ABSTRACT
Stitching frames from a video sequence has become a good alternative to the traditional method with only a few photographs. The big overlapping between consecutive video-frames makes it possible to construct high-quality panoramas. Most stitching methods require a vast amount of computational resources that make them not suitable for mobile devices.

We present a panorama stitching method that is able to create panorama images from a video sequence. The application is able to stitch frames on-the-fly, with low memory consumption. The resulting images can cover a field of view in excess of 360 degree images. Lack of a floating-point-unit in most mobile phones has made necessary the use of fixed-point mathematics to achieve better performance and speed.

The complete application has been implemented for several Symbian mobile phones. Features of the software include among others: different resolution choices, on-the-fly stitching, best frames selection and inter-frame luminance balancing.

1. INTRODUCTION
Modern mobile phones present some built-in devices and a good amount of computational resources. However, the lack of floating-point arithmetic units and the small memory size directs to use carefully tailored algorithms and implementations to make them usable in mobile phones.

To solve this issue, we started converting a previously published algorithm [1, 2] to standard C code that could later be executed on a Symbian environment. Then, a framework application that is able to capture video frames with low memory needs has been created. A full fixed-point arithmetic library with assembly functions specially made for ARM processors has been developed to achieve a much faster application capable of create mosaics from a video sequence while capturing the frames.

2. RELATED WORK
Although it is possible to find some elementary mosaicking capabilities in some devices, an advanced solution to quality stitching is lacking. SonyEricsson has had a panorama shooting mode [3] in their phones for a while now, although it is only capable of joining a few images. However, some commercial mosaicking applications have been appearing recently, there is only a restricted amount of information available about them. The application that most closely resembles our software is BitSide Panoman [4]. Nevertheless, in Panoman, a fixed size panorama image is constructed while in our application the whole panorama is constructed on the fly with no theoretical limit on size and resolution.

3. THE FLOATING-POINT VERSION OF OUR PROGRAM
The floating-point version of our program takes several raw video-frames from the built-in mobile phone camera as an input. After a frame is acquired it is immediately cropped from left and right side where the quality of the image is usually much worse. Then the frame is stored on an array of frames.

Our program starts then to register every cropped frame against the previous one, previously converting it to gray-scale. The method of Vandewalle [5] is used for this purpose. We compute also the blur of every frame to be able to choose the most suitable ones.

In final phase of the algorithm, the selected frames are stitched together into one image and saved in a mass storage device.

3.1. Projection and Camera Motion Model
The manifold projection, originally introduced by Peleg [6] has been used as a camera motion model. In manifold projection, a thin stripe is taken from the center of each frame to be used in the construction of the mosaic image. If there is no significant change of scale
or motion parallax, the frames can be registered accurately by a rigid camera motion model. Manifold projection is quick to process since frames do not have to be projected onto a different surface and offers excellent image quality in the resulting mosaic.

3.2. Image Registration

The method used for the image registration is based on the one recently published by Vandewalle [5] that is able to register aliased, rotated and translated frames. Due to the lack of computational resources for that task, we use a mask of about half of the image height of the frames to perform the computation.

It has been experimentally found that a 128x128 mask in a 320x240 frame-sequence is sufficient to obtain good results. To make the registration algorithm much more robust, a Tukey window function has been implemented to filter the gray-scale image.

To estimate the shift between two frames with the Vandewalle’s method, a general-purpose two-dimensional FFT has been used. The result of the frequency components of the two frames, are divided element by element and this result is normalized, dividing again each element by its amplitude.

Then the inverse FFT function is called and the amplitude of the result is computed to obtain a registration matrix. The greatest element of that matrix will show the number of pixels that we should take as the shift.

3.3. Blur Detection

The amount of blur in the frame is acquired by summed derivatives. This method was used recently by Liang [7] and consists in summing together the derivatives of each row and each column.

This method of blur calculation produces a single number that expresses the amount of high-frequency detail in the image. When taken from the same area of the image, this usually means that the frame with more high-frequency detail is sharper than the others. In some occasions, the difference between consecutive frames due to changes in the landscape (i.e. a moving object) can distort the result.

3.4. Frame Selection

The frame selection is done in a simple way to improve speed and performance. Based in the method published by Boutellier [1], a shift threshold of about half of the width is selected. The best frame between the previous selected one and the one meeting the threshold is chosen.

The only factor to choose the best frame in our method is the result of the blur detection algorithm.

3.5. Exposure correction

Exposure correction is needed because most mobile phones’ video devices adjust automatically the camera’s aperture and exposure time to prevent over- or underexposure of the frames. Although automatic adjustment makes it pleasant to watch a video on playback, it makes the creation of a panorama much more complicated.

Best results are obtained if camera API allows keeping the exposure fixed. In this case only local correction that works only on individual pixels is needed. This correction is usually referred to blending and it is described in the next section.

If exposure cannot be fixed during the capture, global correction of luminance must be implemented. To perform this correction, the average luminance value is computed over the pixels belonging to the overlapping area. Every pixel of the current frame is corrected according to the value of the average on the previous one, adding or subtracting this difference to green, blue and red values of the frame to stitch.

3.6. Blending

To blend the final image, the size of the resulting image is reserved on the memory whenever a frame is stitched according to the value of the estimated shift.

Then, a linear function that gradually merges one frame to the next (changing the weight assigned to each one) is used. A similar method has been used in Szeliski [8] under the name of feathering.

This method prevents some vertical stripes to be visible between the individual frames that compose the panorama. A seamless image is created.

4. FIXED-POINT VERSION OF OUR PROGRAM

The first fixed-point version of our program works in a similar way as the floating-point version does. The main differences respecting to the previous version are the speed and performance. Despite obtaining images with the same quality, the use of some fixed-point routines and approximations makes the application from 10 to 20 times faster on a device without a floating-point unit.

4.1. Fixed-point routines

A library that computes basic fixed-point mathematic operations has been developed. Inline assembly
functions have been used for addition, subtraction, multiplication and scalar division.

4.2. Image Registration

Image registration is done in a similar way as in the floating point version. Shift estimation is the most algorithmically complex task and the most costly, what makes it the most suitable to be improved.

To perform this operation, the values of each pixel in the grayscale image, are converted to Q15.16 fixed-point values with a single bit-shifting operation. This conversion works fast and gives enough precision.

4.2.1. Tukey Window Filtering

To avoid repeated calculation of the Tukey window coefficients, a lookup-table is constructed with fixed values for our desired size. This table can be pre-calculated on a desktop computer or just on the startup of our program, whenever the mask size or resolution is changed. To perform the filtering, a table lookup is performed, selecting the value needed to perform the filtering on each pixel.

4.2.2. Direct and Inverse Fast Fourier Transform

A fixed-point version of a general-purpose two-dimensional Fast Fourier Transform has been used.

The performance of this operation in fixed point can be up to 25 times faster than the same algorithm in floating-point for a mobile phone without a dedicated chip.

4.2.3. Complex Number Division

When working with fixed-point mathematics there are two ways of reducing the risk of overflow and underflow. One way is to increase the dynamic range of the fixed-point chosen format and the other is to rewrite the complex division algorithm.

We chose to use the algorithm showed by R.L. Smith [9, 10], achieving a much more robust operation with low computational cost.

4.2.3. Complex Number Magnitude

The calculation of the magnitude of a complex number requires in the traditional algorithm the use of 2 multiplications, one addition and one square root.

To avoid these operations, an approximation has been used. It is possible to get the magnitude of a complex number by adding the biggest of the components (imaginary or real) and half of the smallest one. This approximation gives a maximum error of 20% in worst case (both components are equal). Although this tolerance can affect some other calculations and result, it has been experimentally found that the increment of the error rate is around the 10% of the value with the classic implementation. On the other hand, using this fast method, offers the possibility of working with a higher frame-rate while computing on-the-fly. The increment of the frame rate results in less errors due to the decrease of the shift between frames.

To perform this operation, only a comparison, and addition and a bit-shift are needed, making the speed of this operation hundreds of times faster than the traditional implementation with floating point numbers.

4.5. Frame Selection, Exposure Correction and Blending

The rest of the operations like frame selection, exposure correction and blending, are performed in the same way described in the floating-point program section.

Minor adjusting has been required in order to use only fixed-point data type. Index and weight computing used for blending and exposure correction have been implemented using integers.

4.6. Error Handling in Image Registration

Some of the compromises that have been made accordingly to the mobile environment constraints can lead the image registration to some undesired errors.

To cope with these errors, a simple error-correction method has been used: each time a frame is registered, the motion vector is compared to the motion vector of the previous frame. If the difference between the current motion vector and the adjacent surpasses a certain threshold, the motion vector is replaced with the previous one. This works very well in panorama imaging where the usual motion trajectory of the camera is close to linear.

5. ON-THE-FLY STITCHING VERSION OF OUR PROGRAM

The reduced RAM capacity on mobile phones makes on-the-fly stitching procedures more complicated and calls for a carefully implemented memory management scheme. All the image processing should be done following a cascade structure with the help of some buffers to save previous data.

In this version of the program, first frame is captured and cropped and its frequency components and blur are calculated and stored on the memory. Next frames are acquired then asynchronously. Just after each frame is acquired, blur is computed, frame registration is performed and a decision about the quality of the frame is then made. If quality is better than that of the previous
frame (based on blur detection), exposure correction and blending algorithms are called.

The result image is growing frame by frame each time the blending function is called. There is no need to store several frames in memory at any time; only the last computed one and its features, its frequency components, its shift estimation and the total amount of shift in this moment, as well as the evolving final image.

When the user stops capturing the frame sequence that will become the mosaic, current resulting image is stored on a mass storage device, memory objects are deleted and application is reset.

6. EXPERIMENTAL RESULTS

Our mosaicking program has been tested with a wide variety of freehand sequences and it has proven to be very robust thanks to the excellent registration method. The main strengths of our program are the speed and sharpness, resolution and size of the final images.

Over 360 degree panoramas like the one displayed on Figure 1 have been created on the fly with an average speed of 3 frames per second.

The sharpness of the results can be seen best in Figure 2. A careful comparison between the images created by Panoman and our application shows that our method creates sharper and bigger results.

7. CONCLUSIONS

We have presented an implementation that creates good quality panoramas and runs on low-resource devices like mobile phones. Moreover, our method outperforms a state-of-the-art method in fields as speed, resolution and sharpness. Our application is being improved constantly. Motion detection, rotation correction and better frame correction are under development.

8. REFERENCES

Figure 1. An Example 360 degree panorama.

Figure 2. A detailed section of a panorama.