

GPU Accelerated Contactless Human Machine Interface for Driving Car

Frédéric Magoulès*, Qimeng Zou*

**CentraleSupélec, Université Paris-Saclay, France*

Email: frederic.magoules@hotmail.com

Abstract—In this paper we present an original contactless human machine interface for driving car. The proposed framework is based on the image sent by a simple camera device, which is then processed by various computer vision algorithms. These algorithms allow the isolation of the user's hand on the camera frame and translate its movements into orders sent to the computer in a real time process. The optimization of the implemented algorithms on graphics processing unit leads to real time interaction between the user, the computer and the machine. The user can easily modify or create the interfaces displayed by the proposed framework to fit his personnel needs. A contactless driving car interface is here produced to illustrate the principle of our framework.

Keywords—computer vision, gesture analysis, video analysis, image processing, machine learning, parallel computing, graphics processing unit

I. INTRODUCTION

Nowadays, there is not a domain where people don't try to remove wires from computer devices. So, controlling machines at distance, without any physical contact between the human and any object in communication with the machine, is more than what we can hope. Reviews on human machine interaction based on computer vision can be found in [1], [2], [3]. Within these systems, contactless human machine have been developed such as hand gloves [4] or other tracking devices for human motion capture. Next to wireless mouses and keyboards, technologies such as the virtual keyboard or the WII appeared. Commercialized at the end of the year 2006, the Nintendo' console belongs to the new generation of human-machine interface. The gamepad is the real innovation of this console. Called Wiimote by Nintendo, the gamepad is indeed like a remote control, connected to the console by the Bluetooth technology. But the Wiimote is more than a simple gamepad. Thanks to an accelerometer, it is able to capture the player's hand's movements: left-right, up-down, forward-backward, rotation, etc. It also incorporates a pointer device, like a computer mouse. And, of course, it can also be used as a normal gamepad when it is horizontally handed. The console offers other high-tech functionalities, while being compatible with former consoles (such as NES, SuperNES, Nintendo64, etc.). It gives, between other things, the possibility of playing directly online. This console is mainly destined for entertainment, and behind the gaming aspect, it brings the possibility of making sport without going out one's room, with the WII Sports (Box, Bowling, Golf, Tennis, BaseBall, etc).

With one of our former paper, we have presented previously a framework where the number and the cost of equipment are significantly reduced: only a basic camera is needed, although the same functionalities are available, while opening new perspectives. This framework is an open source framework that enables contactless human machine interaction using computer vision techniques. This framework enables to implement any algorithm requiring a human-machine interaction, without any physical contact. It is developed in C++ and is written in an object oriented approach. Open-MP library and MPI library are used for parallel computations on CPU, and CUDA and OpenCL languages are used for parallel computing on GPU. OpenCV is used for the image processing kernel.

In this paper we present how we have extended this framework for contactless human machine interface for driving car with graphics processing unit accelerated computations. The plan of the paper is the following. Section II presents the architecture of the framework. Section III discusses some implementations issues of the algorithms. Section V illustrate an experiment for the driving. Finally, Section V concludes this paper.

II. ARCHITECTURE

Our framework works in two steps. The first one consists in the initialization. An XML file is read, containing the configuration to use. Then, the FIZI module learns the background without the user, by using some advanced machine learning techniques. Finally, the interface XML file is loaded. The second step consists in the running one. The camera gets an image, that is treated to deal with the hand's movement. A first treatment is applied to the image so that it can be treated by the program. The brightness is also corrected. Then, the FIZI module can isolate the skin color zones, and more particularly, the hand's one. This hand zone is converted into a pointer by the Mouse module. The Interface module can use this pointer to get its movement, its state (click or not). After this step, it applies the actions associated via the Engine module. These four modules are shortly described in the following paragraphs: FIZI module, Mouse module, Interface module and Engine module.

FIZI module: Its aim is to isolate the skin color zones. Therefore, it implements the settings of the luminance and the contour detection parameters, which are easily adjustable to the functioning conditions. After learning the background at the starting of the program, it can isolate a hand.

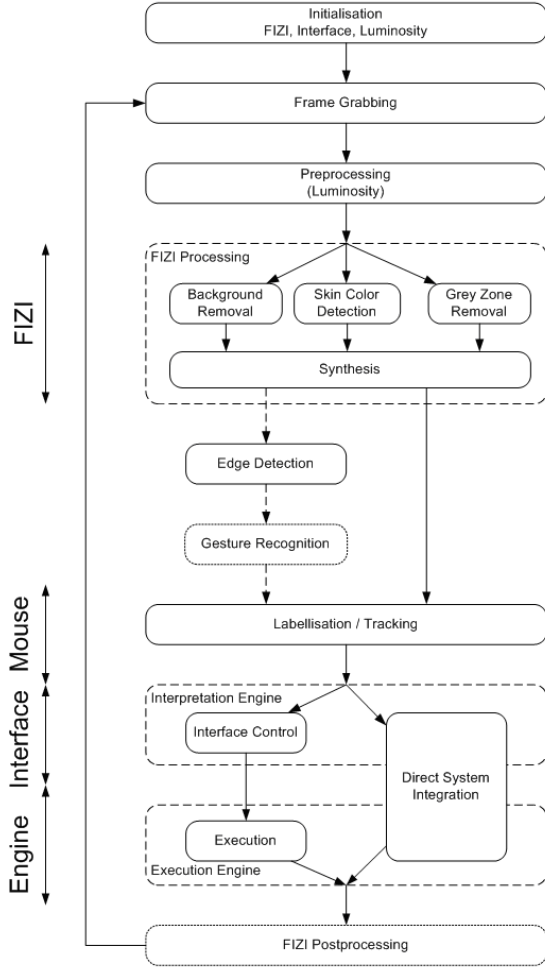


Figure 1. Architecture of the framework.

Mouse module: Getting back the binary image from the FIZI module (skin color zones or not), this module tracks the hand corresponding zone.

Interface Module: On the two previously mentioned modules, can be added the Interface module of which the role is to load a XML configuration file. This file implements a graphic interface in accordance to the wanted application, linking zones to events. With the Mouse module, it determines the interface zone activated by the pointer. The interface is based on the OpenCV library which makes it independent from the Operating System on which it runs.

Engine module: That last module is able to interpret the actions launched by the Interface module, in order to launch system's actions. Opposite to the Interface module, the module is OS dependent.

Workflow: A complete description of the workflow is illustrated in Figure 1. Our framework can be adapted for any particular application. The simplest way is to modify only the XML file. But this enables only to adapt the graphical interface and to launch only some basic

actions on some events defined with the interface. The other way enables to implement more complex algorithm while modifying directly the main function.

III. IMPLEMENTATION

1) *Building the mask of the hand:* To build the mask efficiently with respect to the need of low computational time, FIZI runs three different images in parallel. These three images are the same image captured by the camera but in different color spaces: (i) An RGB image (I_1) is used to remove the background using the learned parameters. (ii) Another RGB image (I_2) is used to remove the gray zones. (iii) An HSV image (I_3) is used to detect and to segment the skin zones in the image. For each of the three images, the processing corresponds to a basic threshold processing, as follow.

(i) In image I_1 , pixels which are between the minimum and the maximum values learned in the initialization step are removed and all the others are kept. The resulting image I_{R_1} is a binary image where white pixels represent the foreground and black pixels the background. This corresponds to the values, $I_{R_1}(x, y) = 1$ if $\min(I_{bg_{\min}}(x, y)) < I_1(x, y) < \max(I_{bg_{\max}}(x, y))$, $I_{R_1}(x, y) = 0$ otherwise.

(ii) In image I_2 , an interactive threshold operator is applied. For each pixel (x, y) of the image I_2 , if the difference between the minimum \min_{RGB} and the maximum \max_{RGB} values of the RGB components are lower than a tolerance S defined by the user, the pixel is considered as a gray color pixel and thus is eliminated from the mask. Indeed with high or low luminosity such pixels are often seen as undefined since their colors are arbitrarily set by OpenCV to zero value, i.e., red color, which unfortunately is included in the range of the skin's colors of European people. This corresponds to the values, $I_{R_2}(x, y) = 1$ if $|\min_{RGB}(I_2(x, y)) - \max_{RGB}(I_2(x, y))| < S$, $I_{R_2}(x, y) = 0$ otherwise.

(iii) In image I_3 , by using the color value in the HSL color space, the zones with color close to the hand's one can be isolated. These pixels represent a mask of skin zones. For the skin segmentation, two thresholds α_1, α_2 have to be defined on the Hue space as illustrated in Figure 2. Defaults values have been learn from experiments but the end user can interactively modified them for his convenience and for the customization of the system. These two thresholds are then used to extract the skin zones in images in the following way. For each pixel (x, y) of I_3 , this corresponds to the values: $I_{R_3}(x, y) = 1$ if $Hue(I_3(x, y)) \in [\alpha_1, \alpha_2] \% 2\pi$, $I_{R_3}(x, y) = 0$ otherwise.

At last, a merging step is processed to build the output meaningful result image. The resulting mask is obtained by applying a logical AND operator between the three images. Some morphological operations, combining erosion and dilatation operators are applied to remove the small noisy objects and to connect to neighborhood zones. As the output of FIZI, an image I_{FIZI} represents the set of the hand zones of the end user.

2) *Managing Luminosity:* It is the biggest issue of the framework, since thresholds depends on the luminosity of

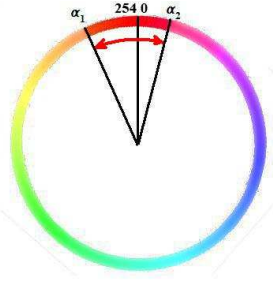


Figure 2. Threshold values on the Hue space.

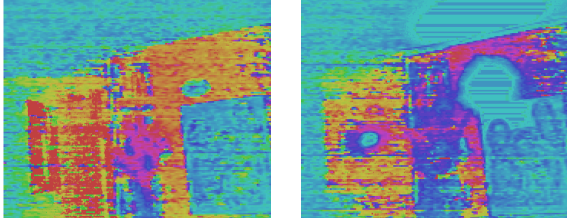


Figure 3. Hand's color upon the luminosity. On the left, the hand looks violet and on the right two minutes after, the same hand looks blue.

the image. While the application is running, the external conditions are subject to change (dawn or simply clouds modify the quantity of light received by the camera). Since most camera adjust automatically, this leads to brutal changes within exposure time, in luminosity and colors. To correct this we can apply an adaptive threshold or modify the dependent values consequently, as illustrated in Figure 3.

Anyway, the automatic customization can trigger discontinuities in the luminosity leading to mistakes in the hand detection. Luminosity raises issues in the process of the hand isolation by shifting the range of values in the different steps. For instance, in a over-exposed environment, the hand is seen by the camera as a white zone but gray zones are removed during the process and thus the hand is not found. A filter has been introduced to modify the Luminosity component in the most obvious cases, as shown in Figure 4 (on the left, the image grabbed from the camera, on the right the image which can then be used as an input for the process).

3) *Acceleration with Graphics Processing Unit*: In addition of the existing modules, mathematical operations and linear systems solvers are implemented by Alinea (An Advanced Linear Algebra) library [5]. The Alinea library

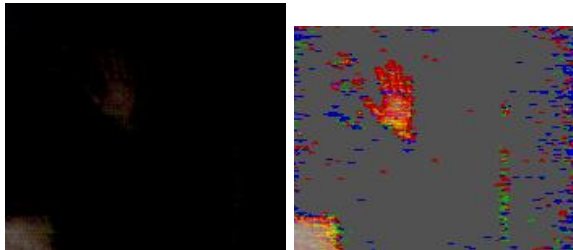


Figure 4. Luminosity treatment.



Figure 5. Contactless interface for driving car.

is implemented in C++, MPI, CUDA and OpenCL. For both central processing unit and graphic processing unit devices, there are different matrix storage formats [6], [7], and real and complex arithmetics in single- and double-precision. It includes several linear algebra operations and numerous algorithms such as iterative methods [8], [9], domain decomposition methods in space [10], together with some energy consumption optimization [11].

It is well known that, dynamically reconfigurable vision-based user interfaces [12] is a key issue to ensure efficient use of any framework. In our framework, dealing with the evolution of the luminosity implies sometimes to re-initiate partly the machine learning techniques which improve the quality of the background removal [13], [14]. Autotuning of the GPU memory is thus implemented [15], [16], [17], [18] to improve the memory size and computation speed dynamically.

IV. APPLICATION FOR DRIVING CAR

Everybody has at least once played with a car driving game on the computer, probably with a console. The algorithm behind mainly consists of controlling three objects: the left direction, the right direction and the speed of the car. Within our framework, the movements of the hand consists on a rotation around an imaginary wheel. To ease this rotation of the hand, an image of this wheel is added on the captured image by the camera to the screen, so the users knows where are his hand on the virtual wheel. Figure 5 illustrates this interface.

V. CONCLUSIONS

In this paper an original approach for driving car has been presented. The proposed approach achieves a virtual wheel using an image acquisition device, for instance a camera. Interaction of the user with the computer without physical contact by the only mean of a camera has been proposed. Accurate luminosity detection and acceleration of the computations with graphics processing unit, make the proposed framework, a robust, accurate and real time human machine interface. Application for driving car has been illustrated.

ACKNOWLEDGEMENTS

The authors acknowledge the numerous students from Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy (France) and from Ecole Centrale Paris (France) who have contributed to this framework since 2003, and in particular N. Vienne, J. Tavernier, the main programmers of the image processing workflow, J. Petin, N. Lambolez, M. Hjalmar, S. Cagnon, C. Mombereau, J. Ott, G. Mathias, S. Massot, D. Miliche, W.

Ken, P. Rémi, B. Rochet, J. Holburn, G. Sauwala, A. Brito Alves da Silva, A. Ortega, A. Vinicius Gonzalves Cardoso, F. Mirieu, P. d'Herbemont, E. de Roux, the main programmers of several modules, H.-X. Zhao, Sidharth GS, the main programmer of the machine learning techniques, and A.-K. Cheik Ahamed, K. Guillo, M. Wang, M. Lubin, T. Stremplewski, E. Balagna, I. Petrovic, N. Foresta, N. Fleury, A. Roullier, B. Allardet-Servent, A. Desmaison, F. Mayer, H. Ben Salem, J.-C., T. Saint-Paul, T. Zhu, A. Manai, L. Beaubois, T. Brendlé, O. Darwiche Domingues, O. Soualhine, S. Beux, K. Towa, S. Ehrhardt, S. Olivotto, A. Bodin, M. Lagattu, M. Botarro, E. de Roffignac, C. Haroune, G. Cochet, the main programmers of GPU acceleration techniques. Since 2006, this framework has been named ViKi (Virtual Interactive Keyboard Interface).

REFERENCES

- [1] R. Cipolla and A. Pentland. *Computer vision for human machine interaction*. Cambridge University Press, 1998.
- [2] J. Joseph and Jr. LaViola. A survey of hand posture and gesture recognition techniques and technology. Technical Report CS-99-11, 1999. Available online at: citeseer.ist.psu.edu/laviola99survey.html (accessed November 2007).
- [3] T.B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006.
- [4] D.J. Sturman, D. Zeltzer, and P. Medialab. A survey of glove-based input. *Computer Graphics and Applications, IEEE*, 14(1):30–39, 1994.
- [5] F. Magoulès and A.-K. Cheik Ahamed. Alinea: An advanced linear algebra library for massively parallel computations on graphics processing units. *International Journal of High Performance Computing Applications*, 29(3):284–310, 2015.
- [6] A.-K. Cheik Ahamed and F. Magoulès. Fast sparse matrix-vector multiplication on graphics processing unit for finite element analysis. In *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC 2012), Liverpool, UK, June 25–27, 2012*. IEEE Computer Society, 2012.
- [7] A.-K. Cheik Ahamed and F. Magoulès. Iterative methods for sparse linear systems on graphics processing unit. In *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC 2012), Liverpool, UK, June 25–27, 2012*. IEEE Computer Society, 2012.
- [8] A.-K. Cheik Ahamed and F. Magoulès. Efficient implementation of Jacobi iterative method for large sparse linear systems on graphic processing units. *The Journal of Supercomputing*, 73(8):3411–3432, 2017.
- [9] A.-K. Cheik Ahamed and F. Magoulès. Conjugate gradient method with graphics processing unit acceleration: CUDA vs OpenCL. *Advances in Engineering Software*, 111:32–42, 2017.
- [10] F. Magoulès, A.-K. Cheik Ahamed, and R. Putanowicz. Optimized Schwarz method without overlap for the gravitational potential equation on cluster of graphics processing unit. *International Journal of Computer Mathematics*, 93(6):955–980, 2016.
- [11] F. Magoulès, A.-K. Cheik Ahamed, and A. Suzuki. Green computing on graphics processing units. *Concurrency and Computation: Practice and Experience*, 28(16):4305–4325, 2016.
- [12] R. Kjeldsen, A. Levas, and C. Pinhanez. Dynamically reconfigurable vision-based user interfaces. *Mach. Vision Appl.*, 16(1):6–12, 2004.
- [13] D.S. Lee. Effective Gaussian mixture learning for video background subtraction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):827–832, 2005.
- [14] M. Piccardi. Background subtraction techniques: a review. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.
- [15] F. Magoulès, A.-K. Cheik Ahamed, and R. Putanowicz. Auto-tuned Krylov methods on cluster of graphics processing unit. *International Journal of Computer Mathematics*, 92(6):1222–1250, 2015.
- [16] A.-K. Cheik Ahamed and F. Magoulès. Energy consumption analysis on graphics processing units. In *Proceedings of the 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Xianning, China, November 24–27, 2014*. IEEE Computer Society, 2014.
- [17] F. Magoulès, A.K. Cheik Ahamed, A. Desmaison, J.C. Lechenet, F. Mayer, H.B. Salem, and T. Zhu. Power consumption analysis of parallel algorithms on GPUs. In *Proceedings of the 16th IEEE International Conference on High Performance and Communications (HPCC 2014), Paris, France, August 20–22, 2014*. IEEE Computer Society, 2014.
- [18] A.K. Cheik Ahamed, A. Desmaison, and F. Magoulès. Fast and green computing with graphics processing units for solving sparse linear systems. In *Proceedings of the 16th IEEE International Conference on High Performance and Communications (HPCC 2014), Paris, France, August 20–22, 2014*. IEEE Computer Society, 2014.