3D structured light reconstruction system

João Santos 76912, *DETI UA,* Samuel Silva 93428, *DETI UA*

***Abstract*—TODO**

***Index Terms*—Infrared, dot pattern, Kinect, speckle, structured light, point cloud**

I. Introduction

THE Xbox 360 Kinect has opened a new era of inexpensive 3D sensing. It was originally intended only for the gaming industry, but it was quickly adopted by scientists, robotics enthusiasts, and hobbyists all over the world. This sensor was introduced as a low-cost, durable, marker-free, and reliable device capable of recording real-time proximity and depth data. It combines a standard color camera with a depth camera that incorporates an Infrared (IR) laser projector as well as an IR camera. The Windows Software Development Kit (SDK) 2.0 has enabled the development of software that use the gesture recognition technology built in the device to integrate joint orientation and skeleton tracking for joint positions in standing and seated positions, among many other applications.

In the scope of this work, we will study and replicate the depth estimation algorithm that is suspected to be implemented on the Kinect system. To achieve so, firstly a set of simulated IR images will be the input for the depth estimation. As a final goal, a real Kinect was intended to be used solely for the projection and detection of the speckle pattern, to then process it into the desired point cloud format.

II. Kinect Model

Accordingly to [1], as soon as the Kinect gathers an image of the projected speckle IR pattern, this image is thresholded to then start the disparity estimation step.

The Kinect depth system is constituted by a class 1M IR laser (called transmitter) and a Complementary metal–oxide–semiconductor (CMOS) sensor (called receiver). Both transmitter and receiver are coplanar and positioned so that their optical axes are parallel and have a baseline distance of 75 mm apart. This baseline $b = 75$ distance is a crucial parameter in the depth estimation since it appears on the triangulation equation 1 (where f_x is the horizontal focal distance and d is the disparity value of a given pixel).

$$z = f_x \frac{b}{d} \tag{1}$$

To compute the also necessary x and y coordinates, one uses equations 2 and 3 in which c_x and c_y are the skew values from the intrinsic calibration, u and v are the pixel coordinates of the pixel and f_y is the vertical focal distance.

$$x = (u - c_x) \times \frac{z}{f_x} \tag{2}$$

$$y = (v - c_y) \times \frac{z}{f_y} \tag{3}$$

Notice that the thresholding algorithm is assumed to be implemented on the Kinect always outputs any given dot on the pattern as a single white pixel.

A. Disparity Estimation

The disparity estimation performed by the Kinect is based on a local, pixel-based correlation algorithm in which a 9×9 window of the mentioned binary image is compared to a sequence of reference (and also binary) images, stored on the device. These reference images are just IR images of the pattern projected against a flat surface, at well known distances.

Given that the epipolar lines of the transmitter and receiver are aligned and that the binarized dots in the speckle pattern are one pixel wide as formerly stated, moving the camera closer or further away from the object actually only results on the received pattern to be shifted left or right. This will be detailed on section II-C, but keep in mind that $N = 45$ pixels is the maximum disparity shift considering the depth of field of the Kinect.

This lateral shift is what, in the end, tells us how far a pixel is from the sensor. If the measured and reference windows are, respectively, centered at (u_Z, v_Z) and (u_{ref}, v_{ref}) then

$$u_Z = u_{ref} + \Delta d_0$$
$$v_Z = v_{ref}$$

where Δd_0 is the change in disparity (measured in pixels). The change in disparity computed by maximizing the cross-covariance C between the measured and reference windows, as in equation 4.

$$\Delta d_0 = \arg\max_{n \in N} C_n \tag{4}$$

The cross-covariance is computed with equation 5, in which W denotes the measured (Z) or a given reference (ref, n) window and $\bar{W}$ is the the average over that window.

$$C_n = \sum_{(u,v) \in W_Z} [W_Z - \bar{W}_Z] \circ [W_{ref,n} - \bar{W}_{ref,n}] \tag{5}$$

B. Subpixel Refinement

Following the computation of the initial integer disparity, a further refinement is carried out that will provide an higher depth accuracy.

Assuming that any given window is at a constant depth, a fractional disparity Δd_s is computed by calculating where the IR dots split pixels. In their work, [1] assume that the Sum of Absolute Differences (SAD) is used.

Starting with the already supplied N binary reference images, a further eight levels are computed between each successive pair. For all pixel positions in each pair, a linearly spaced vector is generated from one value to the next, thus generating the eight intensity levels.

Finally, by knowing the distance at which each of the reference binary images were taken and what subpixel level minimizes the SAD (now using the non thresholded and noisy IR image take), the fractional disparity is determined.

C. Depth Estimation

In this final step, the total depth disparity $\Delta d = \Delta d_0 + \Delta d_s$ is applied in equation 1 to output the final depth estimate, given by

$$z = f_x \frac{b}{d_{\text{off}} + \Delta d} \tag{6}$$

where d_{off} is the characteristic disparity of the selected reference window.

Because the transmitter is physically positioned to the right of the receiver, a positive Δd indicates a pattern shift to the right, resulting in an object mapped closer to the sensor if compared to the reference depth. A negative Δd, on the other hand, denotes a shift to the left and a farther-away object.

III. Methodology

In the scope of the current project, the initial step that the authors proposed to themselves was to implement the already detailed Kinect algorithm with pre-computed (i.e. simulated) IR images gathered from the simulator developed by [1].

To achieve so, a dataset of said images was built and exported, together with the expected reference images. This dataset was built using the simulator freely available on MathWorks File Exchange[1] and is composed by three scenarios,

1) the corner of a room with a wall tilted 60º;
2) a teapot in front of a wall;
3) a Xbox Controller in front of a wall,

that are shown on figures 1, 2 and 3.

After that, the intention was to gathered real IR images from a Kinect and process them so that they would be usable in the implemented algorithms. This implies that the pattern used in the simulator and the real Kinect are the same and, also, that we could actually gather raw IR data from the sensor.

IV. Results

In the end, the fusion between the depth estimation algorithm and the IR images originated by a real Kinect could not be done. The fact is that the tested binarization methods were not robust enough to produce suitable images for the depth estimation. Yet, the goal of understanding how the most well-known structured light system works was achieved. In parallel, we also proved that it is possible to capture IR images from the sensor, thanks to the *libfreenect*[2] library.

TABLE I
Simulation time for each scenario.

Scenario	Time [s]
Wall	46.10
Teapot	70.77
Xbox Controller	125.53

A. Simulated Data

By resorting to the acquired simulated data, we were able to reconstruct the depth map of the scenes. These depth maps are shown on figures 4, 5 and 6. In these depth maps, closer points are in a darker shade while further ones are closer to full white. As a side note, this images are outputted as 10-bit images, so that the full Depth of field (DOF) can be encoded as a single number for each pixel.

With this, we can quite clearly see that, for the teapot and Xbox Controller scenarios, they were placed in front of a flat wall.

The computational time for the three scenarios are presented on table I. Notice that the times refer solely to the computation of the depth map after loading all the necessary data.

With the resultant depth maps, the conversion to point cloud was achieved using the Open3D [2] library. In essence, this library applies equations 2 and 3 to compute the x and y coordinates and directly uses the depth map values as the z coordinate. In our application, we computed the intrinsic parameters of the CMOS sensor as

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{640}{\tan(FOV_x/2)} & 0 & 640/2 \\ 0 & \frac{480}{\tan(FOV_y/2)} & 480/2 \\ 0 & 0 & 1 \end{bmatrix}.$$

The correspondent point clouds are shown on figures 7, 8 and 9.

B. Kinect Data

After being successful at using the simulated IR images to compute depth maps and, then, point clouds, the subsequent step was to acquire and handle the real data provided by a real Kinect. As an example, figure 10 shows a frame, at the same instant, gathered from the video, depth and IR streams.

Unfortunately, we could not integrate this images onto the depth estimation process. As show by figure 10c, much light coming from outside sources severely hinders the binarization of the speckle pattern. This was, as shown earlier, mandatory for the procedure to work.

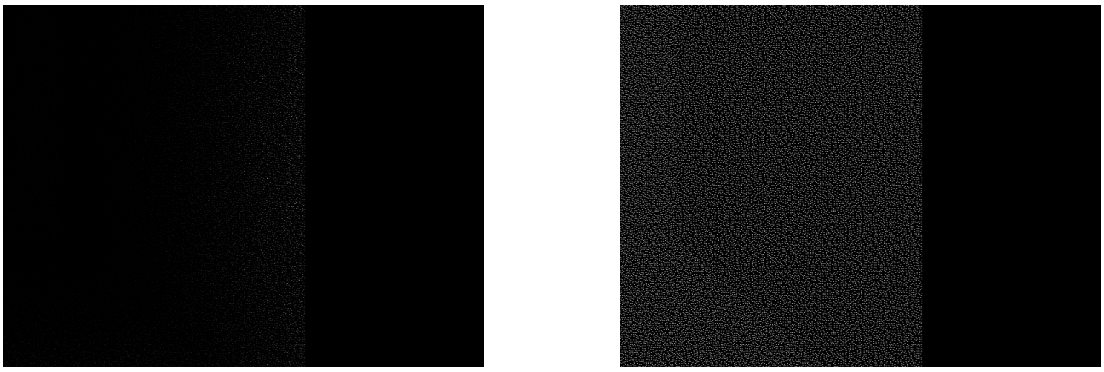
V. Future Work

As guidelines for a future work, two main points can be quite improved.

The main point is the completion of the proposed goal, integrating the depth algorithm with the real Kinect data. This

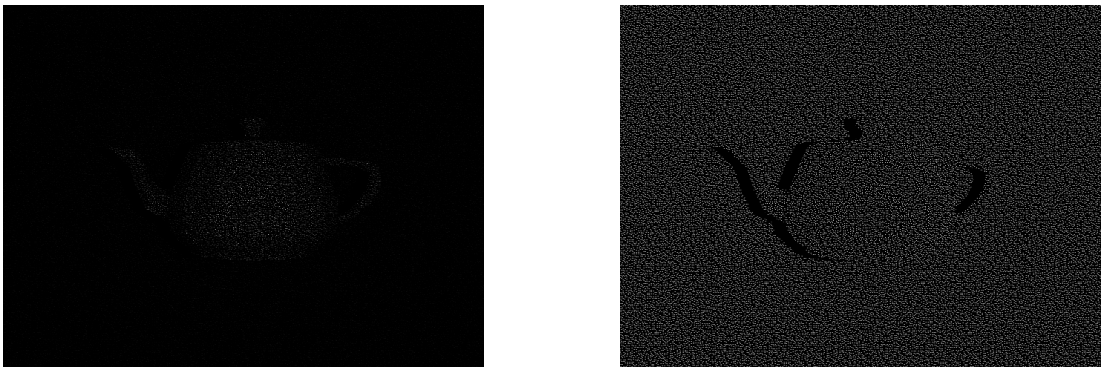
[1] https://www.mathworks.com/matlabcentral/fileexchange/50357-kinect-infrared-ir-and-depth-image-simulator

[2] https://github.com/OpenKinect/libfreenect



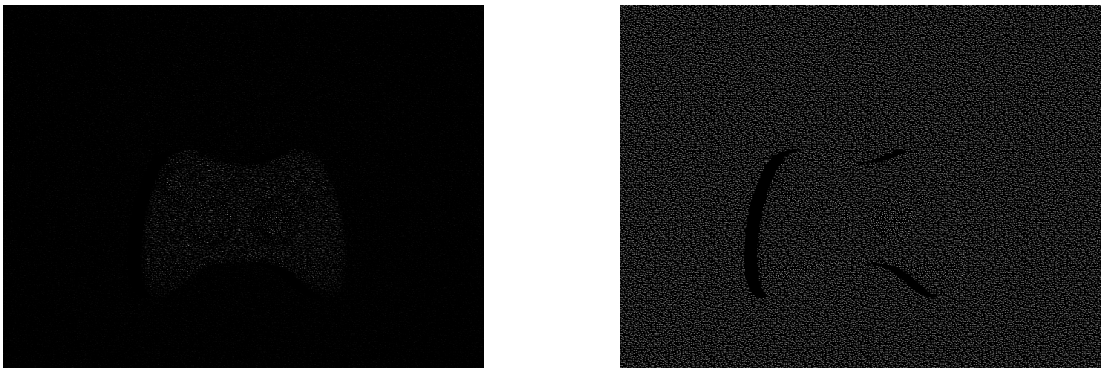
(a) (b)

Fig. 1. Wall corner scenario: (a) IR image and (b) binary.



(a) (b)

Fig. 2. Teapot scenario: (a) IR image and (b) binary IR.



(a) (b)

Fig. 3. Xbox Controller scenario: (a) IR image and (b) binary IR.

poses two main difficulties, i) study if the speckle pattern provided by [1] actually reassembles close enough the actually projected pattern and ii) build a sufficiently robust binarization algorithm for the IR images so that most of the non-pattern portions of the received image are filtered out.

The secondary point should be to improve performance of the implementation. For reasonably small 640×480 images the algorithm is quite slow and would be unusable if applied to a commercial product. One proposal is to use a region growing algorithm and test if this actually produces satisfactory results with reduced computation time.

VI. Conclusion

Although the the goal of building a fully functional, real-time, 3D structured light system was not reached, the authors believe that the current project builds the foundation for future works to accomplish it.

This work not only researched and resumed the essence of how the Kinect (and possible other camera systems based on the PrimeSense technology like the Asus Xtion, Orbec Astra,

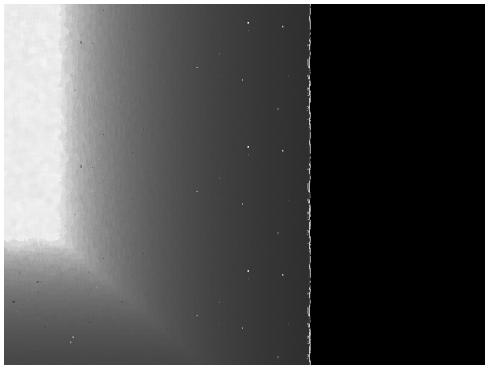


Fig. 4. Depth map for the wall scenario.

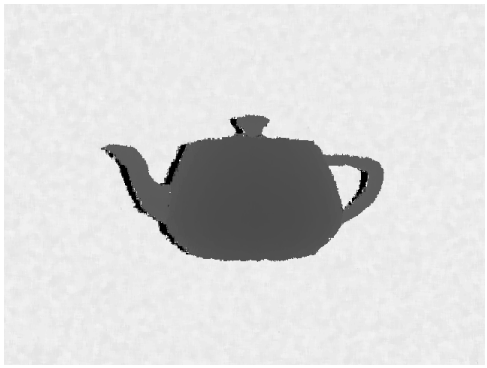


Fig. 5. Depth map for the teapot scenario.

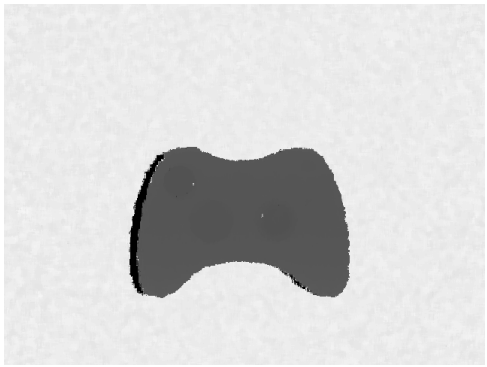


Fig. 6. Depth map for the Xbox controller scenario.

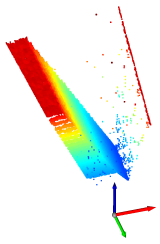


Fig. 7. Point cloud for the wall scenario.

etc.) work but also proved to be possible the direct acquisition of IR images.

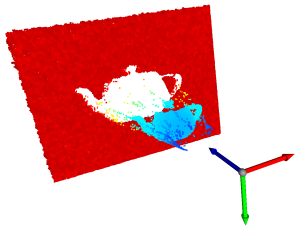


Fig. 8. Point cloud for the teapot scenario.

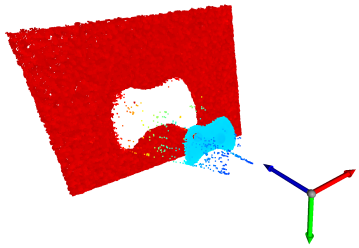
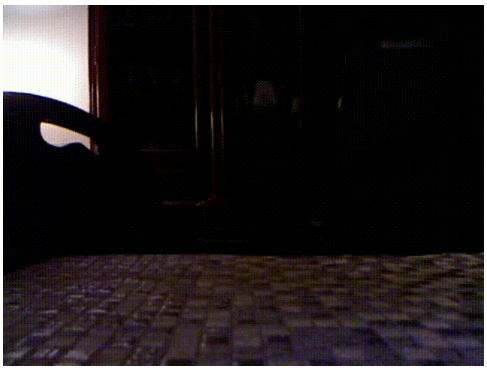


Fig. 9. Point cloud for the Xbox controller scenario.

REFERENCES

[1] M. J. Landau, B. Y. Choo, and P. A. Beling, "Simulating kinect infrared and depth images," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 3018–3031, 2016.

[2] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.



(a)

(b)

(c)

Fig. 10. A frame from the (a) video, (b) depth and (c) infrared streams captured at the same instant from a real Kinect.