Real-time photogrammetric stitching of high resolution video on COTS hardware

Jason de Villiers
Council for Scientific and Industrial Research
Pretoria, South Africa
Telephone: +27 (0)12 841 4860
Email: jdvilliers@csir.co.za

***Abstract*—It is often necessary to stitch images together into a larger image; examples include generating a panoramic image, building an aerial terrain photo, and wide area surveillance. The latter is an example in which this stitching must happen in real time, i.e. less than one frame period. If further constraints are placed on the system such as operation in poor contrast conditions and multiple targets moving against a changing background in the system's field of view, yet it must still operate deterministically, then current predominant methods using image registration do not suffice. These methods typically require a large overlap between adjacent cameras in a static staring array, and involve manipulating the individual video frame images by means of rotation, translation and scaling until detected feature points optimally match. This is non-ideal because more cameras are needed due to the overlap of the adjacent cameras, the requirement to determine salient image points (which may not be evident in poor contrast conditions), the iterative manner used to align images' salient points, the frequent neglecting of lens distortion effects and the inability to align two cameras if the one between them fails. Photogrammetric methods have been used before to rectify these short comings, but have always required expensive, bulky, custom processing hardware. This paper shows that it is possible to stitch an array of minimally overlapped high resolution wide angle cameras irrespective of scene content and lighting changes in a deterministic manner via novel use of commercial graphics processing hardware and a parallelized algorithm.**

I. Introduction

Image registration is the process of determining the transformation required to align two images. These transformations can have varying degrees of freedom (DOF) depending on their complexity, with the search spaces and computation required to perform the transformations increasing accordingly. At each stage in the refinement it is necessary to determine how well the images overlap using a metric such as the sum of squared differences. The landmark Lucas-Kanade algorithm [1] simplifies the search problem using certain assumptions and then converges to the translation delta vector via Newton-Rhapson iteration. The algorithm has a coarse to fine approach using Gaussian filter generated down-sampled images. In this work only translation and rotation of the images was considered for image registration purposes, and implemented [2] as two spatially separated pure translation transformations.

Photogrammetry is the process of determining 3D information from one or more images of an object. It involves the generation of vectors using the known/measured camera parameters. These parameters can include the focal length, field of view (FOV), optical axis intersection point (sometimes also called the principle point), pixel sizes, camera 6 DOF position (i.e. orientation and translation) relative to the world, and lens distortion parameters. Figure 1 depicts this situation for a distortion-free lens and also defines the axis system used in this paper, where the origin is located at the focal/convergence point of the lens. Photogrammetric techniques have been successfully used to stitch images together in real-time, some even having full spherical fields of view [3], [4]. However, the equipment to do this processing can be prohibitively expensive for general use [4].

This is where the modern graphics processor unit (GPU) comes to the fore. With a massive amount of increasingly programmable stream processors (NVidia's GeForce GTX 295 has 480!), many applications are seeing increases in performance of a few orders of magnitude compared to traditional implementations on the central processing unit (CPU) [5]. There are limitations on the type of algorithms which can expect these performance boosts, specifically with regards to the correlation and independence of the outputs. Owen et. al. [5] provide a thorough discussion on these limitations as well as the evolution of the GPU from a dedicated graphics rendering hardware to the flexible general purpose device we have today.

The mathematical notation used in this paper is as follows: A 3-dimensional vector, V_{abc}, is a vector from point b pointing to point a expressed in terms of its projection onto orthogonal coordinate system c. V_{abc} is used when the magnitude of the vector is unknown or unimportant. T_{abc} represents the translation or displacement of point a relative to point b. U_{abc} is a unit vector pointing in the direction of point b to point a. R_{ab} is a 3x3 Euler rotation matrix expressing the rotation of orthogonal axis system a relative to (and in terms of its projections on) orthogonal axis system b. Individual elements of 3 dimensional vectors are referred to as x, y, or z where as two dimensional vectors' elements are referred to as horizontal (h) and vertical (v) to avoid confusion.

The rest of this paper is organised as follows: Section II briefly describes the calibration process used on the array of cameras. Section III details the algorithm used to perform the stitching. Section IV describes the hardware and software used to evaluate the stitching algorithm and section V provides and

978-1-4244-4210-2/09/$25.00 ©2009 IEEE

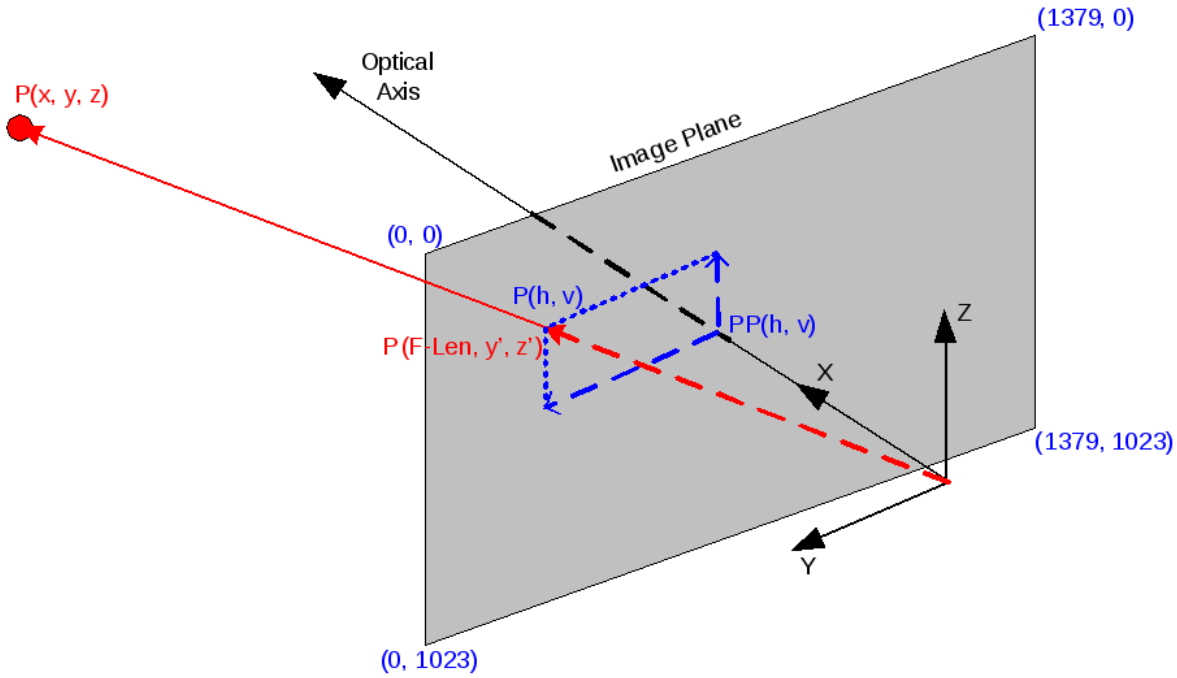

Fig. 1. Photogrammetric principles and axes definition.

discusses the results. Finally, Section VI places the results of this work in context.

II. Camera Calibration

Camera calibration is not the main purpose of this article, and thus the methods used in this work are presented in brief, for completion purposes. The determination of the characterization function from the distorted to the undistorted domains ($f^{undistort}$) which is required for camera calibration is covered in literature[6]. The same is true of the characterization from the undistorted to the distorted domains ($f^{distort}$)which is required in the stitching algorithm described in Section III.

The 6 DOF position of the camera relative to a reference on the staring array, the focal length and the optical intersection pixel position are required. In order to do this either of the metrics expressed in Eq (1) or Eq (2) may be numerically optimized using a robust algorithm such as Fletcher-Reeves [7] or Leapfrog [8]. The first metric is slower but more accurate due to the increased sensitivity to almost-parallel vectors provided by the (computationally intensive) inverse cosine function.

$$Metric = \sum_{i=0}^{n-1} \left(cos^{-1}(U^1_{icc} \bullet U^2_{icc}) \right) \tag{1}$$

$$Metric = \sum_{i=0}^{n-1} \left(1.0 - U^1_{icc} \bullet U^2_{icc} \right) \tag{2}$$

The metrics are a comparison between two bundles of vectors. Eq (3) shows how one of the vector bundles can be generated from the images of the optical reference jig, where the optical references have been accurately located and correctly identified. For this work a 10 by 9 chequer board was used with the intersections found as per Lucchese and Mira [9]. The asymmetry of the board together with heuristics such as 'number of neighbours' and 'distance from average' were used to automatically identify the corners and subsequently allocate the intersections to rows and columns so that they could be identified. The pixel dimensions are known from the data sheet, thus leaving the focal length and optical axis intersection point as the only unknowns. Good initial guesses for the numerical optimization are the manufacturer's claimed focal length and the distortion centre respectively.

$$\begin{aligned} I^u_i &= f^{undistort}(I^d_i) \\ V_{icc} &= \begin{bmatrix} Focal\_Len \\ (P_h - I^u_{i_h})pix\_w \\ (P_v - I^u_{i_v})pix\_h \end{bmatrix} \\ U^1_{icc} &= V_{icc} / \parallel V_{icc} \parallel \end{aligned} \tag{3}$$

where:

I^d_i = the 2D image pixel position of optical reference i,

$f^{undistort}$ = the predetermined lens undistortion characterization function,

P = the optical axis intersection pixel position,

pix_w = the width of the pixels on the camera's imager,

pix_h = the height of the pixels on the camera's imager,

$Focal\_Len$ = the exact focal length of the camera's lens, and

U^1_{icc} = a unit vector pointing from the camera to optical reference i.

The second vector bundle is calculated via Eq (4). It is assumed that the spatial offset of each reference point (T_{ijj}) on the optical reference jig is precisely known. Thereafter the unknown spatial offset of the jig (T_{jcc}) and the (also unknown) Euler rotation of the jig relative to the camera (R_{jc}) are used to determine and then normalize a vector to each optical reference. Initial values for the 6 DOF position of the board relative to the camera can be determined via a combination of approximate physical measurements and good judgement and may require tweaking if the calibration turns out poor. Also note that there is a singularity in both metrics if a planar optical reference jig is used and is placed perpendicular to the camera's optical axis.

$$\begin{aligned} T_{icc} &= R_{jc}T_{ijj} + T_{jcc} \\ U^2_{icc} &= T_{icc} / \parallel T_{icc} \parallel \end{aligned} \tag{4}$$

where:

T_{jcc} = the spatial position of the optical reference jig relative to the camera,

T_{ijj} = the spatial offset of the optical reference mark relative to the jig,

R_{jc} = the Euler rotation matrix of the optical reference jig relative to the camera, and

U^2_{icc} = a unit vector pointing from the camera to optical reference i.

The above determines the position of the reference jig relative to a camera, however if two cameras can be placed so that they see the same reference jig, it is a simple matter to determine their positions relative to each other (see Eq (5)) and this can be cascaded through the system so the cameras are known relative to one camera arbitrarily deemed the reference camera. For this work it was necessary to use temporary interleaving cameras for this cascading calibration process as the overlap of the FOVs was very small and only occurred some distance from the staring array.

$$\begin{aligned} R_{c_2c_1} &= R_{jc_1}R^T_{jc_2} \\ T_{c_2c_1c_1} &= T_{jc_1c_1} - R_{c_2c_1}T_{jc_2c_2} \end{aligned} \tag{5}$$

where:

R_{jc_1} = Euler rotation matrix of the jig relative to camera 1,

R_{jc_2} = Euler rotation matrix of the jig relative to camera 2,

$T_{jc_1c_1}$ = spatial offset of the jig relative to, and in camera 1's axis system,

$T_{jc_2c_2}$ = spatial offset of the jig relative to, and in camera 2's axis system,

$R_{c_2c_1}$ = desired Euler rotation matrix of camera 2 relative to camera 1, and

$T_{c_2c_1c_1}$ = desired spatial offset of camera 2 relative to, and in camera 1's axis system.

III. Stitching Algorithm

This section discusses the stitching algorithm used; it uses the photogrammetric parameters of the cameras in the static staring array in order to perform the stitching. Specifically, it uses the 6 DOF position of each camera relative to the stitching reference, the focal length of the camera, the pixel sizes and pixel coordinate of intersection point of the optical axis with the imager of the camera, and the pre-characterized undistorted to distorted function [6]. It has been designed so that each pixel of the output stitched image is independent of all the others and so can be implemented as a fragment shader on a graphics processing unit (GPU) and attached to a texture. This allows the GPU's horde of stream processors to evaluate the algorithm in parallel.

A. Mesh generation form pixel coordinates

The first step involves generating a mesh of points in free space. Here we generate a mesh on a sphere of radius R centered at the reference point relative to which the 6 DOF positions of the cameras are known. The value of R can be tweaked so that the stitch is in focus in the overlapped regions for the desired operating range. The spherical (instead of planar) mesh allows for FOVs greater than $180°$. Eq (6) uniquely associates an azimuth and elevation with each pixel of the output stitch:

$$\begin{aligned} Az &= Az_{min} + hAz_{step} \\ El &= El_{min} + vEl_{step} \end{aligned} \tag{6}$$

where:

(h, v) = the image coordinate of the pixel in the output stitched image,

Az_{step} = the desired horizontal angular resolution of the final output stitch,

El_{step} = the desired vertical angular resolution of the final output stitch,

Az_{min} = the desired azimuth of the left most column of the final output stitch, and

El_{min} = the desired elevation of the top most row of the final output stitch.

Thereafter Eq (7) generates a point in free space using the calculated azimuth and elevation and the desired stitching radius:

$$T_{prr} = \begin{bmatrix} Rcos(Az)sin(El) \\ Rcos(Az)sin(El) \\ Rsin(El) \end{bmatrix} \tag{7}$$

where:

T_{prr} = position of the point relative to and expressed in the reference system, and

$R =$ the radius of the stitching sphere.

B. Point projection onto camera image plane

Now that we have a point in free space we need to determine which pixels (if any) of each of camera correspond to that point. This step and sections III-C and III-D are done for each of the cameras in the staring array. Eq (8) expresses the point in the camera's axis system:

$$T_{pc_nc_n} = R^T_{c_nr}(T_{prr} - T_{c_nrr}) \tag{8}$$

where:

$T_{pc_nc_n} =$ position of the point relative to and expressed in the camera's axis system,

$T_{c_nrr} =$ displacement of the camera relative to and in terms of the reference system,

$R_{c_nr} =$ Euler matrix generated from the orientation of the camera relative to the reference system, and

$n =$ the index of the current camera being considered.

Eq (9) then clips the vector to this point, so that it refers to a point in the image plane of the camera.

$$V_{pc_nc_n} = T_{pc_nc_n}(Focal\_Len_n / T_{pc_nc_n}.x) \tag{9}$$

where:

$Focal\_Len_n =$ the focal length of the camera, and

$n =$ the index of the camera being considered.

C. Conversion to pixel domain

We now know the spatial offset of the point horizontally and vertically from the point in the image through which the optical axis passes. All that remains is to scale this offset to pixels, account for the intersection point and the different positive directions within the image (typically the horizontal axis increases from left to right, and the vertical axis is positive downwards). Eq (10) shows how this is done.

$$U_n = \begin{bmatrix} I_n.h - (V_{pc_nc_n}.y / pix\_w_n) \\ I_n.v - (V_{pc_nc_n}.z / pix\_h_n) \end{bmatrix} \tag{10}$$

where :

$I_n =$ camera's optical axis intersection pixel position,

$pix\_w_n =$ pixel width of the camera's imager,

$pix\_h_n =$ pixel height of the camera's imager, and

$U_n =$ ideal distortion-free pixel position, and

$n =$ the index of the current camera being considered.

D. Inverse distortion modelling

Real world cameras exhibit some amount of lens distortion which typically increases with larger FOV lenses. Not taking this into account would lead both to blurring in the overlapped regions and to curving of lines that don't pass through the distortion centre of a camera. Worse still is that two parallel lines on either side of an overlap would bend different ways! Eq (11) uses the modelling derived by de Villiers, Leuschner and Geldenhuys [6] to avoid these problems. Note that the distortion function is camera specific.

$$D_n = f_n^{distort}(U_n) \tag{11}$$

where :

$f_n^{distort} =$ current camera's undistorted to distorted domain characterization function,

$D_n =$ the pixel position in the camera's image, and

$n =$ the index of the current camera being considered.

E. Blending of point

Now we have the cameras' pixel positions corresponding to the sphere point generated from the current pixel's ordinate in the stitch. We need to combine these optimally so that the overlap region is as indistinct as possible (especially if the images have different apparent brightness) and so that cameras that cannot see the point do not contribute to, or slow down this last stage of the stitch generation. Eq (12) achieves this by giving higher priorities to pixels closer to the center of an image, thus creating a smooth transition in the overlapped regions.

$$w_n = D_n.h(C_n.h - D_n.h) \tag{12}$$

where:

$C_n.h =$ horizontal centre of camera n in pixels,

$D_n.h =$ horizontal ordinate camera n's pixel

$w_n =$ blending weight associated with camera n, and

$n =$ index of the current camera being considered.

Eq (13) uses the weights from each camera (which have subsequently been clipped to zero if they were negative, or the calculated pixel positions fall outside the camera's valid range) to linearly combine each camera's contributing pixel. The addition of a small value to the denominator removes the necessity of checking for division by zero which would occur when no camera can see the point associated with the current pixel in the output stitched image.

$$S[h,v] = \frac{1}{10^{-12} + \sum_{i=0}^{m-1} w_i} \sum_{i=0}^{m-1} \Big(w_i I_i[D_i] \Big) \tag{13}$$

where:

$I_i =$ camera i's image (2D array of intensity values)

$m =$ the number of cameras in the staring array

$w_i =$ the weight associated with camera i

$D_i =$ the pixel position for the ith camera

$S[h,v] =$ the final intensity value for the current pixel of the stitch

IV. COTS Implementation

The stitching algorithm was implemented on a standard, commercial off the shelf (COTS) personal computer (PC) running Debian Linux. The PC had an Intel Quad Core2^{TM} 2.83GHz processor, 3.2GB of RAM, an NVidia GeForce

GTX280 GPU) and a dedicated gigabit Ethernet link to each of the machine-vision cameras. The algorithm was implemented as a shader in the OpenGL shading language (GLSL) so that use could be made of the GPU. Open Scene Graph (OSG) was used to handle the loading of the shader, and the download of newly received camera images to texture/video memory.

The Prosilica GE1380 cameras that were used have a resolution of 1360 by 1024 pixels and had an external synch creating a 20Hz frame rate, and are capable of 12bit grey-scale colour depth although they were set to provide 8 bits. They had Fujinon HF35SA-1 35mm lenses which provided an approximate 15° horizontal by 10° vertical FOV. Physically, the cameras were arranged as a horizontal 4 by 1 array with approximately 1° overlap.

The stitch output size was set so the horizontal width coincided with the maximum allowed texture dimension of 4096 pixels and a horizontal FOV of 60° and 12° vertically. This yielded azimuth and elevation steps of 0.01465° and a total stitch dimension of 4096 by 820 pixels.

It is worth noting that OpenGL 2.0 and later compliant graphics cards are required to have hardware support to handle out-of-bounds texture access; bi-linear interpolation for non-integer texture value look-ups, matrix and vector addition and multiplication in up to four dimensions, trigonometric functions, swapping and random access to vector elements, and step functions (which are two parameter functions that return either 0 or 1, depending on whether the second parameter is larger than the first). All of these features were used in implementing the algorithm to maximally take advantage of the GPU hardware.

V. Results

Figure 2 shows four images from the calibrated staring array. The contrast in these images is particularly poor, and the moving ocean and clouds are present in all the overlap regions. Figure 3 shows the result of stitching these pictures together using the proposed algorithm, the black outline represents the 60° by 12° FOV.

The hardware described in Section IV was able to perform this stitching at a rate of 360 frames per second (fps) according to the built-in OSG statistics collector. Compared to current commercially available systems the hardware can easily be purchased for 10% of the reported price[4] of the processing equipment yet processes more than 12 times more pixels up from the 100 million pixel/sec[3] to 1.21 billion pixels/sec (although the data from the cameras were only arriving from the quartet of cameras at 113 million pixels/sec).

Compared to registration based stitching methods, a GPU-CPU hybrid Lukas-Kanade implementation [2] using two 256 by 256 regions of interest (to determine translation and rotational differences) was able to stitch two images at 50fps, which is 7 times slower to perform 3 times fewer stitches.

VI. Conclusion

This paper addressed the problem of real-time video stitching. It has been demonstrated that real-time video stitching can be done on freely available COTS hardware at frame rates well beyond that which the human eye can distinguish. In addition this can be done on video that has minimal overlap, has moving objects in the overlapped area, and has low contrast or poorly determinable feature points. Compared to commercially available photogrammetric equipment, higher resolutions are possible on less expensive hardware, and a performance increase of over 1209% was experienced over that reported in recent literature. It was also seen that GPU photogrammetric stitching is up to 21 times quicker than GPU registration based stitching.

Low-cost, high resolution, high frame-rate wide-area surveillance is now feasible and can provide greater situational awareness in a variety of scenarios ranging from home security to crime prevention and automated military asset early warning systems.

Acknowledgment

The author would like to thank Armscor for funding this work and Bernhardt Duvenhage for his advice regarding image registration.

References

[1] B. Lucas and T. Kanade, "An iterative image registration technique with application to stereo vision," *International Journal on Computer Vision and Image Processing*, pp. 674–679, 1981.

[2] P. Jeebodh, "Super resolution," CSIR, Tech. Rep. 6700-OPTO-38601-01, 2008, restricted document.

[3] F. Kahn, M. Chapman, and J. Li, "Camera calibration for a robust omni-directional photogrammetry system," in *Proceedings of the 5th International Symposium on Mobile Mapping Technology*, ser. MMT07, 2005, pp. 1–8.

[4] W. Ma, "Riding shotgun with google street views revolutionary camera," http://www.popularmechanics.com/technology/industry/4232286.html?page=2, 2007.

[5] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Philips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[6] J. de Villiers, F. Leuschner, and R. Geldenhuys, "Centi-pixel accurate real-time inverse distortion correction," in *Proceedings of the 2008 International Symposium on Optomechatronic Technologies*, ser. ISOT2008, vol. 7266, 2008, pp. 1–8.

[7] R. Fletcher and C. Reeves, "Function minimization by conjugate gradients," *Computer Journal*, vol. 7, pp. 140–054, 1964.

[8] J. Snyman, "An improved version of the original leap-frog dynamic method for unconstrained minimization: Lfop1(b)," *Applied Mathematics and Modelling*, vol. 7, pp. 216–218, 1983.

[9] L. Lucchese and S. Mira, "Using saddle points for subpixel feature detection in camera calibration targets," in *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, vol. 2, 2002, pp. 191–195.

(a) (b)

(c) (d)

Fig. 2. Four input images from the Prosilica GE1380s

Fig. 3. Output stitched image