Indexing Uncoded Stripe Patterns in Structured Light Systems by Maximum Spanning Trees

Willie Brink[1,2], Alan Robinson[2], Marcos Rodrigues[2]
[1]Mobile Intelligent Autonomous Systems, CSIR, Pretoria, South Africa
[2]Materials Engineering Research Institute, Sheffield Hallam University, Sheffield, UK
wbrink@csir.co.za, a.robinson@shu.ac.uk, m.rodrigues@shu.ac.uk

Abstract

Structured light is a well-known technique for capturing 3D surface measurements but has yet to achieve satisfactory results for applications demanding high resolution models at frame rate. For these requirements a dense set of uniform uncoded white stripes seems attractive. But the problem of relating projected and recorded stripes, here called the Indexing Problem, has proved to be difficult to overcome reliably for uncoded patterns. We propose a new algorithm that uses the maximum spanning tree of a graph defining potential connectivity and adjacency in recorded stripes. Results are significantly more accurate and reliable than previous attempts. We do however also identify an important limitation of uncoded patterns and claim that, in general, additional stripe coding is necessary. Our algorithm adapts easily to accommodate a minimal coding scheme that increases neither sample size nor acquisition time.

1 Introduction

Structured light is a non-contact active 3D surface acquistion method that projects a predetermined pattern of light onto a target surface and records the resulting deformation [5]. The recorded pattern is retrieved and combined with the geometric relationship between the recording device and light source to infer the shape of the target surface. In some applications it is imperative that sufficient information for dense reconstruction is acquired within the timebase of a single video frame. Examples include non-rigid surface scanning (e.g. human faces in a biometric setting [2]) and measuring surface dynamics (where successive frames of a recorded sequence should reveal changes in surface shape over time [11]).

A useful light pattern for these requirements is a dense set of parallel stripes that covers the region of interest in a single projection. Although this allows fast data acquistion, accuracy in the reconstruction depends on establishing correct correspondence between projected and recorded stripes. Methods have been devised to discriminate between individual stripes, by colour [12], width [1] and time-dependent sequences [4]. See [10] for a review. These coded patterns have limitations however: colour cannot be applied consistently to surfaces with weak or ambiguous reflectance, for width coding the resolution is less than for uniform narrow stripes, and time-coded patterns require multiple scans and are effective only for scanning inert objects.

BMVC 2008 doi:10.5244/C.22.57

In light of these, patterns of *uncoded* (i.e. homogeneous) stripes seem attractive. For these patterns, however, the correspondence problem has proved to be difficult to overcome reliably. Following [9] we refer to it as the Stripe Indexing Problem. Previous attempts assumed some level of surface continuity that preserves stripe adjacency to some extent in the recorded image [8, 9]. The pattern would then be indexed piecemeal, relative to indices already determined in local neighbourhoods.

In this paper a new method is presented that yields a significant improvement in solving the indexing problem for uncoded stripe patterns. We follow similar assumptions but first regard all adjacencies among recorded stripes in the form of a graph before choosing an optimal solution. This, in contrast to the previous methods, avoids errors that could propogate uncontrollably across a solution.

In section 2 the structured light system is described, and a dependency between stripe indices and surface points is derived. Section 3 deals with indexing the recorded pattern and describes our algorithm. A problem inherent to uncoded patterns is identified in section 4, we show how the addition of a minimal coding scheme can eliminate it while retaining high sample density and fast acquistion time, and we adapt our algorithm accordingly. Some results are given and discussed in section 5.

2 Surface scanning with stripe patterns

Figure 1 depicts a structured light system, with a single horizontal stripe on the left and a pattern of parallel stripes on the right. Every recorded stripe indicates a beam of light from the projector that hits the target surface and reflects to the camera's sensor. If the pattern can be found in the image and correctly identified (indexed) points in the image can be mapped to points in space to reconstruct the target surface.

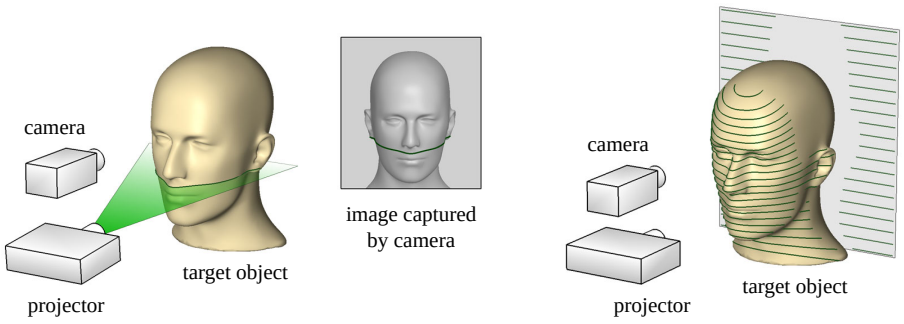

Figure 1: A structured light scanner projects a pattern of light, (left) a single horizontal stripe and (right) multiple stripes, onto a target surface. A camera records the reflected pattern from which the shape of the surface can be obtained.

We define a Cartesian coordinate system in relation to the projector, as shown in Figure 2 (left). The Z-axis coincides with the central projection axis and the X–Z plane with the horizontal sheet of light cast by the projector. The origin is fixed at an arbitrary known distance $D_p > 0$ from the centre of the projector lens.

The camera is positioned on top of the projector such that: its central axis is parallel to the Z-axis and lies in the Y–Z plane; the centre of the camera lens is at point $(0, D_s, D_p)$

with $D_s > 0$; and the camera is not tilted, so that any point in the Y–Z plane is captured to the central image column. This parallel arrangement differs from more conventional ones (see e.g. [5, 9, 12]) where the projector and camera axes intersect. But it simplifies analysis and provides parallel epipolar lines [7] without image rectification.

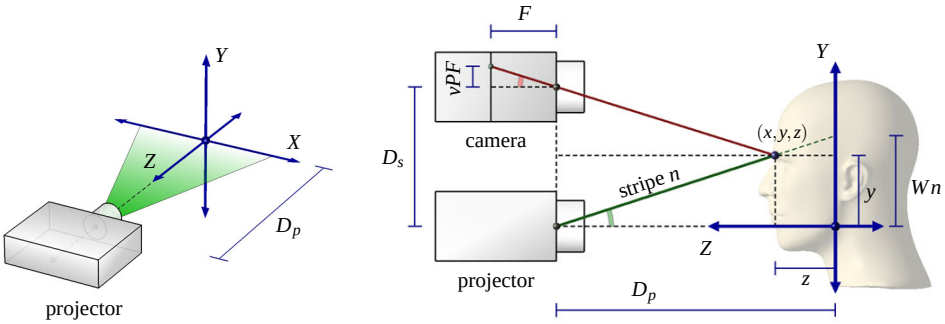

Figure 2: A coordinate system (left) is defined in relation to the projector. The geometry of the scanner, viewed here down the X-axis (right), is used to derive a mapping between points in the image and surface points.

We assume that stripes are evenly spaced on the X–Y plane and parallel to the X-axis. Integer *indices* are assigned to the stripes in the pattern, increasing in the direction of the Y-axis, such that the one in the X–Z plane has index 0. If W indicates the spacing between successive stripes on the X–Y plane then stripe n intersects the Y-axis at $(0, Wn, 0)$.

For a pixel $p = (r,c)$, at row r and column c, in the $M \times N$ image we determine centred coordinates $v = r - \frac{1}{2}(M+1)$ and $h = c - \frac{1}{2}(N+1)$. Supposing that each pixel on the physical sensor plane of the camera is a square of size $PF \times PF$, with F the focal length, p is captured at a distance vPF from the camera's central axis in the vertical direction (see Figure 2), and hPF in the horizontal. The surface point $s = (x,y,z)$ corresponding to p is therefore on the ray through that point and the camera's focal point. If it is further known that p lies on recorded stripe n then s is also on the plane parallel to the X-axis through points $(0,0,D_p)$ and $(0,Wn,0)$. By similar triangles,

$$\frac{y}{D_p - z} = \frac{Wn}{D_p} \quad \text{and} \quad \frac{D_s - y}{D_p - z} = \frac{vPF}{F}, \tag{1}$$

which yield the following $(v,h,n) \mapsto (x,y,z)$ mapping (x is determined through similar triangles in a view of the system down the Y-axis):

$$x = hPD_pK, \;\; y = WnK, \;\; z = D_p(1-K), \quad \text{with} \;\; K = \frac{D_s}{vPD_p + Wn}. \tag{2}$$

It can be shown that for point s to be in front of the projector, such that $z < D_p$, we have $vPD_p + Wn > 0$. Note also that here the camera is modelled as an ideal pinhole camera, and spherical distortion [6] should be dealt with prior to applying (2).

Clearly, successful measuring of the target surface is dependent upon correctly determining the index n of every stripe in the image. Figure 3 shows the required operations

on an example[1]: recorded stripes have to be located and then indexed (different indices are depicted in colours of a repeated sequence), which produces a collection of indexed stripe pixels each having coordinates of the form (v,h,n).

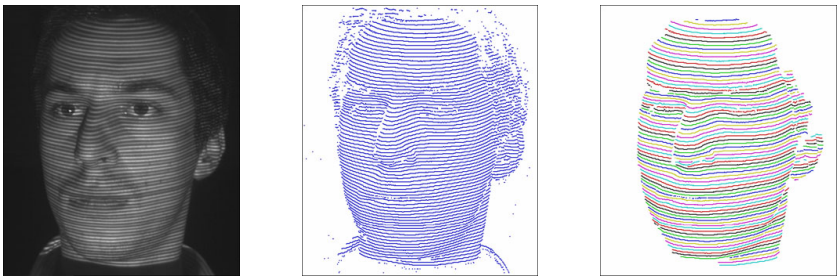

Figure 3: The stripes in a recorded image (left) have to be located (middle) and indexed (right) so that surface points can be determined through (2).

To locate the stripes in the image a simple linear search for local maxima in luminance values of every column can be sufficient. Local thresholding also reduces the effects of noise. A binary matrix P is constructed such that $P(r,c)$ is 1 if the pixel (r,c) is found to be a stripe pixel and 0 otherwise.

3 Indexing by maximum spanning trees

If a stripe is projected onto a flat surface parallel to the X-axis then the recorded stripe is perfectly horizontal and appears in every image column. Loosely speaking, the more the surface "deviates from flatness" the more the recorded stripe varies in the vertical direction. If it is assumed for now that the target surface is sufficiently smooth it should be possible to follow a stripe across a number of columns in the image.

Furthermore, the epipolar constraint [7] ensures that a particular stripe index is present *at most once* in every image column, and that the sequence of stripe indices in a column *decreases monotonically* from top to bottom (if the surface is opaque and free of holes). Note that indices can be absent from certain columns due to occlusions or weak surface reflection.

We proceed to define a set of neighbours for every stripe pixel in P, by heading west (w), east (e), north (n) and south (s). The w-neighbour of stripe pixel (r,c) is the stripe pixel in the preceding column $(c-1)$ that shares at least one corner with (r,c), if such a pixel exists. The e-neighbour of (r,c) is the stripe pixel in the next column $(c+1)$ that shares at least one corner with (r,c), if such a pixel exists. The n-neighbour of (r,c) is the very next stripe pixel encountered when moving upwards in column c, and the s-neighbour is the very next stripe pixel encountered when moving downwards. In Figure 4, for example, the w-, e-, n- and s-neighbours of pixel 4 are pixels 3, 5, 1 and 6 respectively, and pixel 5 does not have an e-neighbour.

[1]The pattern in Figure 3 (left) is not completely uncoded. The darker stripe across the upper lip is a necessary reference whose index is determined prior to scanning. All other stripes can then be indexed relative to it. In section 3, however, we do not assume the presence of any coding and consider the located stripes without any additional information.

Two guidelines for indexing are formulated: (1) a stripe pixel and its *w*- or *e*-neighbour are likely to have the same index; (2) the *n*-neighbour of a stripe pixel with index i is likely to have an index of $i+1$, and the *s*-neighbour an index of $i-1$. We stress that these are guidelines, not rules. In Figure 4, for example, pixel 2 is the *n*-neighbour of pixel 7 but its index may be 2 more than that of pixel 7.

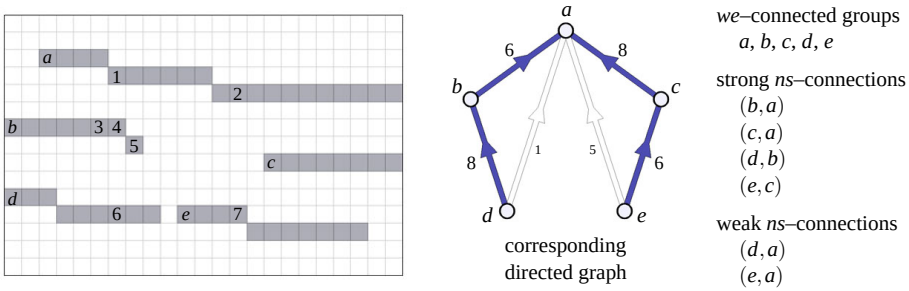

Figure 4: Illustrating stripe pixel neighbours on an example matrix P. Strong *ns*–connections between *we*–connected groups determine a weighted directed graph.

Previous efforts attempted a pixel-by-pixel walk through the image, abiding by similar guidelines [9]. The result however is very much dependent on the specific route taken. The algorithm may reach, say, pixel 7 in Figure 4 with index i, move to its *n*-neighbour pixel 2, index it with $i+1$ and thereby introduce an error which can propagate.

We opt to first consider *all* the neighbourly connections between stripe pixels in the form of a graph and to then decide on a "best" indexing. To this end we define a *we*–connected group to be a sequence of m stripes pixels $p_1,\ldots,p_m$ such that p_1 has no *w*-neighbour, p_j is the *e*-neighbour of p_{j-1} for every $j \in \{2,\ldots,m\}$, and p_m has no *e*-neighbour. The stripe pixels in Figure 4 can thus be split into five *we*–connected groups. In accordance with the guidelines the same index will be assigned to all pixels in a *we*–connected group. We say that an *ns*-connection exists from a *we*–connected group a to a *we*–connected group b if at least one pixel in b is the *n*-neighbour of a pixel in a. Such a connection is denoted by the ordered pair (a,b) and indicates a *possible* increase of 1 in the index from group a to group b. Due to breaks in the recorded stripes some of these connections should in fact not be allowed to influence the indexing in this manner. We say (a,b) is a strong *ns*–connection if and only if a pixel in group b is the *n*-neighbour of a pixel in group a across every column that contains both a and b. The condition is not met when another group of pixels lies "between" a and b.

The connections determine a directed graph G: every *we*–connected group is represented as a vertex, and every strong *ns*–connection as a directed edge between corresponding vertices. Figure 4 shows the graph corresponding to the stripe pixel matrix shown (the edge weights will be explained presently). An indexing can now be generated from G by choosing a starting vertex and visiting connected vertices, heeding the directions of traversed edges: moving with the direction increases the current index by 1, and against the direction decreases it by 1. This yields an index for every connected vertex, thereby for every *we*–connected group, relative to the starting point[2].

[2]In our implementation we then locate a pixel on the darker reference stripe (see Figure 3) to shift these relative indices appropriately.

In general G may not reveal a clear and unique indexing and it is possible, specifically when G contains cycles, that only a subset of the edges can and should contribute. We decide to weigh every edge by the number of columns that its corresponding *ns*–connection spans. If a particular connection is maintained across a large number of columns the chance of that edge influencing the indexing should also be large. It remains to select a suitable subgraph of G that will determine the indexing. The requirements are: to preserve connectivity (connected vertices should remain connected); and to be unambiguous (any two vertices should be connected by exactly one path).

Observe that by definition any spanning tree of G meets these requirements [3]. A graph may not possess a unique spanning tree and we should find one that favours edges with large weights. Finding a *maximum* spanning tree (MST) will ensure that the indexing is dictated by those *ns*–connections with large weights, i.e. those most likely to indicate correct index transitions.

Note that a tree spanning all the vertices is defined only if G is connected. Since no relationship can be established between two disconnected components of G we cannot relate their indices (which is also why some pixels in Figure 3 remain unindexed).

4 Undetectable index shifts

For the indexing of uncoded stripes we have to assume "sufficient" smoothness in the target surface, such that elements of a continuous stripe in the image can be assigned a constant index. Now consider Figure 5 where two stripes n and $n+r$ are projected across a discontinuity in surface depth. Suppose this change in depth, $\Delta z_{(r)}$, causes the left part of stripe $n+r$ and the right part of stripe n to be captured to equal vertical coordinates in the image, as shown. Two different stripes now appear to be a single continuous stripe. We refer to this phenomenon as an undetectable shift in indices.

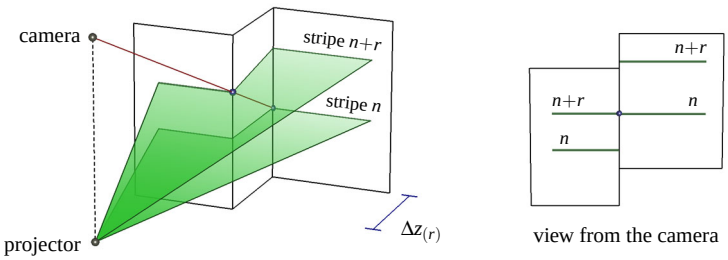

Figure 5: An abrupt change $\Delta z_{(r)}$ in surface depth can cause two stripes n and $n+r$ to align from the camera's point of view.

To express $\Delta z_{(r)}$ explicitly let $z(v,n)$ denote the Z-coordinate of a surface point on stripe n captured to vertical (and centred) image position v. The equations in (1) give $vPD_p + Wn = D_pD_s/(D_p - z)$. We use this with (2) to obtain

$$\Delta z_{(r)} = z(v,n+r) - z(v,n) = \frac{rW(D_p - z)^2}{rW(D_p - z) + D_pD_s}. \tag{3}$$

The expression gives the change in depth required to *perfectly* align stripes n and $n+r$ from the camera's point of view. The image is a discrete space, however, and for stripe

pixels in adjacent columns to be considered connected (as *e*- and *w*-neighbours) their vertical positions can differ by up to $\frac{3}{2}$ pixels up or down. Hence an abrupt change in depth that falls into a certain *interval* around $\Delta z_{(r)}$ can create a problem. The size of this interval can be determined through arguments similar to those above. As an example, we found with our parameters that a target surface 300*mm* from the projector with an abrupt drop in depth between about 5*mm* and 7*mm* can align stripes n and $n+1$.

It should be stressed that erroneous indexing caused by undetectable index shifts is a pitfall of uncoded stripe patterns. Because the stripes are identical in appearance it is impossible for any algorithm to decide with certainty whether or not a seemingly continuous stripe in P (the stripe pixel matrix) should be broken at some point.

If enough of the target surface is known prior to scanning, parameters such as the stripe spacing can be tuned to avoid critical depth changes. It may decrease the resolution however and, because knowledge of the surface is needed, is not a general solution. Alternatively, some additional information from the image, such as the edge of a discontinuity, may be useful. But the stripes typically dominate the image and can corrupt output from feature detectors. Moreover, it is possible that the image may not carry sufficient information for the successful demarcation of an edge.

The main obstacle in dealing with the problem directly is the homogeneity of the stripes, which causes index shifts to be *undetectable*. To eliminate the problem we need to ensure that index shifts will be *detectable*. This necessitates a coding scheme.

To maintain frame rate acquisition, uniform narrow stripe spacing and consistency across variable surface colour, we choose to code stripes by varied light intensity. A code $\{c_1,\ldots,c_q\}$, built from two different luminance levels, is repeated over the stripe pattern. Two levels are used because a larger range can be harder to recognize. If the position in the code of every stripe pixel can be determined, a shift would need to occur over an integer multiple of q indices to be undetectable. The code length needs to be large enough to eliminate the possibility of such shifts.

Our indexing algorithm adapts easily to accommodate a coding scheme. Firstly a position in the code is assigned to every stripe pixel. We achieve this by establishing whether a pixel is light or dark (by comparing luminance values locally), and then attempting to follow the code across a number of stripe pixels in the same column. The definitions of *we*–connected groups and strong *ns*–connections are appended: a *we*–connected group is a group of *identically coded* stripe pixels $p_1,\ldots,p_m$, say all coded with c, such that p_1 has no *w*-neighbour or the code of its *w*-neighbour is not c, p_j is the *e*-neighbour of p_{j-1} for every $j\in\{2,\ldots,m\}$, and p_m has no *e*-neighbour or the code of its *e*-neighbour is not c; an *ns*–connection (a,b) is strong if and only if it is maintained across all columns containing pixels from both groups a and b, and it also *agrees with the coding* of a and b, i.e. the code of b directly succeeds the code of a (modulo-wise). A weighted directed graph is constructed as before and a maximum spanning tree is determined.

5 Results and discussion

In our experiments we utilized a 768×576 monochrome CCTV video camera with a fixed-focus lens, and projected stripes with a standard DLP projector. The parameters were $D_p=790$, $W=3.08$, $D_s=61$, all measured in *mm*, and $P=0.0006$. In the pattern every third stripe was slightly darker, i.e. a code of the form $\{\text{light},\text{light},\text{dark}\}$.

The notion of "correct indexing" is hard to define because an independent set of accurate indices is not available. We decided to generate template indices manually for 10 test images, and results from the algorithms could then be matched to those to measure performance. The test images were taken of a person's face in 5 different poses (looking to the left and right at various angles) and 2 expressions (neutral and smiling). In each case templates were created only for a patch corresponding to the actual face. Other regions, such as hair and clothing, may not reflect the stripes well and are likely to cause poor performance for any algorithm.

Figure 6 shows results for two of the test images, cropped for display purposes. Our MST indexing algorithm was applied, first ignoring the code (MSTIU) and then also incorporating the code (MSTIC). For comparison results from the FloodFill algorithm by Robinson et. al. [9] are also shown.

The lighter pixels in the figure are indexed correctly and the darker ones incorrectly. The nose caused a critical drop and, as may be expected, MSTIU fell victim to an undetectable index shift while MSTIC handled it correctly. FloodFill is a rather strict algorithm and tried to avoid errors by leaving some pixels unindexed in the first image. In the second image, however, an error occured roughly midway through the traversal and propagated across the rest. Both MSTIU and MSTIC are designed to find overall optimal solutions, and errors occur only locally.

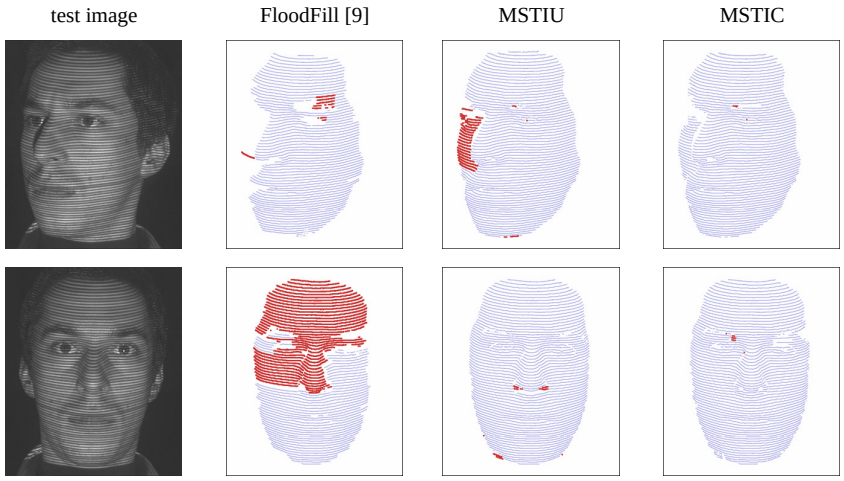

Figure 6: Results on two test images from FloodFill [9], our MST indexing algorithm for uncoded patterns (MSTIU) and for coded patterns (MSTIC). The darker (red) pixels were indexed incorrectly according to manually created templates.

To quantify performance we measured the *coverage* and *error* in the output of each algorithm. The coverage is the percentage of stripe pixels indexed and the error is the percentage of those pixels indexed incorrectly. Table 1 gives these values for each algorithm, averaged over the 10 test images. Note that MSTIC produces a slightly lower coverage on the images than MSTIU but reduces the error by more than an order of magnitude.

The system projects about 75 stripes over a surface the size of a face 800*mm* away, from which roughly 20,000 surface points can be produced. The complexity of locating

FloodFill [9]		**MSTIU**		**MSTIC**	
coverage	*error*	*coverage*	*error*	*coverage*	*error*
89.00 %	11.21 %	99.07 %	3.92 %	95.66 %	0.21 %

Table 1: The coverage (percentage of stripe pixels indexed) and error (in those that were indexed) of the three algorithms, shown as averages over 10 test images.

stripe pixels and building the graph G is linear in the number of image pixels. The number of vertices in G are typically far less than the number of pixels, to the extent that the non-linear complexity of finding an MST becomes negligible. Our implementation in C, executed on a Pentium IV 3.2GHz processor with 2GB memory, requires on average about 0.05s to locate stripes in an image, 0.08s to index them and 0.04s to build a 3D surface. A full 3D reconstruction can thence be performed from a single image in about 0.17 seconds.

Figure 7 shows the surface reconstructed from the image in Figure 3, where we also used a sub-pixel estimator on the indexed stripes to obtain greater depth resolution. Since every surface point corresponds to a pixel in the image, colour data can be added easily. A per-vertex interpolatory colouring effectively skips black stripes between white ones and produces the "stripe-free" texture shown.

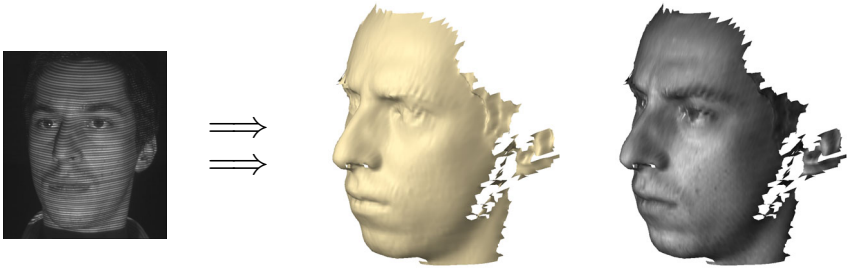

Figure 7: The surface generated from the image in Figure 3. Luminance values of stripe pixels in the original image were used to create a texture map.

6 Conclusions

We have presented a new algorithm for indexing uncoded stripe patterns in structured light systems. All connections and adjacencies among stripe pixels are considered in the form of a weighted directed graph. A maximum spanning tree of the graph favours clear (and likely to be correct) index transitions and yields an overall optimal indexing.

Despite a significant improvement in accuracy over existing methods there is a problem inherent to uncoded patterns. A discontinuity in surface depth can cause different stripes to align from the camera's view point, and can be undetectable because of stripe uniformity. Overcoming this problem in general calls for a coding strategy so that different stripes are distinguishable in the recorded image. We have shown, in tests against manually created templates, that a minimal coding scheme can be effective.

Although the cost of high-accuracy laser scanners are steadily decreasing it is our belief that structured light still competes strongly as a high-speed and flexible solution to 3D surface acquisition. Our current system processes about 6 frames per second. A future goal is to achieve real-time reconstruction, say up to 30 frames a second, which can be beneficial for applications in mobile intelligent systems and immersive media.

References

[1] C. Beumier and M. Acheroy. 3D facial surface acquisition by structured light. In *International Workshop on Synthetic Natural Hybrid Coding and 3D Imaging*, pages 103–106, 1999.

[2] C. Boehnen and P. Flynn. Accuracy of 3D scanning technologies in a face scanning scenario. In *5th International Conference on 3-D Digital Imaging and Modeling*, pages 310–317, 2005.

[3] G. Chartrand and O. Oellermann. *Applied and Algorithmic Graph Theory*. McGraw Hill, 1992.

[4] R.C. Daley and L.G. Hassebrook. Channel capacity model of binary encoded structured lightstripe illumination. *Applied Optics*, 37(17):3689–3696, 1998.

[5] F.W. DePiero and M.M. Trivedi. 3-D computer vision using structured light: design, calibration, and implementation issues. *Advances in Computers*, 43:243–278, 1996.

[6] J. Fryer, T. Clarke, and J. Chen. Lens distortion for simple C-mount lenses. *International Archives of Photogrammetry and Remote Sensing*, 30(5):97–101, 1994.

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[8] M. Proesmans and M.L. Van Gool. Reading between the lines: a method for extracting dynamic 3D with texture. In *ACM Symposium on Virtual Reality Software and Technology*, pages 95–102, 1997.

[9] A. Robinson, L. Alboul, and M.A. Rodrigues. Methods for indexing stripes in uncoded structured light scanning systems. *Journal of WSCG*, 12(3):371–378, 2004.

[10] J. Salvi, J. Pagés, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.

[11] M. Salzmann, J. Pilet, and P. Fua. Surface deformation models for non-rigid 3-D shape recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1481–1487, 2007.

[12] L. Zhang, B. Curless, and S.M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *1st International Symposium on 3D Data Processing, Visualization and Transmission*, pages 24–36, 2002.