Algorithms and techniques for virtual camera control

Session 3: Interactive Camera Control

M. Christie, Univ. Rennes 1
C. Lino, Univ. Rennes 1
R. Ranon, Univ. Udine
When a camera becomes interactive...

...we need to understand:

• the nature of the mapping between the user inputs and the camera parameters (internal constraints)

• the effect of other constraints on the camera parameters (i.e. external constraints such as visibility or surface of objects)
Interactive camera control

4 properties broadly characterize the space of interactive camera control techniques:

- **degrees of freedom of the input device**
  - low degree of freedom input devices (e.g. virtual arcball [Sho92], [CMS88])
  - 6 degree of freedom input devices (direct metaphors)

- **directness of the mappings**
  - control camera parameters, velocity, acceleration,…

- **nature of the constraint on motion:**
  - physical metaphors
  - geometrical
  - task

- world space vs. screen space based control
Enforcing usability

How? by reducing the dimensionality of the problem

- Fixing camera parameters (e.g. roll parameter)
- Automatically computing camera parameters
  - Lookat of the camera fixed to a target
  - Adding physical constraints to the camera
- Constraining camera parameters to a sub-space of possible motions
- Exploiting alternative camera models
Khan et al [KKS*05] developed a “hovercam” metaphor for individual **object inspection**:

- apply user input to the eye point $E_0$ (current camera position) and look-at point $L_0$, to create $E_1$ and $L_1$;
- search for the closest point $C$ on the object from the new eye position $E_1$;
- turn the camera to look at $C$, and
- correct the distance $\delta_1$ to the object to match the original distance to the object $\delta$ to generate the final eye position $E_2$.
- clip the distance travelled
Constraints: Shellcam [Bbk14]

- Boubakeur extended the approach using a smooth motion subspace on arbitrary objects.
- A scale-dependent offset shell is computed around the geometry:
  - it provides tangent directions for pan/tilt camera motions
  - the zoom changes the offset shell
- The *shell* is a low frequency offset of the geometry.
Environment-based control

- methods to assist navigation/exploration are mostly based on motion planning techniques from the field of robotics:
  - e.g., potential fields and vector fields
- methods require significant pre-computations

example: application to virtual colonoscopy [HMK97]
Towards indirect interaction

• multiple approaches implement more elaborate interactions with the camera (i.e. from parameters manipulation to properties manipulation)
  • **through-the-lens techniques:**
    interaction is performed on the content of the screen
    (for specifying camera motions, or screen composition)
  • **reactive techniques:**
    control is operated over targets which indirectly control
    the camera motions
    (typically following avatars [LC08,HHS01])
“Through the lens” control

- indicate desired positions of objects on the screen: **Through-the-lens camera control** (Gleicher & Witkin [GW92])
- difference between the actual screen locations and the desired locations indicated by the user is treated as a velocity
- relationship between
  - the velocity (\(\dot{h}\)) of \(m\) displaced points on the screen
  - and the velocity (\(\dot{q}\)) of camera parameters
- expressed with the Jacobian \(J\) that represents the perspective transformation:

\[
\dot{h} = J\dot{q}
\]
“Through the lens” control
“Through the lens” control

• The Jacobian is generally non-square \((m \times n)\)
  • \(m\): dofs of the camera
  • \(n\): parameters of the visual features in 2D

• Invert of the Jacobian?
  • compute its pseudo inverse with a Singular Value Decomposition (SVD)
    • \(O(mn^2)\) complexity
  • or use some optimization process (e.g. [GW92])

• Visibility needs to be handled separately…
  • by excluding some areas from the camera dofs
Though the lens control with The Toric Space

- Introducing a novel 3DOF representation of a camera [LC15]
- dedicated to viewpoint manipulation of two targets
- Three parameters to control the position:
  - $\alpha$ : angle between targets A and B
  - $\theta$ : horizontal angle
  - $\varphi$ : vertical angle
- the framing of the two targets is implicitly defined in the model

(Unity and C++ code available: ToricCam)
Composition : intuition (2D environment)

Desired on-screen Composition (1D)

Solution = 1-parametric manifold ($\theta$)

Any configuration $c(\theta)$ satisfies the 1D composition

Camera: C
$\alpha = (CB, CA)$
Composition: 3D environment

Desired on-screen Composition (2D)

Solution = 2D manifold surface $\theta, \phi$
(subset of a spindle torus)

Camera: $C$
$\alpha = (CB, CA)$

Any configuration $c(\theta, \phi)$ satisfies the 2D composition
More evolved problems:
⇒ relax the positioning constraint

Generalized model of camera:
• 3-parametric space \((\alpha, \theta, \varphi)\)

Defines the range of all possible manifolds around two targets

(Algebraically) casts 7D camera problems to 3D
Manipulations in the Toric Space

Video: https://www.youtube.com/watch?v=N-hEPkvGSf4
Manipulations in the Toric Space

**Principle:**
- **Manipulation of one target:**
  - while the other is constrained in the screen-space and roll is constrained to 0 (or a fixed value)
- **Interactions:**
  - change on-screen positions, distances, and vantage angles
  - example for on-screen positions:
    - we search for a position on the manifold surface where roll is null and minimizes the change in on-screen position

\[
\min_{(\theta, \varphi)} \left( p_A - p'_A \right)^2 + \left( p_B - p'_B \right)^2
\]
Demonstration

Video: [https://www.youtube.com/watch?v=3kFAlaihIX8](https://www.youtube.com/watch?v=3kFAlaihIX8)
Bibliography