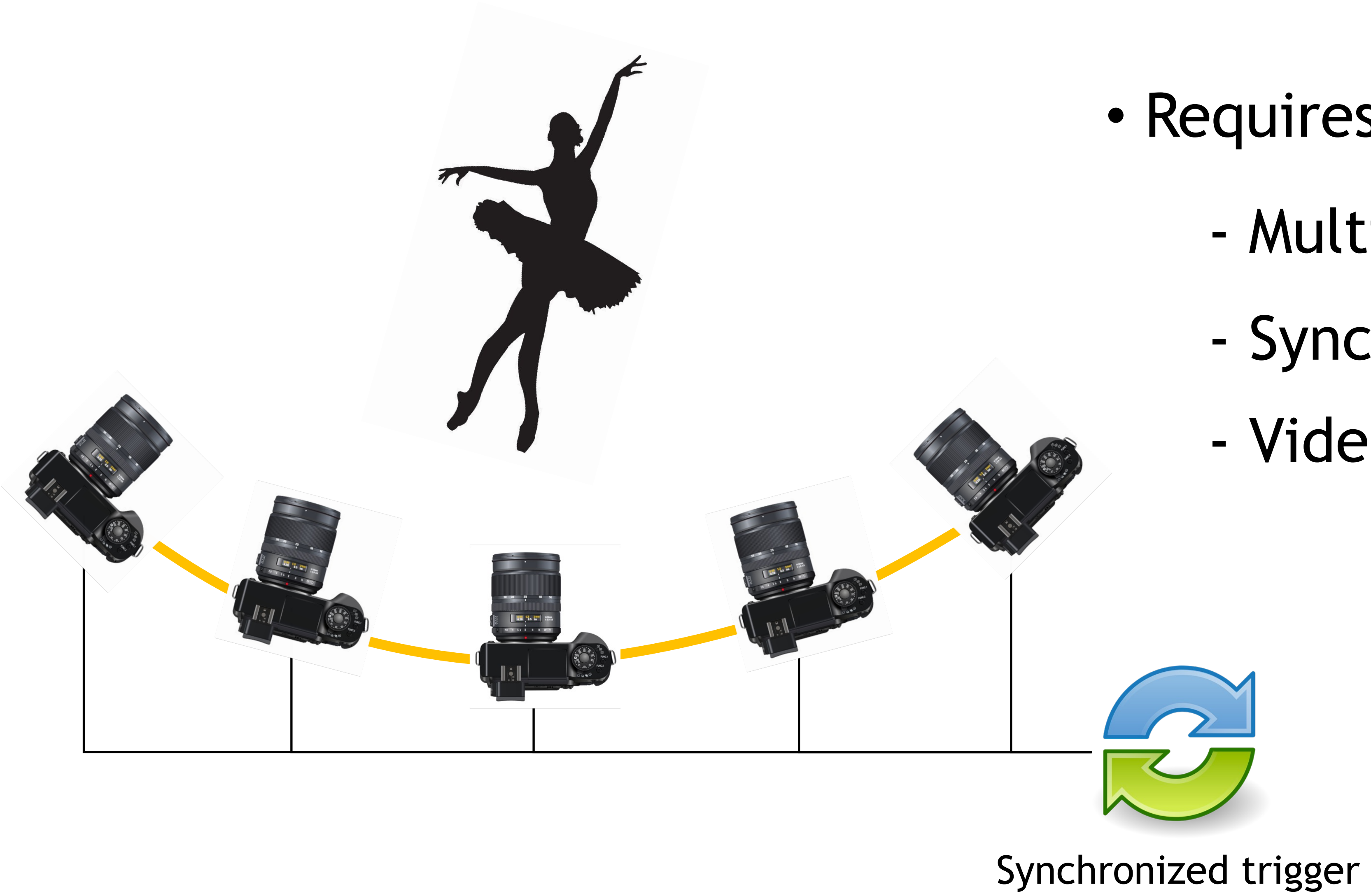
Inchang Choi, Yeong Beum Lee, ***Dae R. Jeong***,
Insik Shin, and Min H. Kim

KAIST

# Bullet Time Effect

# Bullet Time Effect

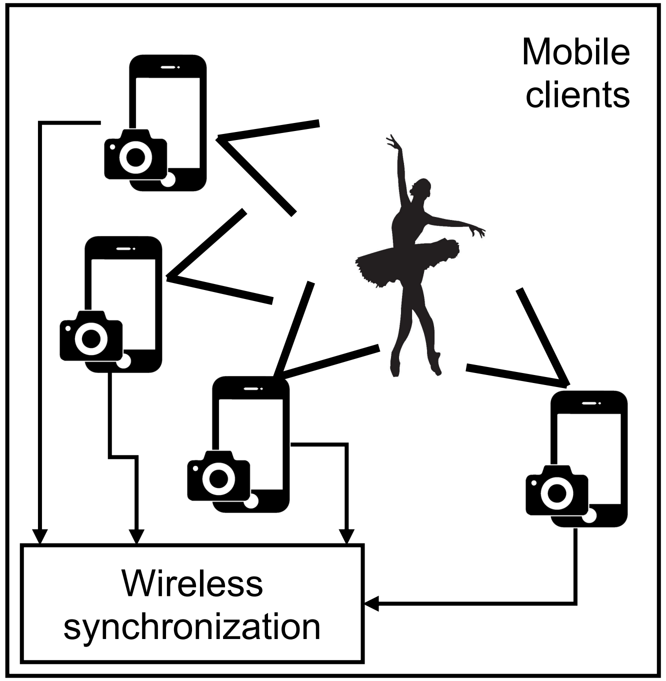





- Requires professional devices
  - Multiple cameras
  - Synchronization hardware
  - Video editing software

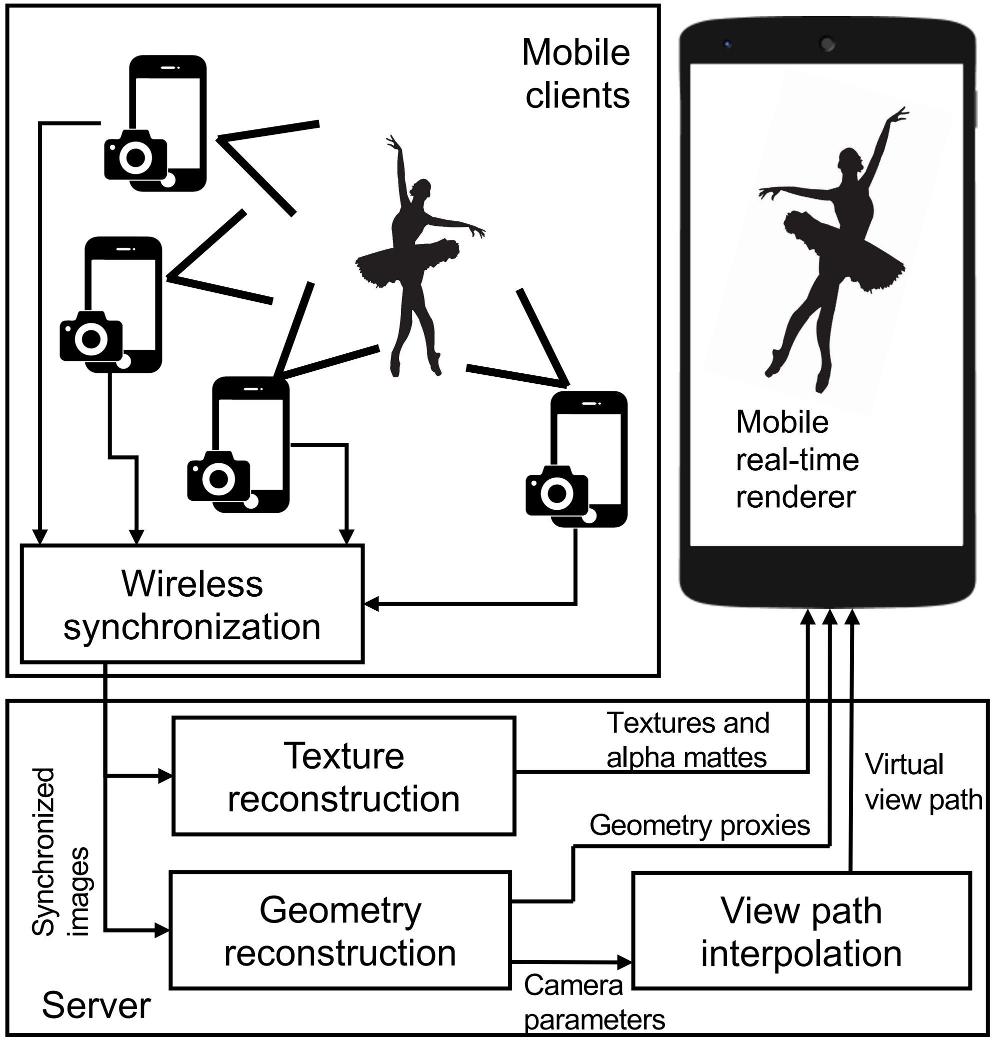
# Our Light-weight Novel View Synthesis






# Our Light-weight Novel View Synthesis






# Related Work

- Light-Field Rendering
- View-Dependent Texture Mapping
- Depth Image based Rendering

# Light-Field Rendering

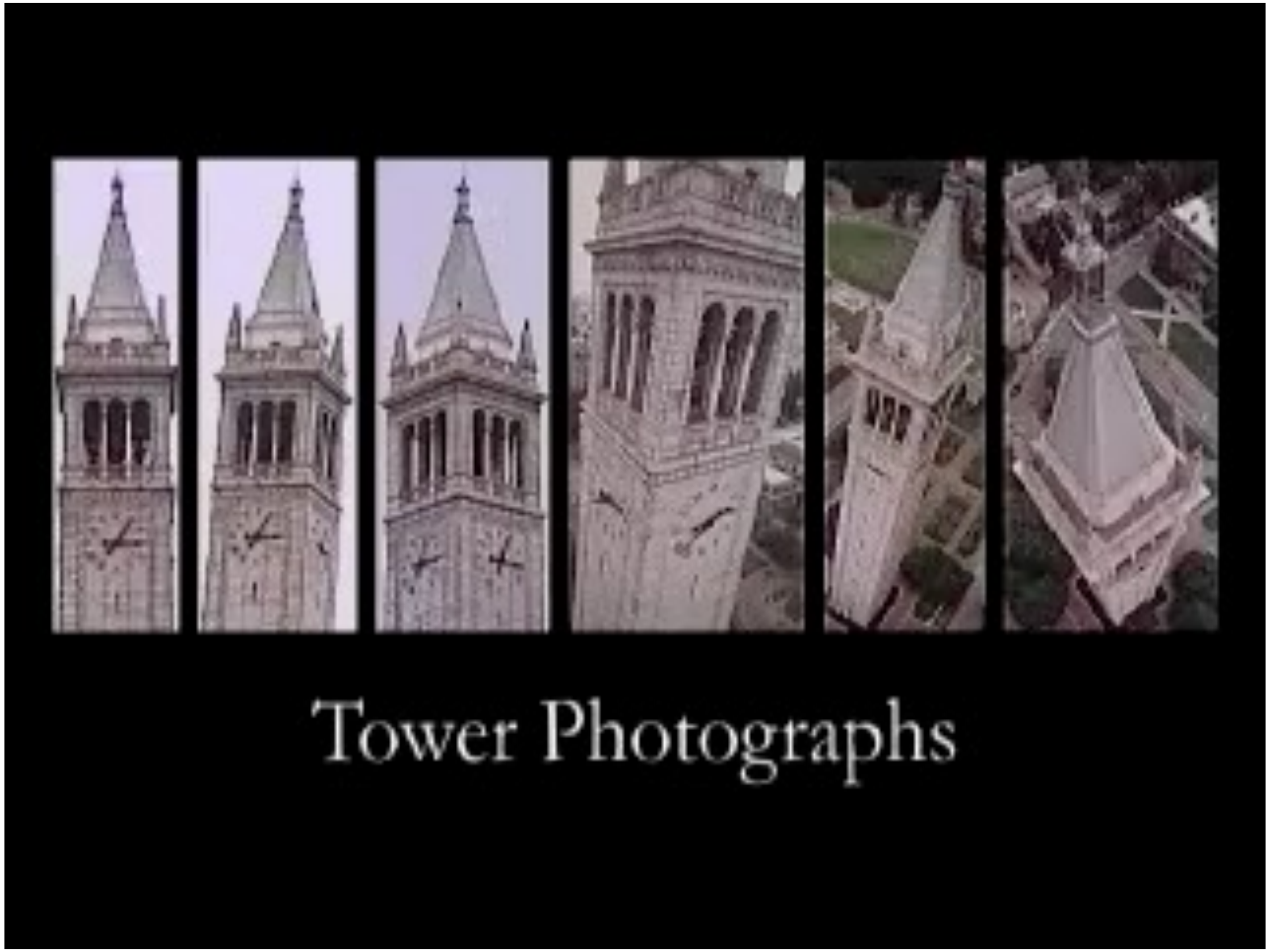

[Buehler et al. 2001]

Light-Field Rendering
- [Seitz et al. 1996]
- [Buehler et al. 2001]

***Requires***
- A large number of images
- Rough geometry

# View-Dependent Texture Mapping





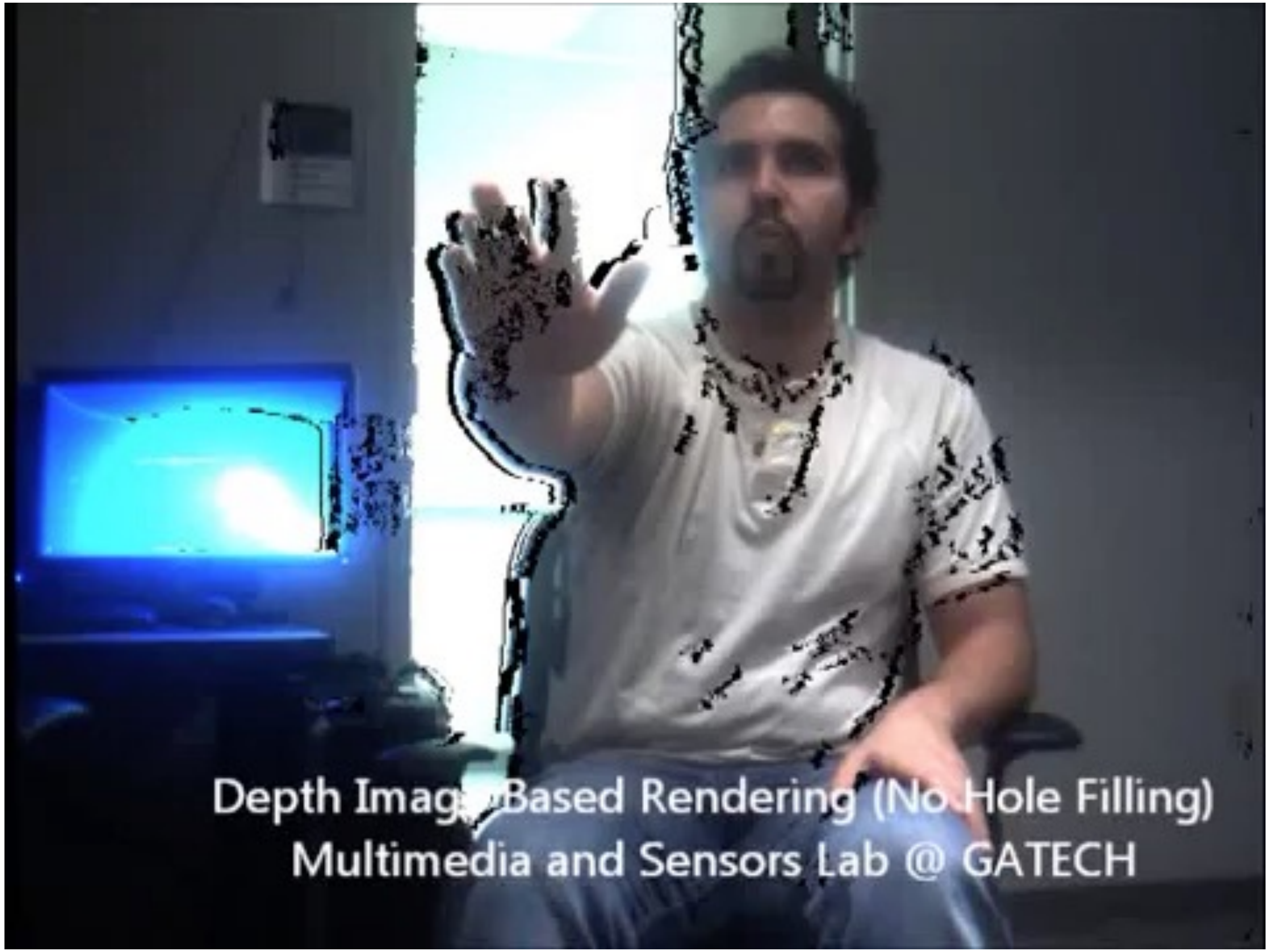

[Debevec et al. 1998]

View-Dependent Texture Mapping
- [Debevec et al. 1998]
- [Siu et al. 2004]
- [Sinha et al. 2009]

*Requires*
- designed
  or 3D-scanned geometry

# Depth Image-based Rendering






[Solh and Alregib, 2010]

Depth Image-based Rendering

- [Shade et al. 1998]
- [Chang et al. 1999]
- [Solh and Alregib 2010]

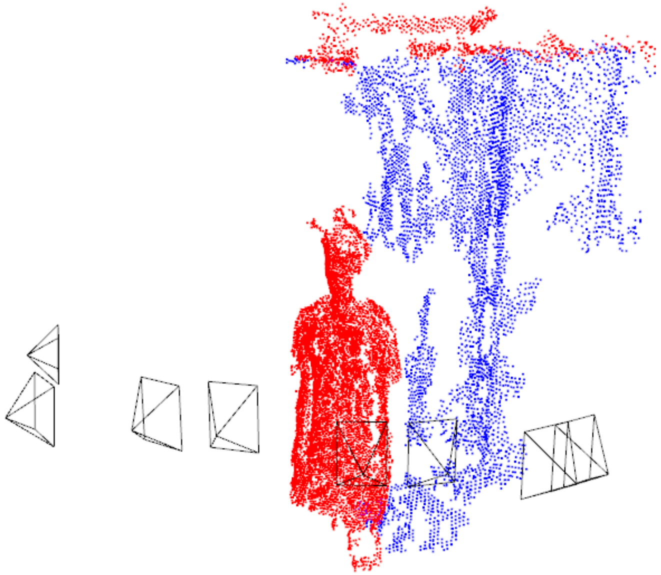
*Disocclusion problem*

- constrains a new perspective to be close to the originals
- requires hole filling algorithm

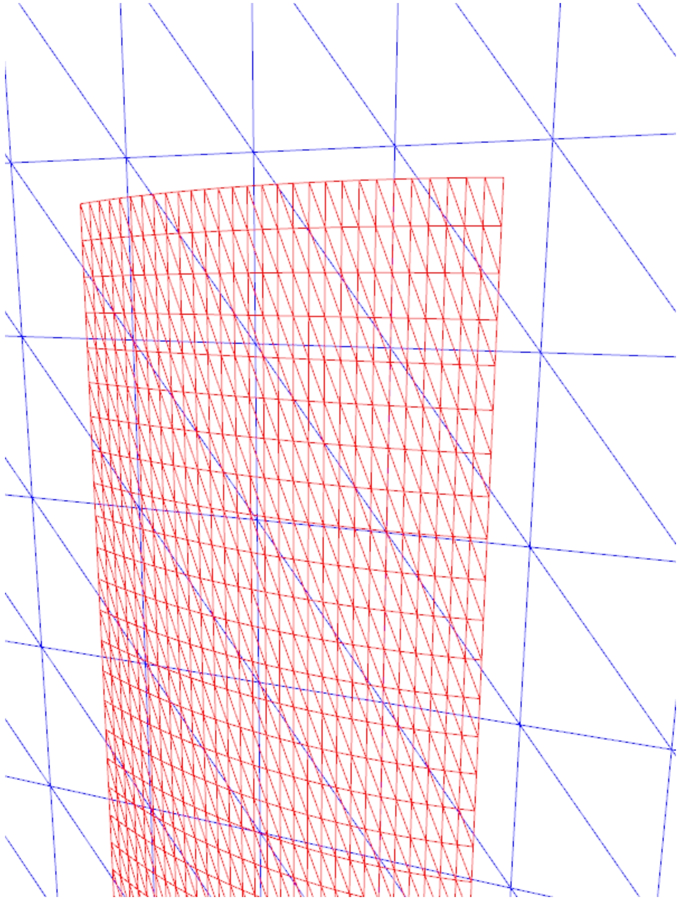
# Our Method



Input images



Point cloud



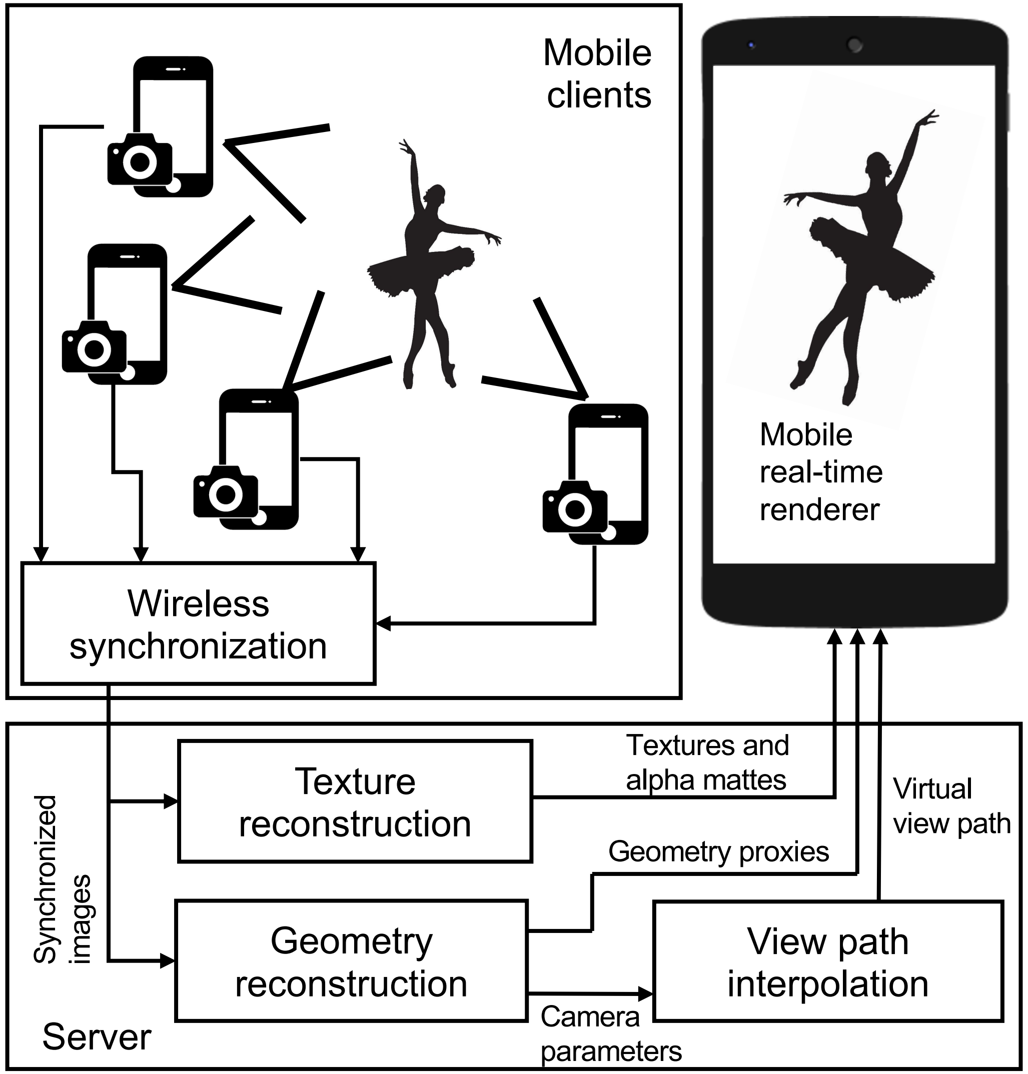
Geometry proxy

# Our Method

## Light-Weight Novel View Synthesis

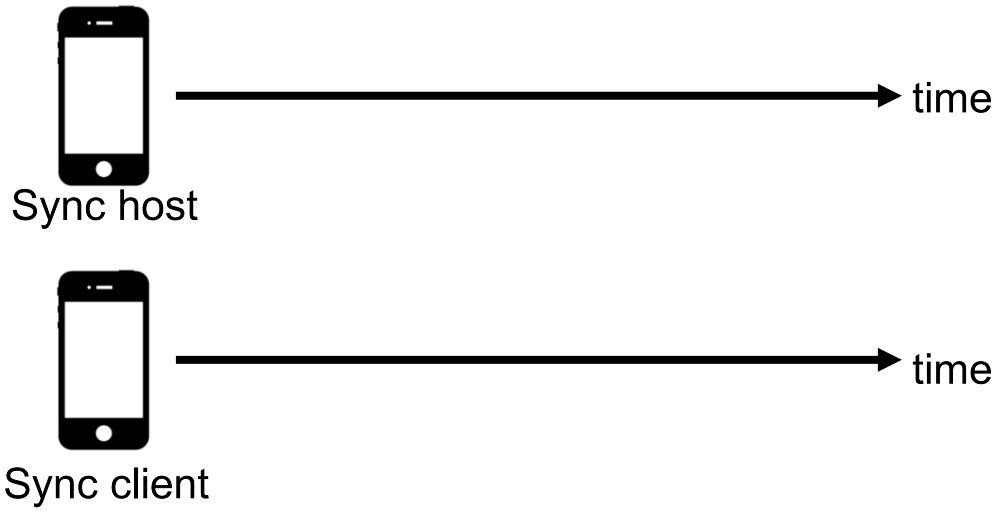
# Wireless Synchronization





# Wireless Synchronization

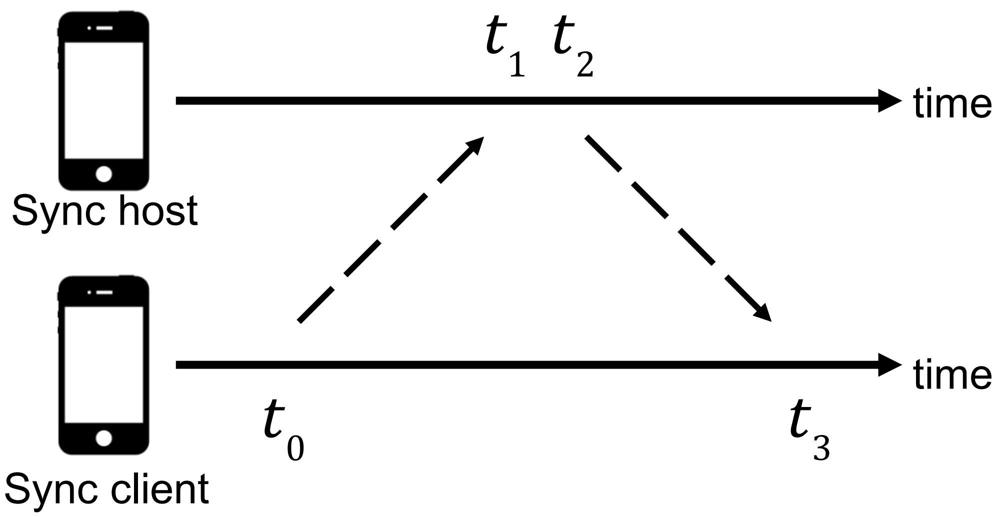





- One device roles a host, the others works as a client
- Use Network Time Protocol (NTP) to synchronize the clock of each device

# Wireless Synchronization





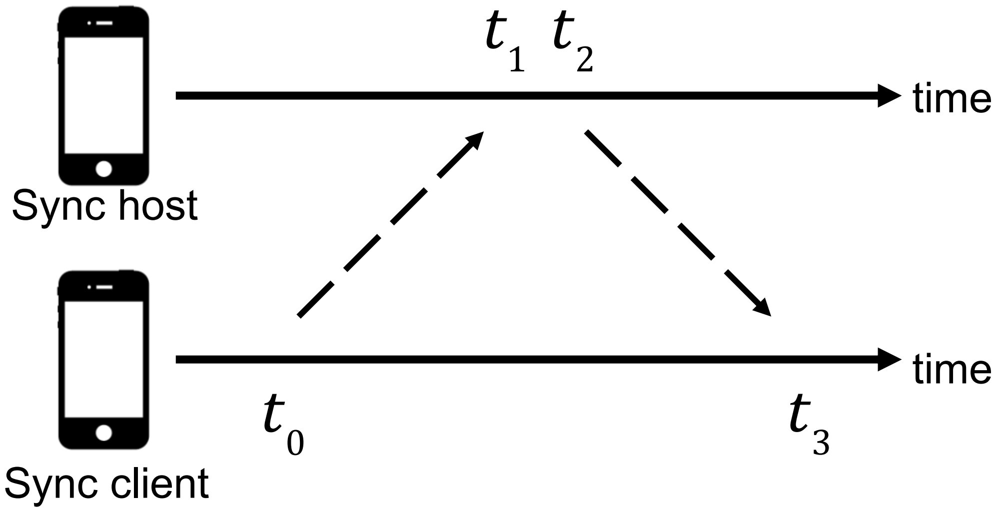

## The procedure for synchronization

1. A client sends a packet with $t_0$
2. The server receives the packet and write $t_1$
3. The server sends back the packet with $t_2$
4. The client receives the packet and write $t_3$
5. Compute the time difference $t_d$ and add it to the client's clock

$$t_d = \frac{(t_1 - t_0) + (t_2 - t_3)}{2}$$

# Wireless Synchronization






## At the capture time

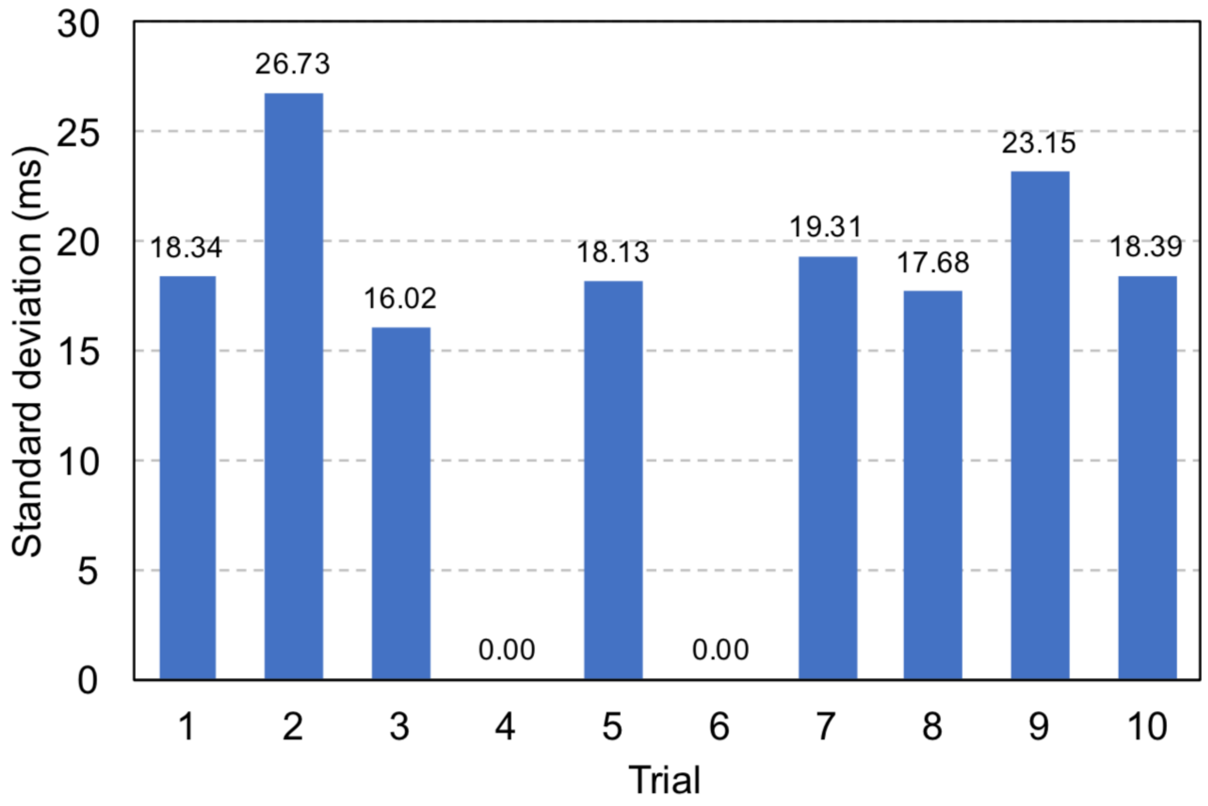
1. The server triggers the capture
2. The clients capture images as soon as the trigger packet is received

*Leave a short delay for the packet trip time*

$$t_r = \frac{(t_3 - t_0) - (t_2 - t_1)}{2} + 100 \text{ ms}$$

# Wireless Synchronization

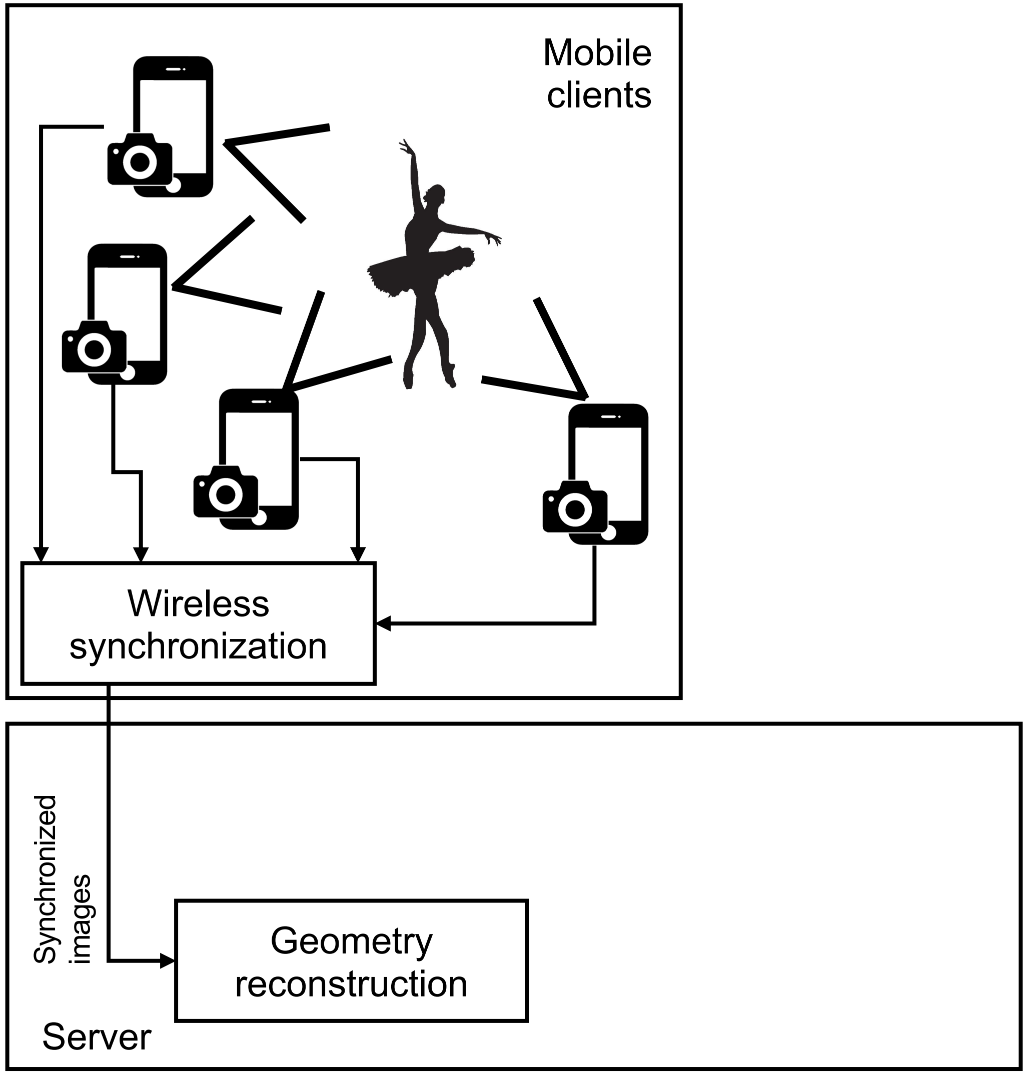





The average std. = 15.77 ms

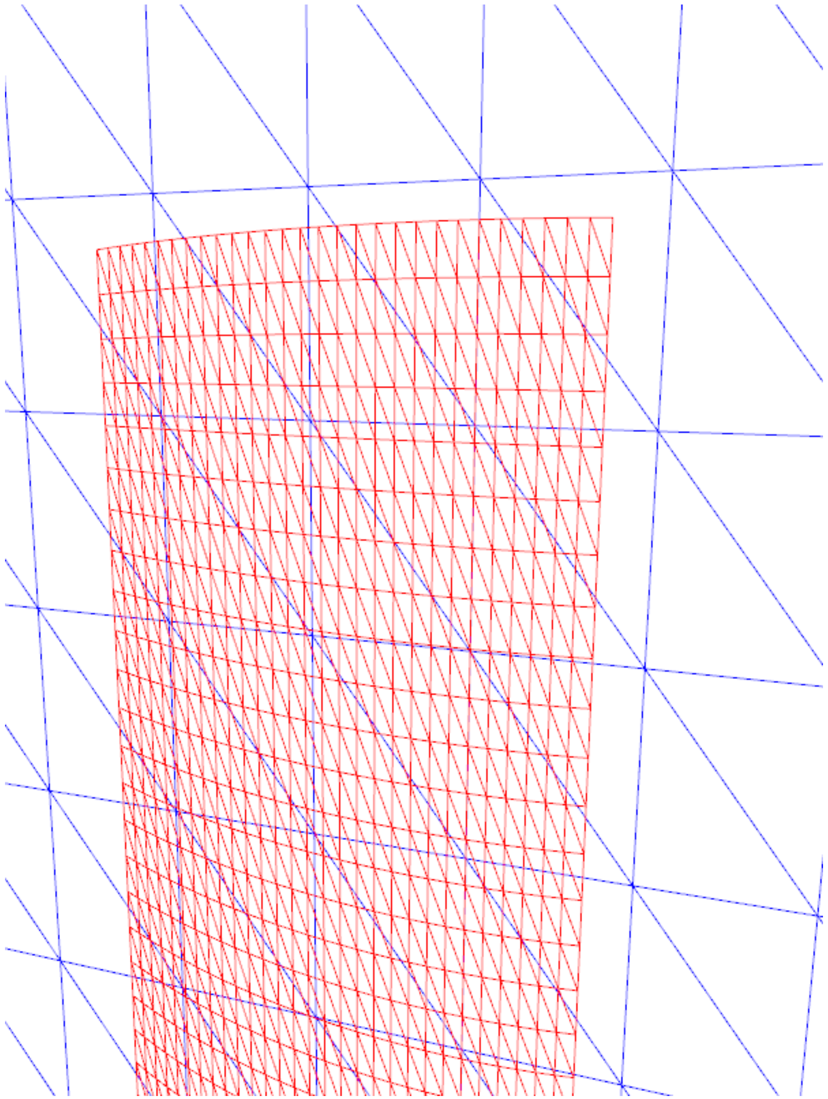
# Geometry Reconstruction

ISVC 2019
Lake Tahoe, Nevada, USA



# Geometry Reconstruction





Given synchronized images

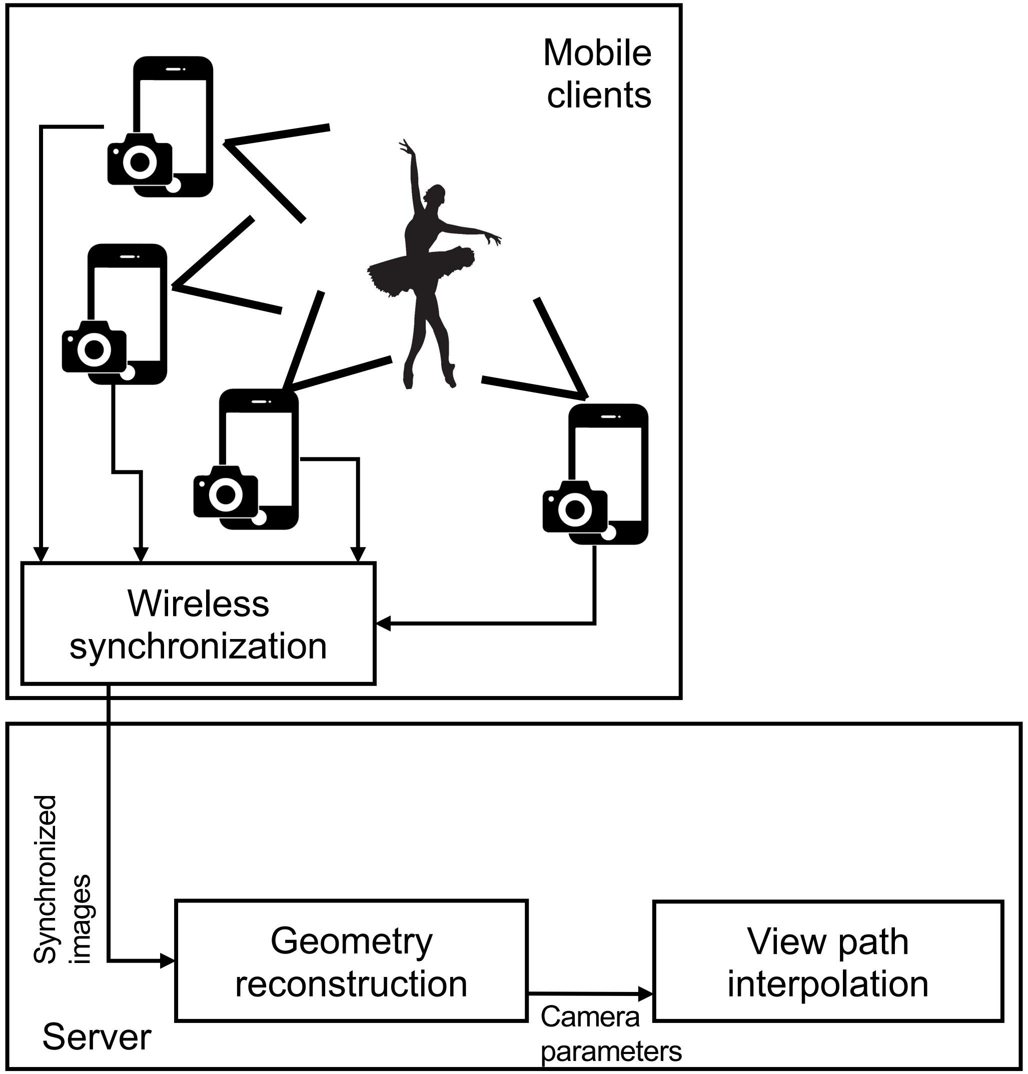
1. Run multiview stereo algorithm
   - [Wu et al. 2011] – Visual SFM

2. Cluster the foreground and the background point clouds
   - K-means clustering algorithm with K=2
   - (x, y, z) coordinates as features

3. Generate geometry proxies for the foreground and the background
   - A cylinder for the foreground
   - A plane fore the background
   - The cluster centers as the centers for proxy

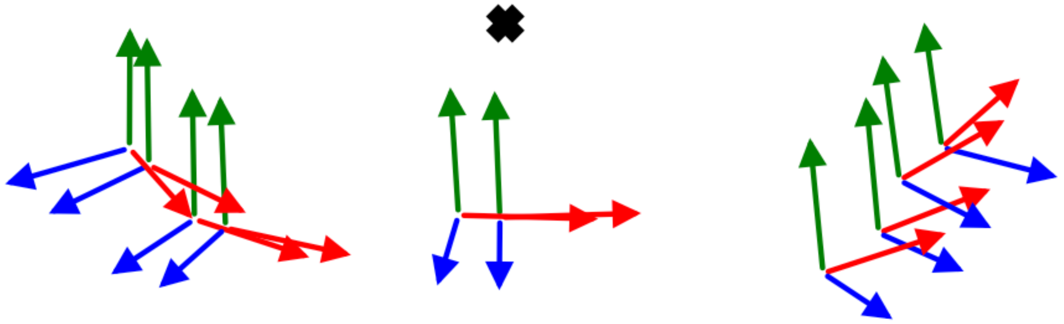
# View Path Interpolation

ISVC 2019
Lake Tahoe, Nevada, USA

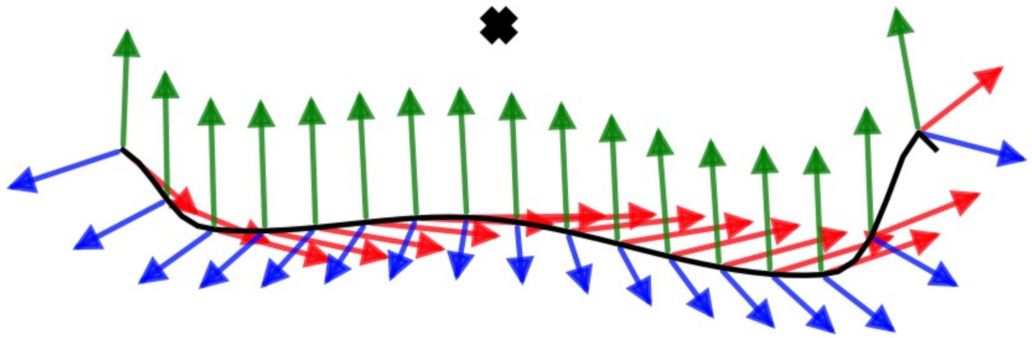


# View Path Interpolation





The original cameras from the multiview stereo



Interpolated virtual cameras

**The positions of virtual cameras:**

1. Fit a spline function

   for the x-y dimension of the real cameras

   - Use a 3rd order polynomial function

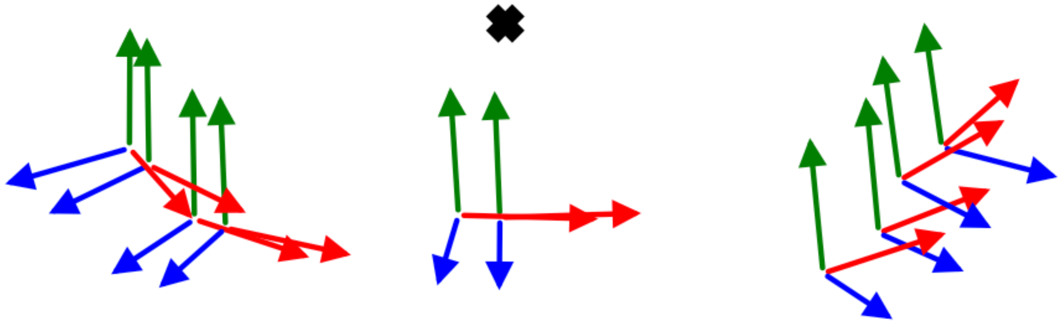
2. Sample new points uniformly along x direction

3. Generate y-coordinates using the spline function

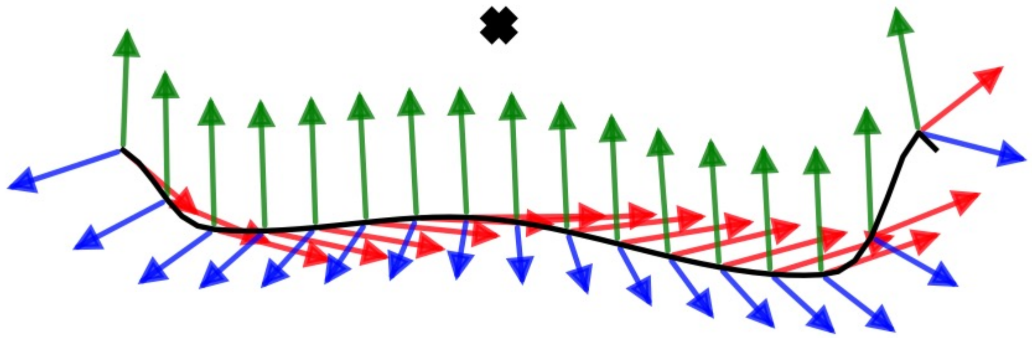
*Repeat this steps for x-z plane*

# View Path Interpolation





The original cameras from the multiview stereo



Interpolated virtual cameras

**The poses of virtual cameras:**

1. Compute the weighted average of the rotation matrices of the real cameras

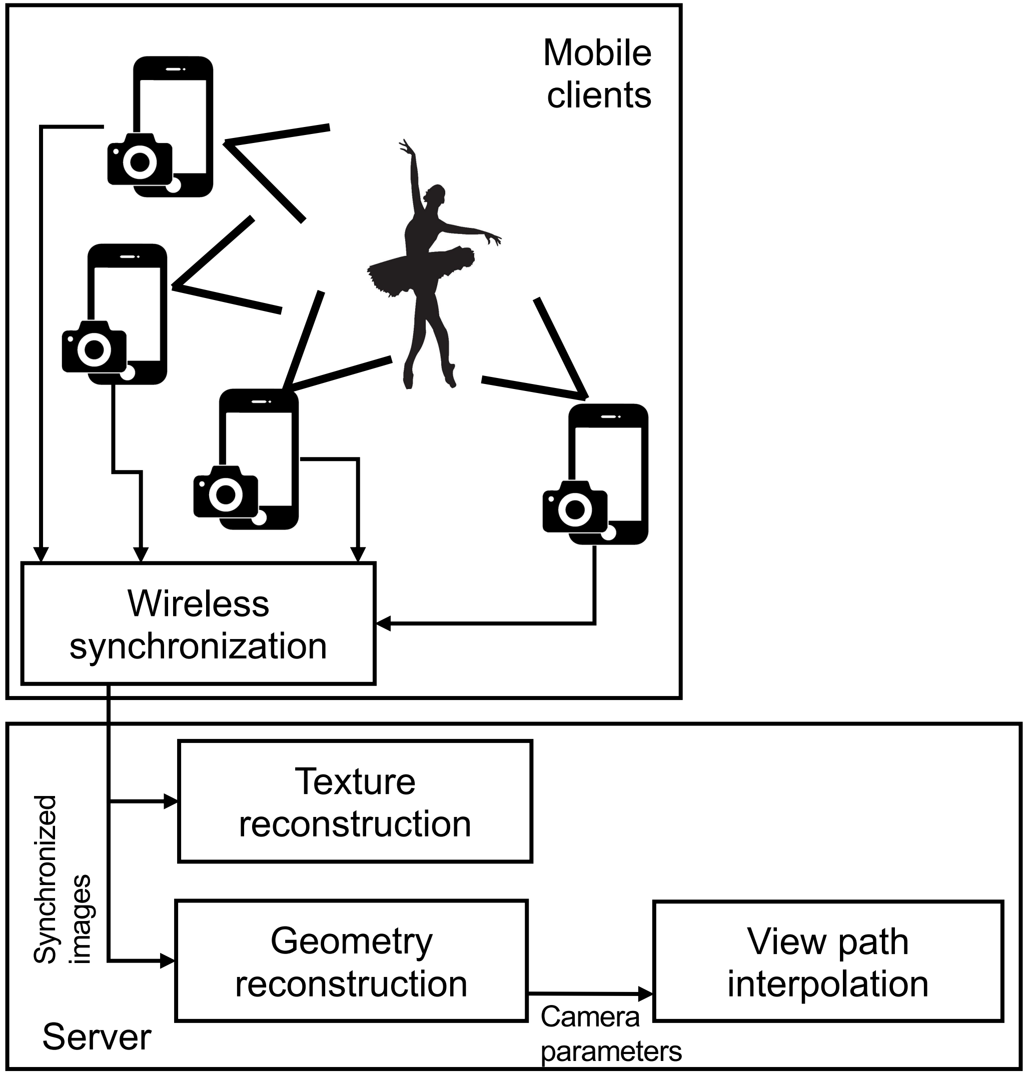
$$R_{V_k} = \frac{\sum_{i=1}^{n} w_{ik} R_{C_i}}{\sum_{i=1}^{n} w_{ik}}$$

2. Use a Gaussian function as the weights

$$w_{ik} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\| T_{V_k} - T_{C_i} \|^2}{2\sigma^2}}$$

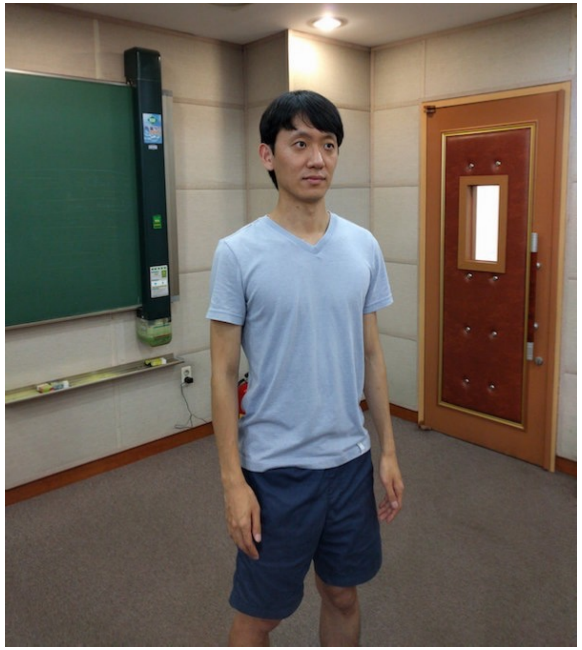
# Texture Reconstruction

ISVC 2019
Lake Tahoe, Nevada, USA



# Texture Reconstruction





One of the captured images

The segmentation map

**Foreground Segmentation**

- Run a CNN-based semantic segmentation [Zheng et al. 2015]
- Assume people as foreground objects
- Use the segmentation map as alpha mattes in the later rendering step

# Texture Reconstruction



Novel view
without inpainting

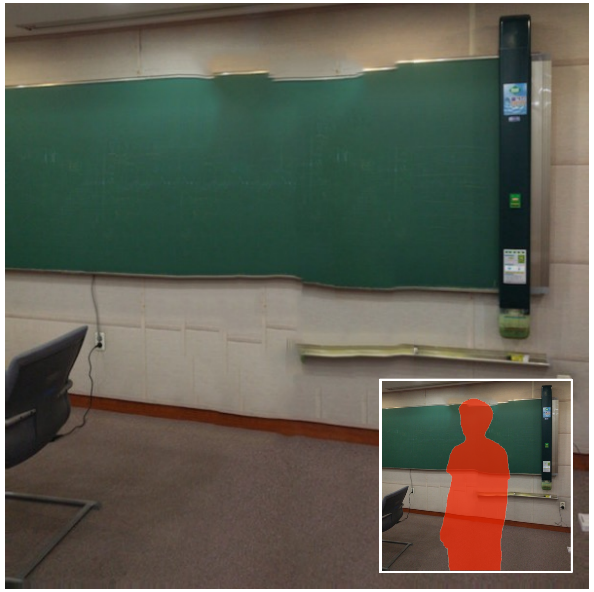
Novel view
with inpainting

**Background Inpainting**

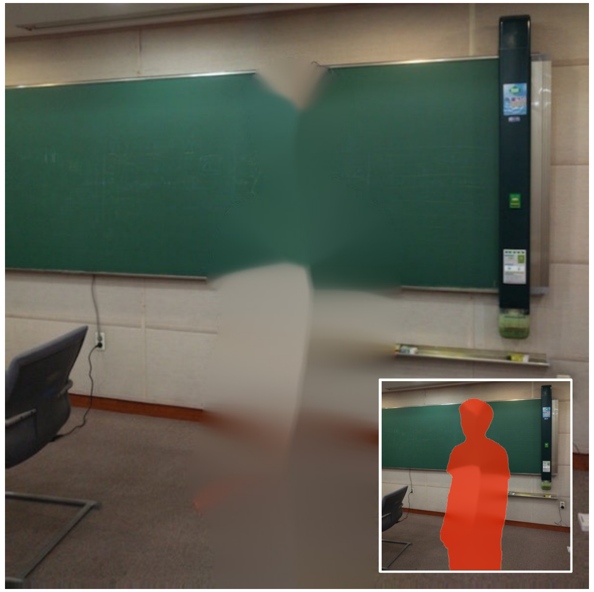
- Use fast marching inpainting algorithm for its efficiency [Telea 2004]

# Texture Reconstruction





Laplacian Inpainting



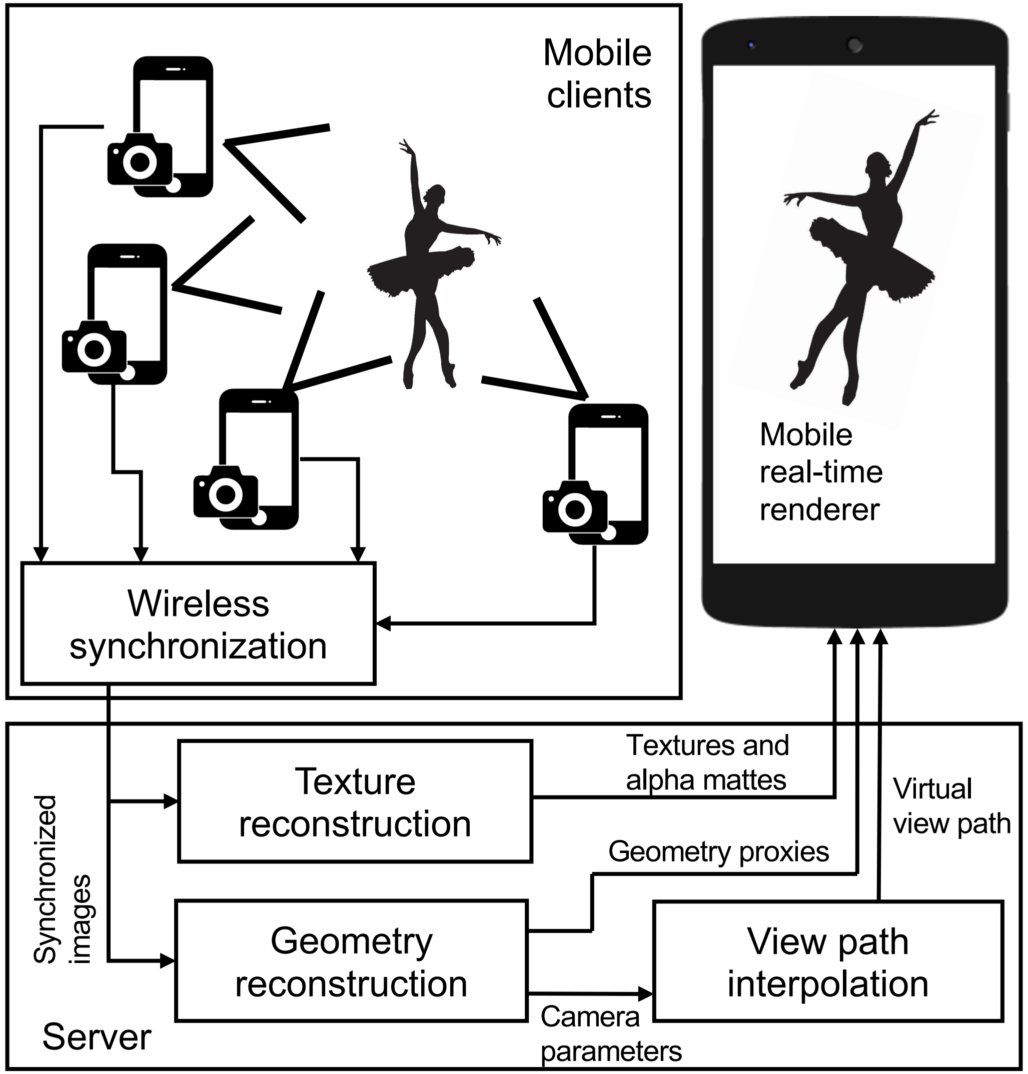
Fast marching inpainting

**Background Inpainting**

- Use fast marching inpainting algorithm for its efficiency [Telea 2004]
- Other methods can be more promising for the quality.
  i.e. Laplacian inpainting [Lee et al. 2016]

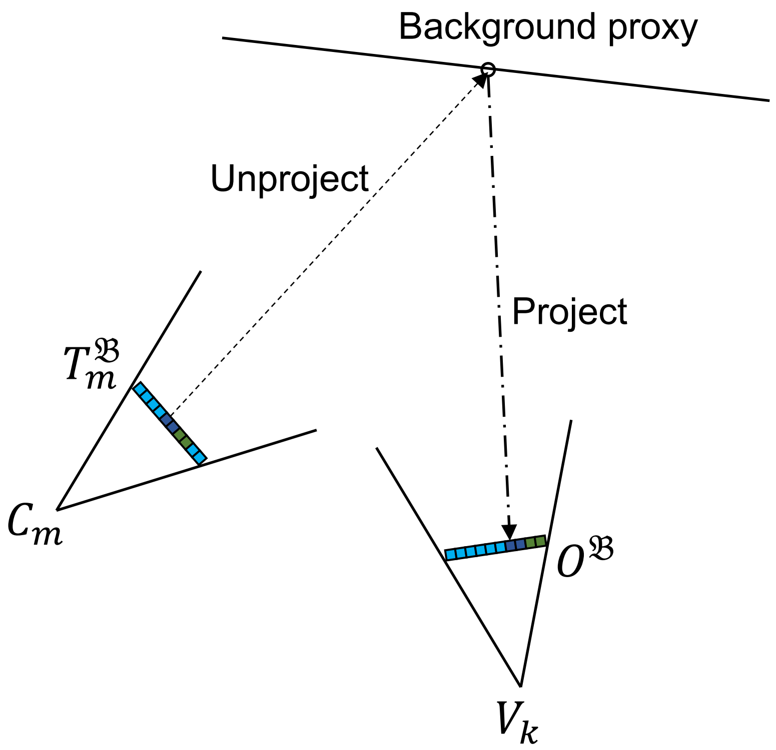
# Real-Time Rendering





# Two-Pass Rendering






## Pass I: Background Rendering

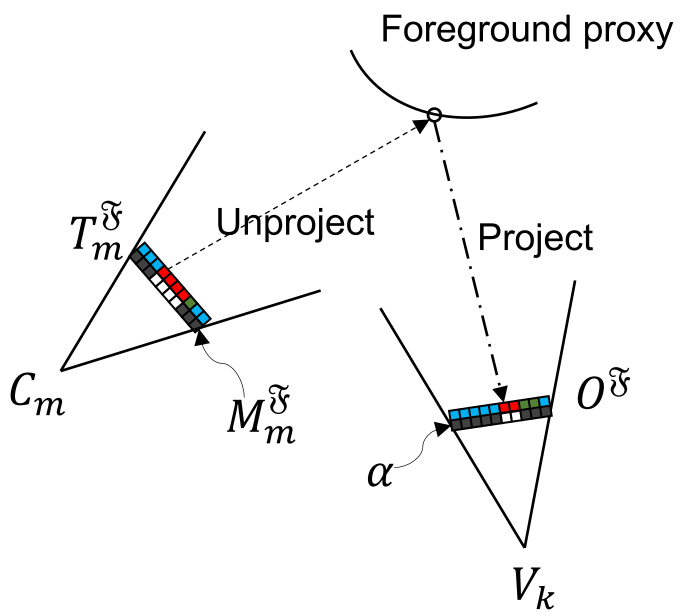
1. Select the closest real camera

2. Unproject the inpainted background texture on the background proxy

3. Project the textured background proxy to the novel camera

# Two-Pass Rendering






## Pass II: Foreground Rendering

1. Select the closest real camera

2. Unproject the image and ***the segmentation map*** of the real camera on the foreground proxy

3. Project the textured background proxy to the novel camera

***Alpha-blend the output of Pass I and Pass II***

# Results

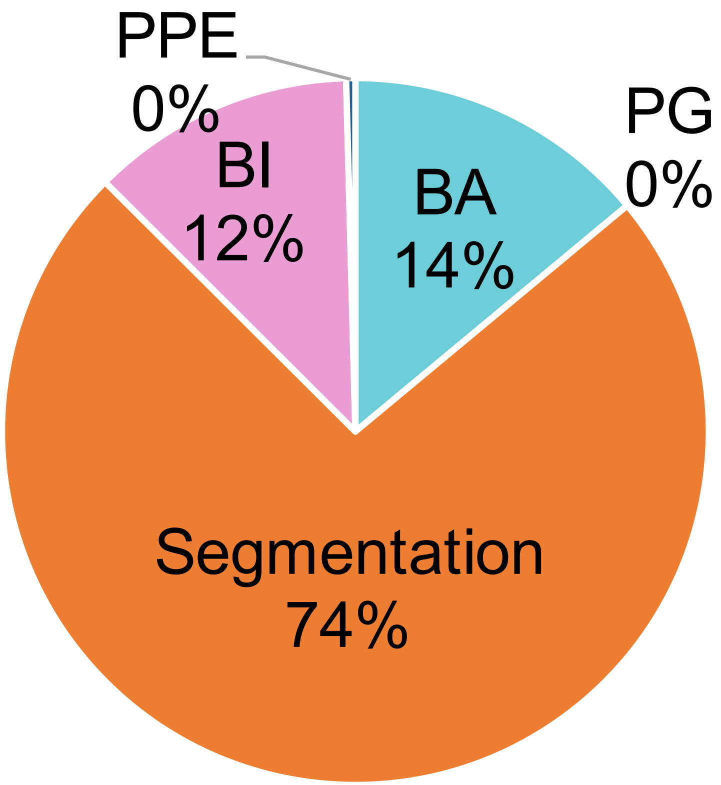
# Synchoronous Input Images

# Discussion & Conclusion

# Running Time Analysis



| Task | | Time (s) |
|---|---|---|
| Geometry processor | Bundle Adjustment (BA) | 11.00 |
| | Proxy Generation (PG) | 0.0020 |
| Texture processor | Segmentation | **57.92** |
| | Background Inpainting (BI) | 9.54 |
| View path generator | Point and Pose Estimation (PPE) | 0.33 |
| Total | | 1min. 18secs. (=78.79 secs) |



# Limitation



- Discontinuous transition when the source inputs change caused by simplified proxies

- Artifacts on the boundaries caused by wrong segmentation

# Conclusion



- Proposed a light-weight novel view synthesis method
- Enabled bullet-time effect only with mobile cameras and efficient fully automatic novel view synthesis algorithm
- Bridged the gap between mobile computing and multiview photography

# Thank You!

KAIST