Fast SFM-Based Localization of Temporal Sequences and Ground-Plane Hypothesis Consensus

Nick Rhinehart
May 2015
Goals

- Estimate ground plane from SFM cameras and image-based evidence
- Quickly estimate pose of new camera in temporal sequence in already seen environment

SFM from [3], Dense points from PMVS [1], Estimate of global plane
Goal 1: Ground plane estimation

- Points from SFM can be noisy, can't necessarily fit plane simply to ground plane
- Vanishing point methods can work but aren't globally consistent (single image)
Goal 1: Ground plane estimation

- Points from SFM can be noisy, can’t necessarily fit plane simply to ground plane
- Vanishing point methods can work but aren’t globally consistent (single image)
- **Use vanishing points & image evidence to estimate ground plane for each image, consolidate all estimates**
Ground plane from vanishing points

- Robust 3-direction vanishing point estimation of Manhattan world
- Also has per-pixel surface estimates

Images from applying method of [2]

Ground plane from vanishing points

- Robust 3-direction vanishing point estimation of Manhattan world
- Also has per-pixel surface estimates
- Can recover ground-plane normal estimate by using calibration matrix estimates from SFM

\[ n_{\text{ground}} = K^{-1} v p_{\text{vertical}} \]

Images from applying method of [2]

Ground plane normals

Most normals are reasonable, some are completely wrong
For most cameras, reasonably good estimate of plane normal. What about plane offset?

\[ \pi = [n, d] \]
Plane offset estimation

- For most cameras, reasonably good estimate of plane normal. What about plane offset?

\[ \pi = [n, d] \]

Apply per-pixel ground estimate as filter to SFM keypoints
Plane offset estimation

- For most cameras, reasonably good estimate of plane normal. What about plane offset?

\[ \pi = [n, d] \]

Apply per-pixel ground estimate as filter to SFM keypoints

Slide plane along normal, measure consensus with “ground” keypoints
Plane offset estimation

Maximum consensus plane and candidate planes formed from estimated normal in consensus with estimated ground keypoints.
Plot for all ground planes

- looks messy, many wrong or offset

http://www.cs.cmu.edu/~nrhineha/.hidden/videos/planes_all.mp4
(https://www.dropbox.com/s/jdffy11b4iudvdg/planes_all.mp4?dl=0)
Finding global best plane

- have set of planes $\Pi = \{\pi_1, \ldots, \pi_N\}$
- RANSAC across estimated planes with consensus test on angular offset and distance offset, then LSQ fit on plane with maximal consensus

$$C(\pi_i, \pi_j; \alpha, \beta) = 1 \left( \cos^{-1} \left( \frac{n_i^T n_j}{\|n_i\| \cdot \|n_j\|} \right) \leq \alpha \right) \cdot 1 \left( |d_i - d_j| \leq \beta \right)$$

Consensus set of maximal consensus plane (largest consensus set)
Finding global best plane

Global best plane estimate:

NSH 4th (sparse keypoints)

Smith 2nd (dense keypoints)
Scale bonus

- Images came from single user (egocentric camera).
- *Apply knowledge of user’s height*, distance between plane fit to camera (using normal from final ground plane)

~1.8 meters
Visualization

http://www.cs.cmu.edu/~nrhineha/.hidden/videos/nsh4rotate_vis2.mp4
(https://www.dropbox.com/s/tnvtp3retuumkw2/nsh4rotate_vis2.mp4?dl=0)

http://www.cs.cmu.edu/~nrhineha/.hidden/videos smith2rotate_vis.mp4
(https://www.dropbox.com/s/w1qzln37zypc54h/smith2rotate_vis.mp4?dl=0)
Goal 2: Fast estimation of temporal sequence in mapped environment
Fast estimation of camera pose in temporal sequence

- Can localize a camera in SFM model by performing all-pairs matching (slow, expensive) + partial bundle adjustment (adjusting only parameters of new camera)

- In large environments, prior on location is very useful to constrain matching
Using prior on location in temporal sequence

<table>
<thead>
<tr>
<th>Algorithm 1: Temporal Sequence Localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Sequence of video frames $F$, SFM model $M$</td>
</tr>
<tr>
<td><strong>Output:</strong> Sequence of camera positions $P$ in global reference frame</td>
</tr>
</tbody>
</table>
Using prior on location in temporal sequence

0. Build KD-tree of positions of model cameras

<table>
<thead>
<tr>
<th>Algorithm 1: Temporal Sequence Localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Sequence of video frames $F$, SFM model $M$</td>
</tr>
<tr>
<td><strong>Output:</strong> Sequence of camera positions $P$ in global reference frame</td>
</tr>
</tbody>
</table>

```python
cur_pos = None

tree = KDTree(M.get_camera_positions())
```

Note: The KD-tree is a data structure that efficiently supports nearest neighbor search and range search in K-dimensional space.
Using prior on location in temporal sequence

<table>
<thead>
<tr>
<th>Algorithm 1: Temporal Sequence Localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: Sequence of video frames $F$, SFM model $M$</td>
</tr>
<tr>
<td><strong>Output</strong>: Sequence of camera positions $P$ in global reference frame</td>
</tr>
</tbody>
</table>

1. Build KD-tree of positions of model cameras
2. First camera in sequence matched against all frames

```python
cur_pos = None

tree = KDTree(M.get_camera_positions());
radii = InitializeRadii(); /* create list of a few small radii */
P = [];

for $f \in F$ do
    for radius $\in$ radii do
        if cur_pos $\neq$ None then
            nearby_cameras = tree.query_radius(cur_pos, radius);
            edges = M.match_pairwise(f, nearby_cameras.get_frames());
        else
            edges = M.match_pairwise(f, M.get_camera_frames());
```

Using prior on location in temporal sequence

0. Build KD-tree of positions of model cameras
1. First camera in sequence matched against all frames
2. Partial bundle adjustment to estimate camera intrinsics & extrinsics

<table>
<thead>
<tr>
<th>Algorithm 1: Temporal Sequence Localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Sequence of video frames $F$, SFM model $M$</td>
</tr>
<tr>
<td><strong>Output:</strong> Sequence of camera positions $P$ in global reference frame</td>
</tr>
<tr>
<td>cur_pos = None;</td>
</tr>
<tr>
<td>tree = KDTree(M.get_camera_positions());</td>
</tr>
<tr>
<td>radii = InitializeRadii(); /* create list of a few small radii */</td>
</tr>
<tr>
<td>P = [];</td>
</tr>
<tr>
<td>for $f \in F$ do</td>
</tr>
<tr>
<td>for radius $\in$ radii do</td>
</tr>
<tr>
<td>if cur_pos $\neq$ None then</td>
</tr>
<tr>
<td>nearby_cameras = tree.query_radius(cur_pos, radius);</td>
</tr>
<tr>
<td>edges = M.match_pairwise(f, nearby_cameras.get_frames());</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>edges = M.match_pairwise(f, M.get_camera_frames());</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>M.partial_bundle_adjustment(edges);</td>
</tr>
<tr>
<td>cur_pos = M.get.localized_position(f);</td>
</tr>
</tbody>
</table>
Using prior on location in temporal sequence

0. Build KD-tree of positions of model cameras
1. First camera in sequence matched against all frames
2. Partial bundle adjustment to estimate camera intrinsics & extrinsics
3. Future frames matched only to nearby cameras

```
Algorithm 1: Temporal Sequence Localization

Input: Sequence of video frames $F$, SFM model $M$
Output: Sequence of camera positions $P$ in global reference frame

1. cur_pos = None
2. tree = KDTTree(M.get_camera_positions());
3. radii = InitializeRadii(); /* create list of a few small radii */
4. P = [];
5. for $f \in F$ do
6.     for radius $\in$ radii do
7.         if cur_pos $\neq$ None then
8.             nearby_cameras = tree.query_radius(cur_pos, radius);
9.             edges = M.match_pairwise(f, nearby_cameras.get_frames());
10.        else
11.            edges = M.match_pairwise(f, M.get_camera_frames());
12.        end
13.     M.partial_bundle_adjustment(edges);
14.     cur_pos = M.get.localized_position(f);
15.     if cur_pos $\neq$ None then
16.         break;
17.     end
18. end
19. P.append(cur_pos); /* cur_pos will be None if failed */
```
Results

- Prior on camera location reduces matching set significantly (orders of magnitude)
- Speedup of full localization usually around 10x for model of ~1500 images (20 seconds -> 2 seconds on my computer), main speedup comes from not performing all-pairs matching. Slower in areas of model with “denser” cameras
Possible improvements

- **Goal 1 extension**: Piecewise planar approximation for larger SFM scenes composed of ground plane at multiple heights.

- **Goal 2 extension**: Localization could fail in SFM models with larger baselines / sparser, instead should match against the cameras that view keypoints nearby the keypoints of those recently viewed.

![Diagram](image-url)
References

Appendix (video links)

Ground planes:
- https://www.dropbox.com/s/tnvtp3retuumkw2/nsh4rotate_vis2.mp4?dl=0
- https://www.dropbox.com/s/w1qzln37zypc54h/smith2rotate_vis.mp4?dl=0

All ground planes:
- https://www.dropbox.com/s/jdffy11b4iudvdq/planes_all.mp4?dl=0