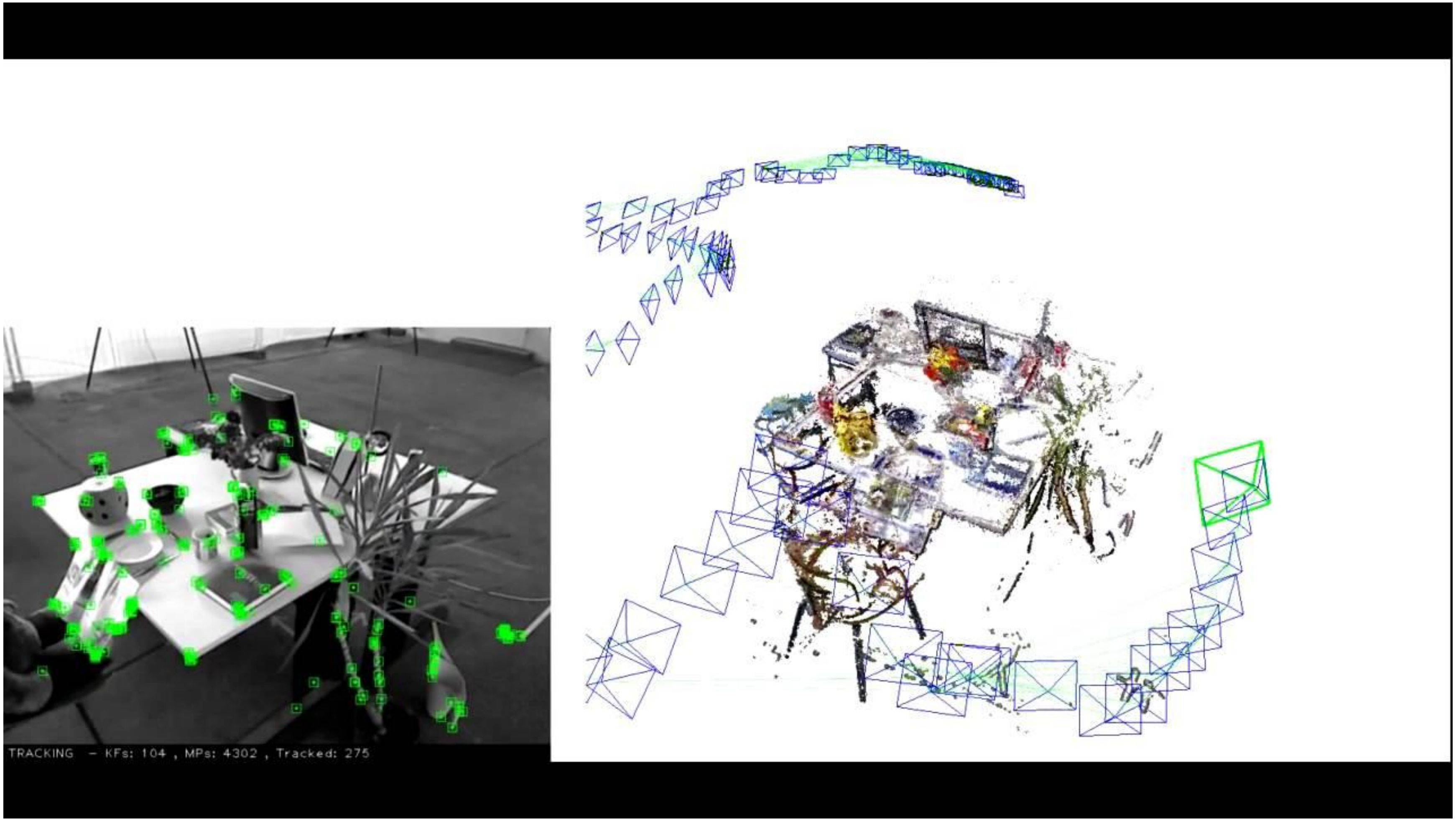
COMPUTER VISION PROJECT SLAM/SFM

Ido Ben Shachar
Ehud Gordon

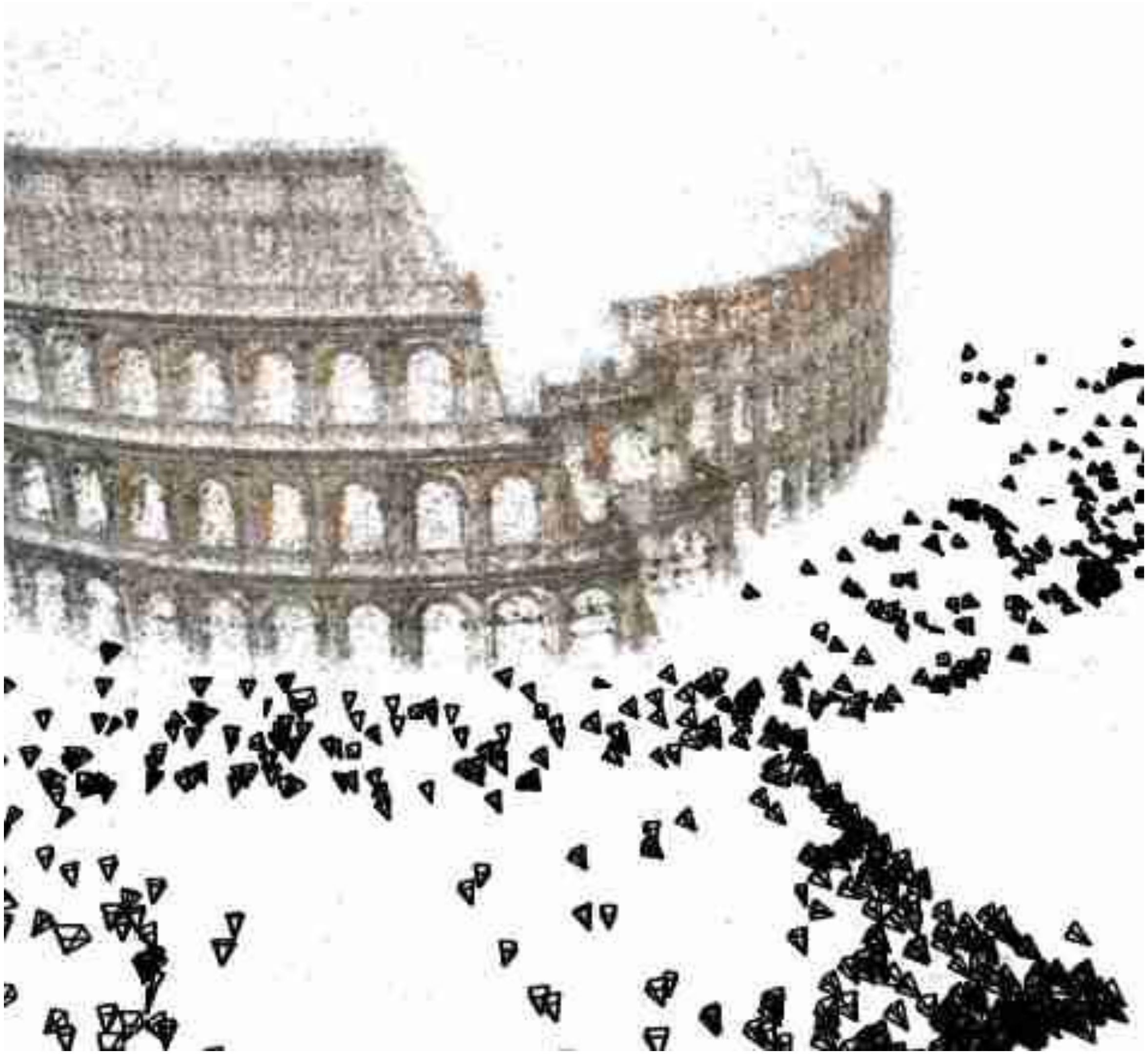
Our project

- Implementing visual SLAM/SFM
- Such as
  - ORB-SLAM
  - Build Rome in a day

ORB-SLAM



Build Rome in a day



Flow

A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow points & calculate Average 3D Coordinates
7. Display Points & Camera Locations

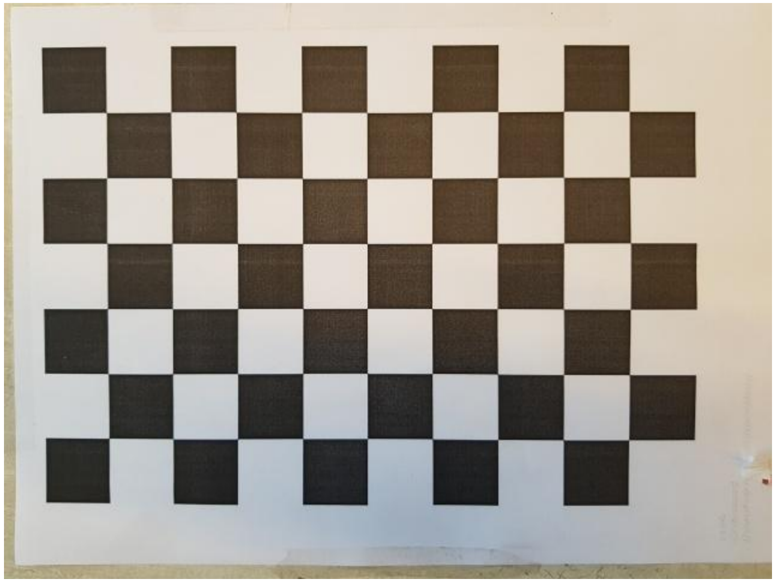
Flow

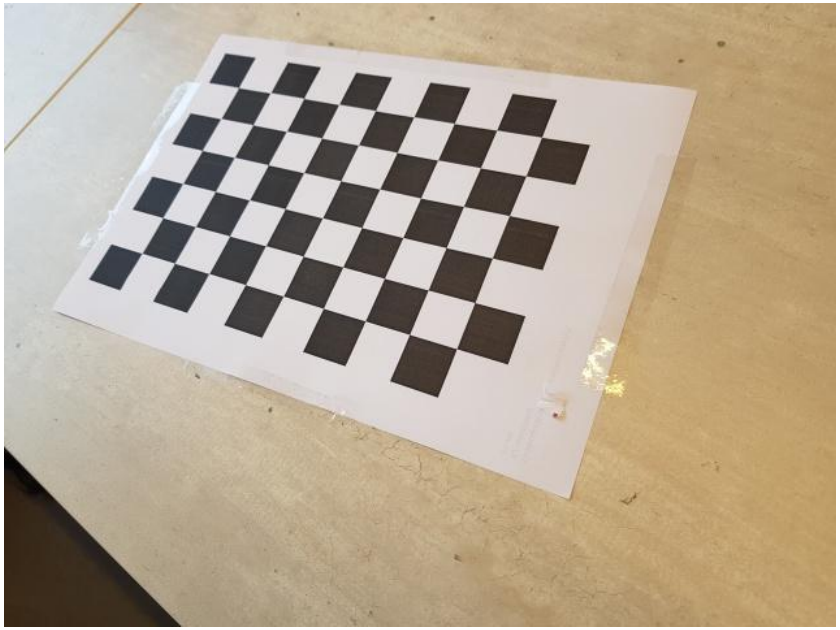
A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow points & calculate Average 3D Coordinates
7. Display Points & Camera Locations

Calibration

- We get intrinsic matrix K and distortion params





Flow

A. Preliminary step- Calibration

1. **Capture Video**
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow Points & Calculate Average 3D Coordinates
7. Display Points & Camera Locations

Flow

A. Preliminary step- Calibration

1. Capture Video
2. **Extract Frames**
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow Points & Calculate Average 3D Coordinates
7. Display Points & Camera Locations

Flow

A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. **Undistort Frames**
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow Points & Calculate Average 3D Coordinates
7. Display Points & Camera Locations

Flow

A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. Undistort Frames
4. **Extract SIFT Points from Key Frames**
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow Points & Calculate Average 3D Coordinates
7. Display Points & Camera Locations

Flow

A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. **Match Consecutive Key Frames**
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. Follow Points & Calculate Average 3D Coordinates
7. Display Points & Camera Locations

Finding initial matches

- For every P_1 in $image_1$ find P_2 and P'_2 of $image_2$ with closest descriptor
- If $|desc_{P_1} - desc_{P_2}| < 0.5 \cdot |desc_{P_1} - desc_{P'_2}|$
  - Keep match

RANSAC

- Finding Essential E and Inlier matches such that for every match of p_1 in $image_1$ and p_2 of $image_2$ it holds that
$$p_2^T \cdot E \cdot p_1 \approx 0$$

Get Pose

- Use points to find which of 4 possible RTs can be extracted from Essential matrix

Triangulate Points

- Use extracted RT and point observations to triangulate and find 3D coordinates of points

Flow

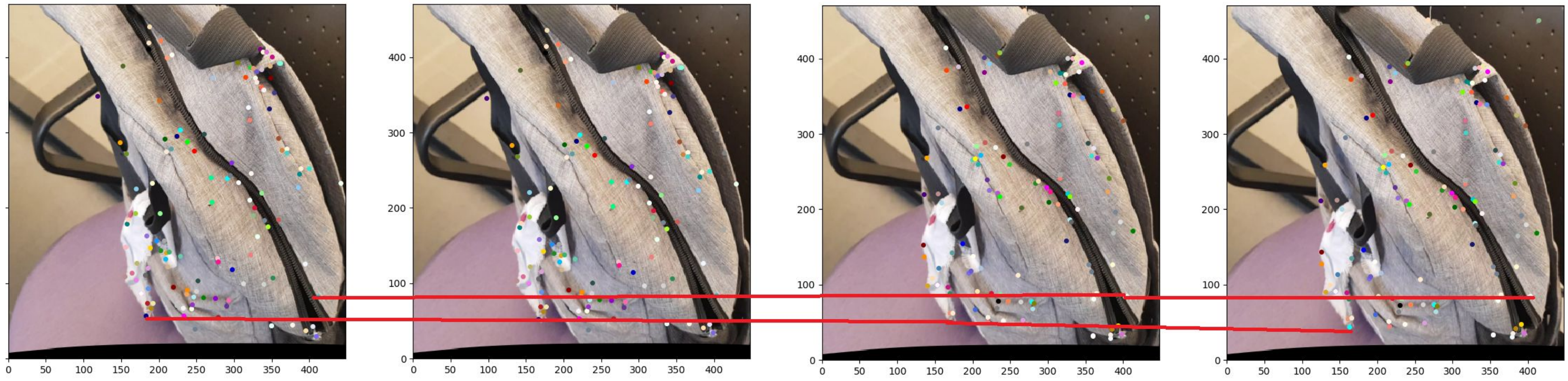
A. Preliminary step- Calibration

1. Capture Video
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
    1. Get Inlier Point Matches + Essential
    2. Get RT
    3. Trialgulate Points
6. **Follow Points & Calculate Average 3D Coordinates**
7. Display Points & Camera Locations

Follow Points

Use average 3D location of point

Color of point is the color where it was first observed



Flow

A. Preliminary step- Calibration
1. Capture Video
2. Extract Frames
3. Undistort Frames
4. Extract SIFT Points from Key Frames
5. Match Consecutive Key Frames
   1. Get Inlier Point Matches + Essential
   2. Get RT
   3. Trialgulate Points
6. Follow points & calculate Average 3D Coordinates
7. **Display Points & Camera Locations**

Display

- We get $RT_{2\to1}, RT_{3\to2}, RT_{4\to3} \dots RT_{n\to n-1}$
- We accumulate RTs
  - $RT_{i\to1} = RT_{2\to1} \cdot RT_{3\to2} \cdots RT_{i\to i-1}$
- We bring all point coordinates to Frame1 coordinate system

Smoothing RT

- RT are found at 10 frame steps
- We would like to have $RT_{1\to1}$,$RT_{1.5\to1}$, $RT_{2\to1}$, $RT_{2.5\to1}\cdots$ $RT_{n\to1}$ for smoother movement
- For that we need $RT_{1.5\to1}$,$RT_{2\to1.5}$, $RT_{2.5\to2}$, $\cdots$
- We assume $RT_{1.5\to1} = RT_{2\to1.5}$, Since $RT_{1.5\to1} \cdot RT_{2\to1.5} = RT_{2\to1}$ we get $RT_{1.5\to1} = RT_{2\to1.5} = \sqrt{RT_{2\to1}}$
- We are thus tasked with solving problem of finding $\sqrt{RT}$ or for more smoothing we would like to find $\sqrt[n]{RT}$

Finding $\sqrt[n]{RT}$

- We break RT to R and $\vec{T}$
- Assuming we found r such that $r = \sqrt[n]{R}$ it holds that for $\vec{t}$
$rt^n \cdot \vec{x} = rt^{n-1} \cdot \left(r \cdot \vec{x} + \vec{t}\right) = r \cdots \left(\cdot\cdot \left(r \cdot \vec{x} + \vec{t}\right) \cdot\cdot\right) \cdots + \vec{t} =$
$r^n \cdot \vec{x} + r^{n-1}\vec{t} + r^{n-2}\vec{t} \ldots r\vec{t} + \vec{t} =$
$R \cdot \vec{x} + (r^{n-1} + r^{n-2} \ldots r + \mathrm{I}) \cdot \vec{t}$
- If $rt^n = RT$ then $(r^{n-1} + r^{n-2} \ldots r + \mathrm{I}) \cdot \vec{t} = \vec{T}$ thus
$$\vec{t} = (r^{n-1} + r^{n-2} \ldots r + \mathrm{I})^{-1} \cdot \vec{T}$$
- Thus we can find $\sqrt[n]{RT}$ assuming we have $\sqrt[n]{R}$

Finding $\sqrt[n]{R}$

- Every rotation can be described by rotation axes $\vec{u}$ and angle α
- The rotation axes is the eigenvector of R corresponding to eigenvalue 1
- We find some $\vec{v}$ that is orthogonal to $\vec{u}$
- We get $\vec{w} = \vec{u} \times \vec{v}$
- U = $[\vec{u}\ \vec{v}\ \vec{w}]$ is an orthogonal matrix changing the basis
- Thus $R' = U^T R U$ is a rotation matrix around the x axes with the same α
- We can easily find α and create r' with $\frac{\alpha}{n}$
- We can now get $r = U r' U^T$ such that $\mathrm{r} = \sqrt[n]{R}$

Possible Improvements

- Bundle Adjust- was implemented but is still buggy
- Smart choises for key frames
- Match not only consecutive key frames to remove noise from RTs
- Find correct scale for translations- was implemented but not used yet
- Filter points differently
- Turn point cloud to mesh
- Many more…

NOW ENJOY THE DEMONSTRATION