Design and implementation of a remapping algorithm for visual prosthesis

Ch. Amerijckx, J.-D. Legat, Ch. Trullemans
Université catholique de Louvain, Microelectronics Laboratory,
3 place du Levant, B-1348 Louvain-la-Neuve, Belgium
amerijckx@dice.ucl.ac.be

Abstract

This paper describes a novel remapping algorithm using two CCD cameras and its implementation on a VLSI chip. This system allows to process these two images in real time and to get the resulting image with a high resolution at the center as it is the case for the central part of the retina. This chip can be used in a constant aid prosthesis for patient suffering of Macular Degeneration (MD) and Retinis Pigmentosa (RP).

1 Introduction

Most of actual aids for MD are optical devices that magnify the image seen by the patients such as hand-held and stand magnifiers, telescopes, microscopes or closed circuits television. These devices change the scale of the image in such a way that the image falls in the remaining field of view of the patient. This allows the patient to recover at least a part of the central field information, even if it is at the expense of losing a portion of the peripheral field. For low magnification, this is not a problem. However, for high magnification, the resulting reduced field size restricts the amount of the world that may be seen at one time. This is the case, for example, for face recognition where the partial view of the image (nose, eye, hairs,...) lead to a difficulty of recognising the face.

At the other hand, for RP patients, the defect field of view is the peripheral retina. For these patients, a reverse telecope is required. This device minimize the image so that the information from the peripheral field of view falls onto the central part of the retina.

However, if these devices could be used to help the patient, they can not be used as a constant aid because it is difficult to localise objects in the field of view. In that case, the patient must alternate between the all visual field and the field resulting of the use of such device. This is not really handy.

In this paper, we describe a remapping algorithm and its VLSI implementation that could be used in a constant aid visual prosthesis. The remapping algorithm is similar to the one described in [1]: the visual field or a part of it is remapped to the remaining part of the retina. The only problem with most of the remapping algorithm is the poor resolution of the central part of the image. If this image has to be projected on the retina of a RP patient, this problem is not acceptable because the central part of the retina has a high resolution in comparision to the peripheral one [2], [3].

To solve this problem, we will use two images coming from two video-cameras with different optics: the first one with a tele-lens, and the second one with a wide-angle. The remapping algorithm will be applied on both images. After that, the images will be clustered together (Figure 1).

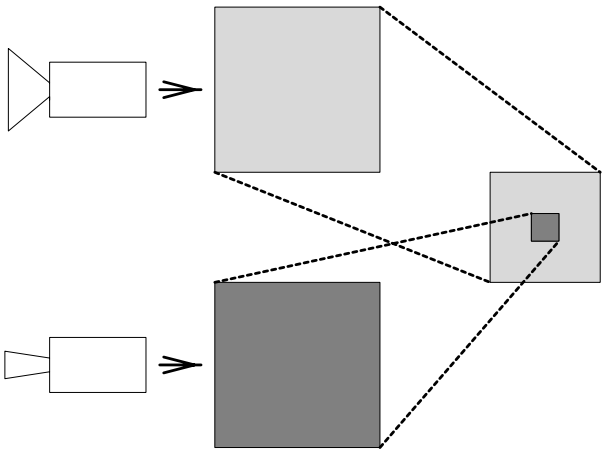

Figure 1: The clustered images

The resulting image will be projected on the remaining retina of the patient: for MD, the central part will not be helpful but for RP, the high resolution of this part will be helpful.

2 Algorithm

2.1 The Remapping Algorithm

According to [4], there is a proportionality between cortical distance, from projection of fovea toward projection along a given meridian, and the logarithm of the related retinal eccentricity. Using polar coordinates (Figure 2), we can write:

$$R = Kr^k$$

$$\Theta = \theta$$

where

- (R,Θ) is the position of a pixel in the image before remapping.
- (r,θ) is the position of a pixel in the image after remapping.
- K is the factor of proportionality and is constant.
- k is the remapping factor.

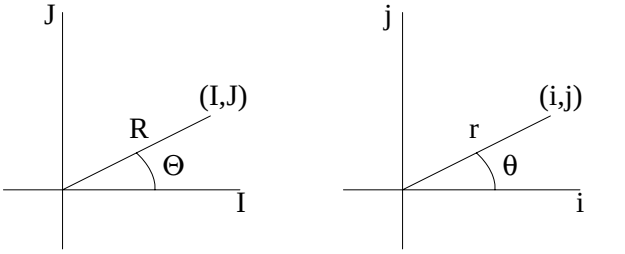

Figure 2: The images parameters before and after remapping

Therefore, by using orthogonal to polar coordinates transformations with the conventions of the Figure 2, it is easy to show that:

$$I = K(i^2 + j^2)^{\frac{k-1}{2}} i$$

$$J = K(i^2 + j^2)^{\frac{k-1}{2}} j$$

or, conversely:

$$i = \frac{(I^2 + J^2)^{\frac{1-k}{k}}}{K^{\frac{1}{k}}} I$$

$$j = \frac{(I^2 + J^2)^{\frac{1-k}{k}}}{K^{\frac{1}{k}}} J$$

These are the equations of the proposed remapping algorithm. Figure 3a shows the original image coming from the CCD and Figure 3b, the same image after the remapping algorithm for k=1.5 and K=1.

Figure 3: (a) the original image coming from the CCD (b) the same image after the remapping algorithm

With such figures, if we are considering a 512x512 CCD matrix at the input, the size of the output image is 80x80 pixels. This is really small and, if we try to enlarge this image, the resolution at the center is really bad. This problem can be solved by using two CCD cameras with two adequate optics: the central image will come from a tele lens camera (Figure 4a) and the peripheral one, from a wide angle camera (Figure 4b).

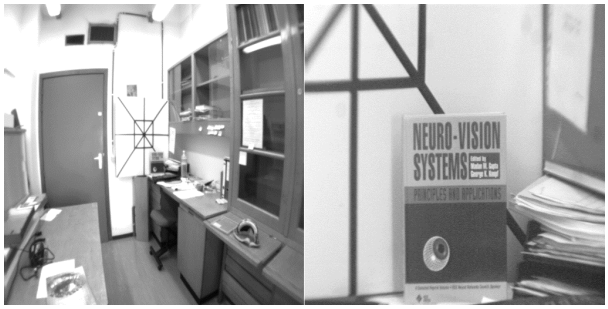

Figure 4: (a) the original image coming from the CCD with the wide angle (b) the image coming from the CCD with the tele lens

2.2 Generalization for two CCD

We can rewrite the equations above for the two CCD:

- Central CCD:

$$R = K_1 r^k \quad (1 \le r \le m)$$

$$\Theta = \theta$$

- Peripheral CCD:

$$R = K_2 r^k \quad (m \le r \le n)$$

$$\Theta = \theta$$

with: m: maximum eccentricity of the output image (central CCD).

n: maximum eccentricity of the output image (peripheral CCD).

If E_{CCD} is the eccentricity of the CCD device, then we can write:

- Central CCD matrix:

$$1 \le R \le E_{CCD}$$

$$1 \le r \le m$$

$$K_1 1^k = 1 \rightarrow K_1 = 1$$

$$K_1 m^k = E_{CCD} \rightarrow m = (E_{CCD})^{\frac{1}{k}}$$

- Peripheral CCD matrix:

$$m \le R \le E_{CCD}$$

$$m \le r \le n$$

$$K_2 m^k = m \rightarrow K_2 = \frac{1}{m^{k-1}}$$

$$K_2 n^k = E_{CCD} \rightarrow n = (E_{CCD})^{\frac{2k-1}{k^2}}$$

If we are using two 512x512 CCD and a k of 1.5, we have: m=40, K_2=0.157 and n=138. Thus the size of the output image is 276x276 pixels. This is represented in Figure 5.

Figure 5: Image after processing with the remapping algorithm using two CCD

2.3 Consideration

Looking at Figure 3b and Figure 5, one could mention that there is a big distortion of the image. But according to [5], the visually impaired person could easily interpret the image after training.

3 Implementation

3.1 Simplification of the algorithm

Looking at the equation above, it seems not easy to implement it due to the exponent. However, it is easy to find a relation between I, i, R and r:

$$I = R\cos\theta \text{ and } i = r\cos\theta \Rightarrow \frac{I}{i} = \frac{R}{r} \Rightarrow i = \frac{r}{R} I$$

$$J = R\sin\theta \text{ and } j = r\cos\theta \Rightarrow \frac{J}{j} = \frac{R}{r} \Rightarrow j = \frac{r}{R} J$$

where $R = \sqrt{I^2 + J^2}$.

For an input image of size mxn pixels, the maximum number of possible radius R is mxn. So the number of different radius R is finite. Knowing that:

$$r = \frac{1}{K} R^k ,$$

the number of different output radius r is also finite. Thus, the number of r/R is finite and is equal to mxn.

An idea would be to put all the possible value of r/R in a ROM and, for a given R, choose the correspondant r/R as it is shown in Figure 6.

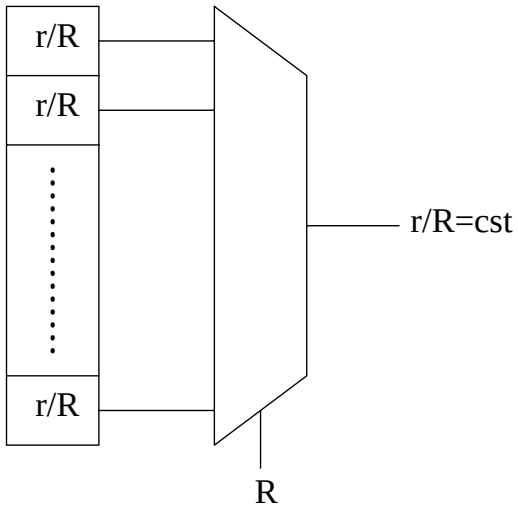

Figure 6: The coefficient r/R stored in a ROM

In that case, we can rewrite the equations above:

$$i = cst\ I$$

$$j = cst\ J$$

However, for large images, this would require a huge memory. A solution to this would be to make a linear interpolation of the remapping function on a certain number of intervals as shown in Figure 7.

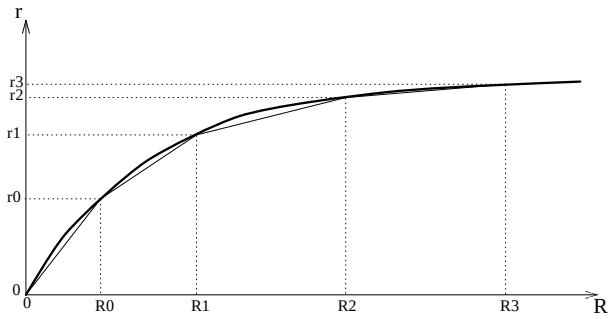

Figure 7: The linear interpolation

For $R_{i-1}<R<R_i$, the constant will be equal to:

$$\frac{r_i + r_{i-1}}{R_i + R_{i-1}}$$

This limits the number of constants to store, to the number of intervals. The number of possible intervals we have choosen after simulations (PSNR versus number of intervals) is equal to 128.

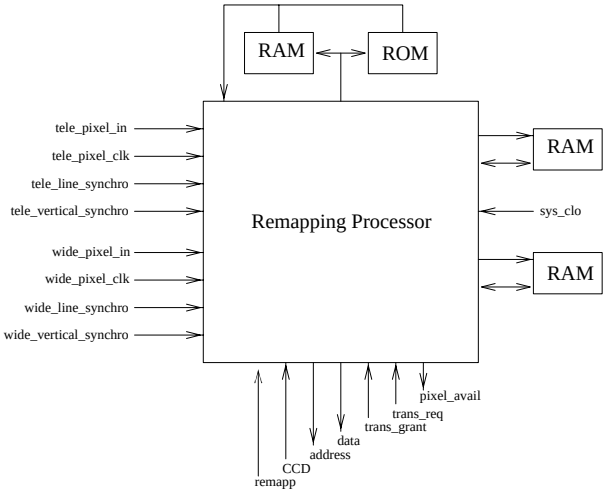

Figure 8: External hardware

3.2 External architecture

The entire remapping processor is configured during the reset stage by an external ROM containing (Figure 8):

- one or more remapping functions
- the weights for he image after remapping
- the initial values of the internal registers

Three additional RAM are required (Figure 8):

- one used as a cache for the weight coefficients (32 kbyte)
- one for the image we are processing (64 kbyte)
- one for the image transfered to the output (64 kbyte)

3.3 Internal architecture

The remapping processor could be divided into five main regions (Figure 9):

- datapaths (on 8 and 16 bits) for the image coming from the camera with the tele lens.
- datapaths (on 8 and 16 bits) for the image coming from the camera with the wide angle.
- a RAM for the remapping function.
- three control units.
- an output interface.

connected with buses on 8 and 32 bits.

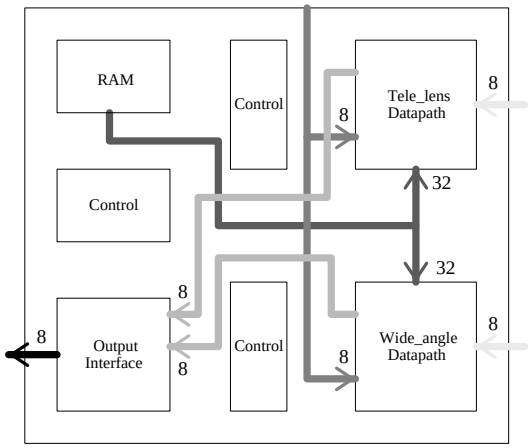

Figure 9: Internal hardware

The datapaths are dedicated to the remapping algorithm and are designed to be able to handle, each of them, 25 images per second. For this purpose, it has been designed as a serial architecture with five small fast stages (Figure 10):

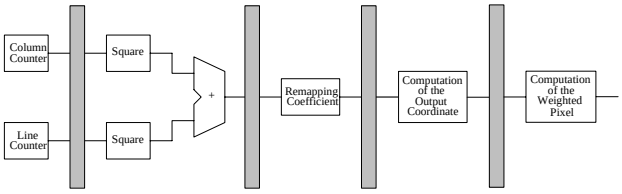

Figure 10: Datapath

- input coordinates computation: this stage includes two counters (one for the columns and one for the lines) computing the orthogonal coordinates of the pixel in the input image. Knowing that we are working with interlaced images, the columns counter is a counter by 1, and the line one, is a counter by 2.

  For a 8x8 pixels image, the sequence is:

  for the columns: -3 -2 -1 0 0 1 2 3

  for the lines: -3 -1 0 2 -2 0 1 3

- square of the radius computation: in this stage, we must take the sum of the square of each coordinate. To avoid the use of multipliers, we use the relations:

$$(x+1)^2 = x^2 + 2*x + 1 \rightarrow x_i^2 = x_{i-1}^2 + 2*x_{i-1} + 1$$

$$(x-1)^2 = x^2 - 2*x + 1 \rightarrow x_i^2 = x_{i-1}^2 - 2*x_{i-1} + 1$$

  for the columns, and:

$$(x+2)^2 = x^2 + 4*(x+1)$$

$$\rightarrow x_i^2 = x_{i-1}^2 + 4*(x_{i-1} - 1)$$

$$(x-2)^2 = x^2 - 4*(x+1)$$

$$\rightarrow x_i^2 = x_{i-1}^2 - 4*(x_{i-1} - 1)$$

  for the line to compute the square of the coordinates.

- remapping coefficient (Figure 11): as already mentionned, the remapping functions is stored in a lookup-table (LUT). The entry of this LUT is the radius of the position of a given pixel, the output is a coefficient we have to use to compute the orthogonal coordinates of the pixel after remapping. This LUT is stored in the internal high speed RAM. The purpose of this stage is to find the interval in the RAM and get the coefficient: this is done by using a fast 32 bits wide bus.

- computation of the output coordinates: we have to multiply each input coordinate by the coefficient from the previous stage. We are using a multiplexed-multiplier to be able to use only one multiplier. This implies the use of a frequency twice higher.

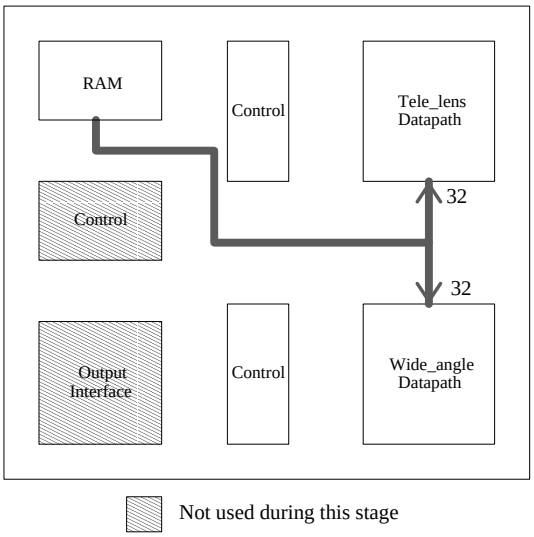

Figure 11: Remapping coefficient flow

- computation of the weighted pixel (Figure 12): in the described algorithm, one pixel in the output image is made of several pixels from the input image (depending on the position of the pixel). The purpose of this stage is, knowing the output coordinates of the pixel, to get the weight from the external RAM and compute a certain amount (depending on the weight) of the value of the pixel in the image after remapping.

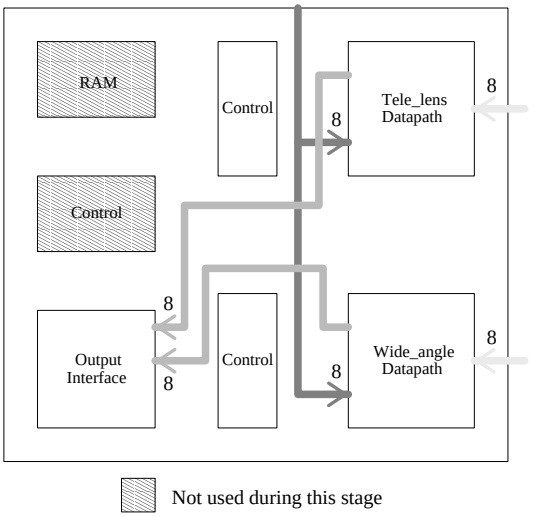

Figure 12: Computation of the weighted pixels

Finally, each amount of pixel is transfered to the output interface (Figure 13). The goals of this one are:

- to compute the final value of the pixel in the image after remapping by adding the previous stored value to the value coming from the datapaths. This implies read and write cycles in the first external RAM.

- to transfer, from the second RAM, the previous image to the output using a standard TMS320C40 protocol.
- to reset the second RAM.

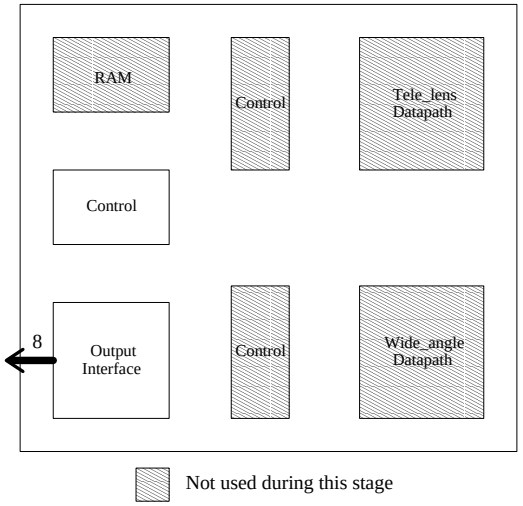

Figure 13: Output inteface flow

3.4 Results

The chip have been realized by using Compass and the ES2 1 micron technology (Figure14).

The size of this chip is 79mm²; it has 152 pads and is packaged in a PGA180. This chip is tested and is working at a frequency of 40Mhz, handling a pair of images at 25 images/s.

4 Acknowledgments

This research have been financed by the european TIDE project POVES (Portable Optoelectronic Vision Enhancement System).

References

[1] R.D. Juday, D.S. Loshin, "The programmable remapper: Clinical applications for patients with fields defects", *Optometry and Vision Science*, vol. 66, pp. 389-395, 1989.

[2] D. Marr, *Vision*, Freeman, San Francisco, California, 1982.

[3] G.K. Knopf, N.M. Gupta, *Neuro-Vision Systems: Principles and Applications*, IEEE Press, 1993

[4] T.N. Wiesel, D.H. Hubel, "Functional architecture of macaque monkey visual cortex", *Ferrier Lecture*, pp. 1-57, 1977.

[5] R.W. Massof, D. Dagnelie, "Toward an artificial eye", *IEEE Spectrum*, pp. 20-29, 1996.

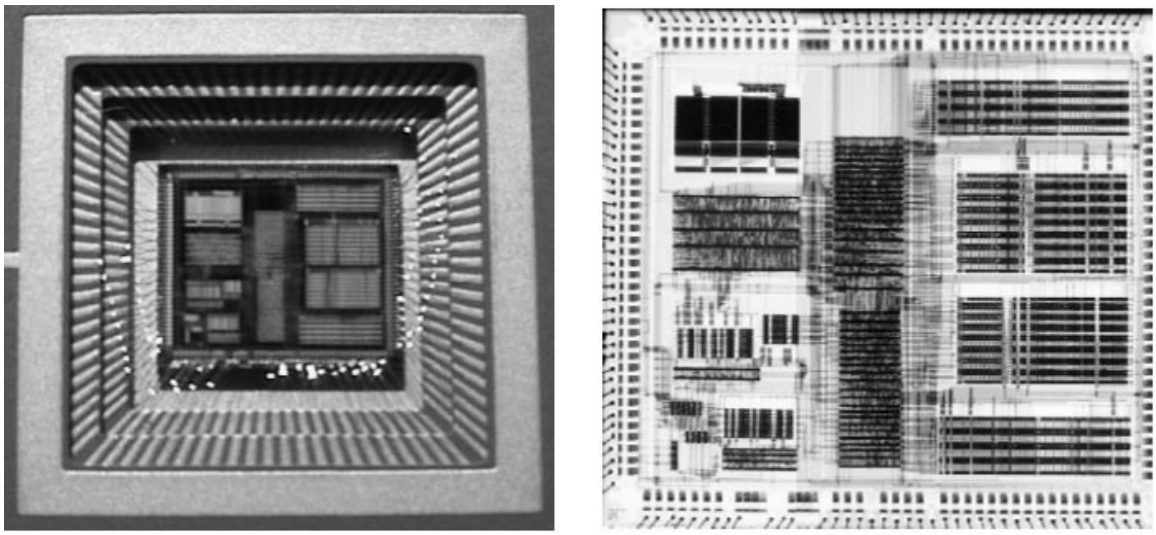

Figure 14: Layout of the chip