Introduction: Today’s demand on computing power can no longer be satisfied by just increasing the clock speed of CPUs. Hence, the approach to parallelise things has become more and more popular these days. NVIDIA provides a technology called CUDA which is an extension to the C programming language. This technology allows programmers to do computations on a CUDA-capable graphics card in a massively parallel way, using thousands of concurrent threads. Image processing algorithms are often quite complex, because computers are not capable of logically interpreting and extracting information as seen by human beings and therefore often need a lot of steps to manage computer vision tasks. Due to this, algorithms can become very computationally intensive and time consuming. Therefore, the idea to use graphical processing units (GPU) to parallelise complex image processing algorithms is quite natural, but highly nontrivial. The performance results need to be analysed in detail to get a good understanding of when CUDA implementations make sense.

Approach / Technologies: Initially, we selected two different computer vision algorithms and tested their suitability for a parallel CUDA implementation. We decided to focus on the Hough transform, an algorithm to detect lines in an image and a template-matching algorithm to find a given pattern in a search image. In a following step, we made performance measurements of these algorithms on the CPU using reference implementations from the OpenCV library. To make the transition from the sequential CPU implementation to a parallel GPU version, we analysed the algorithms and searched for parts which can possibly be parallelised. Afterwards, we developed the algorithms in CUDA in an iterative process of implementation, performance analysis and performance tuning. Finally, we compared the performance measurements of the GPU implementations with their CPU references and interpreted the outcomes obtained.

Result: The results show tremendous performance gains by using CUDA. The two versions of Hough Transform implemented both performed better than their CPU counterparts. An even better result could be observed with the template-matching implementation as we achieved a performance speedup of a factor of 31. Besides the performance evaluation, we gained a lot of experience on how to develop CUDA applications and learned some important points that have to be observed when dealing with GPU programming.