Integrated 4D vision and visualization

MTA SZTAKI, Distributed Events Analysis Research Laboratory
http://web.eee.sztaki.hu/i4d/

Csaba Benedek    Zsolt Jankó

integrated 4D (i4D) System

• i4D: a pilot system for reconstruction and visualisation of complex spatio-temporal scenes by integrating two different types of data:
  o point cloud sequences measured by a Velodyne LIDAR sensor,
  o and 4D models of moving actors obtained in a 4D studio.

4D conception = „dynamic 3D”

• Reality based virtual 3D
  o Virtual city model:
    3D – spatial coordinates
  o Ordinary video sequences:
    2D spatial + 1D temporal dimension
• 4D = data flow, where the time frames are 3D spatial data samples
  o Recognition, interpretation—LIDAR
  o Reconstruction and visualization—4D Studio

4D scene reconstruction

Goal: creating viewpoint free videos

Velodyne LIDAR

• Hardware: Velodyne HDL-64E LIDAR
• Output: 2.5D point cloud sequence from outdoor environments
• Technical data:
  • 64 laser and sensor
  • 120 m distance
  • <2cm accuracy
  • >1.333M points/sec

Lidar (Deva)
Recorded camera image
4D reconstruction studio (GMCV)
Synthetized 4D scene sketch
Surveillance: Courtyard scenario by fixed LIDAR

LIDAR on a mobile mapping platform

Foreground - background separation

Pedestrian detection & multi target tracking

Environment reconstruction

Integration & visualization of spatio-temporal model

Actor videos pre-recorded in a 4D studio

Building 4D models of walking pedestrians

**Foreground detection**

- Velodyne – cylinder projection
- 3D point cloud
- Full view range image (64x1024 pixels)

**Background model**

- Temporal Mixture of Gaussians (MoG) model:
  - MoG approximation of the \( d(s) \) range image value history at each pixel
  - fixed \( K \) number of components (here \( K = 5 \))
  - background: \( k \) largest weighted components
  - Background fitness term of a “pixel”

\[ f_{bg}(s) = \sum_{k=1}^{K} \omega_k \cdot \log \left( f_{bg}^k(s) \right) \]

- Noisy result - errors in textured or dynamic background

**Foreground model**

- Foreground class: non-parametric kernel density model
  - in the neighborhood of foreground pixels, we should find foreground pixels with similar range values

\[ f_{fg}(s) = \sum_{r \in N(s)} \omega_r \cdot \log \left( f_{fg}^r(s) \right) \]

**Dynamic MRF model**

- 2-D pixel lattice \( S = \{s\} \)
- Nodes: image points (\( s \) is a pixel)
- Edges: interactions \( \rightarrow \) cliques
  - intra-frame edges: spatial smoothness
  - inter-frame edges: temporal smoothness
- MRF energy function
- Energy optimization
  - Graph cut based method (real time)
Range image segmentation

- Basic MoG
- MRF with uni fg, probs
- Proposed DMRF model

Label backprojection

- Point cloud labeling based on the segmented range image
  - Problems due to angle quantization for the discrete pixel lattice
  - Misclassified points near object edges and 'shadow' edges

Final point cloud classification

- Point cloud classification based on the segmented range image
  - $\omega(p)$: point cloud label
  - $\omega_r$: range image label of pixel corresponding to point $p$
  - handling the ambiguous point ($p$) - pixel ($s$) assignments

  - $\omega(p) = fg$, iff one of the following two conditions holds:
    - $\omega_s = fg$ and $p$ is closer to the sensor than the background range image value in $s$: $d(p) < d_s$
    - $\omega_s = bg$ and we find a neighbor $r$ of pixel $s$, where $\omega_r = fg$ and $d(p)$ is close to $d_r$

Qualitative results

- Basic MoG: 120fps
- uniMRF: 18fps
- 3D-MRF: 77fps
- Prop. DMRF: 16fps

Quantitative results

<table>
<thead>
<tr>
<th>Sequence name</th>
<th>Point cloud size</th>
<th>F-measure based on 100 frames (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bas. MoG</td>
<td>uniMRF</td>
</tr>
<tr>
<td>Sumn1</td>
<td>60K pts</td>
<td>55.7</td>
</tr>
<tr>
<td>Sumn2</td>
<td>60K pts</td>
<td>59.2</td>
</tr>
<tr>
<td>Sumn3</td>
<td>60K pts</td>
<td>28.4</td>
</tr>
<tr>
<td>Winx1</td>
<td>60K pts</td>
<td>55.0</td>
</tr>
<tr>
<td>Winx2</td>
<td>60K pts</td>
<td>54.9</td>
</tr>
<tr>
<td>Spring1</td>
<td>60K pts</td>
<td>49.9</td>
</tr>
<tr>
<td>Spring2</td>
<td>60K pts</td>
<td>56.8</td>
</tr>
<tr>
<td>Traffic</td>
<td>260K pts</td>
<td>70.4</td>
</tr>
<tr>
<td>Processing Speed</td>
<td>120fps</td>
<td>17-38fps</td>
</tr>
</tbody>
</table>

Pedestrian separation and tracking

- Detection: ground projection + blob separation
- Assignment
- Kalman filter prediction
- Kalman filter correction
- Tracking: state machine
Reconstruction of the background

2.5D point cloud  2D panorama photo

3D surface model  3D textured scene model

Moving avatar creation – 4D Studio

A studio equipped with multiple calibrated video cameras

Objectives:
- capture same scene from various viewpoints
- create dynamic 3D models of moving objects

Panorama of 4D studio at SZTAKI, Budapest

Acquisition of static objects

- One or two cameras sufficient, but more views → higher accuracy
- Cameras (or scanner) can move around the object

Acquisition of dynamic objects

- Dynamic objects
  - simultaneous images from multiple viewpoints → multi-camera system
  - fixed cameras
  - less views → lower quality
  - redundancy of sequence → higher accuracy

Hardware components

- Green box
  - cylinder with dodecagon base
  - massive, firm steel cage
  - 12 cameras uniformly around scene + 1 camera on top
  - green curtains and carpet → homogeneous background

Hardware components

- Cameras
  - wide-angle lenses
  - 1624 x 1236 pixels
  - 25 fps, GigE (Gigabit Ethernet)
- Lighting
  - light-emitting diodes (LEDs) set around each camera
  - can be turned on/off with high frequency
- Micro-controller
  - synchronises cameras and lights
  - opposite light turned off when camera takes picture
  - allows for more flexible configuration of cameras
- Computing power
  - 7 conventional PCs → 2 cameras per PC

Autodesk 3D CATCH

Moving avatar creation – 4D Studio

Moving avatar creation – 4D Studio

Moving avatar creation – 4D Studio

Moving avatar creation – 4D Studio
Hardware components
Adjustable platform with video camera and LEDs mounted on cage

Software components
- Two main software blocks
  - Studio
    - Image acquisition software
    - Video recording
  - ModelMaker
    - 3D reconstruction software
      - Creation of dynamic 3D models
- Complete software system developed at SZTAKI
  - Uses elements from OpenCV

Software components
- Selects cameras for acquisition
  - User can use a subset of cameras
- Configures cameras
  - Focus, gain, white balance, etc.
- Calibrates camera system
  - Intrinsic parameters of cameras (focus, lens distortion, etc.)
  - Positions and orientations of cameras in joint coordinate system
    - Extrinsic parameters
- Synchronises cameras and lights
- Captures synchronised multi-video sequences

Interface of image acquisition software

Camera system calibration
- Based on OpenCV routines (using method by Z. Zhang)
- Operator moves and shows 7×6 flat calibrating chessboard pattern to each camera
  - Pictures taken for varying orientations of calibrating pattern
  - Corners detected and identified unambiguously
    - Intrinsic parameters of each camera
    - Lens distortion parameters of each camera
    - Relative positions and orientations (poses) of neighbouring cameras
- Chessboard pattern put on table and viewed by upper-row cameras
  - Extrinsic parameters of upper-row cameras
  - Relative poses of bottom-row cameras already known
    - Extrinsic parameters of all cameras
- Asymmetric pattern is used to unambiguously identify corners under rotation
- Calibrating pattern is rotated to vary orientation
- Extrinsic parameters are defined in common coordinate system
Reconstruction software step-by-step

1. Extract colour images from raw data captured
2. Segment colour images to foreground and background
3. Create volumetric model by Visual hull algorithm
4. Create triangulated mesh from volumetric model
5. Add texture to model

Reconstruction software step-by-step

• Basic configuration: each video frame is processed separately
  o temporal coherence has been addressed later (not discussed here)
• Binarises colour input images
  o assigns 0 to background, 255 to object
  o binary image provides silhouette of object
• Principles of segmentation
  o assumes that background is larger than object
  o reference background image obtained in absence of objects
  o input RGB image converted to spherical colour representation
    → improves robustness to illumination changes
  o difference between image and reference background calculated
  o object detected as outlier using robust outlier detection

Creating volumetric model

• Apply shape-from-silhouettes to obtain Visual hull
  o maximal volume consistent with given set of silhouettes
• Back-project silhouette images in 3D space
• Take intersection of generalized cones
  o this gives bounding geometry of actual 3D object
    → concave details may be lost
  o more cameras → better geometry

Triangulated mesh generation from volumetric data

• Use the standard Marching cubes algorithm
• For each voxel, take 8 neighbors → cube
• Determine the polygon(s) needed to represent the part of the surface that passes through this cube
• Post-processing: smoothing, decimation
Texturing the triangulated surface

• For each triangle and camera, calculate measure of visibility
  o Triangle is visible from camera
  o Triangle normal vector points towards camera
• Form cost function with visibility and regularisation terms
  o Regularise to reduce sharp texture edges between adjacent triangles
  → Balance between visibility and smoothness
• Minimise the cost function using graph cuts
  → Find best image for texturing the triangle
• Quality of texturing depends on precision of surface geometry
  o Visual hull and Marching cube may yield imprecise normals
  → Texture details may be lost or distorted

Virtual pedestrians - 4D studio

Output of the integrated model

i4D workflow

Registered Lidar and camera sensor
Multi target tracking and person re-identification based on LIDAR

Velodyne on the moving platform

Horizontal LIDAR: street object and traffic monitoring

Tilted LIDAR: reconstruction of building facades

Lidar

Input point cloud frames (1,2,...,N)

Grid based segmentation of each point cloud (1,...,N)

Point cloud registration

Merged cloud

Surface reconstruction

Large planar regions

Moving object detection and classification

Other objects

Grid based re-segmentation and connected component analysis (merged cloud)
Preprocessing – point cloud segmentation

- A grid based method.
  - Uniform grid defined in the 2D space along the ground plane.
  - The grid is segmented as an image first.
  - Runs in real time.

- Point classes:
  - Noise and sparse data: grid cells with a few data points.
  - Ground surface: cells of points with small elevation differences (used threshold: 25 cm).
  - Tall objects (e.g. walls): cells with large elevation differences (more than 310 cm) or large maximal elevation (used 350 cm).
  - Short street objects: everything else (cars, pedestrians, street furniture, etc).

Street scene analysis from moving platform

- Street scene segmentation

Registration

- Only the points in the Wall or Tall Static Object class are used.
  - Noise and dynamic data are removed.
  - Reduced number of points.
  - Remaining points are strong features.

- Registration techniques
  - Normal Distributions Transform (NDT) – used for most of the following results.
  - Trimmed Iterative Closest Point algorithm (TrICP, Chetverikov at all, ICV 2005) – alternative method used in some tests.

Registration: results

Tilted configuration– Great Market Hall, Fővám tér

On the streets of Budapest

- Our office

- Kende utca (MTA SZTAKI)
Kálvin square

Tree crown removal
- Overhanging trees can corrupt object detection
- Registered data is dense, thus sparse regions with large scattering (such as leaves) can be detected
- Overhanging tree crowns can be removed

Result of upper vegetation detection

Vehicle detection
Frame #1
Frame #2
Frame #3
30 merged frames
2D recognition
3D backprojection

Distinguishing moving vs. static objects
- Moving objects result in blurred blobs in the merged cloud
- Solution: preserving the time stamp information for each point

Separating moving and parking vehicles
Center point sequence for a moving vehicle
Center point sequence for parking vehicles
Analysing motion tracks

Turning vehicle:

Trajectory of point cloud centers

Point cloud sequence color = time stamp

Pedestrians:

Road mark detection – zebra crossing

Point cloud with intensity coloring

Ground points after intensity based thresholding

Detected zebra

Horizontal histogram

Vertical histogram

Street object recognition

Road signs

Vehicles

Pedestrians

Tree

Surface reconstruction

Poisson triangulation of the obtained point cloud

Figures: main and southeastern facades of the Great Market Hall

NDT vs TrICP

• NDT is more robust for ‘featureless’ buildings (like office houses)
• TrICP gives superior results for surfaces containing characteristic features.

NDT

TrICP

• TrICP gives superior results for surfaces containing characteristic features.
Surface models

Surface + texture

Great Market Hall, Budapest

Data fusion

• Roofs (aerial) + facades (terrestrial scanning)

Mixed reality

4D scenario in front of the Great Market Hall

Acknowledgement

• Funding: The i4D Project was funded by the internal R&D grant of MTA SZTAKI, Budapest, Hungary

• Participating laboratories of MTA SZTAKI:
  o Distributed Events Analysis Research Laboratory
  o Geometric Modeling and Computer Vision Laboratory

http://web.eee.sztaki.hu/i4d

• Contributing people:
  • DEVA Lab.: Csaba Benedek, Attila Börcs, Csaba Horváth, Oszkár Józsa, Gábor Mészáros, Dömötör Molnár, Balázs Nagy, Tamás Szirányi
  • GMVC Lab.: Dmitry Chetverikov, Iván Eichhardt, Zsolt Jankó
Aerial Lidar: height map

- Optical aerial image
- LIDAR height map

3-D visualization

Auxiliary channels

- Usage: texturing, vegetation detection

Optical aerial image
- LIDAR intensity map
- LIDAR echo map

Patent (HUN)


Publications