Parallel Monte Carlo search for Hough transform

Raul H. C. Lopes[1]
Virginia N.L. Franqueira[2]
Ivan R. Reid[1]
Peter R. Hobson[1]

[1]Brunel University London, UK
[2]University of Derby, UK

October 12, 2016

Table of Contents

Motivation

- The problem
  - The problem: Detection of straight lines from a set of pixels obtained from a digitized image.
  - Theoretical foundation: variations of Hough Transform (**HT**)
  - Target computing environment: parallel architectures in general.
  - Input assumptions: given is a set of pixels, representing image points abstracted from R^2 plane. We do not deal with how the pixels were obtained.
  - Output: a set of pairs of coordinates defining lines.

Motivation

- The problem
  - The problem: Detection of straight lines from a set of pixels obtained from a digitized image.
  - Theoretical foundation: variations of Hough Transform (**HT**)
  - Target computing environment: parallel architectures in general.
  - Input assumptions: given is a set of pixels, representing image points abstracted from R^2 plane. We do not deal with how the pixels were obtained.
  - Output: a set of pairs of coordinates defining lines.
  - Theoretical limits: solvable in $O(kN^2)$, where k is a constant that reflects the accuracy and is made explicit as it can get arbitrarily large.

Good for HEP and Physics?

- Atlas project tracking of muons traversing the Cathode Strip Chambers [1].
- Track reconstruction in CMS[2]
- Astronomy [4].

Proposed investigation

- Survey **HT** variations.
  - Many variations from initial Hough's idea, not just one.
  - Many algorithms for each of them.
  - Not many parallel algorithms and none with documented tests.

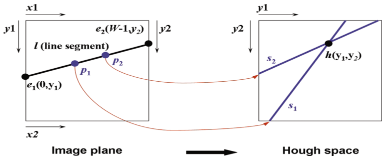
Proposed investigation

- Survey **HT** variations.
  - Many variations from initial Hough's idea, not just one.
  - Many algorithms for each of them.
  - Not many parallel algorithms and none with documented tests.
- Work in this paper
  - Investigate parallel algorithms for variations of **HT**.
  - Different parallel architectures: multi-GPU, multicore, Xeon-Phi, distributed memory.
  - Different parallel programming paradigms: arrayFire, Thrust, OpenAcc, OpenMP, MPI.

Table of Contents

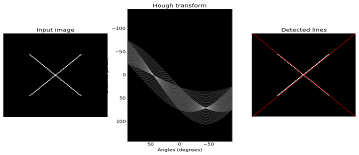
Standard transform as introduced by Hough

- Based on map from the (x, y) plane of the given points to (m, k) plane m and k ranging over the i possible slopes and intercepts of the lines.
- Each point in (x, y)-plane determines a line in (m, k)-plane.
- A bundles of lines intercepting in the (m, k)-plane define collinearity for the points associated with them in the (x, y)-plane.
- A combinatorial algorithm is immediately suggested. Complexity?



Duda-Haart-Hough Transform (**DHHT**)

- Introoduced by Richard Duda nad Peter Haart as an alternative mapping to the Hough transformi [3].
- Represent lines and points by their polar coordinates.
- Map in point in (x, y) to the (ρ, θ)-plane.
- Each point in the (x, y)-planedefines a sinusoid in the (ρ, θ)-plane.
- The intersection of two or more sinusoids indicates collinearity of the respective points.
- Algorithm: compute the sinusoid defined by each point and scan for intersections between sinusoid.



The Radon transform

- It can be seen as generalized form of the Hough transform.
- For a discrete image, a projection is computed as the sum of data lying in a unit-wide strip.
- Replaces HT vote counting with intensity addition.
- As in HT, integral along $x \cos\theta + y \sin\theta$

Table of Contents

Dynamic Combinatorial **HT**

- Implement **DHHT**.
- Exhaustive scan of each point to detect other collinear points.
- Target: find peaks of $(r, theta)$ pairs, where many lines intersect.
- Consider points in sequence and for each point p compute (r_i, θ_i) for the line defined by p and any other point p_i.
- Intersection of maximum peak $(r, theta)$ considered to define a line. All its points remove from further computations.
- Computation of all pairs (r_i, θ_i) can be made in parallel, however work for each point can be quadratic.
- Work for all points can be cubic.
- Prohibitive for example for a sample with 2^{20} foreground pixels?
- Work complexity can be improved by sampling.

Standard Duda-Haart-Hough algorithm

- Computation performed over discrete values for $(r, theta)$.
- For each discrete θ and each (x, y), a pair $(r, theta)$ is computed and a matrix of intersection counts is accumulated.
- Matrix $(r, theta)$ scanned for peak.
- Threshold can be used to decided which pairs $(r, theta)$ are considered to define lines. Example: OpenCV, and the Intel implementation of the Hough Transform.
- Accumulation and scan of matrix can be parallel.
- Problem: in principle, there is no mathematical limit for the granularity of θ intervals. No upper limit for work to be performed.
- Solution: θ scanned with finer granularity only in regions with higher density of data. It demands point indexing or clustering.

Fast and adaptive **HT**

- Sample voting by regions of space.
- Increase *resolution* of θ for regions with many accumulated votes.
- Split points in space to consider string cluster or to postpone computations in well separated regions.
- Standard **DHHT** or dynamic combinatorial **DHHT** can be considered.

Table of Contents

The essential steps of in a **DHHT** algorithm

- Voting: for each point and each possible angle θ find the respective $(r, theta)$ pair.
- Vote counting: count number of intersections for each $(r, theta)$.
- Vote scanning:
  - OpenCV: select all pairs $(r, theta)$ that exceed a given threshold.
  - Hough: find pairs that represent local maxima.

Complexity limits

- Voting
  - The easy step.
  - No interference involved.
  - Theoretical work upper bound in $O(n)$, for n pixels.
  - Practical scaling: $O(n/p)$, p being number of processors.

Complexity limits

- Voting
  - The easy step.
  - No interference involved.
  - Theoretical work upper bound in $O(n)$, for n pixels.
  - Practical scaling: $O(n/p)$, p being number of processors.
- Vote Counting
  - Data interference in accumulators as any line will see several (many?) points incrementing the same $(r, theta)$ accumulator.
  - Solutions
    - Impose mutual exclusion: mutex, compare-and-swap, others.

Complexity limits

- Voting
  - The easy step.
  - No interference involved.
  - Theoretical work upper bound in $O(n)$, for n pixels.
  - Practical scaling: $O(n/p)$, p being number of processors.
- Vote Counting
  - Data interference in accumulators as any line will see several (many?) points incrementing the same $(r, theta)$ accumulator.
  - Solutions
    - Impose mutual exclusion: mutex, compare-and-swap, others. Sequential algorithm, in worst case can become quadratic.
    - Concurrent writing for a leader election. Same upper bound as mutual exclusion.
    - Avoid data interference. For example, parallel on range of θ.

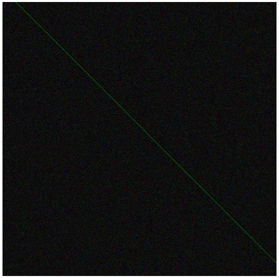
The parallel architectures and their programming models

- Multi-core
    - OpenMP
    - Thrust
    - OpenACC
- Multi-gpu
    - OpenMP
    - Thrust
    - ArrayFire
    - OpenACC
- Intel Xeon-phi
    - OpenMP
    - OpenACC
- Distriubuted memory
    - MPI
    - Coarrays

Table of Contents

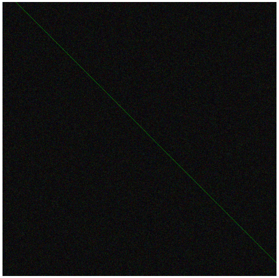
A line with Gaussian noise at 37 amplitude

- Image generated with cimg *noise(37)*



A line with Gaussian noise at 37 amplitude

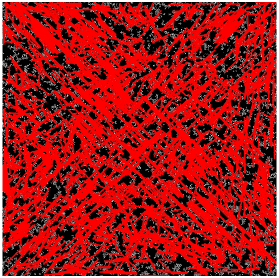
- Image generated with cimg *noise(37)*



- 200K lines detected with standard HT, using OpenCV.
- Disclaimer: Noise removal could have been applied.

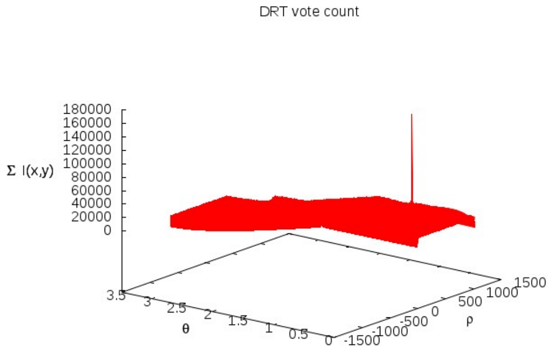
Probabilistic HT in OpenCV

- More than 2000 lines detected.



- Again: no tweaking done with noise removal or threshold setting.

DRT: The vote count



- $\theta \in [0..\pi)$
- $\sum I(x, y)$ sums intensity in one image strip.

Preliminary performance tests

- Run on gcc (-fopenmp) $-O3$, GNU Linux Centos 7, Intel Xeon E5-2680 v3 (2.50GHz).
- Best speed-up at 5.04 on 6 cores, (efficiency 84%).
- Drop in efficiency from 6 to 12 cores. Explained by cache/hit miss.
- Drop from 12 to 24 cores: cache hit/miss and non locality of data across physical processors.
- Huge drop in performance for 4096x4096 due again hit/miss ratio in processor's caches.

Pixels	1 core	6 cores	12 cores	24 cores
512x512	1.011s	0.249s	0.155s	0.096s
1024x1024	7.762s	1.548s	0.843s	0.460s
2048x2048	87.31s	24.07s	14.92s	8.181s
4096x4096	1144s	272.3s	154.6s	83.92s

Preliminary performance tests

- Run on gcc (-fopenmp) $-O3$, GNU Linux Centos 7, Intel Xeon E5-2680 v3 (2.50GHz).
- Best speed-up at 5.04 on 6 cores, (efficiency 84%).
- Drop in efficiency from 6 to 12 cores. Explained by cache/hit miss.
- Drop from 12 to 24 cores: cache hit/miss and non locality of data across physical processors.
- Huge drop in performance for 4096x4096 due again hit/miss ratio in processor's caches.

Pixels	1 core	6 cores	12 cores	24 cores
512x512	1.011s	0.249s	0.155s	0.096s
1024x1024	7.762s	1.548s	0.843s	0.460s
2048x2048	87.31s	24.07s	14.92s	8.181s
4096x4096	1144s	272.3s	154.6s	83.92s

- Tests ongoing for Aproximate DRT show 4Kx4K pixels on 4 cores at: 4.6s.
- Tests also being performed for a dual gpu architecture and 24 multi-core.

Table of Contents

On going work

- Approximate DRT now being test.
- Both HT and DRT can be improve by expoiting locality in regions with higher density of data.
- Parallel indexing higher order data is theme of our companion poster in CHEP'2016.
- DRT algorithm can used to compute inverse Radon Transform. An efficient parallel algorithm offers promise in radio-astronomy and electon microscopy, for example.
- Question: Can *reals* instead of floats improve either HT for DRT?
  - Unums for *reals*.
  - Sparse matrices needed.

References I

Nir Amram.
Hough Transform Track Reconstruction in the Cathode Strip Cambers in ATLAS.
PhD thesis, Tel Aviv University, 2008.

D. Cieri et al.
L1 track finding for a time multiplexed trigger.
Nucl. Instrum. Meth., A824:268–269, 2016.

Richard O. Duda and Peter E. Hart.
Use of the hough transformation to detect lines and curves in pictures.
Commun. ACM, 15(1):11–15, January 1972.

C. Hollitt and M. Johnston-Hollitt.
Feature detection in radio astronomy using the circle hough transform.
Publications of the Astronomical Society of Australia, 29(3):309–317, 001 2012.

Table of Contents

On going work

Lopes, Reid and Hobson are members of the GridPP collaboration and wish to acknowledge funding from the Science and Technology Facilities Council, UK.