Hamming Color Code for Dense and Robust One-shot 3D Scanning

Shuntaro Yamazaki[1]
http://www.dh.aist.go.jp/~shun/
Akira Nukada[2]
nukada@matsulab.is.titech.ac.jp
Masaaki Mochimaru[1]
m-mochimaru@aist.go.jp

[1] National Institute of Advanced Industrial Science and Technology
Tokyo, Japan

[2] Tokyo Institute of Technology
Tokyo, Japan

We propose a novel color code, Hamming color code, designed for rapid 3D shape acquisition using structured light projection. The Hamming color code has several properties which are desirable for practical 3D acquisition as follows. First, the Hamming distance of adjacent colors is always 1, which makes the color detection robust to color blending due to defocusing, subsurface scattering, or chromatic aberration. Second, the substrings of a certain length is guaranteed to be unique. In other words, the Hamming code can be viewed as a subset of de Bruijn sequence. Third, a one-dimensional coordinate can be encoded for each pixel, which enables dense 3D reconstruction from a single pattern projection. Thanks to the uniqueness and robustness of the substrings, the structured light can be decoded stably by dynamic programming. We have implemented parallel dynamic programming on GPU and achieved the speed-up by a factor of 630 compared to the CPU-based implementation, and accomplished video-rate 3D acquisition using commodity hardware. Several experiments have been conducted to demonstrate the stability and performance of our algorithm. Finally we discuss the limitation and future direction of this work.

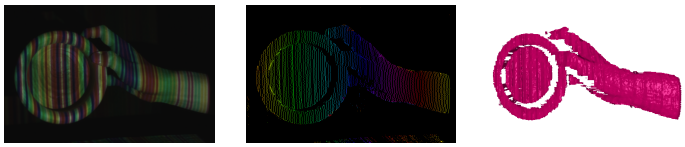
De Bruijn sequence has been widely used for one-shot 3D scanning using structured light projection. The sequence guarantees the discriminability of the subsequence of a certain length, which enables unique identification from a partial observation. Zhang et al. [1] propose a color stripe code based on the de Bruijn sequence embedded into color transition. Their method is prone to erroneous correspondence and unbearably poor 3D reconstruction, for the following two reasons. First, an acquired image often undergoes imaging noise such as chromatic aberration and defocusing, resulting in a color sequence which is different from the one projected. Second, the projected color code can be partially shifted, occluded or duplicated due to the depth discontinuity in the scene.

To remedy the drawbacks of the prior method, we propose a novel pseudo-number sequence which is similar in spirit to de Bruijn sequence. Our color pattern has the property that the hamming distance between adjacent colors is always one. Thus, we call the code as **Hamming color code**. This property guarantees that, even when two stripes are blended at their boundary, the resultant color matches either of the two colors. We encode one-dimensional coordinate into a color, instead of color transition, to alleviate the second drawback mentioned in the previous section. In decoding, correspondence is determined at the center of each color band detected by a gradient operator.

We generate the Hamming color code using the RGB colors except black, whose channel values are either 0 (minimum) or 1 (maximum). The elements in the code are represented by 3-bit RGB colors, or equivalently by 1-digit number as $\{1,2,3,4,5,6,7\}$. The sequence is characterized by a single parameter, the order of the sequence k which is the length of unique subsequence in the sequence. The sequence starts with any of the elements, and the following digits are generated by flipping one of the color channel in the previous color. If the generated number is zero or the last k digit has already included in the sequence, the number is discarded and other bits are flipped. The order-k Hamming color code $H(k)$ is the longest sequence generated by repeating the above algorithm. Apparently

$H(4) =$
6457326751376457646231546267323137515457
5467545132645157315762376264645454646731
3151513132326262323757376757576767373 75

Figure 1: Hamming color code. (top) Code sequence $H(4)$ used in our experiments. The sequence starts with 6, contains 1 to 7, and consists of distinctive substrings of size 4. (bottom) Color structured light pattern.



Acquired Correspondence 3D reconstruction

Figure 2: Result of 3D shape reconstruction: (left) Acquired images. (middle) Estimated correspondences shown in color. (right) Surface rendering of reconstructed 3D shapes.

Table 1: Average computation time of 100 dynamic programming on CPU and GPU in milliseconds.

	CPU	1 GPU	2 GPUs
Data upload	—	0.286	0.285
Color normalization	5.1	0.097	0.053
Consistency computation	809.6	8.533	5.130
Table update	12948.4	11.756	5.892
Backtrack	13.5	0.817	0.440
Data download	—	0.376	0.340
Total	13776.6	21.865	12.140

$H(k)$ is not unique, but any of the sequence can be used for our purpose. We use $H(4)$ of length 119 shown in Figure 1.

The correspondence from an acquired image to the projected color pattern can be solved by dynamic programming [1]. We apply this technique to each scanline in the acquired image and solve the correspondence. While dynamic programming allows us to estimate the global optimal solution, it is not fast enough to achieve video-rate reconstruction on standard PC platform. This problem is resolved by using GPU to parallelize the computation.

Figure 2 summarizes the result of 3D shape reconstruction. The 3D shapes are reconstructed by our GPU software running at a video-frame rate. We do not remove the background and use all pixels for computing correspondences. Image rows are processed independently, and no inter-scanline smoothness is considered. No postprocessing such as noise reduction has been applied. We can see two kinds of artifacts in the reconstruction results. First is the jittering effects across the scanlines. Second is systematic vertical patterns on the 3D surface. The first problem can be drastically improved by considering inter-scanline smoothness by using additional optimization such as dynamic programming in vertical direction or belief propagation. The second problem is caused by the quantization in our correspondence. In dynamic programming, the correspondence is limited to pixel coordinate. We can improve the correspondence using the gradient of pixel intensity, which we are now working on.

Table 1 shows the computation speed of our 3D reconstruction. We show the timing of dynamic programming since our focus is on estimating correspondence from an acquired image to a projected color pattern. The overall speed-up of our GPU implementation is about 630 times against CPU implementation, which CPU software is not fully optimized. We can also see that the GPU implementation scales nearly linearly without any changes to the implementation. The frame rate of our 3D reconstruction system, including image acquisition, triangulation, and rendering, is about 60 Hz which we believe is sufficient for many graphics and vision applications.

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