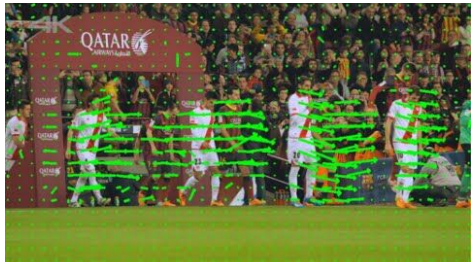
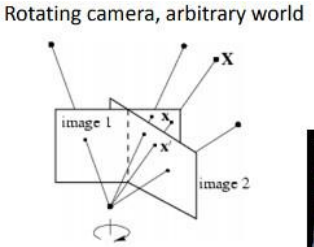
Parallel Corner Detection

Kunal Jobanputra and Ryan Stentz

Applications of the Harris Corner Detector

Optical Flow





Homography Estimation

Object Recognition

Feature Matching

Harris Corner Detection Algorithm

Original

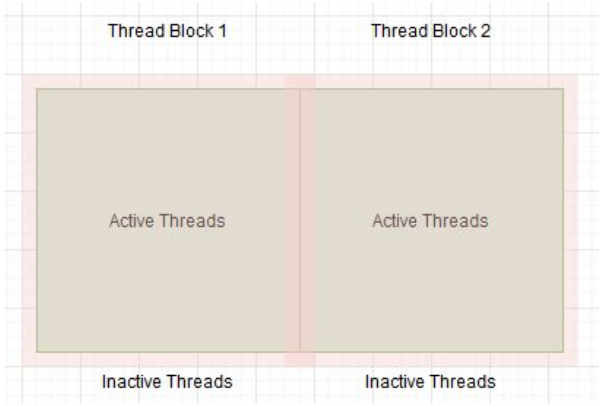
Gaussian Blur + Partials

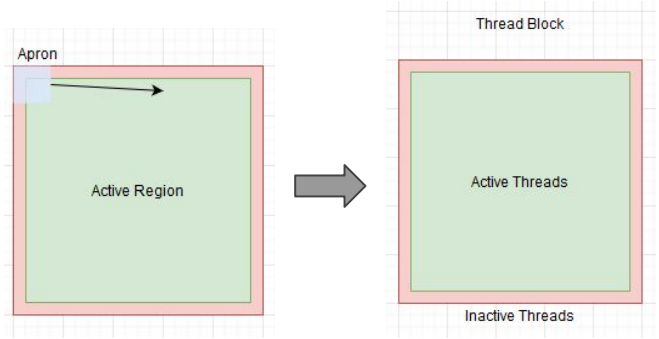
Corners Detected

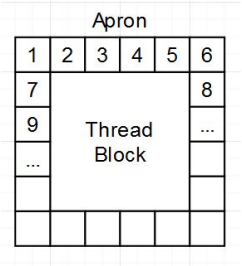
GPU - Approaches

Partial Derivatives (Sobel Convolution):

- Unroll Kernel Loop
  - Reduces data accesses and arithmetic
- Utilize shared memory
  - Each thread loads their pixel into memory
  - Creates idle threads
  - More thread blocks per image
- Utilize shared memory - 2nd approach
  - Each thread loads their pixel into memory, some threads load an additional pixel in the apron
  - No idle threads



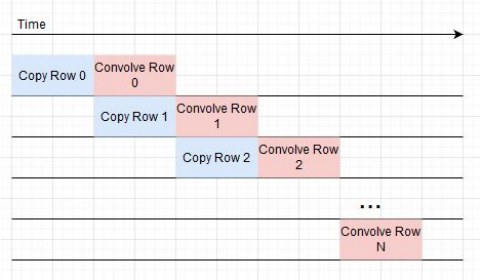




GPU - Approaches

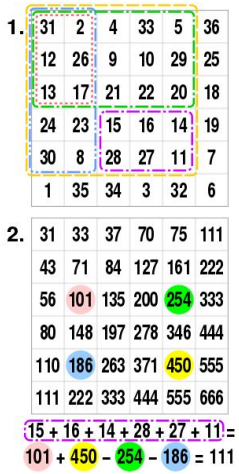
Partial Derivatives (Sobel Convolution): Streaming approach

- Helps hide memory overhead
- Utilizes CUDA streams to copy and process rows of an image independently



Cornerness Computation:

- Utilized the shared memory approach on the previous slide
- Compute and use integral images (pictured below)



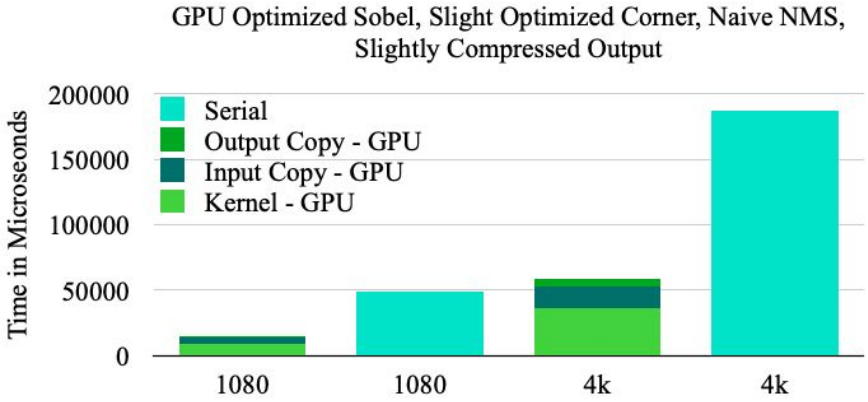
Output Compression:

- Switch from a float image to a byte image
- Compress the output into a vector of pixel locations instead of a binary image mask

GPU - Results

Our most optimized GPU implementation consisted of:

1. Shared memory sobel filter
2. Shared memory cornerness calculator
3. Non-maximal suppression
4. Slightly compressed output



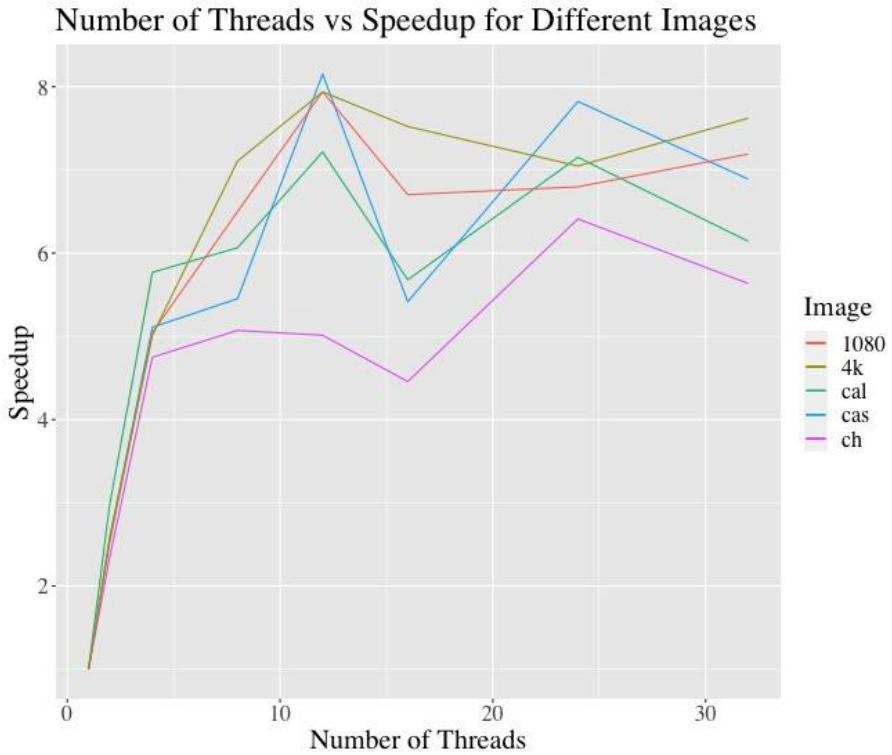
We achieved a speedup of more than 3x, making 1080p and 4k image processing on embedded devices feasible in real time

Shared Memory - Approach

- Static assignment
- Rewriting the loop to better use resources

```
#pragma omp parallel for num_threads(numThreads)
  for (int it = 0; it < srcGray.rows * srcGray.cols; it++) {
    int i = it / srcGray.cols;
    int j = it % srcGray.cols;
    harris.at<float>( i0: i,  i1: j) =  c(i, j);
  }
}
```

Shared Memory: Up to 8.15x Speedup



- Initial superscalar speedup
- Highest speedup around 12 threads, with one anomaly
- Spikes again round 24 threads, with anomalies from bigger images
- A tough apples to apples comparison to GPU implementation due to cost disparity