**Objectives**

* Introduce GPU hardware and features available for imaging & vision

1. 10-Series Architecture

* Each SM:
  + 8 Thread Processors
  + 1 double precision unit
  + 16kb shared memory, 16394 registers
* Series of 3\*10 streaming multiprocessors(SM)
* Each streaming multiprocessor has 8 thread multiprocessors
* Total 240 thread processors
* Total 4.0 GB of RAM
* 1 TFLOPS single precision
* 87 GFLOPS double precision (IEEE 754 floating point)

1. Texture Hardware

* Texture engine is a custom hardware block for image data access
* texture memory is another variety of read-only memory that can improve performance and reduce memory traffic when reads have certain access patterns.
* graphics applications involve memory access patterns which exhibit a great deal of spatial locality.
* **spatial locality of reference** -- having data that is stored close together
* texture cache system assumes that if data is accessed, there is a high probability that data nearby will be accessed soon.
* By minimising the distance between related data, more data can be retrieved from the faster cache (a cache hit) and you minimise the data that has to be retrieved from the slower source (a cache miss).
* Texture Features
  + Cached on chip
  + Bilinear interpolation
  + Data-type conversion
  + Boundary clamping (Automatic boundary handling)
  + Multi-channel aware
  + Handles 1D, 2D, or 3D spatial locality

3) Shared Memory (SMEM)

* Ultra-high speed “on-chip” memory
* Load an image tile into SMEM and share pixels between threads
* 

1. Visualisation : Graphics Interop

* Image and Video frequently need interactive display :Visualisation
* Graphics Interoperability introduces the various functions the runtime provides to interoperate with the two main graphics APIs, OpenGL and Direct3D.
* CUDA Interop with OpenGL and D3D allows display without additional data transfer overhead

**Imaging Pipeline: with respect to Panorama Stitching**

* Image stitching is a process of combining several individual images having some overlap into a high resolution smooth composite image.



1. Radial Distortion Removal

* Radial distortion is most visible when taking pictures of vertical structures having straight lines which then appear curved
* Radial Distortion Removal methods include interpolation
* Improved performance of interpolation methods implemented on GPU (table)
* Linear Interpolation is directly done from texture memory with the excellent Texture Cache behavior ; and we can Apply hardware linear interpolation to approximate higher order interpolation

1. Keypoint Detection & Extraction
2. Corner Detection

Compute matrix A, in region (u,v) around a point (x,y), of Gaussian weighted (w(u,v,)) image derivatives (Ix, Iy)

1. Dynamic

No single threshold can eliminate clutter and maintain weaker features

A weak Threshold gives indistinct cluttered features

A high Threshold leads to Loss of salient points

Dynamic Thresholding:

1. Compute Eigenvalues

Dynamic Threshold: Take ratio of min(λ1, λ2) to its neighbourhood (regional (u,v) minimum)

This involves additional Computational Costs for : One 2D convolution and a division or comparison

More complex algorithms gives better quality/performance with GPU

**Pixels to Points: HistoPyramids**

GPU HistoPyramids : help us to go from pixels to ( x,y ) point coordinates on the GPU

* helps to generate a list of feature coordinates completely on the GPU
* we can determine- What is the location of each point? And How many points are found?
* Applicable for quadtree data structures

HistoPyramid working:

The image buffer contains 1’s (points) and 0’s (non-points)

Do a reduction: each level is the sum of 2x2 region “below” it

The top level is the total number of points found

The pyramid is now a map to where the point locations are

Traverse down the pyramid, counting past points, and populate the list

Time Taken on : (C1060 GPU)

1024x1024: 0.27 ms

4096x4096: 2.1 ms

8192x8192: 7.7 ms

1. Keypoint Matching
2. Generating Descriptors

**A Feature Descriptor is a distinct numerical representation of an image point for matching eg. SIFT(Scale Invariant Feature Transform)**

**Feature computation involves Sparse point processing.**

**HistoPyramid tree organization gives good spatial locality .**

**Previous fragment shaders could do this but limited to one fragment processor (thread) per point but now parallel processing of single descriptor is possible via thread cooperation achieved via :**

**Shared Memory**

**Thread Synchronization**

**data dependent array indexing in compute shaders**

**Good texture cache usage, constant cache (Gaussian weights)**

**Feature Descriptor Computation :**

1. Calculate feature orientation *(shared memory reduction)*
2. Lookup rotated samples (texture cache)
3. Generate local orientation histograms (one thread per histogram, shared memory, pointer indexing)
4. Normalize Histogram *(shared memory reduction)*
5. **Threshold**
6. Re-normalize Histogram *(shared memory reduction)*
7. Feature Matching (SIFT Descriptors):

**SIFT Descriptor is a 128 element floating point vector (“key”)**

Matching operation : For SIFT the nearest Euclidean distance between keys(vectors) is the best match

Tesla C1060

1. Recover Homography (RANSAC)
2. Create Laplacian pyramid

**Introduce libraries and resources dedicated to GPU imaging & vision**