A self-controlling active camera system for a free viewpoint video

Yous Sofiane
Artificial Intelligence Lab.

Introduction and motivation

- Free viewpoint video
  - A spatial extension of the traditional video.
  - A viewer can freely choose and change his viewpoint within the scene.
  - *Virtualized reality [Kanade et al. 1997]*
  - *Free-Viewpoint Video of Human Actors [Carranza et al. 2003]*
  - *Video-based rendering for an interactive viewpoint control [Zitnick et al. 2004]*

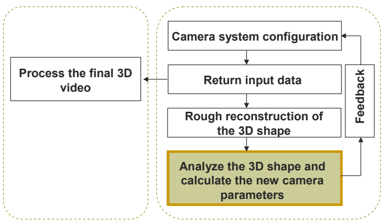
Moving object — Fixed camera system

A continuous global view of the scene is necessary → Input image with limited resolution

Introduction and motivation

- Our aimed system

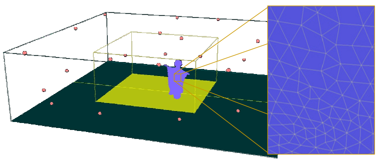
Input image with limited resolution → Active camera system: Moving object, Long focal-length lenses (partial views)



Unsupervised active control of the camera system in necessary

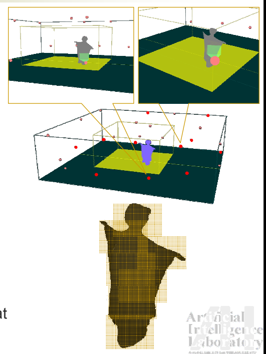
Introduction and motivation

- Input 3D shape
  - Polyhedral surface (mesh surface) defined by:
    - A set of facets
    - Each facet is defined by a set of vertices
    - The order of the vertices defines the outward face (outward normal vector)



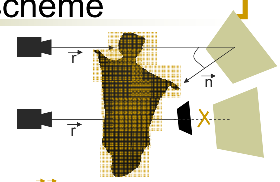
Self-controlling scheme

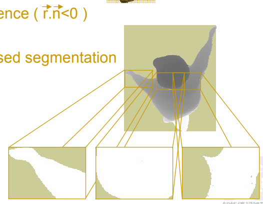
- Goal:
  - Follow the moving object within the scene
  - Assign each camera to a specific part of the subject
- Global scheme
  1. Project the 3D surface to the appearance plan of each camera
  2. For each camera evaluate the visibility to each part according to its field of view
  3. Seek for the best assignment (camera, part) configuration that maximize the global visibility



Self-controlling scheme

- Main problems
  - Windowing scheme
  - Visibility quantification
  - Assignment scheme
- Constraints
  - Visibility → Angle of incidence ($\vec{r}.\vec{n}<0$)
  - Accessibility → Occlusion
  - Connectivity → Depth-based segmentation
  - Mean distance

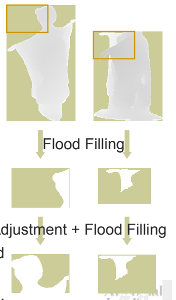




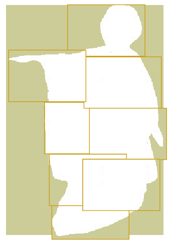
Windowing scheme

- Windowing constraints
  - Cover the entire object area
  - Obey the connectivity constraint
  - Be as optimized as possible
- Algorithm
  1. calculate the bounding rectangle
  2. Set a window to an expected position (pre-defined)
  3. Depth-based segmentation (Flood Filling)
  4. Repeat:
     - Adjust the window to the connected region
     - Re-apply the flood filling

     in so far as tow adjacent borders of the window fit with the contour of the connected component
  5. Delete the connected component area
  6. If the depth image still contain points go to 1



Windowing scheme



Visibility quantification

- Facet-wise level
- Local level
  - Constraints:
    - Respect the mean distance constraint.
    - Involve the facet area
    - Normalization
- Global visibility

$V(f,c)=|\vec{r}.\vec{n}|=|\cos(\alpha)|$; $\vec{r}, \vec{n}$ normalized *(Facet f, Camera c)*

$$L(c,w) = \frac{\bar{D}(c,w)}{\bar{S}(c,w)} \cdot \sum_{f \in Vp(c,w)} \frac{V(c,f).S(f)}{D(c,f)}$$

$$\bar{D}(c,w) = \frac{\sum_{f \in Vp(c,w)} (D(c,f))}{cpt_{c,w}}$$

$$\bar{S}(c,w) = \frac{\sum_{f \in Vp(c,w)} (S(f))}{cpt_{c,w}}$$

S (f) : The surface area of facet f
V p(c, w) (resp. , cpt(c, w)): The set (resp. the count) of facets visible from c and corresponding to the window w
D(f, c) : The mean distance of the facet i from j.

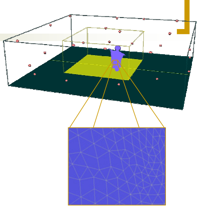
$$G = \sum_{c,w} L(c,w)$$

Assignment scheme

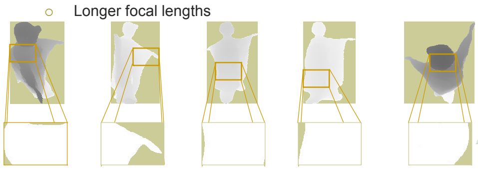
- Algorithm
1. *For each facet f compute Vf(f): the set of camera from where f is visible.*
2. *For each camera c:*
   1. *compute Vc(c): the set of visible facets*
   2. *Split Vc(c) into Vc(c, w): Windowing.*
   3. *For every couple (c,w), Calculate L(c,w)*
3. *repeat:*
   1. *Select the pair (c,w) having the highest local L(c,w) and assign c to w.*
   2. *Delete all facets chosen twice and accordingly, update L(c, w) for all*
   3. *windows comprising the deleted facets.*

until no camera or no facet left.

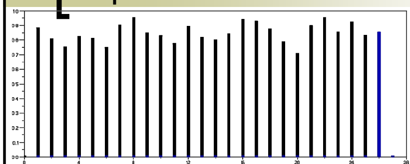
Experimental results

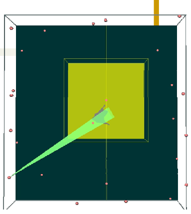


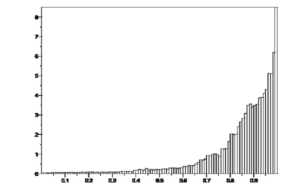
- Environment:
  - 25 cameras
  - Reconstruction: PPP scheme [Matsuyama et al. 2004]
  - Mesh surface
- Considerations
  - The same environment
  - Longer focal lengths

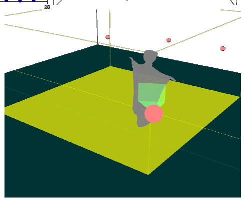


Experimental results









Camera movement optimization

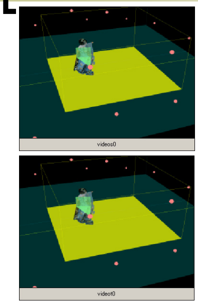
- Goal:
  - Keep assigning the same camera to the same part if the gain in visibility is not significant.
- How:
  - Involve a new parameter in the formulation of the local visibility formula.
- Advantage
  - Smooth, fast and optimized inter-frame camera movement

$$\acute{G} = G \cdot (\vec{P}.\vec{N})$$

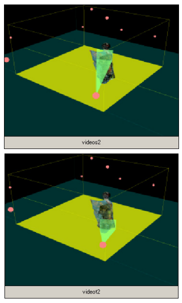
$\vec{P}$: *The last camera orientation vector*

$\vec{N}$: *The camera orientation vector to the current window*

Experimental results



Without optimization



With optimization

Conclusion and perspectives

- Conclusion
  - The fixed camera system limits the resolution of input images
  - Self-control is necessary for an active camera system
  - Two problems needs to be solved in order to achieve self-control:
    - Windowing scheme
    - Visibility qualification and evaluation
  - The experiments have shown the effectiveness of our proposed assignment scheme.
  - The inter-frame camera movement optimization permits a smooth, fast, and optimized camera movement
- Future works
  - Shape simplification
  - Parallel GPU-based implementation