Problem 0

The researcher’s challenge

From a very high-level perspective, developing a hand gesture control system for human-computer interaction generally boils down to designing an efficient and robust pipeline: detect gestures from raw RGB data and map them to some sort of an application event. Nonetheless, depending on the target platform (computer, mobile, custom embedded device) and on the embodiment of the system (a standalone application or an SDK), numerous design choices and even some simplifying assumptions must or can be considered in order to ensure an optimal development process. However, there are several inherent challenges for such a task, regardless of a particular implementation. Some of these include:

Noisy data - camera sensors are not perfect; different types of noise (e.g. likely gaussian noise) and artifacts can appear in the acquisition step and, unless accounted for, they highly affect the quality of the later stages (e.g. detection / recognition)

Real-time - in order to deliver a smooth experience, the system must be able to perform acquisition, detection, tracking and to offer useful output in *almost* real-time (or with very little delay[1]). This is difficult to achieve, considering that many factors do not directly concern the machine learning / computer vision algorithms, for instance, the threading model, which comes with the hassle of synchronisation.

Robustness - bad lighting, high or low contrast (especially during night time), scale (i.e. distance), orientation, losing the tracked target etc are several factors that directly affect the accuracy and the precision of algorithms.

There are several approaches to this task, some of which do not include *deep learning* algorithms, and mainly rely on *classic* computer vision methods, such as filtering, thresholding, contour extraction, convex hull, convexity defects (which determine the gesture), SIFT features, Deformable Part Model (hand-crafted features) etc. The benefit of this approach is that it can be seen as a controlled (debuggable) chain of transformations, it

[1] Imperceptible to / not distracting the user

can run on CPU, and achieves decent results under certain conditions (e.g. plain background), but it might be harder to design (accounting for many corner-cases) and maintain (components might become tightly coupled).

On the other hand, a gesture recognition system can be implemented as an end-to-end (deep) learning process from video, that is, mapping raw pixels to gestures (SwipeLeft, RotateClockwise etc). This approach is more straightforward, allows for various lines of research, but it requires loads of annotated frames, it may take a long time to train, and it's more computationally expensive than the *classical* approach (it must definitely run on a GPU), so it's not a viable option for mobile / embedded devices.[2]

For an initial prototype, some use cases / scenarios must be devised. As mentioned in the beginning, this will aid and guide the development. This is the answer to the *What do we want to achieve?* question. Next, the main architecture must be constructed, starting from a high-level (i.e. components and data flow) point-of-view, up to certain algorithmic details (e.g. what deformable parts to use, what activation function to use etc). Implementations details (such as platform-dependent aspects, low-level optimizations techniques) are also extremely important and must not be neglected - for example, *How is the computer's load when nobody is using the system? Does the system continuously scan for hands? How is the system triggered?* These are genuine, real-world questions for which we must give an answer even for the early prototypes, because *Ok, it works, but it makes the system unusable since this process eats all the CPU cycles.* - is not so good. In addition to that, for an initial prototype, the user interface is not so important, the performance is what makes a good first impression.

A prototype should also allow for easy development on top of what's currently implemented. Therefore, the initial prototype is a good exercise of respecting the architecture, and not just bodging things together. Furthermore, the prototype must permit to easily change components, in case some groundbreaking research result is found.

[2] Of course, lightweight architectures can be designed and deployed for such devices.