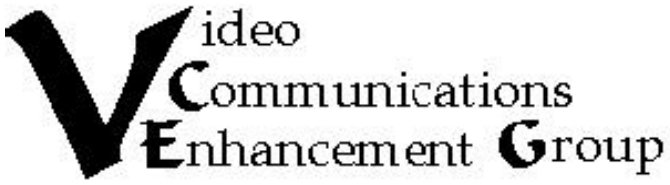
BACK

TI DSP and Analog Challenge 2000



Enhancement of Videotelephone Imagery Using the TMS320C6711 DSP

Dr Tim Morris
Julian McCollin
Marlon Ducille, Deborah Maxwell
Christopher Piggin, David Rajakumar

30/04/2001

Abstract

With the increasing bandwidth available to mobile telephones has come the ability to transmit video data. The increasing bandwidth may be utilized to create a mobile video-telephone.

However, this model ignores some of the practical aspects of the mobile videophone's use, which arise because the videophone will be hand-held by a caller, who will aim it at himself and may be moving whilst making a call. Using the videophone in this way can introduce several artefacts to the video stream.

i. The phone may be incorrectly aimed, consequently the caller will be incorrectly framed in the receiver's display.

ii. The caller may move his phone during the course of a conversation, resulting in his image being unstable: the head and shoulders will move laterally and change apparent size.

iii. The background behind the caller may contain distracting detail. If the background contains illumination sources, then the exposure of the image will alter dynamically.

We have developed a system to address these issues. The head region of the image is located using colour properties and facial features. Background distractions are suppressed by dimming the non-head region. The image is resampled such that the apparent head size is standardised. The image is then redrawn so that the head appears in the centre of the output window.

This report describes the algorithms in detail, examines their implementation on the TMS320C6711 Digital Signal Processor (DSP) and presents results of both experimentation into the achieved processing rate and the degree of subjective improvement to the data.

Table of Contents

Introduction

There are some practical aspects of a mobile videophone's use that are not currently accounted for. These arise because the videophone will be hand-held by a caller, who will aim it at himself and may be moving whilst making a call. Using the videophone in this way can introduce several artefacts to the video stream.

The phone may be incorrectly aimed, consequently the caller will be incorrectly framed in the receiver's display. The caller may move his phone during the course of a conversation, resulting in his image being unstable: the head and shoulders will move laterally and/or vertically and change in apparent size. The background behind the caller may contain distracting detail. The background scene will naturally change, given that the application is in mobile video telephony.

In order for the receiver of the videophone transmission to be presented with a stable image that is not distracting, the artefacts must be removed from the video stream, or their effect reduced in the video stream. The Video Communications Enhancement Processor (VCEP) removes some of the artefacts from the video stream before transmission to the receiver. VCEP addresses the problems of unstable images due to motion of the phone and distracting detail in the background. VCEP does not process any audio.

VCEP consists of three main modules that locate the face, stabilize the video stream and suppress the background. The FindFace module uses the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm to locate the face. The AntiJitterFace module resamples the capture image so that the face size is standardized. This module also redraws the resampled image so that the face is at the centre of the display image. The SuppressBackground module suppresses background detail by dimming half of the background and replacing the other half with a uniform grey colour.

An improvement of VCEP is for it to process YCrCb colour directly so as to avoid conversions between RGB and HSV. Also, YCrCb is more suited for current video compression techniques that facilitate transmission.

The VCEP system is due to be completed on June 7, 2001 so there are no quantitative results available at present with regards to the DSP. The system, however, performs correctly under normal conditions at 12 frames per second (fps) on the PC platform.

This report defines the problems that are addressed by VCEP, presents an overview of the top level design of the system, describes the algorithms used in the system and their implementation on the TMS320C6711 DSP and presents an evaluation of the system.

Problem Definition

Relative motion occurs between the camera and the user's face so the face can move around somewhat erratically in the output image. This results in jitter in the video stream and a fluctuation in the size of the user's face in the video stream. Also the background will be moving around in the output image. Such movements result in a display that can cause motion sickness to the receiving party, which is undesirable. The phone will sometimes be held at an angle with respect to the horizon, which will produce a display image that is tilted. When there is activity in the background of the capture image it will be present in the display image and will distract the receiving party from the user's face. This is because human perception operates such that moving regions are more easily detected than stationary regions in a scene. In any video communication system it is undesirable for the receiving party to be distracted from the person that they are communicating with. As the user moves around the background lighting can change significantly, which causes fluctuations in the contrast of the display image.

The VCEP system addresses the problems of face jitter, face size fluctuation and background distraction.

VCEP will run at a minimum of 10 fps. The solution to the face jitter is to maintain the face at the centre of the display image. The face size fluctuation is reduced by maintaining the height of the face at 95% of the height of the display image or the width of the face at 95% of the width of the display image. Suppressing the background in the display image solves the background distraction problem. Dimming half of the background and replacing the other half with a

shade of grey suppresses the background. The shade of grey is complimentary to the shade of grey that corresponds to the average colour of the background.

Top Level Design

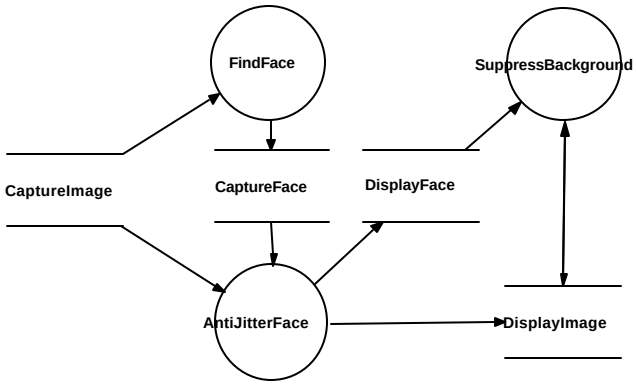


Figure 1 VCEP Level 0 Data Flow Diagram

The VCEP system captures colour images at a resolution of 352 x 288 pixels and displays colour images at 176 x 144 pixel resolution. The captured face and display face data stores contain the left, right, top and bottom edges of a rectangle that represents the face in the capture image and display image, respectively.

VCEP consists of three main modules that locate the face, stabilize the video stream and suppress the background.

The FindFace module locates the user's face in the capture image by use of the CAMSHIFT algorithm. The capture face is found as a rectangle that outlines the user's face in the capture image.

The AntiJitterFace module produces the display image and the display face. The display face is at the centre of the display image and its height or width occupies 95% of the height or width of the display image.

The SuppressBackground module suppresses the background of the display image. Half of the pixels in the background of the display image are replaced with a grey colour. This has the effect of superimposing a mesh on the background of the display image, which suppresses any detail in the background. The other half of the background pixels are dimmed, which too has the effect of suppressing background detail.

Face Region Location

Face region location is necessary so that the user can be centred in the transmitted image. Therefore, for the system to be usable in normal conditions the software must locate the face for different users (without prior knowledge of user – e.g. no training of system on user's face), be robust (remain relatively unaffected by lighting conditions) and consistently find the user. Essentially this can be considered to be face tracking. Face tracking can be undertaken in a variety of different ways: by using skin tone, edge detection, feature recognition, the use of neural networks and ellipse matching to name a few. However, as the system must run in real-time and not only track faces but manipulate the found face, speed of the face-tracking algorithm is of the highest importance. Thus, skin tone based on colour pixels of the captured image is being used in this project implementation.

Using colour components is a relatively easy and fast method of determining skin in images, however it is not necessarily completely accurate, hence a variant of CAMSHIFT [BRAD98] has been implemented, which grades colour pixels based on the probability that they are 'skin'.

CAMSHIFT can be considered as two separate stages:

i) Creating a flesh probability image
ii) Using this flesh probability image area to climb the 2-D probability gradient of a search window (via mean shift algorithm) in the image area hence finding the highest probability window location. Iterating through the flesh probability image area and adjusting the search window size accordingly, (i.e. continuously adaptive) for each frame finds the face.

Creating the flesh probability image area
Skin colours are sampled from images of different users and combined to give a mean and standard deviation of skin colours. This is then used to create a look up table which gives the probability that each colour value is skin – ranging from 0 to 100%. It is now a simple matter to transform a captured image to a flesh probability image area (it is not necessary to process the whole image for every frame – see next section).

Due to the inherent nature of the probability image, there is no definite cut-off to the probabilities and a more gradual flesh probability image is created compared to a basic thresholded image. This helps the robustness of the algorithm.

Finding the face from the probability image
For the first frame in a video sequence, the flesh probability image area is created for the whole captured image. The face is then found using this flesh probability image by first finding the zeroth moment (zeroth moment is: $M_{00} = \sum_x \sum_y I(x, y)$) and the first moment for x and y ($M_{10} = \sum_x \sum_y xI(x, y)$ and $M_{01} = \sum_x \sum_y yI(x, y)$ respectively, where $I(x, y)$ is the pixel probability at position (x, y) in the image and x and y range over the search window).
The centroid of the search window is calculated by:

$$x_c = \frac{M_{10}}{M_{00}}; y_c = \frac{M_{01}}{M_{00}}$$

And the initial window size, s, is created by: $s = \left(\sqrt{\frac{zerothMoment}{k}} \right) * 2$, where k is the maximum pixel value.
Using the search window and (x_c, y_c), we find the zeroth and first moments for x and y within the search window, then update values for (x_c, y_c) and s until (x_c, y_c) converges, or lies within a negligibly small distance of a previous consecutive iteration of the image. (A variation of this is to mean shift for a certain number of iterations within the one window size, before adjusting the window size.) This entire process can be repeated for every frame; or for faster convergence, we may store the zeroth moment and (x_c, y_c) for consecutive frames and hence it is not necessary to construct whole captured flesh probability image, but merely a larger than s window sized flesh probability. This speeds up the face-tracking algorithm as knowledge is used from previous frames.

The diagrams, in figure 2, illustrate the convergence properties of CAMSHIFT. The graphs represent a 1-D image with skin probabilities shown in red. The blue marker indicates the centroid position of the search window, size s, which itself is represented by the yellow window spanning skin probabilities. As can be seen, for each iteration through the image the centroid marker and search window changes in order to cover the largest peak of the graph. In this example

CAMSHIFT requires six iterations, but on subsequent frames this number should be reduced, as previous window size and centroid marker positions are stored.

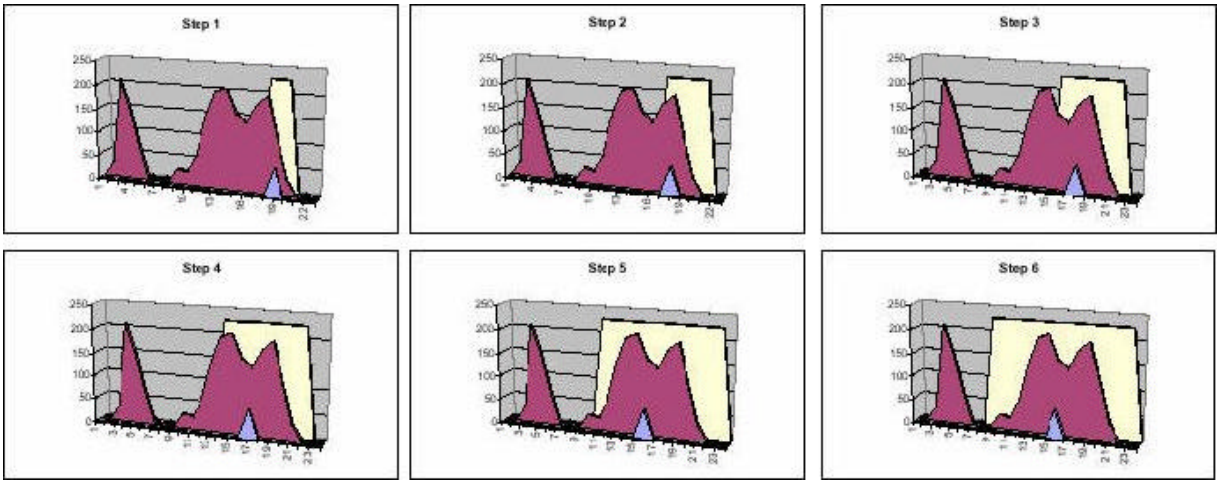


Figure 2 Convergent Properties of CAMSHIFT [BRAD98]

Find Face Sample Results

Figure 3 Capture Image

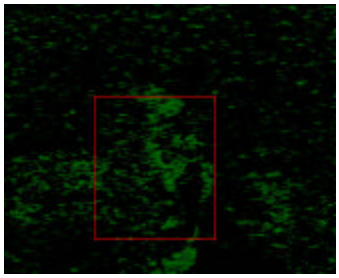


Figure 4 Flesh Probability Image

Figure 5 Located Face

The FindFace module correctly locates the face from the flesh probability image.

Face Region Manipulation

The position of the face is maintained at the centre of the display image so that it does not move around in the display. This eliminates the effect of left-right and up-down relative motion between the camera and the user's face. The area of the face is maintained at a fixed percentage of the display image so as to reduce the effects of away-towards relative motion between the camera and the user's face.

The AntiJitterFace module zooms out from the capture image to form an intermediate image and then copies part of the intermediate image to the display image such that the face is at the centre of the display image and its height or width occupies 95% of the height or the width of the image. Alternately, part of the capture image can be copied to an intermediate image such that when the intermediate image is zoomed in to obtain the display image, the face is in the centre of the display image and its height or width occupies 95% the height or width of the image. Under normal hand held use of the phone the size of the capture face is sufficiently large that no 'zooming in' has to take place. Only the

'zooming out' aspect is presented here although the system can 'zoom in' should the phone be in some kind of hands free configuration.

The face manipulation is achieved by 'zooming out' from the capture image to form an intermediate image, from which, pixels are copied from a viewport, represented by the dashed box, to the display image (in figure 6).

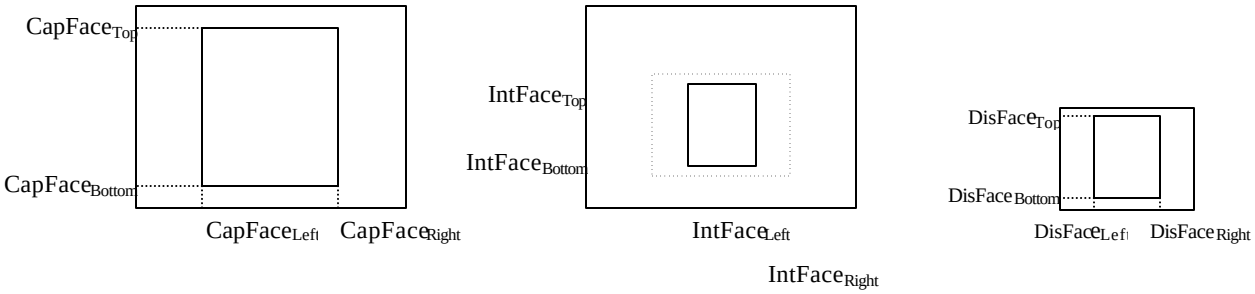


Figure 6 Capture, Intermediate and Display Images and Faces

The AntiJitterFace module operates in 7 steps as follows:

1. Compute the zoom factor

```
Zx = display image width  * 90% / (capture face right - capture face left)
Zy = display image height * 90% / (capture face top - capture face bottom)
Zoom factor = smaller of Zx and Zy
```

2. Compute the display face

```
Display face width = capture face width * zoom factor
Display face height =capture face height * zoom factor
Display face left = (display image width – display face width) / 2
Display face right = display face left + display face width
Display face bottom = (display image height – display face height) / 2
Display face top = display face bottom + display face height
```

3. Compute the intermediate face

```
Intermediate face left = capture face left * zoom factor
Intermediate face right = capture face right * zoom factor
Intermediate face bottom = capture face bottom * zoom factor
Intermediate face top = capture face top * zoom factor
```

4. Zoom capture image by zoom factor to intermediate image

```
FOR each pixel in the capture image
   Position in intermediate image = position in capture image * zoom factor
   Copy pixel from capture image to position in intermediate image
```

5. Compute viewport on intermediate image

```
Viewport right = intermediate face right + (display image width – display face right)
Viewport left = Viewport right – display image width
Viewport top = intermediate face top + (display image height – display face top);
Viewport bottom = Viewport top – display image height
```

6. Check viewport

```
IF viewport left < 0 THEN set to 0
IF viewport bottom < 0 THEN set to 0
Intermediate image width = capture image width * zoom factor
Intermediate image height = capture image height * zoom factor
IF viewport right > intermediate image width THEN set to intermediate image width
IF viewport top > intermediate image height THEN set to intermediate image height
```

7. Copy viewport from intermediate image to display image

```
Compute horizontal offset and vertical offset
Set input row to bottom of viewport
Set output row to bottom of display image + vertical offset
FOR each row in viewport
   Copy to display image at horizontal offset
   Increment input row and output row
```

8. FillImage

```
IF display image needs filling to the left THEN
  FOR each row between viewport bottom and viewport top
    Copy pixel from viewport left to each pixel between viewport left and the left edge
IF display image needs filling to the right THEN
  FOR each row between viewport bottom and viewport top
    Copy pixel from viewport right to each pixel between viewport right and the right
edge
IF display image needs filling to the bottom THEN
  Copy row from viewport bottom to rows between viewport bottom and bottom edge
IF display image needs filling to the top THEN
  Copy row from viewport top to rows between viewport top and top edge
```

AntiJitterFace Sample Results

Figure 7 Capture Image

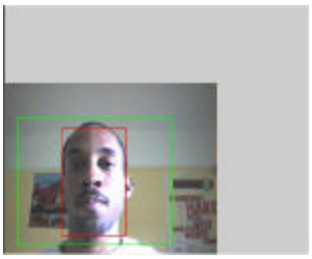


Figure 8 Intermediate Image

Figure 9 Display Image

The AntiJitterFace module correctly centres the face at a standardised size in the display image.

Background Manipulation

Distracting details and effects can be removed from the background by replacing the entire background with one colour, superimposing a mesh over the background, dimming the background, blurring the background or some combination of these. The VCEP system suppresses the background by replacing the even numbered pixels on even numbered lines and odd numbered pixels on odd numbered lines with a shade of grey (pixel interleaving). Also the pixels that are not replaced by interleaving are dimmed. Together, these techniques produce the best balance of computational speed and quality of result.

Whereas the face was represented by a rectangle for the purposes of face manipulation, an ellipse that is bounded by the rectangle is used to represent the face for the purpose of background manipulation.

The SuppressBackground module manipulates the background as follows:

```
Interleave colour = complimentary colour of average grey colour of background
Dim percentage = percentage average grey colour of background of max grey level
FOR each even row in display image
   Set each even background pixel to interleave colour
   Dim each odd background pixel by dim percentage
FOR each odd row in display image
   Dim each even background pixel by dim percentage
   Set each odd background pixel to interleave colour
```

SuppressBackground Sample Results

Figure 10 Display Image Before Suppression

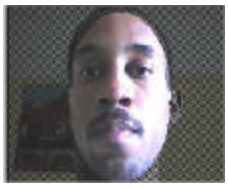


Figure 11 Display Image After Suppression

The SuppresBackground functions correctly, in that a viewer's attention is drawn to the user's face and not the background. There is still, however, sufficient detail in the background that the user's face does not appear to be floating in the centre of the display image. Motion in the background is not distracting in the display image but this cannot be shown in this report.

Implementation

The system used was the recently released Texas Instruments Image Development Kit. This is based around the TMS320C6711 DSP and contains the C6711 DSK board. The video capture hardware comes in the form of a daughtercard, which fits into the EVM compatible daughtercard interface. The daughtercard captures information in composite video (we have the PAL version) and formatting to 4:2:2 YCrCb is done by an on-board Field Programmable Gate Array (FPGA). The daughtercard also allows video data to be output to a monitor in RGB 565 format (16 bit colour).

How the system was implemented

The system was implemented entirely by the use of Texas Instruments provided software. The Code Composer Studio environment was used to enter and compile the software and code for the system. Debug and analysis tools within the software were used for debugging and analysis.

The code was initially designed and prototyped for the PC platform. Once a satisfactory solution for a module had been gained, TMSC6x code was written that performed the same function. This was due to the fact that the team was initially very familiar with the PC platform. Ideas and theories could quickly be implemented and tested. This kind of development cycle also allowed us to compare what was possible on the DSP architecture and where we could or could not improve over its PC counterpart. It should be noted however, that later in development the C6x code became very familiar and it was easier to implement the modules directly in the C6x code.

Once the modules are in place, they will be integrated and tested. Due to the fact that the system as a whole was designed to correct undesirable aspects of video-telephony, the system needs to be relatively robust in all different kinds of conditions. This led us to use our test-plan, which was drawn up during the design stages of the project.

As a result of the testing, there will be an extensive period of re-design and implementation. Approaches that were prototyped upon the PC platform may prove infeasible on the DSP architecture and thus the system may need to utilise other algorithms to achieve its goal.

Once final testing takes place the system will be complete and robust in most conditions.

Design and implementation details

During the course of implementation there were two main issues regarding the design and structure of the PC prototype that we came across.

Firstly, in the algorithms that were used a number of different colour spaces were called upon to perform various areas of the algorithm. The daughter-board capture card was able to return images in the YCrCb colour space and libraries were available to perform an RGB conversion. For some major areas of the PC algorithm HSV was used and thus to perform a straight conversion without optimisation required the use of more processing cycles per loop. Also, the colour depth used for most of the PC algorithm was 24 bit per pixel (bpp), however the daughter-board was able to capture at 16 bpp. This is usually quite normal. As a result our algorithms and how they fit together may need to be re-structured.

Another hurdle we came across was the limitation as to the number of pixels available at once to the DSP. The TMS320C6711 has a limited amount of internal memory compared to the other processors in the C67x range. As a result, it is impossible to copy an entire frame into memory and manipulate it. Subsequently, this may make it more difficult to perform algorithms that perform non-sequential searches on an image, such as the face tracking algorithms.

Optimisations

The main route of optimisation is that of “linear assembly”. Optimisations are generally done on a small loop level basis. Code Composer Studio allows the inspection of small C code segments in assembly. This allows us to write code that was possibly more efficient. The pre-processor allowed us to insert linear assembly into our code which would be caught by the assembly pre-processor at link time (actually the “linking” stages are more complex than this and this is why such operations can be performed). Linear assembly could be described as “C-like” assembly and one can omit things such as parallelism directives and register/memory locations. Instead a memory location may be referred to by its variable identifier. This allows us to streamline code because we know what the compile was achieving and what actually needed to be achieved and thus we could omit or carry out any necessary steps to make it more efficient.

The FindFace module does not actually process the entire captured image. The FindFace module processes a psuedocapture image that contains a quarter of the pixels of the capture image. When the face is found, its dimensions are scaled up to match the dimensions of the capture image. This results in 75% less pixels to be processed by the FindFace module, which compensates for the overhead of sub sampling the capture image to the psuedocapture image.

Floating point operations have been replaced with integer operations, where applicable, such as the processing of the saturation and value components of the HSV colour model in the FindFace module. Integer operations are faster than floating point operations, although the C6711 has floating point capabilities.

Pointers to image pixels and pointer arithmetic were used to access the image pixels in order to avoid the overhead of multiplications, for each pixel in an image, that is associated with the standard method of indexing 2-dimensional arrays. Such optimisations were especially crucial to process the face ellipse in the suppression of the background. The background is first processed by moving a pointer along the region of the image that is below the rectangle that bounds the face ellipse. One pointer is then used to find the edge of the lower left quadrant of the face ellipse. Three pointers, one for each of the other three quadrants, are used in conjunction with the symmetric properties of the face ellipse to process the regions between the left and right edges of the image rectangle and the ellipse. The computationally inefficient operations of squares and square roots only have to be performed for the lower left quadrant of the face ellipse rather that every pixel in the display image. A pointer is finally used to process the region of the background that is above the rectangle that bounds the face ellipse.

Evaluation

VCEP is scheduled to be completed on June 7, 2001, therefore the evaluation of the system has only been performed on a PC platform to date.

Figure 12 Left Face Capture

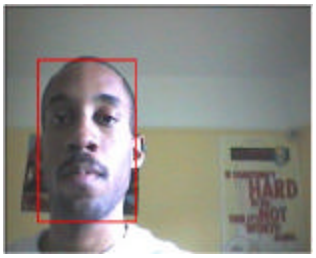


Figure 13 Left Face Found

Figure 14 Left Face Predisplay

Figure 15 Left Face Display

The system correctly centres the face from the left of the capture image in the predisplay image. Since the face is close to the left edge of the capture image the left side of the predisplay image had to be filled to the left. The background is suppressed in the display image.

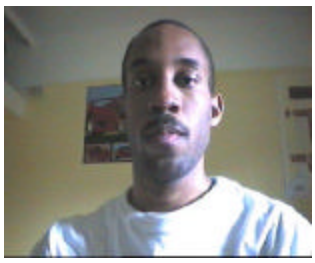


Figure 16 Top Face Capture

Figure 17 Top Face Found

Figure 18 Top Face Predisplay

Figure 19 Top Face Display

The system correctly centres the face from the top of the capture image in the predisplay image. The background is suppressed in the display image.

Figure 20 Centre Face Capture

Figure 21 Centre face Found

Figure 22 Centre Face Predisplay

Figure 23 Centre Face Display

The face is already centred in the capture image but it is closer to the camera than in the other capture images. When the face is centred in the predisplay image, the bottom, left and top of the image needs to be filled. When the background is suppressed in the display image the effects of the filling in the predisplay image are eliminated.

The VCEP system functions correctly on still images and runs at 12 fps on the PC platform.

Conclusion

The Video Communications Enhancement Processor (VCEP) system has been completed on the PC platform and is currently being implemented on the TMS320C6711 DSP.

The VCEP system removes some of the artefacts from the video stream before transmission to the receiver. VCEP addresses the problems of unstable images due to motion of the phone and distracting detail in the background. VCEP does not process any audio.

The VCEP system consists of three main modules that locate the face, stabilize the video stream and suppress the background. The FindFace module uses the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm to locate the face. The AntiJitterFace module resamples the capture image so that the face size is standardized. This module also redraws the resampled image so that the face is in the centre of the display image. The SuppressBackground module suppresses background detail by dimming half of the background and replacing the other half with a uniform grey colour.

The VCEP system currently runs at 11 fps on the PC platform and correctly performs the tasks for which it was designed.

Bibliography/References

[BRAD98] Computer Vision Face Tracking For Use in a Perceptual User Interface – Gary R. Bradski, Microcomputer Research Lab, Santa Clara, CA, Intel Corporation – Intel Technical Journal Q2 1998

[FINK98] Finkelstein, D.A, "A Comparative Evaluation of the Algorithms for Colour Identification", M.Sc Dissertation, UMIST Department of Computation, 1998

[GOWI87] Gonzalez, R, Wintz, P., "Digital Image Processing", Addison-Wesley, 1987

[MORR00] Morris,T "Multimedia Systems", Springer-Verlag London Limited.

[BRAD] http://www.cs.ucsb.edu/PUI/PUIWorkshop98/Papers/Bradski.pdf

[FC] www.ece.cmu.edu/~jkapur/face.html

[HIPR] http://www.dai.ed.ac.uk/HIPR2/wksheets.htm

[INTE] http://developer.intel.com/technology/itj/q21998/articles/art_2.htm

[MASO] http://mason.gmu.edu/~mvasque1/img_prob.html

[TR] www.is.cs.cmu.edu/ISL.multimodal.modelgaze.tracking.html

Appendix: Project Plan