Low-cost embedded hardware for computer vision

06 April 2021, 02:00

With the massive progress made in computer vision and the easy availability of cheap computational power in recent years, the huge demand for smart solutions to be delivered on small single board computers could be easily met. More and more, computer vision algorithms are being deployed for vision on a variety of edge use cases such as drones, security cameras, mobile applications, retail analytics, ...

Recently, while building a proof of concept for a company, our goal was to detect certain objects in real-time.

Proof of concept

In the proof-of-concept set-up, a camera continuously captures images and feeds them to a mini computer, in this case a Raspberry Pi 4. This minicomputer processes the images using the Python library OpenCV. To begin with, we started to detect Lego blocks of a particular colour, say red.

We used Raspberry Pi 4 as the main processing unit, which is a low-cost, lightweight, credit-card-sized, open-source hardware platform with the capacity of functioning as a full-fledged computer with a powerful CPU and many more useful functionalities. This makes it perfect for mobile or other applications, where size, weight, cost are crucial. The camera used in this set-up for image capture is Raspberry Pi Camera v2, a low-cost and high quality 8-megapixel Sony image sensor,

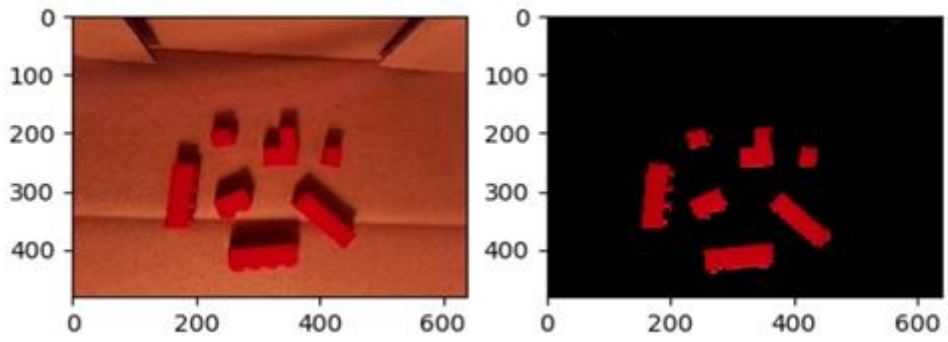
capable of 3,280 x 2,464 pixel static images and 1080p video as well.

This set-up achieved the requirement of continuously capturing images and processing them at 100 fps (frames per second). We could capture the video at 100 fps using the Pi camera very easily. Also image processing could easily be achieved at the right speed. By evaluating the speed of both processes separately, one would conclude that the hardware should be able to run perform both processes combined at the required speed. However, when we also had to segment out the Lego blocks from each captured frame, our setup initially failed to attain the processing speed of 100fps. Instead it reduced to 30 fps.

Profiling and multithreading

Performance is critical in most real-time systems. As any programmer knows, finding the causes of poor performance can be a difficult and time-consuming task. Often, the problem is easy to fix, if and when the root cause is discovered. **Profiling** is the act of monitoring an application at various levels of running to understand where resources are being utilised. This is done among others by determining when a method is executed and how long the method is taking to do a particular task. After profiling our application to determine the time each function takes, we adopted **multithreading**. This method improves the execution performance of a process by the use of concurrency, that is allowing more than one thread to run independently of each other within that program. The main thread was capturing the images at 100 fps, while a second thread was processing the captured images at 180 fps. Eventually after synchronisation of both threads, our set-up was successful in achieving both capturing and processing at 100 fps.

The segmented output of the Lego block detection algorithm running on Raspberry Pi:



Let’s end with the following two statements:

1. Never underestimate the capabilities of such cost-effective systems for real-time intelligent applications.
2. It is possible to achieve high-performance efficiency by adopting conventional programming techniques, rather than just blindly investing in expensive hardware.

Struggling with similar issues? Contact us!

(Source picture above: https://www.dreamstime.com)