Hardware/Software CoDesign of an Acceleration SoC for Interactive Digital Painting

Luc Claesen, Andy Motten, Peter Vandoren, Tom Van Laerhoven, Frank Van Reeth
Expertise Center for Digital Media
Hasselt University, tUL, iBBT
Diepenbeek, Belgium

Yun Pan, Jingjin Hu, Ning Zheng, Xiaolang Yan
Department of Information Science & Electronic Eng.
Zhejiang University
Hangzhou, China

Abstract—**Novel digital paint systems use brush-like or real brush input for computer assisted graphics- and art-creation. This paper presents the hardware-software implementation of the real-time video processing algorithms for brush position, shape and bristle texture detection. The system has been implemented using a 5 megapixel camera and an FPGA system. The system consists of a dynamically reconfigurable image processing streaming architecture. It is heavily pipelined to keep the propagation delays within the strict timing requirements in order to enable the processing at camera video rate. Video processing blocks are realized in a modular way and interconnected by a dedicated programmable switch fabric. A 32 bit RISC processor controls the settings and parameters of the image processing, the camera settings as well as the communication with the host PC. Segmented brush footprints are forwarded by an Ethernet link to the paint rendering host PC. The system can be used for real-time virtual brush based painting.**

Keywords: digital painting, tangible interfaces, HCI, FPGA, System-on-Chip, SoC, Image Processing, video pipline

I. Introduction

Since the beginning of computing history drawing, sketching and painting systems have been developed [1]. Modern computers are all equipped with some sort of paint software. The advantage of painting on a computer is that intermediary results can be saved and that the “undo” function reduces the effect of errors in the ultimate painting result.

In the most common case, a computer mouse is used as a paint input device. Modern tablet computers can use fingers and pens to paint, but as only a contact position and size can be detected on a capacitive or resitive touch screen, the capability to really track a brush shape or foot print is not available. The expressiveness of a user while drawing or art creation is limited in comparison to real pencils and brushes. As a pen stylus better resembles traditional painting instruments, pen and tablet input devices are often used in professional digital painting systems.

Part of this research has been sponsored by the FWO (Flanders, Belgium) - MOST (China) project nr. G.A.063.10

The Wacom Cintiq system [6] is probably one of the best pen tablet input devices commercially available as it combines the input tablet with a flat LCD screen. The position is detected by a magnetic field interaction between loop wiring in the tablet and the drawing pen. The wire grid implemented in magnetic field based tablets is realized as a transparent ITO (Indium Tin Oxide) wiring on the LCD screen.

State-of-the-art pen and tablet systems [6] allow to sense the x- and y-position, the tilt and a one dimensional force along the stylus shaft. Although already very useful, the artistic expressivity of such systems is still far off from real painting instruments like brushes. A pen is stiff, while a brush with flexible and wet bristles in the tuft can take all kinds of shapes and footprints while moving the brush over a paint canvas.

In common painting software the graphics are bit-map based. In recent years, physics based fluid dynamic models and pigment diffusion mechanisms have been used to simulate the behavior of wet paint on a textured canvas [2-5]. This allows to accurately model a painting process where several colored pigment particles are mixed in a wet solvent and binder glue. The wet paint can move by advection and diffuse inside the painting canvas. The drying of the paint solvent can also be modeled like in real paint. Even the thickness of the paint layer and texture can be modeled accurately. In the project Gustav, MicroSoft is putting a lot of emphasis on physics based painting software [14,15].

Novel brush-based input methods have been presented. The IntuPaint system [7] is an innovative system employing special electronic brushes with flexible fiber bristles. The fiber bristles result in different imprints on a screen canvas depending on the tilt angle of the brush, the force and hand movement by the user. The IntuPaint system can be seen in action in [8]. A projector on the back of the virtual canvas can render the painted image and interactively show the artwork being produced. IntuPaint is based on infrared light that is conducted through the transparent fibers by means of total internal reflection and that exits at the bristle tips. An infrared camera behind the canvas can capture the image of the shape and intensity of the fiber bristle tips. This image can be used as a more accurate painting input.

IntuPaint brushes require special electronics hardware and the bristles behave like bristles in a dry brush. The tuft does not stick together like a real wet brush tuft. To cope with this deficiency, the FluidPaint system [9] has been developed. FluidPaint is making use of real brushes as familiar to artists. It is based on a layered sensor canvas (Fig. 1), where infrared light is entered at the side of a very thin (0.6mm) transparent optical wave guide. The light is normally propagated to the other edge side of the wave guide plate by means of total internal reflection. Dry brushes in contact with the sensor plate do not disturb the propagation of the infrared light captured in the wave guide by means of total internal reflection. This is because there is always a thin layer of air between the brush and the waveguide, maintaining the total internal reflection.

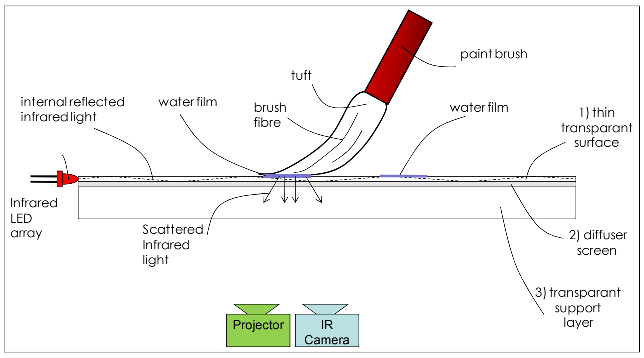

Figure 1. Operational principle of the FluidPaint System

Only when a wet brush (or another wet object) is put in contact with the sensor plate, it is possible that the infrared light is exiting the waveguide into the transparent fluid (water) until it reaches the object in contact with the sensor plate. There the light will diffuse with the object in the normal way of light reflection. This diffused light can then be detected by an infrared camera positioned behind the virtual canvas. This has as a result that the camera will only track the footprint of a wet painting brush.

A projector can be positioned behind the canvas to interactively visualize the painting being produced. As the FluidPaint system uses real brushes, it is very natural for artists to use the system. As the brush tuft is wet during the painting, just as painting with real paint, its feel in the hand and its expressiveness is the same as with a real brush. Artists are able to express similar nuances as with real brushes. As with real painting, the interaction of the brush with the canvas is only in the wet footprint. While (virtually) painting with FluidPaint, the brush tuft will become dry and has to be dipped in water regularly, just like in real paint. The FluidPaint system can be seen in action in [10].

In both the IntuPaint and the FluidPaint systems, an infrared camera has to detect the position, shape, and actual footprint with the distribution of tuft bristles on the contact surface with the canvas. To enable the real-time detection of the brush footprints of the 1024 by 768 pixel image, an FPGA based architecture has been developed. In this FPGA the image enhancement, segmentation and footprint position detection is performed in real time.

The FPGA hardware consists of a video streaming based architecture, supported by dedicated memories to compensate for background effects, footprint images, camera normalization etc… The architecture is built up in a reconfigurable switch fabric. Individual image processing operations can be tuned by parameter settings. The overall control on the camera, the parameter settings in the image processing operators, the switch fabric interconnect as well as the PC/Ethernet communication on the FPGA is done by means of a 32-bit RISC soft-core microprocessor. Brush footprint image data is sent to a host PC by means of a UDP based Ethernet connection. On the host PC, the settings of the FPGA board can be controlled. The host PC, equipped with a GPU (Graphics Processing Unit) performs the fluid dynamics based simulation of the paint.

The basic setup of the SoC hardware has been published in [13]. The underlying paper specifically presents the video and image processing algorithms and their implementation in a combined hardware/software architecture.

In section II the overall brush capture architecture is described. Section III explains the individual aspects in the real time video and image processing flow. Section IV describes the applications and use. Conclusions are made in Section V.

II. Overall Brush Capture Architecture

A. *Operation Principle*

Both the IntuPaint [7] and the FluidPaint [9] systems are real-time interactive and in-place painting systems. The conceptual setup is shown in Fig. 2.

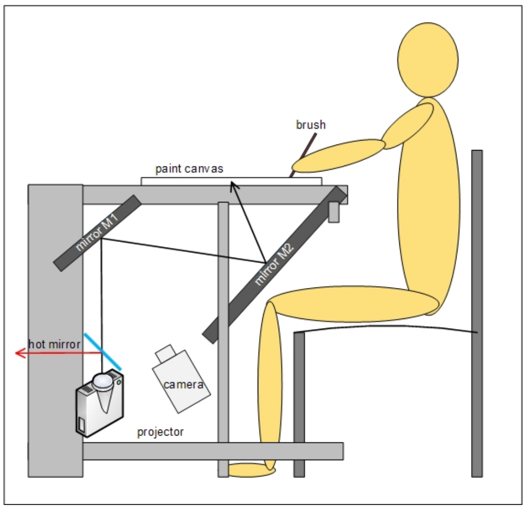

Figure 2. Digital paint system setup

A brush is used for interactive painting on a paint canvas. As explained in the previous section, the brush is either an active brush [7] emitting infrared light or a passive – real brush [9]. Although the canvas setup and layering is considerably different, the brushes manifest themselves as infrared images on the paint canvas. The brush footprint is tracked by an infrared camera specifically sensitive to the wavelength of the infrared light of the brush footprints. The camera monitors the footprint to determine the position of the brush on the canvas as well as to image the detailed footprint of the brush. This paper specifically explains the image processing SoC hardware/software architecture. The SoC system communicates via Ethernet to a host PC equipped with GPU accelerators. On the host PC physical model based software [12] is executed. The interactive results of the painting are displayed by a projector positioned below the virtual canvas.

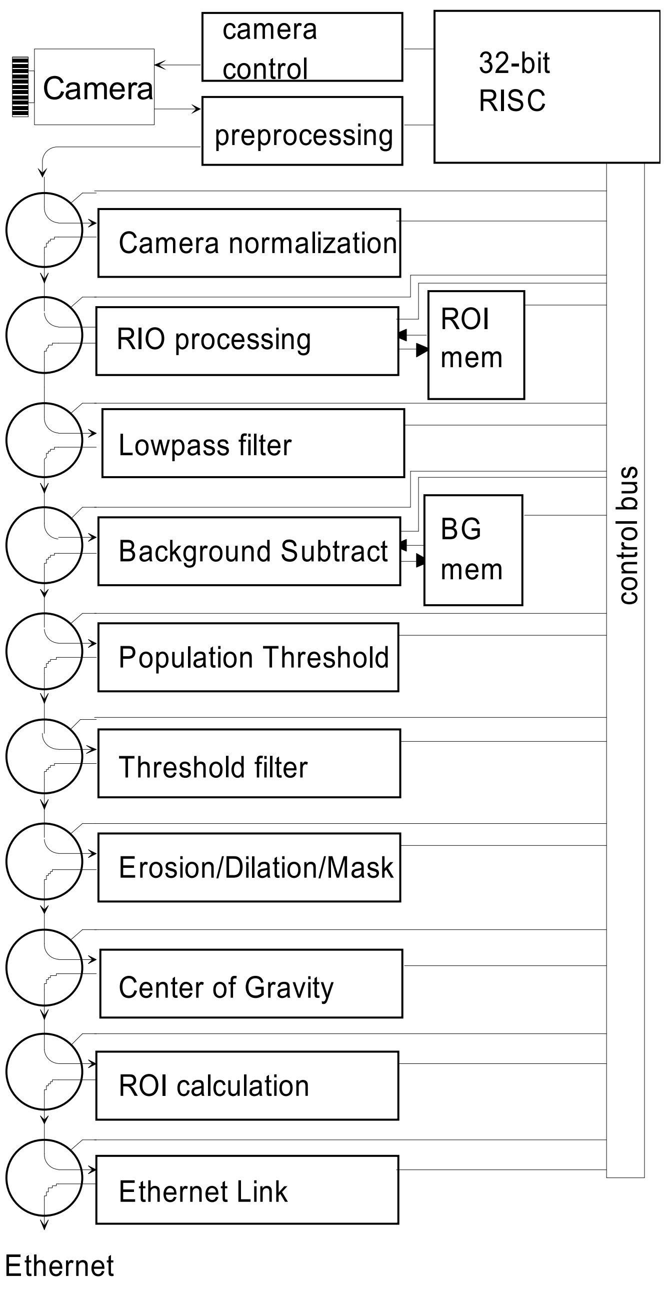

Figure 3. Brush video signal processing fabric architecture.

The result is that the effect of brush strokes are directly shown on the virtual canvas, taking into account the color pigment mixing and the detailed effects of the bristles in the tuft of the painting brush.

A key aspect in accurate painting with brushes is the detection of the brush position and the detailed bristle shape. For this the SoC based real-time video and image processing architecture is presented in this paper.

The overall architecture is shown in figure 3. It consists of a digital camera and a streaming architecture. In the streaming architecture a number video processing modules are directly implemented in hardware. Their interconnection and use is available in a programmable switch fabric. The functionality and parameters of the image sensor and hardware modules is controlled by a 32 bit on chip soft-core RISC processor. The RISC processor takes care of the local settings and control as well as the communication with the host PC. On the host PC a GUI interface can be used to control individual settings in the system.

III. SIGNAL PROCESSING FLOW

A. Camera preprocessing

The drawing canvas is visualized by an infrared CMOS camera. The silicon diodes in CMOS cameras have a sensitivity in the wave length range of 300nm to 1000 nm. This includes the visual as well as the near infrared spectrum. The light used for visualizing brushes in both the IntuPaint as well as the FluidPaint systems is of a known wavelength as generated by the infrared LED's used. Therefore the cameras are equipped with an optical band pass filter at the 940 nm wave length light emitted by the LEDs. All of the pixels of the camera are entered in the architecture in streaming mode.

The camera settings are controlled via an I2C interface, allowing all of the internal registers of the image sensor to be written and read. The I2C interface is controlled by a 32-bit on-chip RISC processor. As the RISC processor is connected to the host PC via USB and Ethernet, settings can also be controlled from the graphical user interface on the host PC.

B. Camera Normalization

The data generated by the image sensor in the camera is heavily influenced by the image sensor and lens setup. The camera normalization module therefore provides a normalization of the image data and a uniform sensitivity over the whole imager area. The normalization parameters can be calibrated and controlled from the host PC.

C. Region of Interest processing

The amount of data in one video frame is too large to be stored on the SoC itself. The active region around the brush is however interesting to be stored on chip. Based on the location **c**(t - 1) (the center of gravity) of the brush during the previous video frame at time t - 1, a bounding box or region of interest RIO(t) in the current frame at time t is predicted. Figure 4 illustrates the drawing canvas with the current RIO(t) predicted by the previous brush center of gravity **c**(t – 1). The region of interest should be large enough to fully enclose the footprint of the brush during the current camera frame, as indicated by the current center of gravity **c**(t).

The size of the region of interest is a compromise between the available on-chip memory, the brush size, the brush speed and the frame rate. The region of interest determined from the previous frame should enclose the current brush footprint. The displacement **d**(t) of the brush center of gravity is determined by:

$$\mathbf{d}(t) = \mathbf{c}(t) - \mathbf{c}(t-1)$$

The size of the displacement vector |**d**(t)| is determined by the velocity of drawing stokes on the canvas divided by the frame rate. If frame rates are too low or if strokes are made by very fast movements, it can occur that the current brush position will not fully be enclosed in the predicted region of interest RIO(t) during the frame at time t. In the current implementation parts of the brush will go undetected during such fast movement.

The region of interest processing has the advantage that immediately after the detection of the brush by the image sensor, the image data of the brush footprint can be transmitted over the Ethernet link to the host PC for use by the physical model drawing software [12].

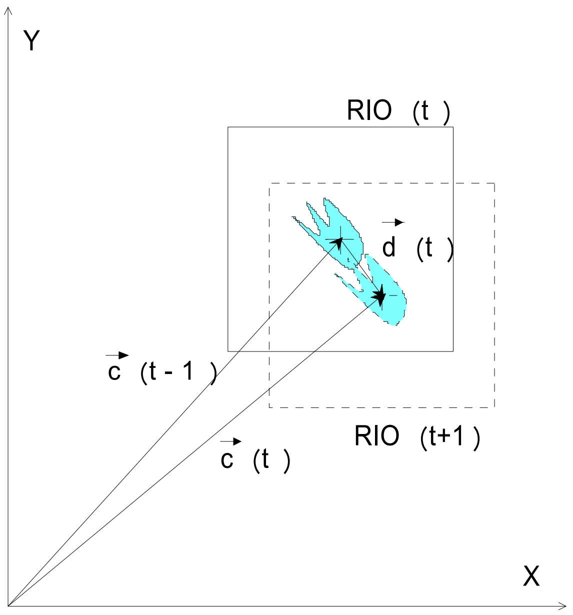

Figure 4. Region of Interest around brush footprint. The blue area indicates the brush footprint on the canvas. ***d(t)*** *is the brush displacement from the previous frame to the current frame.*

Therefore the brush footprint can be available in the painting software a few scan lines after the brush has been detected in the image sensor. This reduces the latency between movements on the canvas and the processing on the host PC.

D. Lowpass Filter

To reduce the effect of noise resulting from the image sensor, a low pass filter can be used. Either a 3x3 or a 5x5 convolution filter [11] can be used here.

On a streaming based SoC architecture, these convolution filters can be implemented by means of 2 (for the 3x3) or 4 (for the 5x5 filter) line buffers and 3x3 (or 5x5) register files. For speed and complexity reasons the coefficients are chosen as powers of 2. Because of this, no multipliers are required, as a multiplication by a power of 2 only consists of shifting the bits in a binary number. In hardware this consists of only routing the appropriate bits. Therefore the convolution filtering can be implemented in a pipelined way as a number of additions.

The filter coefficients predefined for the 3x3 low pass filter, scaled by 2^{-4} are:

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

The filter coefficients predefined for the 5x5 low pass filter, scaled resp. by 2^{-5} and 2^{-7}, are:

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix} \quad \text{and} \quad \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 8 & 4 & 1 \\ 1 & 8 & 64 & 8 & 1 \\ 1 & 4 & 8 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$$

After the additions in the convolution filters, the results are scaled by the power of 2^{-m}. This again consists of a simple shift of the resulting binary number. In hardware this is free as it only requires a different routing of bits.

The use of this low pass filter is controlled by the RISC processor and/or the host PC.

E. Background Subtraction

Because of the 940nm optical band pass filter in the camera, most of the environmental light is filtered away. Nevertheless, the environment light, e.g. reflected sun light, also has fractions of 940nm light. This results in an unwanted background image. The parasitic background 940nm image is stored in a background memory frame buffer. (BG MEM in Fig. 3) The background image can then be subtracted from the camera input to result in an image of the brush only.

The background image is usually changing, due to movement of people and objects in the room, change of external lighting, opening of doors etc… But most of the time this background image is only changing slowly in comparison to the video frame rate. Therefore the background image intensity BG(t) is updated in an adaptive way with the current foreground image FG(t):

$BG(t) = (1 - \alpha)\, BG(t - 1) + \alpha.FG(t)$

This is a first order IIR filter of the image. The update coefficient α is programmable by the RISC processor as well as the host PC

F. *Population Thresholding*

After the lowpass filtering to remove the input noise and the adaptive background subtraction the image with the brush footprint will stand out if present. In the next few steps the area of the footprint will be segmented out. A population thresholding is available to filter out regions with too few and unrelated pixels in a neighborhood. The percentage of population thresholding is controlled by the RISC processor and the host PC.

G. *Threshold Filter*

A threshold filter selects those regions with sufficient intensity. After the thresholding filter the resulting pixel values are normalized again, using an amplification factor. Threshold and amplification values are settable by the RISC and host PC.

H. *Erosion/Dilation/Masking*

If necessary the footprint region can be further processed by means of programmable erosion, dilation and masking functions under control of the RISC and host PC.

I. *Center of Gravity Calculation*

On the whole frame generated in the previous steps, the center of gravity is calculated. This is done in two steps. During the streaming of the processed video information I (x, y) a running value of the moments $M_x\,(x, y)$ and $M_y\,(x, y)$ in the x- and y- directions is calculated:

$M_x\,(x, y) = \Sigma_{\text{all } x}\, \Sigma_{\text{all } y}\;\; x \,.\, I\,(x, y)$

$M_y\,(x, y) = \Sigma_{\text{all } x}\, \Sigma_{\text{all } y}\;\; y \,.\, I\,(x, y)$

as well as running values for the weights $W_x\,(x, y)$ and $W_y\,(x, y)$:

$W_x\,(x, y) = \Sigma_{\text{all } x}\, \Sigma_{\text{all } y}\;\; I\,(x, y)$

$W_y\,(x, y) = \Sigma_{\text{all } x}\, \Sigma_{\text{all } y}\;\; I\,(x, y)$

At the end of each frame, after $x = x_{max}$ and $y = y_{max}$, where x_{max} is the image width and y_{max} is the image height, the center of gravity $\mathbf{c} = (c_x, c_y)$ can be calculated during the vertical blanking period of the camera:

$c_x = M_x(x_{max}\;, y_{max}) \;/\; W_x(x_{max}\,, y_{max})$

$c_y = M_y(x_{max}\;, y_{max}) \;/\; W_y(x_{max}\,, y_{max})$

The center of gravity **c**(t) is used during the next frame to store the adaptive region of interest RIO(t + 1). The image of the region of interest can be used as a basis for the brush footprint.

As the brush footprint is also available in the ROI MEM on-chip memory, the brush footprint can be masked by the segmented outline calculated in the previous steps.

The resulting center of gravity is communicated to the RISC and the host PC.

J. *Region of Interest calculation*

Based on the center of gravity the new region of interest is determined.

K. *Ethernet link*

An Ethernet link is used to transmit the brush image within the region of interest RIO(t) to the host PC. In the physical paint software on the host PC the brush footprint is used in rendering the paint result.

The Ethernet communication is done via the UDP protocol. Segmented footprint image data as well as side channel data (e.g. center of gravity etc.) is bundled in IP packages.

IV. Application and Use

The FluidPaint digital painting system setup is shown in Fig. 5. The effect of some random brush strokes on the virtual canvas is shown in Fig 6.

The virtual painting in action is shown in Fig. 7. By using real brushes, the system is very easy to use. The intuitive graphical user interface using the well known painting tools such as paint, water and brushes makes the system immediately usable by anybody. Even pre-school children have succesfully used the system without any preparation in a matter of seconds.

The current camera and image processing system has been implemented on an Altera Cyclone II FPGA with 68,000 logic elements. The design is done using Quartus-II using the verilog language. The 32-bit RISC processor is a Nios-II. The processor, the Avalon bus structure and the peripheral modules have been designed using the SOPC system.

The SoC architecture presented is a tradeoff between factors of usage as well as available hardware resources. Due to the constant technological evolution illustrated by Moore's law, hardware resources are steadily growing. Larger SoC's or FPGA's with more resources on chip can drastically improve the overall performance.

The specific processing of the region of interest is currently only impacting the SoC hardware architecture itself. In fact it can also be exploited in the camera, as the region of interest can be programmed in the image sensor itself. The frame rates of image sensors are limited by the bandwidth of the digital video interconnection. Due to this bottleneck, larger amount of pixels per frame will reduce the frame rate. In the current implementation a camera resolution of 1024 x 768 equal to the resolution of the projector is used. If the smaller predicted region of interest RIO(t) is used for adaptively reading out the image sensor, the frame rate of the camera can be increased. To guarantee that the RIO frames will include the brushes, overall detections of the brush positions should be interleaved to take care of the case when the current brush position **c**(t) would not be included in the predicted RIO(t). Such optimizations are planned for future versions of the system.

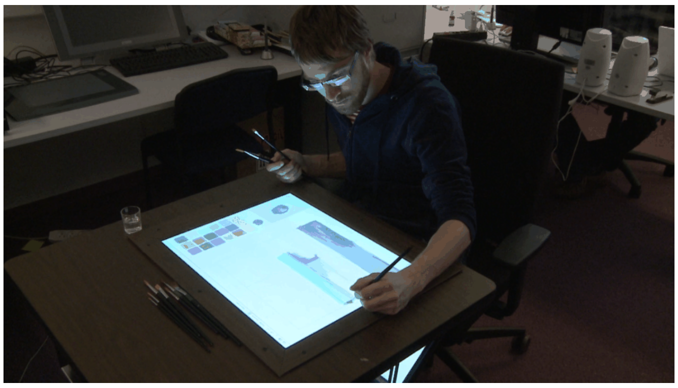

Fig. 5. FluidPaint System Setup.

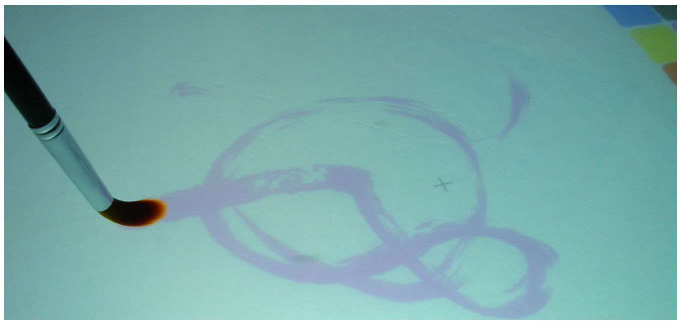

Fig. 6. Random brush strokes and result on virtual canvas

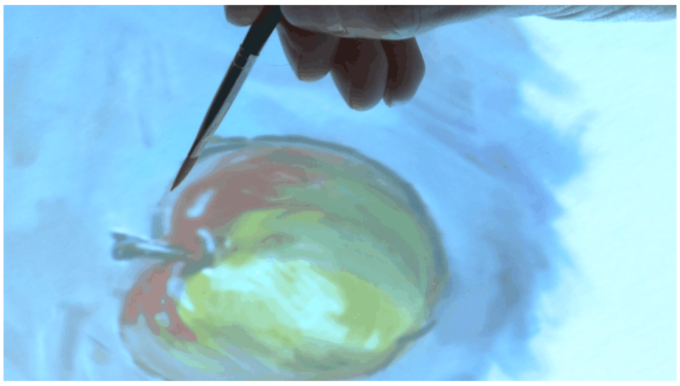

Figure 7. Close-up of the virtual painting with real brushes on the FluidPaint system [10].

V. CONCLUSIONS

In this paper the video and image processing SoC architecture is presented. It is a combination of dedicated hardware, application specific embedded processor, embedded software and host PC application software design. By hardware/software codesign the real-time preprocessing and brush footprint segmentation is done. Future work will concentrate on implementing the paint simulation methods directly in hardware [16] thereby increasing the processing speed and reducing the latency between paint strokes and the interactive results rendered by the image projector.

REFERENCES

[1] Smith, A.R. . Digital Paint Systems: An Anecdotal and Historical Overview, IEEE Annals of the History of Computing, 23, 2(2001), pp. 4-30.

[2] C.J. Curtis, S.E. Anderson, J.E. Seims, K.W. Fleischer, D.H. Salesin, "Computer-generated watercolor", ACM Siggraph, Proc. 24th ann. conf. on Computer Graphics and Interactive Techniques, pp. 421-430, 1997.

[3] William V. Baxter, Jeremy Wendt, and Ming C. Lin, "IMPaSTo: A realistic, interactive model for paint." In Stephen N. Spencer (ed.), Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France, June 5-7, 2004.

[4] T. Van Laerhoven, F. Van Reeth, "Real-time simulation of watery paint", CASA 2005, Volume 16, Issue 3-4, July 2005, pp. 429-439.

[5] Nelson S.-H. Chu and C.-L. Tai, MoXi: Real-Time Ink Dispersion in Absorbent Paper, ACM Transactions on Graphics (SIGGRAPH 2005 issue), Vol. 24, No. 3, August 2005.

[6] Wacom Cintic. http://www.wacom.com/cintiq

[7] Peter Vandoren, Tom Van Laerhoven, Luc Claesen, Johannes Taelman, Chris Raymaekers, Frank Van Reeth, "IntuPaint: Bridging the Gap Between Physical and Digital Painting" Proc. of TABLETOP 2008, pp. 71 - 78, IEEE, ISBN 978-1-4244-2897-7, Oct. 2008.

[8] Peter Vandoren, Tom Van Laerhoven, Luc Claesen, Johannes Taelman, Chris Raymaekers, Frank Van Reeth, “IntuPaint Digital Painting”, http://www.youtube.com/watch?v=n2IW4koT7Gw

[9] Peter Vandoren, Luc Claesen, Tom Van Laerhoven, Johannes Taelman, Chris Raymaekers, Eddy Flerackers, Frank Van Reeth, "FluidPaint: an Interactive Digital Painting System using Real Wet Brushes" Proc. of ITS2009, pp. 53 - 56, ACM, ISBN 978-1-60558-733-2, Nov. 2009.

[10] Peter Vandoren, Luc Claesen, Tom Van Laerhoven, Johannes Taelman, Chris Raymaekers, Eddy Flerackers, Frank Van Reeth, "FluidPaint: an Interactive Digital Painting System using Real Wet Brushes", http://www.youtube.com/watch?v=fkl782OqqmA

[11] Rafael. C. Gonzalez, Richard E. Woods, "Digital Image Processing", Prentice Hall, 2002, ISBN 0-201-18075-8.

[12] Tom Van Laerhoven, “An Extensible Simulation Framework Supporting Physically-based Interactive Painting”, Ph.D. Thesis, University Hasselt, 21 june 2006

[13] Luc Claesen, Peter Vandoren, Tom Van Laerhoven, Andy Motten, Domien Nowicki, Tom De Weyer, Frank Van Reeth, Eddy Flerackers, "Smart Camera SoC System for Interactive Real-Time Real-Brush based Digital Painting Systems", Proc. IEEE VLSI-SOC 2010 Conference, Madrid, 27-29 Sept. 2010, pp. 247-252.

[14] N. Chu, W. Baxter, L.-Y. Wei, and N. Govindaraju, "Detail-Preserving Paint Modeling for 3D Brushes", in Procedings of the 8th international symposium dedicated to non-photorealistic animation and rendering (NPAR 2010), Association for Computing Machinery, Inc., 7 June 2010

[15] W. Baxter and N. Govindaraju, "Simple Data-Driven Modeling of Brushes", in Proc. Symposium on Interactive 3D Graphics (I3D 2010), Association for Computing Machinery, Inc., February 2010

[16] D. Nowicki, L. Claesen, "SoC Architecture for Real-Time Interactive Painting based on Lattice-Boltzmann", Proc. 17th IEEE ICECS 2010, Athens, Dec. 12-15, 2010. IEEE Catalog Number: CFP10773-CDR, ISBN 978-1-4244-8156-9, pp. 239-242.