Real-Time Affine-Invariant Feature Extraction: Object Recognition Under Extreme Viewpoint Change

Presented by Valeriu Codreanu
What is object recognition?

Viewpoint-invariant keypoint extraction

GPU-ASIFT case study
  ◦ Multi-GPU implementation
  ◦ Timing/accuracy results

Real-time GPU-ASURF

Viewpoint-invariance overhead

Conclusions
Object Recognition

2-Step Process:

- Identify distinctive features (keypoints) using a feature extractor (SIFT/SURF/ORB/etc)
Object Recognition

2–Step Process:

- Find matching keypoints in source and query images
ASIFT is the first fully affine-invariant CV algorithm
- Proposed in 2009 by Morel et al.*

Based on the SIFT algorithm
But addressing its main deficiency
- Low invariance to camera viewpoint change

By simulating multiple views
But it generates/processes a lot more information than SIFT
- This makes it computationally expensive
- Unfeasible for real–time feature extraction on multi–core CPUs

Variation of the latitude angle (rotation)

Anti-aliasing filtering kernel. Gaussian

Variation of the longitude angle (tilt)

SIFT/SURF/ORB keypoint extraction
Parameter sampling

- Geometric sampling of $t$: $\Delta t = t_{k+1}/t_k = \sqrt{2}$.

<table>
<thead>
<tr>
<th>$t$</th>
<th>1</th>
<th>$\sqrt{2}$</th>
<th>2</th>
<th>$2\sqrt{2}$</th>
<th>4</th>
<th>$4\sqrt{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta$</td>
<td>$0^\circ$</td>
<td>$45^\circ$</td>
<td>$60^\circ$</td>
<td>$69.3^\circ$</td>
<td>$75.5^\circ$</td>
<td>$79.8^\circ$</td>
</tr>
</tbody>
</table>

- Arithmetical sampling of $\phi$: $\Delta \phi = \phi_{k+1} - \phi_k = \Delta \phi = \frac{72^\circ}{t}$.

Key idea:
- Although a tilt distortion is irreversible, it can be compensated by simulating a tilt of the same amount in the orthogonal direction.
ASIFT motivation

Viewpoint-invariant features (ASIFT)

Traditional features (SIFT)
All functions have a significant weight!
- So all have to be moved to the GPU
- Otherwise, Amdahl’s law limits the achievable gain
Data transfers between stages should be minimized

Percentage of runtime

- Compute SIFT keypoints: 56%
- Gaussian Blur: 18%
- Tilting: 15%
- Rotation: 8%
- Other: 3%
The goal is to have as little CPU<->GPU traffic as possible:
- Copy image data
- Readback keypoints

Implemented the image transformations as 3 CUDA kernels
- Rotation
- 1D Gaussian convolution
- Directional sub-sampling

A variation of SiftGPU* is used for SIFT keypoint extraction

---

* Changchang Wu: SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT)
CUDA implementation of SIFT
Generates the traditional set of 128-dimensional SIFT descriptors

Modified to fit our needs:
- Load image data from device memory
- Improved performance through stream concurrency
  - 20% performance increase on SiftGPU
Basic idea: The image transformations are fully independent
Total number of transformed images is divided by the number of GPU devices
Each GPU device extracts keypoints in parallel
Keypoints are then aggregated on the host
Scales almost linearly w.r.t. the nr. of GPU devices
  ◦ 1.75–1.9x for 2 GPUs
6 tilts generate 43 independent image transformations
  ◦ Can be distributed to multiple GPUs
Timing results

- Images originally sampled at 3MP
  - Upscaled and downscaled for performance measurements
- Features from 6x image area are extracted in 4x SiftGPU time
- 4x image area is computed in 2x time on GPU, 4x on CPU

\[ A(t) = 1 + (|\Gamma_t| - 1) \frac{180^\circ}{72^\circ} \]

\[ |\Gamma_t| = |\{1, \sqrt{2}, \ldots, \sqrt{2}^{t-1}\}| \]

Test system
- CPU: Intel Core i7–2600K
- GPU: Nvidia GTX690

<table>
<thead>
<tr>
<th>Resolution</th>
<th>single-core</th>
<th>quad-core</th>
<th>single-GPU</th>
<th>dual-GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>480x640</td>
<td>8861</td>
<td>2262</td>
<td>390</td>
<td>222</td>
</tr>
<tr>
<td>600x800</td>
<td>13634</td>
<td>3464</td>
<td>483</td>
<td>272</td>
</tr>
<tr>
<td>1024x1280</td>
<td>35101</td>
<td>8414</td>
<td>765</td>
<td>425</td>
</tr>
<tr>
<td>1200x1600</td>
<td>54039</td>
<td>13136</td>
<td>921</td>
<td>492</td>
</tr>
<tr>
<td>1536x2048</td>
<td>90465</td>
<td>21840</td>
<td>1342</td>
<td>710</td>
</tr>
<tr>
<td>1936x2584</td>
<td>141711</td>
<td>34086</td>
<td>2075</td>
<td>1092</td>
</tr>
</tbody>
</table>
Accuracy can be further improved by upsampling

- Much lower computational penalty than for the CPU algorithm
Send input image to GPU

Simulate each rotation/tilt on the GPU

Store image results in GPU memory

Match keypoints

Readback keypoint list to the CPU

Extract features for each transformation
Implemented image transformations using the OpenCV GPU functionality
- `gpu::rotate`
- `gpu::GaussianBlur`
- `gpu::warpAffine`

Data transfers are minimized
- Transformed images reside in GPU memory

The SURF/ORB GPU implementations are applied on the resulting images
- The framework can be used with any other feature extractor

Keypoint matching is done with kNN
Performance comparison

- Multi-GPU implementation provides good performance benefits
  
  - 1.75x–1.9x acceleration from 2 GPUs

- GPU–ASURF applied on low-resolution images extracts features in real-time on GTX690

<table>
<thead>
<tr>
<th>Resolution</th>
<th>GPU–ASURF</th>
<th>GPU–AORB</th>
<th>GPU–ASIFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>480x640</td>
<td>89</td>
<td>107</td>
<td>195</td>
</tr>
<tr>
<td>600x800</td>
<td>128</td>
<td>122</td>
<td>242</td>
</tr>
<tr>
<td>1024x1280</td>
<td>291</td>
<td>155</td>
<td>382</td>
</tr>
<tr>
<td>1200x1600</td>
<td>456</td>
<td>182</td>
<td>461</td>
</tr>
<tr>
<td>1536x2048</td>
<td>740</td>
<td>224</td>
<td>671</td>
</tr>
<tr>
<td>1936x2584</td>
<td>1145</td>
<td>341</td>
<td>1038</td>
</tr>
</tbody>
</table>
## Accuracy comparison

<table>
<thead>
<tr>
<th>Method</th>
<th># correct matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU–ASIFT (t = 2)</td>
<td>159</td>
</tr>
<tr>
<td>GPU–ASURF (t = 2)</td>
<td>75</td>
</tr>
<tr>
<td>GPU–AORB (t = 2)</td>
<td>24</td>
</tr>
<tr>
<td>GPU–ASIFT (t = 4√2)</td>
<td>497</td>
</tr>
<tr>
<td>GPU–ASURF (t = 4√2)</td>
<td>354</td>
</tr>
<tr>
<td>GPU–AORB (t = 4√2)</td>
<td>146</td>
</tr>
<tr>
<td>SIFT</td>
<td>1</td>
</tr>
<tr>
<td>SURF</td>
<td>2</td>
</tr>
<tr>
<td>ORB</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th># correct matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU–ASIFT (t = 2)</td>
<td>34</td>
</tr>
<tr>
<td>GPU–ASURF (t = 2)</td>
<td>15</td>
</tr>
<tr>
<td>GPU–AORB (t = 2)</td>
<td>8</td>
</tr>
<tr>
<td>GPU–ASIFT (t = 4√2)</td>
<td>151</td>
</tr>
<tr>
<td>GPU–ASURF (t = 4√2)</td>
<td>50</td>
</tr>
<tr>
<td>GPU–AORB (t = 4√2)</td>
<td>21</td>
</tr>
<tr>
<td>SIFT</td>
<td>0</td>
</tr>
<tr>
<td>SURF</td>
<td>2</td>
</tr>
<tr>
<td>ORB</td>
<td>2</td>
</tr>
</tbody>
</table>
Overhead of viewpoint-invariance

- Image transformation overhead is negligible in comparison to feature extraction
  - 3–11% depending on resolution
- Scales very well on multiple GPUs
  - All transformations are independent
- Programming overhead is minimal

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Overhead [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>480x640</td>
<td>10.8</td>
</tr>
<tr>
<td>600x800</td>
<td>13.5</td>
</tr>
<tr>
<td>1024x1280</td>
<td>16.9</td>
</tr>
<tr>
<td>1200x1600</td>
<td>20.2</td>
</tr>
<tr>
<td>1536x2048</td>
<td>25.8</td>
</tr>
<tr>
<td>1936x2584</td>
<td>38.6</td>
</tr>
</tbody>
</table>
Conclusions

- Simple OpenCV framework to develop viewpoint-invariant object recognition algorithms
- GPU-ASIFT provides a high level of accuracy while extracting features in near-real-time
- Real-time ASURF extraction using the OpenCV GPU framework

- The transformation framework:
  - has acceptable overhead
  - is scalable to multiple GPUs
  - can be applied to any “non affine-invariant” feature extractors
Thank you
Q&A