Structured-Light Based Acquisition (Part 2)

CS635
Daniel G. Aliaga
Department of Computer Science
Purdue University
Acquiring Dynamic Scenes

• Scene: object (or camera) is moving and/or object is deforming

• Acquisition: capture as much information as possible in one to a few frames
  – By exploiting coherence
  – By exploiting several “channels” of information (e.g., color, infrared, etc...)
Acquiring Dynamic Scenes

• “Capturing 2½D Depth and Texture of Time-Varying Scenes Using Structured Infrared Light”, Frueh and Zakhor, PROCAMS 2005
  – Use single-frames and infrared illumination...

• “Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming”, Zhang et al., 3DPVT 2002
  – Use single-frames and colored patterns...

• “Real-time 3D Model Acquisition”, Rusinkiewicz et al., SIGGRAPH 2002
  – Use patterns producing local correspondences over a few frames and merge...

• “Fast 3D Scanning with Automatic Motion Compensation”, Weise et al., CVPR 2007
  – Use phase shifting and motion compensation over a few frames...

• “Real-time 3D Reconstruction at Scale using Voxel Hashing”, Niessner et al. 2013
  – Voxel-based RGB-D integration

• “Real-time Non-Rigid Reconstruction using an RGB-D Camera”, SIG 2014
  – Combine RGB and D and some template smarts
Capturing 2½D Depth and Texture of Time-Varying Scenes Using Structured Infrared Light
V-lines

Line defined at “middle” of IR strip

How do you know which line is which? Ideas?
H-lines

H-line sweeps up/down at 2Hz and enables an ordering of (a subset of) the V-lines and thus permits their correspondence.

Figure 6: Reconstructing the depth along V-lines. (a) IR frame; (b) V-lines from intra-frame tracking only; (c) V-lines with additional forward inter-frame tracking; (d) final result after V-lines with both forward and backward inter-frame tracking, and line counting.
Additional Steps

- Grab color image
- Foreground segmentation, and dense depth interpolation
- IR camera at 30Hz, color camera at 10Hz (probably faster today…)

Put it all together…
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

• Recall: how do we correspond lines?
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

- Use color transitions to define features
- Define lines at the transitions from color A to color B
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

• What is a notable problem?
• Resolution. Why?
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

- Only have three color channels (R,G,B) and can only robustly differentiate “strong” color changes
- This reduces the number of colors to use, and
- Often results in ambiguity in the color coding
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

• Challenges
  – Given a color code, how to do “best” correspond the stripes?
  – With the above in mind, how do we design a good color code?
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

• Challenges
  – Given a color code, how to do “best” correspond the stripes?
  – With the above in mind, how do we design a good color code?
How to “best” correspond the stripes

• Solution
  – Dynamic Programming
How to “best” correspond the stripes?

(rectified images)
How to “best” correspond the stripes?
How to “best” correspond the stripes?

Multiple match hypotheses \( \phi = \left\{ \left( \begin{array}{c} j_1 \\ i_1 \end{array} \right), \left( \begin{array}{c} j_2 \\ i_2 \end{array} \right), \ldots, \left( \begin{array}{c} j_H \\ i_H \end{array} \right) \right\} \)

Similarity score (of color) between edge \( e_i \) and transition \( q_j \) is \( s(q_j, e_i) \)

Score of the entire match sequence \( f(\phi) = \sum_{k=1}^{H} s(q_{j_k}, e_{i_k}) \)

Dynamic programming objective is: \( \arg \max_{\phi} f(\phi) \)
How to “best” correspond the stripes?

Dynamic programming objective is: \[
\arg \max_{\phi} f(\phi)
\]

However, the space all possible \( \phi \) is very large: \( O(M^N) \)

Solution?

Assume monotonicity (of the depth ordering):

\[
i_1 \leq i_2 \leq \ldots \leq i_H
\]

Great! But this monotonicity does not hold in what situation?

Occlusions! Oh well...

But it holds for individual fragments, which we can combine
How to “best” correspond the stripes?

Dynamic programming objective is: \( \arg \max_{\phi} (f(\phi)) \)

Let optimal \( \phi \) be called \( \phi^* \)

\[
f(\phi^*_{ji}) = \begin{cases} 
0 & \text{if } j=0 \text{ or } i=0 \\
\max & f(\phi^*_{j-1,i-1}) + s(q_j, e_i) \\
& f(\phi^*_{j-1,i}) \\
& f(\phi^*_{j,i-1}) 
\end{cases}
\]

\( f \) found through a recursive search and some optimizations to further reduce the search space (e.g., assume at most small depth changes from one column to another)
Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming

• Challenges
  – Given a color code, how to do “best” correspond the stripes?
  – With the above in mind, how do we design a good color code?
How do we design a good color code?

- De Bruijn sequence $B(k,n)$
  - (Dutch mathematician: Nicolaas Govert de Bruijn)
  - is a cyclic sequence of a given alphabet $A$ with size $k$ for which every possible subsequence of length $n$ in $A$ appears as a sequence of consecutive characters exactly once
  - thus it is optimally short as well
- $B(k, n)$ has length $k^n$
- Example: $A=\{0,1\}$
  - $B(2,2) = 01100$
    - All possible strings of length 2 (00, 01, 10, 11) appear exactly once as sub-strings in $A$
  - $B(2,3) = 00010111$ (or 11101000)
    - All possible strings of length 3 (000, 001, 010, 011, 100, 101, 110 and 111) appear exactly once as sub-strings in $A$
De Bruijn sequence $B(k,n)$

- Can also be constructed by a Hamiltonian cycle of an $n$-dimensional De Bruijn graph over $k$ symbols; e.g.,

(Hamiltonian cycle means each vertex is visited once)
Color Sequence

• Colors = \{000, 100, 110, ..., 111\} total of 8-1=7 because 000 is useless
• Color sequence is created by \( p_{j+1} = p_j \text{ XOR } d_j \)
  – XOR’ing effectively “flips bits” using \( d_j \)
  – \( p_0 \) is a chosen initial color (e.g., 100)
• Want 3 letters sequences \( d_j \) to be unique
• In practice about 125 stripes is sufficient
• Thus, a \( B(5,3) \) is adequate
Examples
Structured Light using Phase shifting

• Binary/gray code structured light
  – limited to (integer) projector resolution
  – has trouble with the LSB bits

• Alternative is phase shifting
  – Able to produce subpixel accuracies
  – Able to capture LSB bits more robustly
Phase Shifting

- Project 3 or more “phase shifted” sinusoidal patterns
- For example:
  \[ I_- = I_b + I_v \cos(\phi - \theta) \]
  \[ I_* = I_b + I_v \cos(\phi) \]
  \[ I_+ = I_b + I_v \cos(\phi + \theta) \]
- The variable \( \phi \) is the phase of each column
Phase Shifting

• Since binary/gray code good for MSB (and phase shifting less reliably for large ranges), we define absolute and relative phase shifts:

\[ \phi(x, y) = 2\pi k(x, y) + \phi'(x, y) \text{ where } k = [0, N - 1] \]

(from binary codes) (relative phase shift)
Phase Shifting

\[ I_- = I_b + I_v \cos(\phi' - \theta) \]
\[ I_* = I_b + I_v \cos(\phi') \]
\[ I_+ = I_b + I_v \cos(\phi' + \theta) \]

• How do you remove dependence on \( I_b, I_v \)?

• One option: use 3 images and compute

\[
\frac{I_- - I_+}{2I_* - I_- - I_+}
\]
Phase Shifting

\[
\frac{I_- - I_+}{2I_* - I_- - I_+} = \frac{2 \sin(\phi') \sin(\theta)}{2 \cos(\phi')(1 - \cos(\theta))} = \frac{\tan(\phi')}{\tan(\frac{\theta}{2})}
\]

thus

\[
\phi'(0, 2\pi) = \arctan(\tan(\frac{\theta}{2}) \frac{I_- - I_+}{2I_* - I_- - I_+})
\]
Real-Time 3D Model Acquisition

(slides and videos of this section by Syzmon Rusinkiewicz @ Princeton)
Real-Time 3D Model Acquisition Pipeline

- **3D Scanner**
- **View Planning**
- **Done?**
- **Display**
- **Alignment**
- **Merging**

The process involves:
1. **3D Scanner**
2. **View Planning**
3. **Done?**
4. **Display**
5. **Alignment**
6. **Merging**

A human interacts with the process at each step.
Real-Time 3D Model Acquisition Pipeline

Challenge: Real Time

3D Scanner

Alignment

Merging

Display

Done?

View Planning
Part I: Structured-Light Triangulation
Recall Triangulation...

- Depth from ray-plane triangulation
Recall Triangulation...

• Faster acquisition: project multiple stripes
• Correspondence problem: which stripe is which?
Codes for Moving Scenes

- Assign time codes to stripe boundaries
- Perform frame-to-frame tracking of corresponding boundaries
  - Propagate illumination history

Illumination history = (WB),(BW),(WB)

[Hall-Holt & Rusinkiewicz, ICCV 2001]
Designing a Code

- Want many “features” to track: lots of black/white edges at each frame
- Try to minimize ghosts – WW or BB “boundaries” that can’t be seen directly
Designing a Code

[Hall-Holt & Rusinkiewicz, ICCV 2001]
Space-Time Boundary Code
Implementation

- Pipeline:
  
  - DLP projector illuminates scene @ 60 Hz.
  - Synchronized NTSC camera captures video
  - Pipeline returns range images @ 60 Hz.
Real-Time 3D Model Acquisition Pipeline

Part II: Fast ICP
Aligning 3D Data

• ICP (Iterative Closest Points): for each point on one scan, minimize distance to closest point on other scan...
Aligning 3D Data

• ... and iterate to find alignment
  – Iterated Closest Points (ICP) [Besl & McKay 92]
ICP in the Real-Time Pipeline

• Potential problem with ICP: local minima
  – In this pipeline, scans close together
  – Very likely to converge to correct (global) minimum

• Basic ICP algorithm too slow (~ seconds)
  – Point-to-plane minimization
  – Projection-based matching
  – With these tweaks, running time ~ milliseconds

[Rusinkiewicz & Levoy, 3DIM 2001]
Real-Time 3D Model Acquisition Pipeline

Part III: Voxel Grid

View Planning → 3D Scanner → Alignment → Merging → Display

Done?
Merging and Rendering

• Goal: visualize the model well enough to be able to see holes

• Cannot display all the scanned data – accumulates linearly with time

• Standard high-quality merging methods: processing time ~ 1 minute per scan
Merging and Rendering
Merging and Rendering
Merging and Rendering
Merging and Rendering
• Point rendering, using accumulated normals for lighting
Example: Photograph

18 cm.
Postprocessing

• Real-time display
  – Quality/speed tradeoff
  – Goal: let user evaluate coverage, fill holes

• Offline postprocessing for high-quality models
  – Global registration
  – High-quality merging (e.g., using VRIP [Curless 96])
Fast 3D Scanning with Automatic Motion Compensation

Figure 1. 3D reconstructions of a static (left) and a moving (right) hand. Motion compensation (bottom right) removes the ripples from the reconstructed surface (top right).

• Higher resolution/quality than previous method
• Uses phase-shifting and motion-compensation
Fast 3D Scanning with Automatic Motion Compensation

Figure 7. Reconstruction of a complex scene containing several objects (phone, teapot, figure, fruit): a) texture image, b) reconstructed phase, c) geometry, d) textured geometry, e+f) close-ups.

Figure 8. Reconstruction of a waving cloth. Motion correction correctly removes the ripples (right).

Figure 9. Reconstruction of a person speaking.

Figure 10. Reconstruction of moving hands in front of the torso. On the right with motion compensation.

Figure 11. Online reconstruction of hand gestures.
Motion Compensation

- Since phase shifting assumes a static scene, **correlation-based stereo** is used to compensate for motion.

- An additional modification is proposed to **handle discontinuities** (which also plague standard phase shifting).
Motion Compensation

(projector) -> (moving) object -> (camera)
Real-time 3D Reconstruction at Scale using Voxel Hashing

Niessner et al. 2013 (TOG)
Real-time 3D Reconstruction at Scale using Voxel Hashing
Real-time 3D Reconstruction at Scale using Voxel Hashing

https://www.youtube.com/watch?v=XD_UnuWSaoU
Real-time Non-Rigid Reconstruction using an RGB-D Camera

[Zollhoefer et al. 2015]
Figure 2: Main system pipeline. **Left:** the initial template acquisition is an online process. Multiple views are volumetrically fused, and a multi-resolution mesh hierarchy is precomputed for the tracking phase. **Right:** in the tracking phase, each new frame is rigidly registered to the template, and a sequence of calls to the GPU-based Gauss-Newton optimizer is issued from coarse to fine mesh resolution. At the finest resolution, detail is integrated using a thin-plate spline regularizer on the finest mesh.
Real-time Non-Rigid Reconstruction using an RGB-D Camera

https://www.youtube.com/watch?v=qNiPirnvMHC