mobica

# COMPUTER VISION

INTEGRATING TECHNOLOGY WITH THE REAL WORLD

## Abstract

In this white paper, Mobica covers computer vision from the perspective of a software services company. An overview of the type of projects performed by Mobica and some of the lessons learned are provided, followed by a discussion of how Mobica expects the field of computer vision to develop. Mobica see the biggest challenges as coming from two aspects: firstly, there need to be improvements in performance and efficiency; secondly, the functionality needs to be made more easily available to developers.

# Introduction and Technology Overview

**What is computer vision?**

The field of computer vision allows a computer to emulate human vision by showing perception and understanding of the content of an image or video. The computer can then make decisions based on this understanding: for example, detecting the collision of an object coming towards it or identifying parts of an input image such as a face or type of object.

A computer goes through a process of analysing the information from images or videos to produce numerical or symbolic data. This data is then used by accompanying computer programs to make decisions on the data.

Uses for computer vision systems include:

- Object detection and recognition
- Creating a model of an object, for example in medical imaging
- Image processing and filtering
- Navigation, for example for an autonomous vehicle
- Organising image management databases

**Why do we think it is important?**

Computer vision is a key technology in a number of industry sectors such as the automotive industry as part of a driver assistance strategy (ADAS), the medical industry in the application of better diagnosis, treatment and prediction of diseases, and the entertainment industry as an integral part of gaming systems. Use of computer vision can make tasks safer (e.g. lane detection and warning to assist a driver), more efficient (e.g. automation of routine medical procedures) or more pleasurable (e.g. a natural user interface to a gaming system).

**Relevant technologies**

Most developers who have worked with computer vision will have some familiarity with the open source computer vision and machine learning software library called OpenCV (Open Source Computer Vision Library). According to their web site [OCV], “OpenCV was designed for computational efficiency” and is “written in optimized C/C++”. However, our experience leads us to believe that there is a lot of scope for further optimisation when considering specific use-cases on specific hardware platforms.

One way to improve the performance of applications that use computer vision is to offload some of the processing to alternative processing units, such as a GPU. This is often done through APIs such as OpenCL and CUDA; for example, OpenCV contains OpenCL, CUDA and CPU implementations of many algorithms. OpenCL is an “open standard for parallel programming of heterogeneous systems” [OCL] that is maintained by the Khronos Group. Depending on the implementation on the target device, OpenCL can be used as an API to a wide range of devices (including DSPs, CPUs, and FPGAs) but GPUs are generally the most prevalent target processor. In addition to the “Full” OpenCL profile, the

OpenCL specifications define requirements for an “Embedded” profile that is allowed to support, for example, a smaller range of data types. This embedded profile is typically supported by lower power devices - thus, OpenCL implementations of computer vision algorithms that require the full profile will not run on these devices.  CUDA is a proprietary parallel computing platform and programming model invented by NVIDIA [CUDA] that targets the GPU.

Khronos has introduced another level of abstraction with the OpenVX API that is intended to provide access to “Vision Acceleration” [OVX, KHRONOS]. This API allows the creation of directed graphs for vision processing. (In other words, the developer can express their solution as a chain of OpenVX nodes that are functional blocks representing image operations.) The nodes in the graph can be implemented in software or use accelerator hardware, depending on the target device - the implementation detail is hidden behind the API. For example, a node could be implemented using OpenCL to push the work to the GPU. Alternatively, the same node could be implemented on another device by using software running on the CPU or by using dedicated hardware. An application developer could access the OpenVX APIs directly but might also use them indirectly through middleware such as OpenCV.

It should be noted that these technologies can be used on a local device or on a server. There are many use-cases where there is an option to send data to a server for analysis and then receive the results. Using a server to perform work that needs either intensive computation or access to a large database is an approach that can be used to allow computer vision functionality on less capable devices but, of course, this relies on a connection to the server from the device. An example of this would be object recognition on a mobile phone, where it does not make sense or is not feasible to store and maintain a large database of objects on the handset. In this example, to reduce the amount of data being sent over the network, it might be sensible to do some processing on the handset so that a small amount of metadata can be sent to the server rather than complete images. The server can then compare the metadata against its large database of objects and return a result to the handset. The stage at which the processing is handed over between the handset and the server would be application-specific and the handover is likely to use a proprietary protocol.

## Mobica’s Work with Computer Vision

Mobica has a long history of working with graphics; this stretches from simple devices where a framebuffer is solely managed by the CPU, through 2D graphics accelerators, to fully-fledged modern GPUs. We work with IP vendors (e.g. ARM and Imagination Technologies), silicon manufacturers (e.g. Intel and NVIDIA), standards organisations (e.g. Khronos) and device manufacturers on graphics drivers and related software. Mobica has found that many of the skills used in graphics acceleration are transferrable to computer vision.

Solutions to computer vision problems (object detection, image stitching, augmented reality, etc.) can usually be considered as a pipeline, with a set of discrete stages that move from the raw image data to useful information. A generic pipeline is illustrated in Figure 1, below.  In some cases, Mobica engineers have found it helpful to view computer vision as the inverse of the graphics rendering pipeline. Graphics tends to work from a model through to a rendered image, whereas computer vision starts with an image and often ends with some form of model.

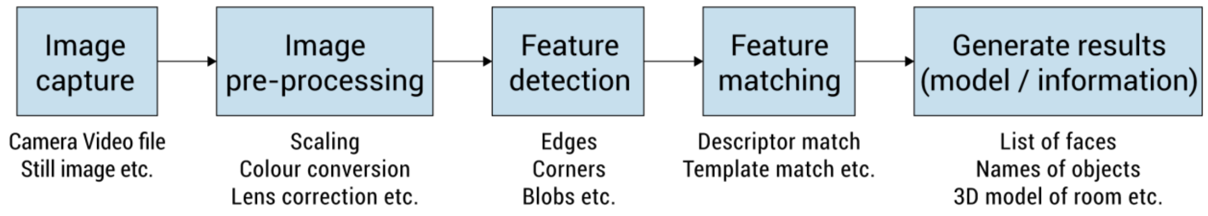

*Figure 1: A generic computer vision pipeline*

The symmetry that Mobica has observed in use-cases such as augmented reality applications is illustrated in Figure 2, below. As the data passes through the pipeline, from left to right, the volume of data (as represented by the width of the shaded area) is reduced from an input image through to a model. Early stages in the computer vision side of the pipeline are likely to be image processing operations, probably including resampling to a lower resolution, followed by identification of points of interest. As well as involving large amounts of data, the early stages of computer vision typically involve per-pixel calculations that only make use of nearby pixels; just like the final stages of the graphics, this allows for high levels of parallel processing and thus is ideal for accelerators.

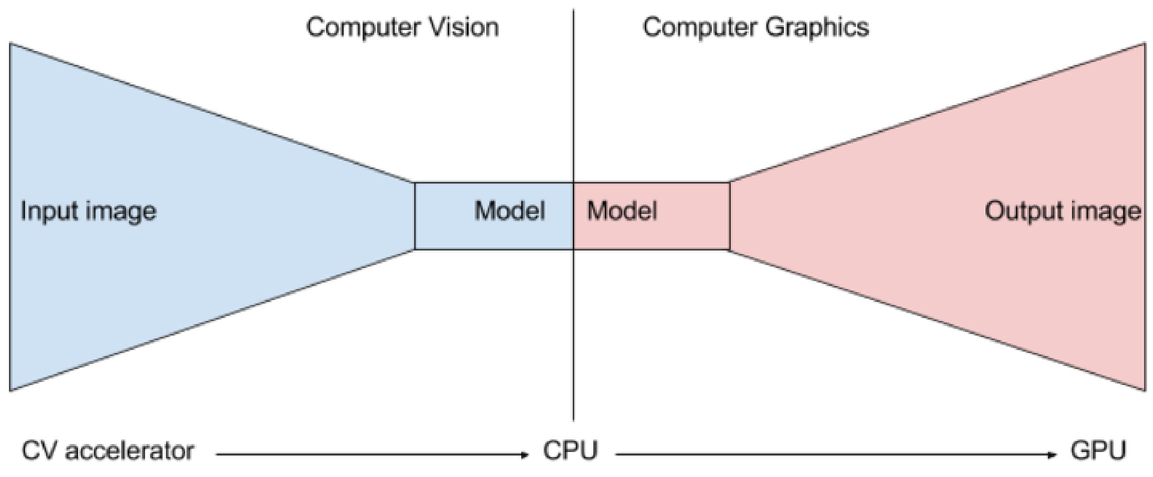

*Figure 2: A conceptual representation of an augmented reality pipeline, illustrating the volume of data at each stage.*

Typical computer vision projects executed by Mobica either involve using existing computer vision solutions as part of a product or involve optimisation of existing computer vision solutions. This includes “blue-skies” research, which is normally performed within academia or within the research divisions of large corporations and is rarely outsourced. However, our experience in this field has led to our services being used in several research projects for a variety of customers.

- Implementation and optimisation of computer vision algorithms using OpenCL and NEON on Android-based tablets. Profiling of performance and CPU off-loading Comparison of the OpenCL and NEON versions with the OpenCV CPU implementations
- For a leading automotive chipset maker, the implementation of an object recognition demo to demonstrate the capability of their latest (currently unreleased) hardware
- Re-implementation of selected OpenCV computer vision algorithms using OpenCL for a major mobile handset vendor. This included face detection, object tracking and image processing
- Evaluation of various GPU-accelerated Augmented Reality (AR) & Visual Computing frameworks. Studying the efficiency of running AR tasks on the GPU, to identify which CV kernels, from the detection-tracking pipelines, are better suited for GPU acceleration
- Investigative study for an OSGi Gateway to monitor output from connected devices (counting people), perform pre-processing/aggregation on the data and publish the results to M2M integration
- Implementation and optimisation of an image stitching application for Android using OpenCV for a tablet vendor
- Working with a Programmable Image Signal Processor (ISP) vendor to create a proof of concept where tasks such as face detection are performed within the ISP and a stream of metadata is passed to the CPU in parallel with the image data from the camera

- Working on a proof of concept of a vending machine with a gesture-based user interface. This uses a camera to provide user input, demonstrating a real-life application of Intel’s RealSense technology to provide computer vision
- Gesture recognition backend development to develop a portable library that provides hand gesture recognition using a web camera. It is implemented in C++ using many routines from the OpenCV library. The created solution has two main modules:
    - Hand image segmentation - This module extracts a binary image of a hand from a webcam image. It performs background subtraction followed by contour and feature points finding before using a statistical approach for segmenting out the hand (palm) from the rest of the body
    - Gesture classification - The classification module uses the segmented hand image to decide which gesture is currently being performed. As a pre-processing step, rotation, position and scale of the hand segment is calculated and the segment is transformed to a normalized pose. The classifier itself utilizes the OpenCV machine learning framework. It is trained on a set of normalized hand segments and takes the same type of data as input for the classification

## Optimisation

As indicated above, the most common computer vision projects at Mobica are those that involve optimisation of existing code. The starting point may be open source code, such as OpenCV, but could also start from our customer's proprietary code.

Optimisation can be just a matter of improving the performance of C or C++ code but even a CPU-only solution can potentially show large gains when there is scope to use vector operations such as those provided by NEON or SSE, albeit at the cost of increased complexity. These CPU vector operation instruction sets operate on one-dimensional arrays of data in a "Single Instruction, Multiple Data" (SIMD) way, performing the same operation on multiple data items in parallel. This allows us to introduce a limited amount of parallel processing without needing to share the data with a different compute unit.

Many computer vision algorithms result in large numbers of iterations across a data set; small optimisations to the most frequently used sections of code can give significant cumulative performance gains. It is very rare to need to handcraft assembler but we often inspect the intermediate representations to check that compilers are doing what we expect.

Moving away from the CPU to offload work to some form of accelerator can provide many more options. Mobica projects using computer vision accelerators have included the use of:

- FPGAs
- Programmable Image Signal Processors
- GPU via OpenCL

An accelerator has been used in these projects for one or more of a number of reasons:

- Performance - the accelerator may be able to do the task faster than the CPU
- Power consumption - the accelerator may be able to do the task using less power than the CPU
- CPU bandwidth - the CPU may be overloaded and offloading tasks to other processors will help

## Memory bandwidth

A recurring theme that Mobica has observed in both graphics and computer vision is that memory bandwidth becomes the limiting factor. Computer vision is far more efficient if accelerators have direct access to the data from sensors. This allows the accelerators to deal with the raw data and only send processed data to the CPU, typically reducing the volume of data that hits the main memory bus.

One of the more frustrating experiences for Mobica in the field of computer vision is where certain stages of pipelines are ideal for offloading to another processor (accelerator) but that stages in between require CPU processing. There has often been a significant overhead in handing over tasks between the CPU and an accelerator; this means that on current systems it may only be worth using the accelerator if a large and continuous portion of the pipeline can be executed on the accelerator. This is changing as improvements are made in the mechanisms for passing work back and forth between CPUs and accelerators, such as those offered by the standards and software provided by the Heterogeneous System Architecture Foundation [HSA].

## The Future of Computer Vision

One way to predict the future of computer vision is by looking at the relationship between the suppliers of Computer Vision solutions (hardware and/or software) and the manufacturers of devices that use Computer Vision. Mobica believes that device manufacturers will always want to avoid locking themselves in to a particular supplier; this leads to standardisation of the interfaces to the computer vision pipeline. We expect to see greater use of standard interfaces, such as those specified by Khronos, between each stage of the common use-cases.

The APIs will continue to evolve to meet the needs of developers. For example, a standard computer vision API at a higher level than OpenVX could help open up the use of computer vision functionality for those who do not have domain expertise. The "Mobile Vision API" from Google [MOBVIS] is an example of such an API but it only supports face and barcode detection and is not intended as a standard. Such an API would co-exist with (and probably make use of) OpenVX.

Computer vision processing needs to become both faster and less power hungry, particularly for mobile device applications. Mobica believes that this is likely to be solved through dedicated accelerators and improvements in the way that CPUs and accelerators work together. These accelerators will have direct access to the data from sensors, with only the processed data hitting the main memory bus and the CPU. As examples, the Myriad 2 "Vision Processing Unit" from Movidius [MYRIAD] already allows programmable vision processing and various high-end ISP devices provide features such as face detection as fixed-function IP blocks. However, these accelerators use custom data structures to share metadata with the CPU rather than adhering to some standard format.

Mobica believes that server-side processing for computer vision is likely to be restricted to object recognition and similar cases, where small amounts of metadata are sent from device to server and then compared against large databases. Most computer vision processing will be performed locally, to reduce latency and dependence on the network, but it is not practical to store and maintain large databases on many types of device.

It is clear that some applications of computer vision will be classed as "safety critical", particularly in the automotive and medical fields. Khronos already have a safety critical version of OpenGL. The Khronos Safety Critical working group is developing open compute acceleration standards for markets such as avionics and automotive displays [GLSC]. This leads us to expect safety critical versions of Vulkan and OpenVX in the near future. (The Vulkan API offers both graphics and compute APIs. We expect Vulkan to be used instead of OpenCL in safety critical applications.) It is typical for safety critical versions to be at least a generation behind the cutting edge, so it is reasonable to expect the safety critical versions of Vulkan and OpenVX to offer a subset of the functionality of the current Vulkan and OpenVX APIs.

## Conclusions

Computer vision is an exciting and fast developing area of Computer Science, with the most advanced applications of this technology being in automotive (ADAS and autonomous cars) and medical applications. We are also seeing this technology in everyday common applications, such as counting people walking into a department store and general image processing, for example on social media websites etc.

The application of this technology is already saving lives, making people and organisations more productive and creating new entertainment experiences. The continuing development of faster, more efficient hardware platforms, as described above, means that current applications are only scratching the surface of the technology's capabilities. Mobica are well positioned as experts in the component technologies of computer vision, underpinned by its foundation in graphics technology, to assist in the continued development of applications and develop computer vision platform technology. We are excited to be able to further contribute towards the growth of this technology, by developing new and existing partnerships with some of the world's cutting edge companies.

## References

| Reference | Title | Author | Date |
|---|---|---|---|
| [CUDA] | NVIDIA web site (http://www.nvidia.com/object/cuda_home_new.html) | NVIDIA | Retrieved May 2016 |
| [GLSC] | Khronos Safety Critical working group web site (https://www.khronos.org/openglsc/) | Khronos | Retrieved May 2016 |
| [HSA] | Heterogeneous System Architecture (HSA) Foundation web site (http://www.hsafoundation.com/) | HSA Foundation | Retrieved May 2016 |
| [KHRONOS] | Khronos OpenVX 1.1 press release (https://www.khronos.org/news/press/releases/khronos-releases-opengl-4.3-specification-with-major-enhancements) | Khronos | May 2nd 2016 |
| [MOBVIS] | Introduction to Mobile Vision web site (https://developers.google.com/vision/introduction) | Google | Retrieved May 2016 |
| [MYRIAD] | Movidius Vision Processing Unit web site (http://www.movidius.com/solutions/vision-processing-unit) | Movidius | Retrieved May 2016 |
| [OCL] | OpenCL web site (https://www.khronos.org/opencl/) | Khronos | Retrieved April 2016 |
| [OCV] | OpenCV web site (http://opencv.org/) | OpenCV | Retrieved April 2016 |
| [OVX] | OpenVX web site (https://www.khronos.org/openvx/) | Khronos | Retrieved April 2016 |

## About Mobica

Mobica is a world-leading software services company who specialises in working with leading technology providers in the connected device market to bring next generation, time critical products to life. We do this through technology consulting, bespoke software engineering and a wide range of services including UX, UI and QA. Mobica operates in a host of sectors including; Automotive, Mobile, Semiconductor, Fintech, TV and Broadcasting and Marine, Aviation and Defence. Many of our solutions are used in everyday life, from smartphones and TVs, to cars, intelligent buildings and the internet of things.

To find out more about Mobica, visit us at www.mobica.com or contact sales@mobica.com

## List of Contributors

- Sean Jesson – Author
- John Ward – Co-Author
- Jim Carroll – Co-Author

## Disclaimer

## Trademarks