Low-Cost Visually Intelligent Robots with EoT

David Moloney[1], Dexmont Pena[1], Aubrey Dunne[1], Alireza Dehghani[1], Gary Baugh[1], Sam Caulfield[1], Kevin Lee[1], Xiaofan Xu[1], Maximilian Müller[1] , Remi Gastaud[1], Ovidiu Vesa[2]
[1] Movidius Ltd., 1st Floor, O'Connell Bridge House, D'Olier St, Dublin D02 RR99, Ireland
[2] Movidius Inc., 1730 S El Camino Real, San Mateo, CA 94402, United States
Email: {firstname.surname@movidius.com}

Oscar Deniz Suarez, José Luis Espinosa Aranda, Noelia Vallez Enano
VISILAB Grupo de Visión y Sistemas Inteligentes, Av. Camilo José Cela, s/n 13071 Ciudad Real, Spain
Email: oscar.deniz@uclm.es

***Abstract*—this paper describes the development of a low-cost, low-power visual-intelligence and robotics platform based on the H2020 EoT (Eyes of Things platform) and the Movidius Myriad2 VPU (Vision Processing Unit), associated machine vision, communications and motor-control libraries and the Movidius Fathom deep-learning framework. The platform allows a variety of low-cost robots to be built using kits and programmed in micropython. The ultimate goal of the EoT robotics project, which is still in development, is to enable the mass-production of advanced and highly programmable robots for developers, students and hobbyists at a sub $100 price-point that includes robot, control electronics, software and an eBook.**

Keywords-component; Vector Processor, Vision Processing Unit, VPU, ISA, Computer Vision, Computational Imaging, Hardware Accelerator, SIMD, heterogeneous, Multicore, VLIW, VLLIW, Multicore Multiported Memory Subsystem, EoT, machine-vision, deep-learning, low-cost, low-power, MQTT, micropython, Motion control

I. INTRODUCTION

Typical compute platforms used in embedded robotics include the Odroid XU4, Nvidia Tegra K1 and Tegra X1 platforms shown in *Figure 1*. The Odroid platform is restricted to running Deep-Networks based on OpenBLAS which runs on the multicore ARM subsystem at low rate while the Nvidia platforms are much more capable and support networks accelerated using CuDNN. In all cases the incumbent platforms consume 5-15W depending on the network being used for inference. The level of integration is also low and they require additional boards such as a Teensy 3.1 and level shifters etc. to control motor drivers and receive inputs from sensors.

Robots based on these platforms used in academia like MIT's Racecar Project [1] often cost thousands of dollars which is clearly beyond the reach of most students and hobbyists. In order to address this need this paper describes the design of a low-power machine-vision and deep-learning module. The EoT platform exposes the underlying deep-learning, machine-vision, motor-control and wireless communications functionality via micropython reducing the barrier to entry for students and hobbyists. The resulting platform is glueless and low-power allowing longer term autonomy without the overhead of fans, multiple power supplies etc.

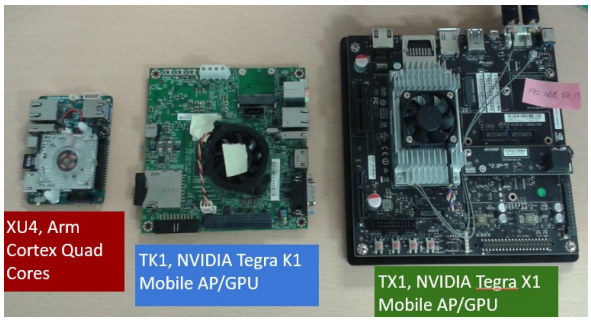

Figure 1 Typical Embedded Robotics Compute Platforms

I. EYES OF THINGS

The Horizon2020 funded Eyes of Things (EoT) project [2] targets always-on embedded vision applications through the development of a low-power wearable form-factor hardware coupled with a custom software framework for visual processing, communications and control.

A. EoT Hardware Platform

The EoT platform consists of a custom designed 35 x 25mm 10-layer high density PCB that is optimised for size and power efficiency. All processing and control is controlled by the low-power Myriad2 MA2150 VPU. The primary visual sensor is a 1x1x1.7mm NanEye2D sensor from Awaiba [3], which is capable of capturing 250x250 pixel images. This is a module package CMOS image sensor and integrated lens that is fully self-timed, consuming less than 5mW at 60 frames per second (fps). Additional peripherals such as a triaxial gyroscope, triaxial accelerometer, magnetometer, and microphone enable a cascade filtering approach for 'interesting' event detection and reaction. Based on the sensor data and the information extracted from visual processing and neural inference, decisions coupled with relevant metadata can be communicated over an integrated WiFi module to external devices or to the cloud. In personal assistant use-cases, audio cues for prompting and notification are enabled via a full onboard audio codec. Integrated level shifters expose motor control pins for direct, yet generic, robotic control (GPIOs, PWMs, I2C, and UART) without the need for external

components. By combining Myriad2, NanEye, low-power board design and energy efficient component selection, the EoT platform can run for up to 24 hours from a fully charged Lithium-ion Polymer (LiPo) battery. USB and micro-SD functionality support rapid development and data logging modalities. A block diagram of the complete system is shown in *Figure 2*.

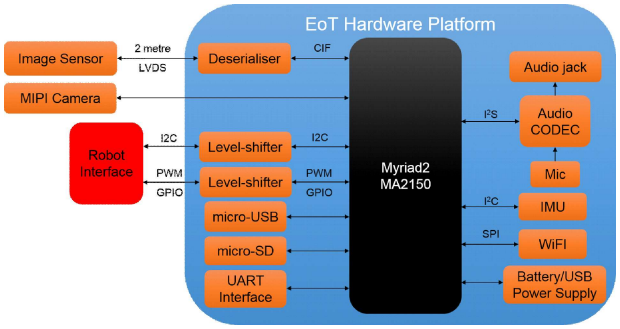

Figure 2 EoT Hardware Platform

B. *Myriad2 Vision Processing Unit (VPU)*

The Myriad2 MA2150 VPU [4] is a heterogeneous, multicore always-on System-on-Chip supporting computational imaging and visual awareness for mobile, wearable, and embedded applications. Also called a Vision Processing Unit (VPU), Myriad2 is based on the proprietary 128-bit SHAVE vector-processor and hardware acceleration pipeline backed by shared multicore memory subsystem and peripherals, and occupies 27mm^2 in 28nm HPM-CMOS. The device has been designed to operate at 0.9V for nominal 600MHz operation, and contains 17 different power-islands coupled with extensive clock-gating under a software API to minimise power-dissipation. The twelve integrated SHAVE processors combined with video and ISP hardware accelerators achieve 1000 GFLOPS (fp16 type) at 600mW including peripherals and 128MB LP DDR2 DRAM die stacked in package. The VPU incorporates parallelism, ISA and microarchitectural features such as multi-ported register-files and native hardware support for sparse data-structures, video hardware accelerators, configurable multicore and multiported memory banks; thus it provides exceptional and highly sustainable performance efficiency across a range of computational imaging and computer vision applications including those with low latency requirements on the order of milliseconds.

II. EoT Platform Software

C. *Fathom Deep-Learning Framework*

Many large companies have invested heavily in machine intelligence for vision and other application applications over the past five years. To date these applications have largely confined to the cloud. Given the difficulties in scaling vision based solutions which include power dissipation, latency, continuity of service, bandwidth and privacy, there is huge interest in migrating such applications in whole or in part to the network edge.

In order to address these needs Movidius has developed a framework, called Fathom, for implementing CNNs of various configurations at ultra-low power targeting the Myriad2 VPU. The Fathom framework accepts xml representations of trained networks from Caffe or TensorFlow as shown in *Figure 3*. Fathom parses these existing and optimally translates them to run on the Myriad2 VPU architecture. For complex networks such as GoogleNet, Fathom enables performance at a nominal 15 inferences/sec with fp16 precision on Myriad2 MA2450 (512MB stacked LPDDR3 die in package).

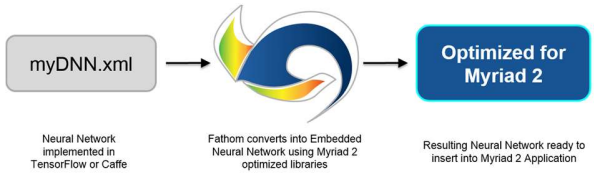

Figure 3 Fathom Deep-Learning Framework

The Fathom framework builds on top of software library (MvTensor) and hardware acceleration support in present and future Myriad devices as shown in *Figure 4*.

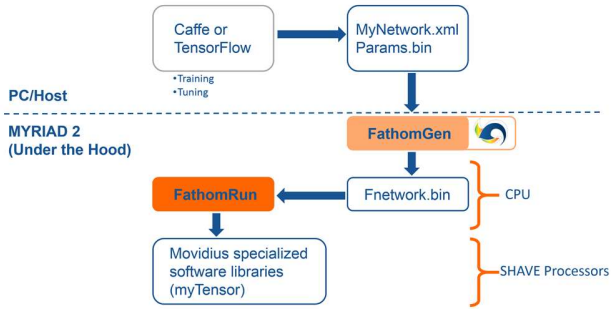

Figure 4 Fathom Framework Details

The mVTensor library is built on top of the mvMatMul library, and MvTensor library provides highly efficient 3D/4D convolution with an initial focus on strided small convolutions required by GoogleNet. The basic concepts and scheduler used in mvMatMul are described in [5]. Building on this foundation the MvTensor library provides highly efficient 3D/4D convolution with an initial focus on strided small convolutions required by GoogleNet.

In order to seed adoption of the Fathom framework and Myriad2 for embedded deep-learning applications Movidius has launched a unique USB-key based accelerator which allows any device with a standard USB port to support deep-learning networks trained in the cloud. The Fathom key enables rapid prototyping, validation, and fine-tuning of neural network models built in standard frameworks.

The Fathom tool flow runs in real-time Fathom USB-key and Myriad 2 EVM, so it can be used to gather detailed per-layer statistics and to validate network models with native hardware precision. Fathom also accelerates DNNs when using online TensorFlow with a standard USB2 or USB3 connection.

D. *Computer Vision Libraries*

While libraries such as OpenCV [6] and libccv [7] have been ported to the Myriad2 platform as part of the EoT framework they have not been performance optimised for the SHAVEs. For high performance and low-power computer vision Movidius provides a proprietary library called MvCV which takes advantage of the various features of the Myriad architecture to achieve maximum performance at minimum power. One of the key optimisations is to operate on stripes or tiles from a frame rather than going round trip to DRAM each time which greatly reduces power and maximizes performance.

The MvCv library has been used across dozens of customer projects and contains hundreds of commonly used CV functions including: a high performance multi-frame point tracker called vTrack, Visual Odometry, HoG, Kalman/ROI, h264 video-codec, SGBM, BLIS, JPEG codec, Image Warp, RANSAC, FREAK and various linear algebra solvers.

E. *Pulga MQTT Broker*

Typical Internet communication protocols are not appropriate for an IoT context. HTTP, for example, is text-oriented, which means that more bits are needed to transfer the same information. EoT communication with the outside world is based on MQTT [8], an open lightweight publish/subscribe protocol based on TCP/IP. Devices that support this protocol can open a connection, maintaining it using very little power while receiving commands with as little as 2 bytes of overhead. While HTTPS is slightly more efficient in terms of establishing connection, MQTT is much more efficient during transmission.

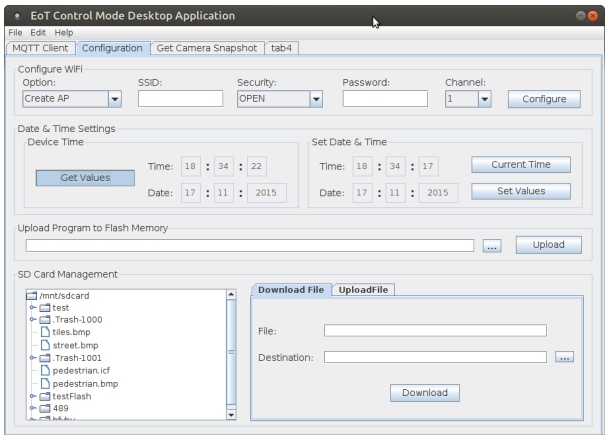

Figure 5 Pulga MQTT client application running on a PC

With MQTT, the classic approach is that an embedded client connects to an MQTT server (broker) in the cloud. In this scenario the server has to be leased by the user, or else it has to be installed on a locally-managed server. In EoT, the device actually implements a broker (called Pulga), which means that an external server is not needed (although it can be used). This approach allows other EoT devices to subscribe to another EoT. Furthermore an external device can be used to send/receive configuration commands to the EoT using MQTT messages. This approach has been followed in EoT, which exposes a number of basic configuration commands to external devices (PC, tablet or smartphone). The following figure shows the client application running on a PC.

F. *Motion-Control API*

In order to communicate with a wide range of devices such as robots, drones, actuators and sensors the Motion-Control API encapsulates the hardware details of the communication into an easy to use library. The API makes it is possible to control small robot cars and flying drones; open and close doors as required; monitor devices such as fridges, vending machines; or control systems like air conditioning and lighting which might not have access to internet or wireless connectivity. The API is a C/C++ library which was initially developed for Arduino, ported to Raspberry Pi and finally ported to Myriad2. The current implementation of the Motion-Control API only provides functions for controlling ground vehicles including: Move forward, Move backward, Tank-turn left, Tank-turn right and Stop. It is planned that wheel and IMU odometry and additional facilities to support a broader range of robots, drones and devices over the longer term.

G. *Micropython*

The Python programming language's flexibility and ease-of-use have made it wildly popular in academia and in general in many realms where speed of development and flexibility is more important than performance. Python is an interpreted language, which means that it is not in principle a good fit for embedded systems. In the past, a number of attempts have been made at porting it to embedded platforms. MicroPython is currently the most successful port [9]. It is a C language port of the Python 3 language. The EoT board supports MicroPython. A REPL system is already available on top of the MQTT communication protocol. This allows sending of instructions and receiving responses remotely, from a PC or tablet, see figure below. Work is currently underway to extend the language to use the EoT hardware capabilities.

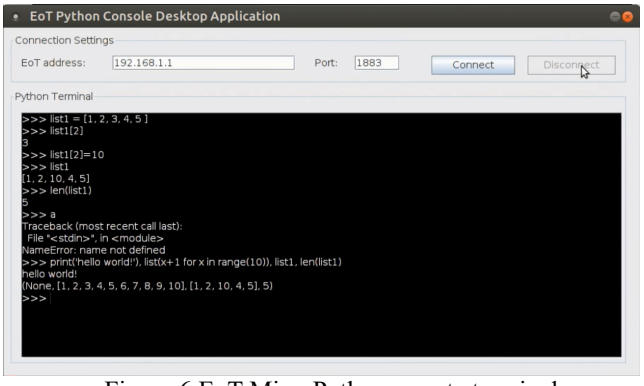

Figure 6 EoT MicroPython remote terminal

III. OPTIMISED ALGORITHMS AND DATA-STRUCTURES

Simultaneous Localisation and Mapping (SLAM) is the process by which a mobile device or robot can construct a map of an environment while simultaneously computing its

location in that map. While previously confined to the field of robotics like the Mars rover, SLAM has more recently been driven by the latest wave AR and VR applications as well as devices like Google's Project Tango [10]. While SLAM systems have existed for some 30 years, SLAM maps have typically been produced offline using post-processing. Enabled by the introduction of low-cost depth cameras [11] there has been an explosion in terms of interest in so-called Dense SLAM algorithms such as KinectFusion [12]. By dense SLAM we mean that the map produced is composed of surfaces rather than sparse point geometry. However, since the introduction of the first project Tango devices the emphasis has moved to real-time SLAM [13].

Incumbent methods for computing dense SLAM include KinectFusion [12], Kintinuous [15] and ElasticFusion [16], all of which are designed for high performance desktop systems and as a result they require much higher FLOPS performance and memory than is practical to integrate in a low-cost wearable device. For instance the portable implementation of KinectFusion known as SLAMbench [18] achieves less that 2fps on an Odroid XU4 processor dissipating 6-7W with a memory requirement of 512Mbytes to store 32-bit TSDFs (Truncated Sign Distance Function) for each voxel in a 512 element on a side cube with 1cm resolution.

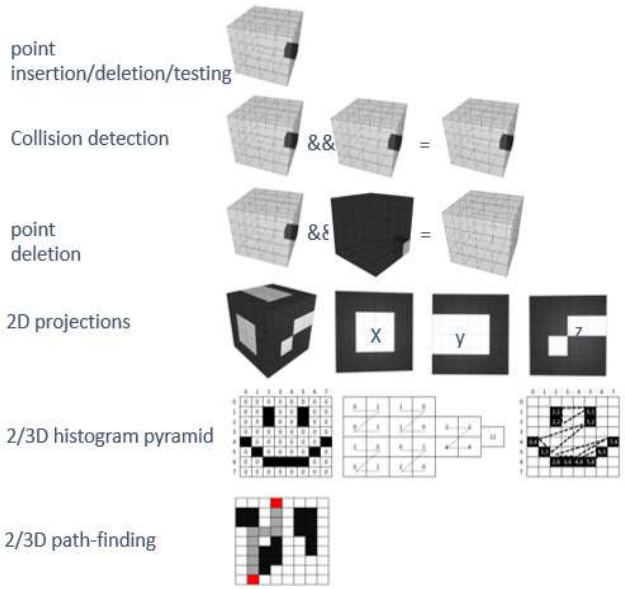

Figure 7 VOLA Acceleration for Dense SLAM Systems

The present work describes the development of a lightweight Octree-based volumetric storage format with hardware acceleration for access methods, {x,y,z} projections, 2 and 3D pathfinding, coarse and fine updates using a parallel data-structure for patching the Octree based on TSDF updates, and the exchange of compact volumetric maps for AR/VR and autonomous vehicles via a wireless channel etc. The basic storage element in the proposed volumetric solution is a 4^3 volume packed into a 64-bit integer (a good match for the internal data-path widths in the Myriad2 VPU load-store ports and shared CMX memory). Using the VOLA format achieves 99.6% compaction when compared with the SLAMbench dense 3D array format. Indeed the format can also be extended to support both graphics and audio rendering on the fly for hybrid volumes of real and virtual components modelled in volumetric format while requiring only 2 bits per voxel in the proposed sparse voxel format as shown in *Figure 7*.

IV. Training Deep Networks For Use In Robots

A major question for deep-learning in robotics is how they can be trained to work reliably in a huge variety of domestic settings and to adapt to their surroundings. The approach proposed here, which is still in its infancy, has been to extend the ICL-NUIM POVray based dataset [14] used in SLAMbench [12] to generate synthetic data, initially focusing on depth from passive stereo with promising results. We believe that synthetic data will play a role in training networks for more reliable operation.

H. Per-pixel Depth-Labelling Network

The network takes in the synthetic stereo images as input for training and testing. The images are captured from left and right eyes of the camera. These 2 images are combined together as a batch acting as input to the network. Therefore the input contains 6 channels, i.e. RGB channels for both images. This network contains 5 convolutional layers, 2 max pooling layers and 4 ReLU layers. The first three convolutional layers contain16, 64 and 256 kernels of size 7*7 respectively. The last two convolutional layers contain 64 and 16 kernels of size 1*1 respectively. In max pooling layers, max pooling of 2*2 is applied to each feature maps achieved from the convolutional layers. After testing 200 different batch of test input, the trained network achieved accuracy around 90%. This means for given unseen stereo images from left and right eyes of the camera, the network can produce an output of depth image based on each pixel of the input with around 90% accuracy. The figure below shows the input and output of the network. Different colour in the output represents different depth information.

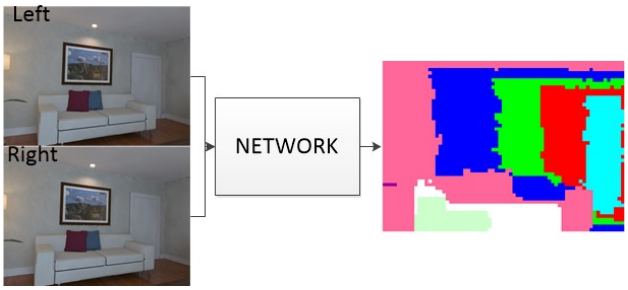

Figure 8 Per-Pixel Depth-Labelling Network

V. Experimental Results

A. K-Bot

This $999 robot from autonomous.ai is based on a Kobuki base [20]. This robot was augmented with extra hardware to capture stereo video and volumetric data. The current configuration is shown in *Figure 9*.

The additional hardware consists of two RGB camera sub-systems implemented using Movidius' MV0182 development boards, containing Myriad2 VPU, daughter cards with mono and stereo image sensors, and communication support hardware for 1Gbit Ethernet and USB 3.0. Video is streamed from the Stereo and Ceiling *Figure 9* Camera sub-systems to the NVIDIA TK1 Linux computer using USB 3.0. There is also an ASUS Xtion 3D sensor (similar to Microsoft's Kinect) connected to the TK1 via the USB port.

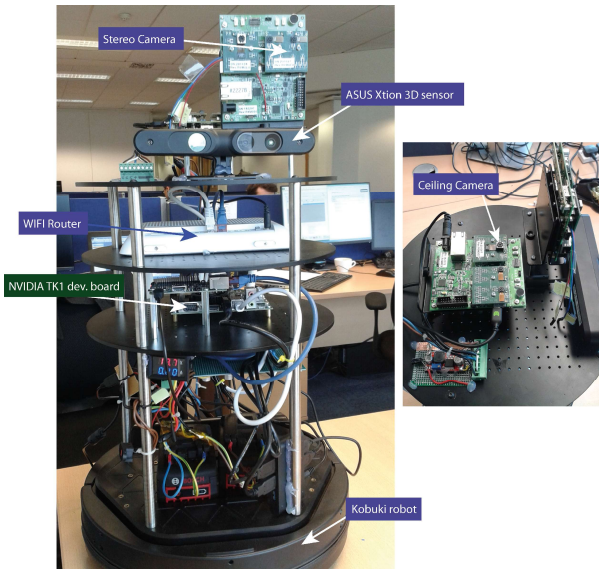

Figure 9 K-Bot robot with Stereo and Volumetric capture

Using depth information from the 3D sensor, dense volumetric maps of the environment are created using KinectFusion [12]. The image on the right of *Figure 10* shows a render of the geometry learnt for the scene in the left image.

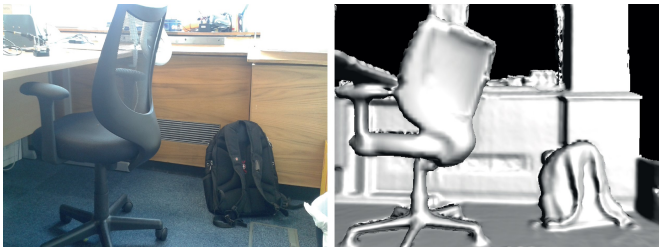

Figure 10 captured scene (left) with learnt geometry (right)

The intention is to shortly replace the TK1 and migrate all functions to Myriad2 as well as using a depth map generated from passive stereo video as the primary input to KinectFusion using the cross-platform SLAMbench port [14]. A Ceiling Camera and associated video processing pipeline tracks specially coded marker on the ceiling for precise determination of the point of origin/arrival. The output of the tracker reports the position of the robot relative to a fixed world reference coordinate frame. The ceiling marker pattern is shown on the left of *Figure 11* and corresponding segmentation on the right.

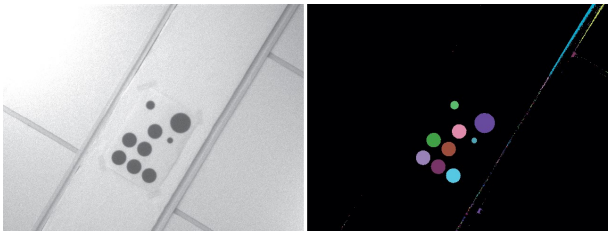

Figure 11 Ceiling marker tracking

B. C-Bot

The DFrobot Cherokey 4WD [21] was selected for this robot as it is low-cost and has a robust aluminium frame. In addition to the motion-control API an Android App was developed to allow the remote control of a ground vehicle via Wi-Fi.

Figure 12 C-Bot

The robot is controlled by four 5V logic signals (ENABLEs for controlling direction of the wheels and PWMs to control the speed) which are compatible with the signal levels in the EoT Motor Control Header. The signal connections the between the Cherokey 4WD and the EoT board is shown in Table 1. Figure 4 shows a schematic of the hardware setup. An Android application (GRApp) provides a user interface for controlling a ground robot using a custom designed protocol, called the Mobile Robot Control (MRC), to communicate between app and the EoT. This protocol uses UDP sockets to send and receive control and status data. The user interface of the app has two modes for controlling a Ground Robot. The first mode uses the touchpad for forward and backward motion and to (tank) turn left or turn right. The second mode uses the phone gyroscope to allow phone orientation to control the robot for one-handed operation.

The vision pipeline for object learning and tracking was implemented on the Myriad2 VPU as shown in *Figure 13*.

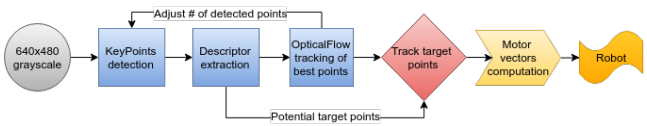

Figure 13 C-Bot Vision Pipeline

A monocular grayscale VGA live feed coming from one of the cameras connected to Myriad2, at a rate of 30Hz is used as the primary input to the vision pipeline. The first step is to detect interest points which is done by applying the Harris

corner detector to each pixel in the image, using a specialized hardware block. Then the points are filtered, only the highest (Harris) scoring points remain. In the second step FREAK descriptors are extracted for all remaining points, which are subsequently used for target learning and tracking. The third step tracks all the points that made it through the selection process using Pyramidal Lucas-Kanade (PLK) Optical Flow. This final step in the tracking pipeline provides feedback to adjust various thresholds and point detection strategies. The output of PLK is fed to the next module which performs target locking and tracking based on the tracked points that fall within a region previously selected by a human user using an application. The tracking module has the capability to reject or accept new points, and is therefore self-managing based on the learned model of the target. Finally the position of the target is input to the motor vector computation equations which aim to smooth the movement and determine the direction and acceleration of the robot.

The vision pipeline runs at 40-100fps on Myriad2 using highly optimized algorithms which consume about 30% of the computing resources on the platform. The frame-rate varies with the environment and was left deliberately floating in order to study the behavior of the robot in various conditions. It was found that even in worst case conditions the tracking be performed with very low latency, enabling the vision-system to keep track of human motion.

VI. CONCLUSIONS

In addition to allowing the creation of low cost robots with advanced computer vision and deep-learning capabilities the EoT robotics platform can also capture datasets for creating algorithms that support the migration to increasingly autonomous behaviour in robots. In addition to this large scale synthetic data-sets have also been created that facilitate training of deep networks to replace traditional computer vision algorithms such as stereo vision.

The hardware system is built with low cost, off the shelf parts while the software system leaves plenty of raw horsepower to run more complex algorithms. However, currently the Cherokey robot places severe limitations on the system due to its mechanical shortcomings. Its weak motors are unable to keep pace with the movement of the target even in a straight line, and worse still the tank-track steering creates strong friction between the tires and flooring such as carpets resulting in the robot not being able to turn fast enough to track targets as they turn.

Considering the mismatch between the latency of the software and the C-bot robot hardware, two approaches could be considered:

1. reduce the frame-rate to better match the Cherokey
2. use a more agile robot

To this end a new robot based on a $29.90 Sinoning robot kit [23] with steerable front wheels is under construction which should overcome the mechanical limitations of the Cherokey-based C-Bot.

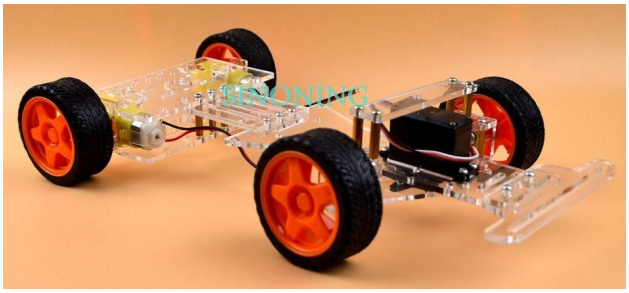

Figure 14 S-Bot

VII. FUTHER WORK

The ultimate goal of the EoT robotics project is to enable the mass-production of advanced and highly programmable robots for developers, students and hobbyists at a sub $100 price-point that includes robot, control electronics, software and an eBook. Additionally work continues to add to the capabilities of the software by adding additional deep-networks, computer vision algorithms and reinforcement learning capabilities.

ACKNOWLEDGMENT

This work has been supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 643924.

REFERENCES

[1] http://sertac.scripts.mit.edu/racecar/wp-content/uploads/2015/01/slides.pdf

[2] http://www.eyesofthings.euhttp://www.eyesofthings.eu/

[3] http://www.cmosis.com/products/product_detail/naneye_module

[4] B. Barry *et al*., "Always-on Vision Processing Unit for Mobile Applications," in *IEEE Micro*, vol. 35, no. 2, pp. 56-66, Mar.-Apr. 2015

[5] M. H. Ionica and D. Gregg, "The Movidius Myriad Architecture's Potential for Scientific Computing," in *IEEE Micro*, vol. 35, no. 1, pp. 6-14, Jan.-Feb. 2015

[6] http://opencv.org/

[7] http://libccv.org/

[8] http://mqtt.org/

[9] https://micropython.org/

[10] http://techcrunch.com/2014/02/20/inside-the-revolutionary-3d-vision-chip-at-the-heart-of-googles-project-tango-phone/

[11] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," in IEEE Multimedia, vol. 19, no. 2, pp. 4-10, Feb. 2012

[12] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon, KinectFusion: Real-Time Dense Surface Mapping and Tracking, in IEEE ISMAR, IEEE, October 2011

[13] Richard A. Newcombe, "Dense Visual SLAM: Greedy Algorithms", http://frc.ri.cmu.edu/~kaess/vslam_cvpr14/media/VSLAM-Tutorial-CVPR14-P14-GreedyDenseSLAM.pdf

[14] http://www.cs.nuim.ie/research/vision/data/rgbd2012/

[15] http://www.thomaswhelan.ie/Whelan15rss.pdf

[16] http://arxiv.org/abs/1410.2167

[17] A. Handa, T. Whelan, J.B. McDonald and A.J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM", IEEE Intl. Conf. on Robotics and Automation, ICRA, Hong Kong, May 2014

[18] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Luján, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber. “Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM”. In IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2015. ArXiv: 1410.2167.

[19] http://research.microsoft.com/en-us/projects/surfacerecon/

[20] https://www.autonomous.ai/deep-learning-robot

[21] http://www.dfrobot.com/wiki/index.php/Cherokey_4WD_Mobile_Platform_(SKU:ROB0102)

[22] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825

[23] https://www.sinoning.com/steering-engine-4-wheel-2-motor-smart-robot-car-chassis-kits-diy-for-arduino-with-futaba-3003.html.html