Using FPGAs to Accelerate 3D Vision Processing: A System Developer's View

VanGogh Imaging

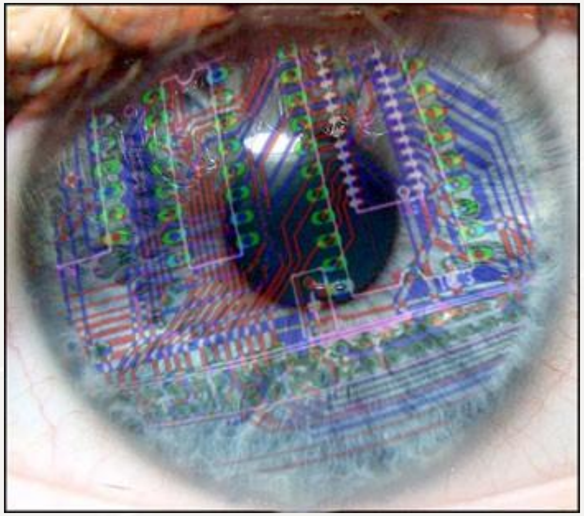
Ken Lee, CEO

October 2, 2013

Company Background

- Founded in 2007
- Located in McLean, VA
- Mission: “Provide 3D embedded and mobile vision technology for high volume applications”
- Computer Vision Technology Focus
  - 3D object recognition, feature detection, measurement
- Existing Customers: Medical, Gaming, Agriculture
- System Developed to Date
  - Measurement device for agriculture industry
  - 3D printing for medical industry
- Upcoming Products
  - Android Unity plug-in module

3D Computer Vision System

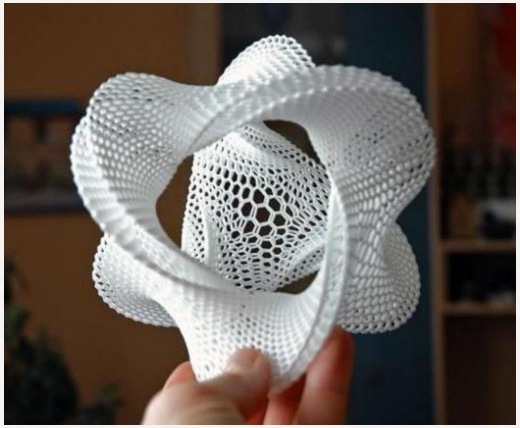


The CV-Measurement System

- Problem: Use 3D camera and computer vision technology to
  - Detect object
  - Check for scan quality
  - Identify features
  - Make measurements
- Product goals
  - Reliable, accurate, low cost
  - Need to capture and analyze in real-time
- Hardware
  - PrimeSense 1.08 3D Camera (with OpenNI drivers)
    - Low-cost 3D sensor
- Ideal frame rate requirement = 30 frames per second
- Use in harsh outdoor environments

Technology Overview

- Software Development
  - In-house library “Vincent” (in C and C++)
  - MATLAB for additional math functions
  - Open-Source library for Windows
- Types of Computer Vision Modules
  - Object Recognition
    - Shape detection, curvature estimation, noise cropping
  - Quality Control Check
    - Hole detection, scan obstruction, shape comparison
  - Feature Detection
    - Feature matching
  - Measurement Tools
    - Auto-calibration, spatial reference

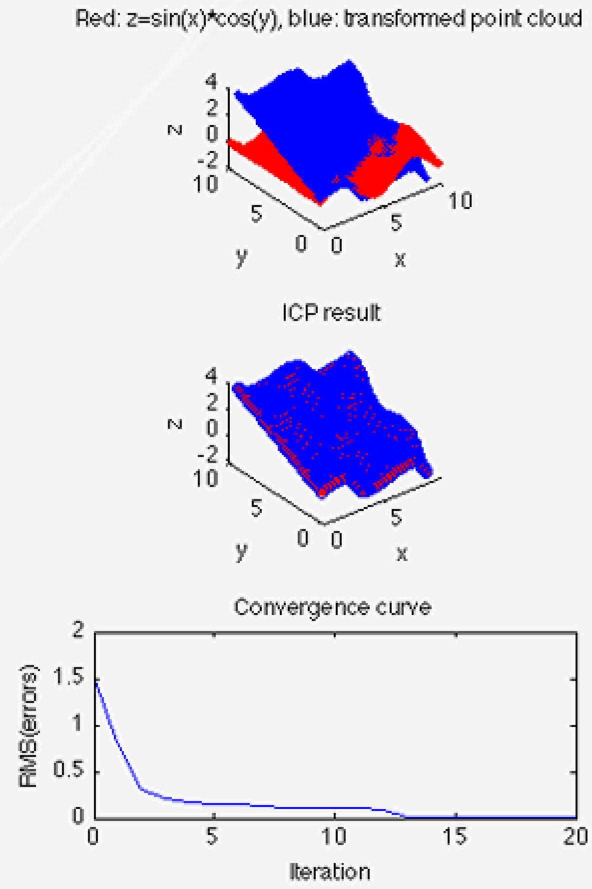


Note: (patent-pending)

Algorithm Implementation Example

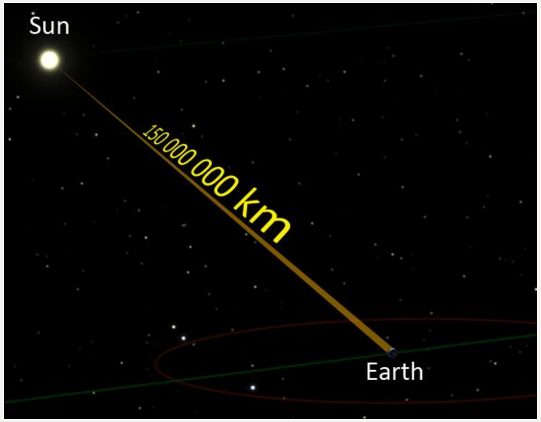
Algorithm Example—ICP

- Feature Detection - ICP (Iterative Closest Point)
- Match Captured 3D Scan to 3D Model
- Every Point in Scan And Model Compared
    - Find nearest neighbor for scan points
    - Covariance matrix
    - Find translation and rotation matrix
    - Apply to scan and repeat process
        - Typical number of iterations for us is 50
        - VERY PROCESSING INTENSIVE!



Module Example: Distance Function

- Within ICP Algorithm
  - Most computationally intensive portion is
    - "Find the nearest neighbor"
- Algorithm: ICP (Iterative Closest Point)
  - Function: Nearest Neighbor
  - Distance Formula
  - $D = (x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2$



Feature Detection Implementation

- Initial Implementation: Prototype - PC
  - C++, MATLAB, open-source libraries
  - Intel i7 quad-core, 8G RAM
  - Problems:

    Cost, environment (temperature, power, dust)

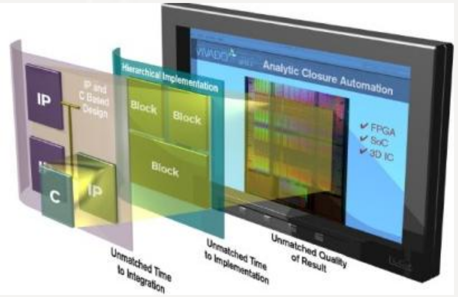
- Production Implementation: Micro-Linux (Android) with ARM processor
  - C++
  - ARM – A15, 1G RAM
  - Most libraries had to be re-written or developed from scratch
    - Does not compile or too slow
  - Changed the data structure
  - Problem: Works, but not as fast as desired

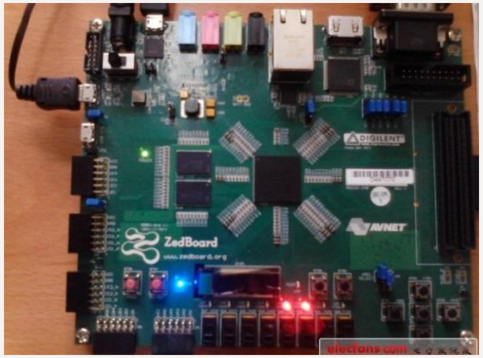
Adding FPGA to ARM

- Discovery of **Xilinx Zynq** processor!
  - 2 ARM processors
    - One ARM for the application
    - Second ARM for CV functions
  - FPGA programmable core
- Processing intensive functions can be off-loaded to the FPGA!
  - Good example is the ICP function
    - Runs faster on FPGA than on ARM
    - Can process multiple sub-datasets in parallel

Steps—ARM to FPGA

- Select Functions to Be Implemented in FPGA
  - FPGA - Nearest neighbor (78% of the processing)
  - ARM – Rest of the functions
- Using the Xilinx Vivado-HLS (converts C++ to gates)
  - Easy initial implementation
  - Optimize the resource versus performance
  - Parallel processing replication
- Prototype with Avnet Zedboard
  - Challenges
  - No floating point calculation
    - Changes to the algorithm required
  - Balance between resource versus speed
  - Multi-threading implementation





VanGogh Imaging

Intel i7 versus ARM versus FPGA

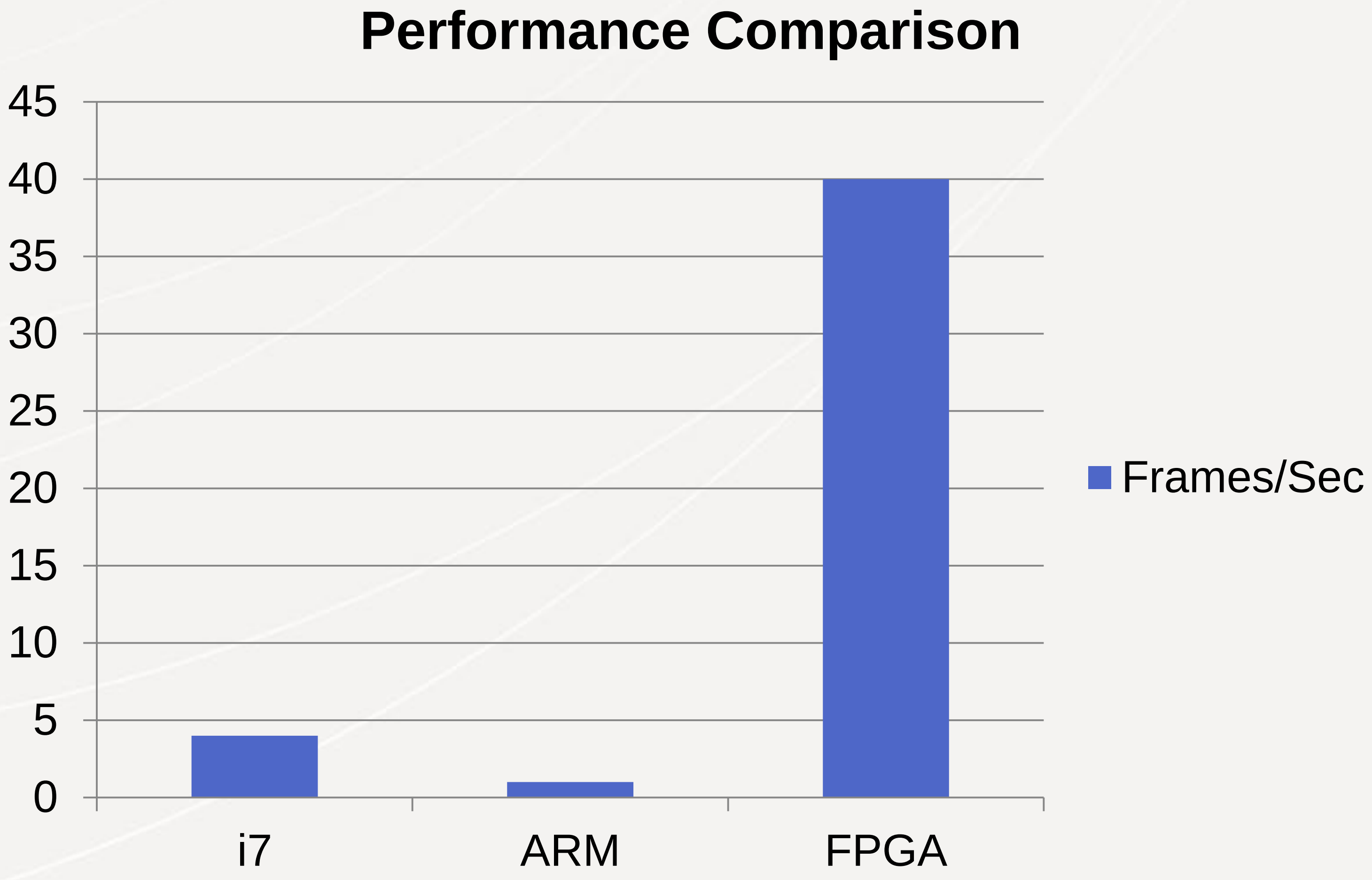
Nearest neighbor Implementation

- Function: “distance calculation” between two (x,y,z) points
  - D = (x1-x2)^2 + (y1-y2)^2 + (z1+y2)^2
- # points in the model = 1000
- # points in the scan = 1000
- For each iteration, there are 1000 x 1000 = 1M distance calculations
- Number of iterations (typically 50)
- Total number of “distance calculations”
  - 50 iterations to ‘converge’
  - 1M x 50 = 50 million

Nearest Neighborhood Performance Comparison (Measured)

- On PC (i7-2600 running 3.4 Ghz with one thread used)
  - 248 millisecond for 50 million distance calculations
  - Or, **4 frames per second**
- On ARM (A15 1.7Ghz with one thread used)
  - ~1 sec for 50 million distance calculations
  - Or, **1 frame per second**
  - Solution: Down-sample or use multiple cores => 4 frames per second
- On Zynq SoC 7020
  - Nearest Neighbor (100 x 100 points) using 3% of the resource
    - We can put 20 in these in parallel in a single chip
  - 25 milliseconds for 50 million distance calculations per function
  - Or, **40 frames per second**

Nearest Neighborhood Performance Comparison (Measured)



Summary

Summary

- PC (Intel) → ARM only → ARM + FPGA
- ARM is a pretty good solution but bit slow
- FPGA (Xilinx Zynq) is good for
  - Easy transition from ARM-based system
  - Significant performance boost
- IMPORTANT
  - Algorithm has to be designed to run on FPGA
    - Most libraries (e.g., PCL) cannot be easily imported
    - Efficient and simple algorithm needed
    - Small datasets needed for parallelization
  - Trade-off between software and hardware resources
- Positive experience so far
  - Good tools (Vivado-HLS, evaluation board) and SoC approach

Resources

- www.vangoghimaging.com
- Android 3D printing: http://www.youtube.com/watch?v=7yCAVCGvvso
- “Challenges and Techniques in Using CPUs and GPUs for Embedded Vision” by Ken Lee, VanGogh Imaging—http://www.embedded-vision.com/platinum-members/vangogh-imaging/embedded-vision-training/videos/pages/september-2012-embedded-vision-summit
- “Using FPGAs to Accelerate Embedded Vision Applications”, Kamalina Srikant, National Instruments—http://www.embedded-vision.com/platinum-members/national-instruments/embedded-vision-training/videos/pages/september-2012-embedded-vision-summit
- “Demonstration of Optical Flow algorithm on an FPGA”—http://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/videos/pages/demonstration-optical-flow-algorithm-fpg

Thank you