5th International Conference, VAMR 2013, Held as Part of HCI International 2013,
Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part I
http://www.springer.com/computer/hci/book/978-3-642-39404-1
The original publication is available at www.springerlink.com

Enhancing Metric Perception with RGB-D Camera

Daiki Handa, Hirotake Ishii, and Hiroshi Shimoda

Kyoto University, Graduate School of Energy Science, Kyoto pref., Japan
{handa,hirotake,shimoda}@ei.energy.kyoto-u.ac.jp

Abstract. Metric measurement of environment has fundamental role in tasks such as interior design and plant maintenance. Conventional methods for these tasks suffer from high development cost or unstability. We propose a mobile metric perception enhancement system which focuses on interactivity through user locomotion. The proposed system overlays geometric annotations in real-time on a tablet device. The annotation is generated from RGB-D camera in per-frame basis, alleviating the object recognition problem by effectively utilizing processing power of human. We show a few illustrative cases where the system is tested, and discuss correctness of annotations.

Keywords: Augmented Reality, Augmented Human, Mobile Device, RGB-D Camera, Geometric Annotation, Per-frame Processing

1 Introduction

Real world tasks such as interior design and plant maintenance rely on knowledge of geometric properties of surrounding objects. In these scenarios, measurement of environment often forms the basis of higher level sub-tasks. We propose metric perception enhancement through overlaying geometric annotation extracted from RGB-D data in real-time.

Existing augmented reality solutions for these tasks mostly depend on the idea of overlaying suitable pre-made virtual objects such as furniture or CAD model [1]. While this approach can potentially provide tailored user experience, these applications tend to add little benefit compared to required application development and deployment cost. These costs may occur from employment of artists to create virtual object, setup of markers to track the device, or maintainance of up-to-date CAD data of the environment.

On the other hand, there are many methods to create 3D model of the environment on-the-fly, generally called Simultaneous Localization and Mapping (SLAM) [2]. But these methods are either not robust enough, only able to provide sparse model, or computationally expensive. So dynamic content creation through automatic modeling of environment is not feasible.

Recent introduction of consumer-grade RGB-D sensors such as Microsoft kinect enable us to use it on mobile devices such as tablets, making it possible to robustly acquire local 3D point cloud in real-time.

We propose geometric annotation application that can be used with little constraint on the environment. We generate salient annotations from RGB-D data, and overlay them on per-frame basis. The proposed system can annotate straight line in sight with lengths, and surfaces with contour lines. While the quality of output is lower than that of perfect CAD data, tight interaction loop created by per-frame presentation of data can compensate the downside, and can provide reasonably good user experience at very low cost. The key insight is that human can easily associate flickering or duplicated annotations to real world structure, while it is very difficult for computers to accurately create coherent model of the environment from raw data.

We will illustrate a few cases where our system would be useful and discuss correctness of generated annotations.

2 Generating Annotation

To visualize metric properties of environment, two complementary kinds of annotations are generated. The *length annotations* enable the user to perceive lengths of straight edges abundant in artificial objects. The *height annotations* are contour lines to help understanding featureless or curved surfaces where straight edges are absent and thus length annotation is unavailable.

Dataflow of the annotation process is shown in Fig. 1. The input of the system is QVGA frames from a RGB-D camera and the gravity vector [1] from a tablet.

2.1 Edge Detection and Refinement

The goals here are extraction of straight edges and calculation of their lengths. Output of current depth sensors typically contains jagginess of several pixels near object outline, while RGB image have effective angular resolution of nearly one pixel. So, three-dimensional edges are estimated from line segments in RGB image.

Line segments are extracted from RGB image by first converting it to grayscale, and then applying LSD [3] detector. The detected line segments contain *Number of False Alarms* (NFA) values, which are used as saliency in later optimization phase.

Detected edges can be categorized to three classes as shown in Fig. 2; a textural edge lies on planar surface, and a structural edge corresponds

[1] Mobile platform such as Android provides gravity sensor based on low-pass filtering of accelerometer data.

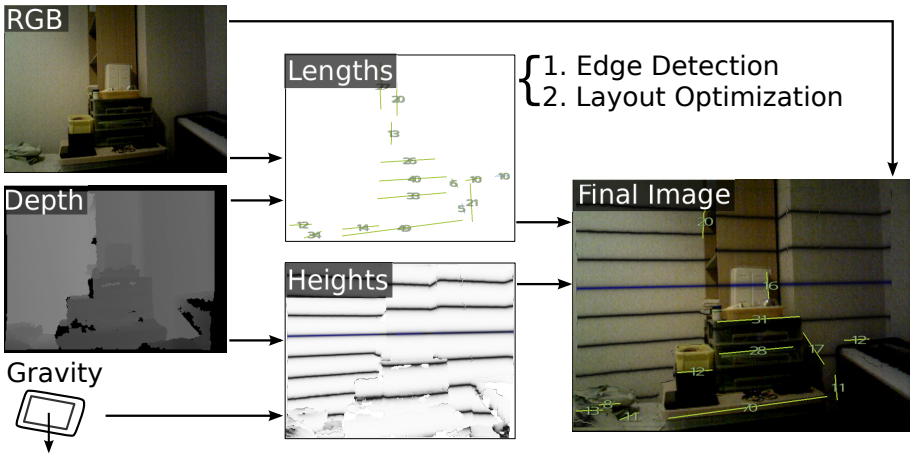

Fig. 1. Upper Middle: Lengths from line segments, Lower Middle: contours from per-pixel depth coloring

a ridge or a cliff of an object. Structural edges are further divided to continuous or discontinuous by whether two sides of the edge are on a same object (continuous) or not (discontinuous). Discontinuous strucutral edges need special treatment when calculating length, since depth is ill-defined on the discontinuous edge.

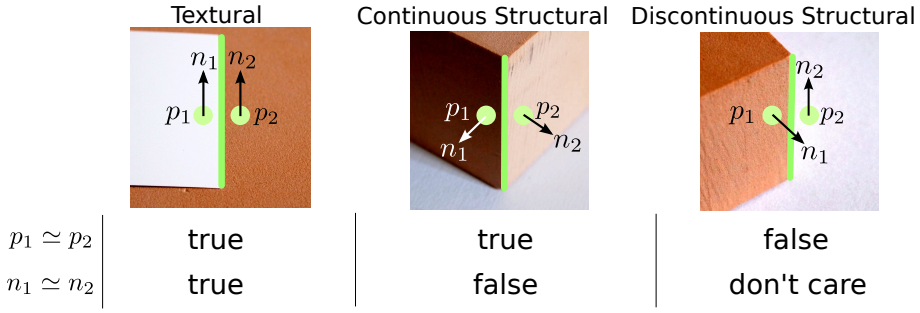

Fig. 2. Edges can be classified by comparing positions and normals near midpoints. In reality, occulusions, shadows and noise make distinction unclear.

To check discontinuity of an edge, $p_1 \simeq p_2$ condition (in Fig. 2) is used. When depth is continuous at an edge, 3D distance d between two symmetric points near the midpoint is linear to that of screen space. Pair-distance $d(s)$

for points $2s$ apart in screen space is defined as follows:

$$d(s) = |T(p_{\text{mid}} + sn) - T(p_{\text{mid}} - sn)| \tag{1}$$

where p_{mid} is the midpoint of the edge, $T(p)$ is 3D position of the pixel, and n is the normal of the segment. By using $d(s)$, the discontinuity conidition is approximated by $\frac{d(5\text{px})}{d(2\text{px})} < \frac{5}{2}\alpha$, where $\alpha \simeq 1$ is a sensitivity constant.

After edge classification, discontinuous edges are refined by moving toward the nearer (i.e. foreground) side to avoid jagged region. After edge refinement, length is calculated respectively from two endpoints of the segments. If depth at an endpoints is unavailable due to depth camera limitation, the edge is discarded as false one.

2.2 Layout Optimization

In complex scenes, edge annotations may become unreadable due to overlap. To mitigate this problem, annotation density distribution on screen is represented by a lattice, and edges are picked sequentially in order of decreasing saliency. The greedy selection process is depicted in the following pseudocode:

```
def select_edges(edges):
  bool[][] density = {{false,...},...}
  edges_to_show = []
  for edge in sort(edges, order_by=NFA, decreasing):
    if not any(density[x,y] for (x,y) in cells_on(edge)):
      for (x,y) in cells_on(edge):
        density[x,y] = true
      edges_to_show.add(edge)
  return edges_to_show
```

Here we use $20\,\text{px}$ for cell and lattice size where frame size is $320\,\text{px} \times 240\,\text{px}$. To allow edges with a shared vertex like a corner of a box, cells corresponding to endpoints are excluded when computing `cells_on(edge)`.

2.3 Height Annotation

Normalized gravity vector n_{gravity} is used to show contour lines. To draw a single contour line with camera-relative height h, intensity $I(p)$ at pixel p in screen coordinates is determined by Eq. 2.

$$I_h(p) = \frac{1}{1 + (T(p) \cdot n_{\text{gravity}} - h)^2 w^2} \tag{2}$$

where $T(p)$ is 3D position of p in camera coordinates, and w is a constant controlling the line width. In this paper, height annotations are drawn with $20\,\text{cm}$ interval.

The decision to draw height annotations relative to device position instead of automatically detected floor, ensures smooth temporal behavior of lines by avoiding non-robust floor detection step. It is up to the user to hold the device at appropriate height to get meaningful readings.

3 Implementation

In this section, implementation details which can affect performance and mobility are described.

3.1 Hardware

The device consists of an Android tablet and a RGB-D camera as shown in Fig. 3. Since the camera is powered by USB from the tablet, there is no need for an external power supply. Mobility of the system is further increased by modifying the camera shell and cable. This results in a device with total weight of under $450\,\mathrm{g}$, which can be used portably with a single hand.

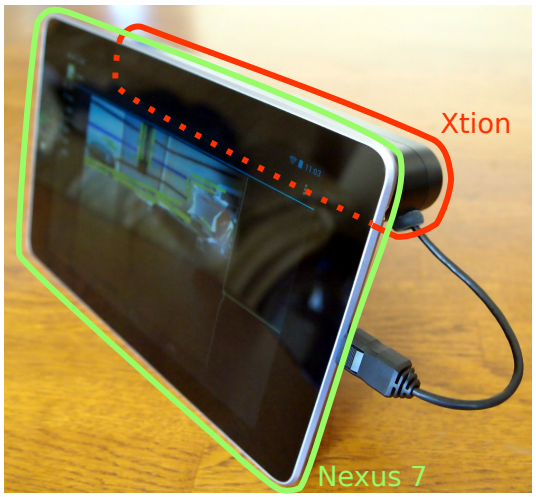

Fig. 3. Nexus 7 tablet and modified ASUS Xtion PRO LIVE RGB-D camera connected via USB

Note that a Nexus 7 contains an accelerometer, so the only external component is the RGB-D camera.

3.2 Software

The system is implemented on Android 4.2.1, and most part is coded in Java. A screenshot in Fig. 4 shows the UI and a typical result of annotation.

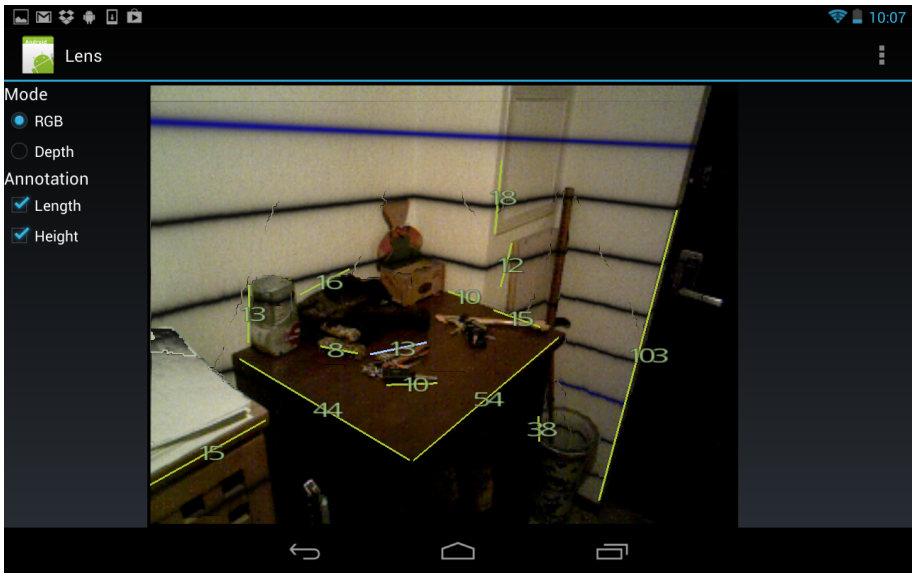

Fig. 4. A screenshot of the system

To maximize performance, the line segment detector [3] is compiled for ARM NEON instructions and called via Java Native Interface. Rendering of annotations is performed on GPU, and particularly, height annotation is implemented as a fragment shader.

The UI allows respective switching of length and height annotations to increase framerate by turning off unnecessary annotations. To limit the mode of interaction to moving in the real world, controls for parameters such as detection threshold are intentionally excluded.

4 Experimental Evaluation

We illustrate several use cases by showing example of operation and evaluate accuracy of annotations. All examples were run at real-time frame rate.

4.1 Interactive Usage

Figure 5 shows the change in display when the user moved toward an object. Initially invisible small features (e.g. lattice-like object in 4) become visible

with a closer look. In this example, natural user movement cause scale to change and show what the user would want to see. In general, it is often possible to read the length of an arbitrary edge by viewing from an appropriate angle and position. It can be argued that this kind of minimal-guessing (on the computer side) approach is more effective and feasible than trying to acquire detailed model of the environment and construct a GUI to choose what to see in the model.

Fig. 5. 1-4: Scale of length annotation changes as the user moves toward the Android mascot

Figure 6 illustrates how height annotation can complement length annotation for a curved object.

In these cases, two kind of annotations are used separately to see the effect respectively. Using both annotations simultaneously as in Fig. 4 does not cause a clutter, so we can omit GUI switches and make real world locomotion a sole, yet complete mode of interaction. This property would be useful when using the proposed technique with a HMD or a mobile projector like [4].

Fig. 6. Left: Length annotation cannot display height of the round-end cylinder Right: Height annotation reveals height of 1.5 units, which corresponds to $30\,\mathrm{cm}$

4.2 Latency

Important to interactivity is the latency. Typical latency to process a single frame is shown in table 1. Note that actual framerate is somewhat higher than determined by the total latency, since the code is multi-threaded.

Table 1. Typical Latency

Section	Time[ms]
Line Segment Detection (QVGA)	481
Edge Analysis & Refinement	8
Layout Optimization	4
Rendering & CPU-GPU Transfer	25
Total	518

Line segment detection is taking significant time and clearly needs a faster implementation, possibly on GPU. However, the system runs at nearly $30\,\mathrm{fps}$ when only height annotation is used.

4.3 Discussions on Correctness

Ultimately, precision would be limited by depth camera error, for which a detailed analysis exists [5]. However, incorrect lengths from false edges are far more noticeable in the current implementation.

In Fig. 5, there are roughly two kinds of false edges; edges corresponding to no structure nor texture, and fragmented or incomplete edges along long lines. An example for each kind is shown in Fig. 7.

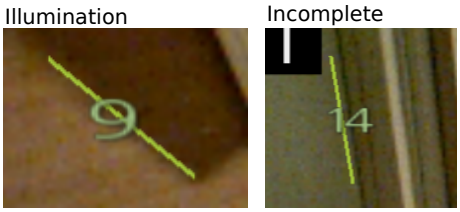

Fig. 7. Left: false edge along shadow Right: edge is structural, but too short

The former is caused by shadow or gradation due to illumination, but human is so good at distinguishing illumination and texture (i.e. *lightness constancy* effect [6]) that difference between human and machine perception becomes noticeable. This kind of false edges are relatively harmless since they appear where other real edges are absent.

The latter is more problematic, since shorter edges can hide original long edge in layout optimization. Solution to this would be giving long edges higher scores in optimization, or using depth-guided line segment detection.

5 Conclusion

In this paper, we have shown that the conveying geometric information directly to the user is useful in various settings and relatively simple to implement compared to conventional approaches like in [1]. We proposed a method to annotate lengths and contours and implemented in a truly mobile way.

The experiment shows the ability of the system to explore edges by real world locomotion of the user. It also shows that per-frame processing can augment perception more cost-effectively than conventional methods by creating tighter interaction loop. Also, this kind of real world interaction would be beneficial to hands-free implementations in the future.

Inaccuracy and slowness of line segment detection is found to be a limiting factor in the current implementation. This could be remedied by depth-guided segment detection or a fast GPU-accelerated implementation in conjunction with more sophisticated layout optimization.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number 23240016.

References

1. Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., Ivkovic, M.: Augmented reality technologies, systems and applications. Multimedia Tools and Applications **51** (2011) 341–377
2. Aulinas, J., Petillot, Y.R., Salvi, J., LladÃş, X.: The SLAM problem: a survey. In: Catalonian Conference on AI. (2008) 363–371
3. Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, Gregory Randall: LSD: a Line Segment Detector. Image Processing On Line (2012)
4. Mistry, P., Maes, P.: Sixthsense: a wearable gestural interface. In: ACM SIGGRAPH ASIA 2009 Sketches. SIGGRAPH ASIA '09, New York, NY, USA, ACM (2009) 11:1–11:1
5. Khoshelham, K., Elberink, S.O.: Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors **12**(2) (2012) 1437–1454
6. Adelson, E.H.: Lightness perception and lightness illusion (1999)