References

[1] A. Saxena, M. Sun and Y. Andrew, "3-D Depth Reconstruction from a Single Still Image", *International Journal of Computer Vision (IJCV)*, vol. 76, no 1, January 2008.

[2] C. Martin, "Evolving Visual Sonar: Depth From Monocular Images", *Pattern Recognition Letters*, 27, 2006

[3] J. Cardillo and A. Sid-Ahmed, "3-D position sensing using a passive monocular vision system*", IEEE transactions on pattern analysis and machine intelligence*, vol. 13 no 8, August 1991.

[4] J. M Loomis, "Looking down is looking up". *Nature News and Views*, 2001, pp. 155–156.

[5] R. Kumar, S. Sawhney and R. Hanson, "3D model acquisition from monocular image sequences", *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 1992.

[6] S. H. Schwartz, "Visual perception (2nd ed.)". Connecticut: Appletonand Lange, 1999

[7] S. Sawhney and R. Hanson, "Identification and 3D description of 'shallow' environmental structure in a sequence of images", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 1992, pp. 179-186

[8] S. Sawhney and R. Hanson, "Affine Trackability aids Obstacle Detection", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 1992, pp. 418 – 424

[9] The foundation for intelligent physical agents, FIPA Specifications, Available at: http://www.fipa.org/repository/aclspecs.html

[10] E. de Croon, E. de Weerdt, C. de Wagter, W. Remes, "The appearance variation cue for obstacle avoidance", *Proc. IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2010, pp. 1606 - 1611

[11] V. Leroy, T. Simon and F. Deschênes, "An efficient method for monocular depth from defocus", *Proc. 50th International Symposium(ELMAR)*, IEEE Computer Society, 2008, pp. 133 – 136

[12] X. Lin and H. Wei, "The Depth Estimate of Interesting Points from Monocular Vision", *Proc. International Conference on Artificial Intelligence and*

Computational Intelligence(AICI 2009), IEEE Computer Society, 2009, pp. 190-195

[13] Y Fujii, K. Wehe, E. Weymouth, "Robust Monocular Depth Perception Using Feature Pairs and Approximate Motion", *Proc. IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 1992, pp. 33 – 39

[14] B.K.P. Horn and B.G. Schunck, "Determining optical flow", *Artificial Intelligence*, vol. 17, 1981, pp. 185-203.

Appendix A - Optical Flow estimation

A.1 Introduction

In this chapter it is going to discuss the Lucas-Kanade Optical Flow estimation, which is one of the widely used Optical Flow estimation methods.

A.2 Lucas–Kanade method for Optical Flow estimation

Optical flow methods try to compute Optical flow using two images taken in time *t* and *t+ δt*. Lucas–Kanade is such a optical flow estimation method developed by Bruce D. Lucas and Takeo Kanade. Initial algorithm works for small displacements of interested points and it assumes that the intensity of the same object location is a constant over time. This is also called as the image constraint equation, which is mathematically expressed as:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

Above statement refers to the same object location, but having two different image coordinates in two different consecutive image frames. Based on this constraint, Optical flow algorithms can track interested points between two consecutive image frames. One drawback of this assumption is that it requires the displacement of the object point to be very small and does not work for object points having a large speed. Assuming the movement to be small, the Taylor expansion of the above equation becomes

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T.$$

Where H.O.T are the higher order terms, which can be safely ignored. From these equations it follows that:

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \qquad \frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

Or simply as:

$$I_x \cdot V_x + I_y \cdot V_y = -I_t \longrightarrow \mathrm{S}$$

Where *Vx* and *Vy* are the x and y components of the 2D optical flow vector associated with the considered pixel.

In addition to the image constraint equation, Lucas–Kanade method holds an additional hypothesis where given a pixel *P*, all the pixels in the neighborhood of *P* have the same velocity as *P*. this allows us to consider an image window m x m centered at pixel *P* and allows to write the following equation.

$$\begin{aligned} I_{x_1}V_x + I_{y_1}V_y &= -I_{t_1} \\ I_{x_2}V_x + I_{y_2}V_y &= -I_{t_2} \\ &\vdots \\ I_{x_n}V_x + I_{y_n}V_y &= -I_{t_n} \end{aligned}$$

This can also be represented by the following matrix:

$$\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}$$

Above equation can be represented as:

$$A\vec{v} = -b$$

Where

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix} \qquad \vec{v} = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \qquad -b = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}$$

The least square method holds:

$$A^T A\vec{v} = A^T(-b) or \vec{v} = (A^T A)^{-1} A^T(-b)$$

By expanding this equation we get:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{x_i} I_{y_i} & \sum I_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix}$$

Given a window *W*, This is same as writing the equation S as:

$$f(u,v) = \int\int (I_x u + I_y v + I_t)^2 dxdy$$

Since we assume that the velocity is same over *W*, We can further simplify this equation as:

$$u\int\int I_x^2 dxdy + v\int\int I_x I_y dxdy + \int\int I_x I_t dxdy = 0$$
$$u\int\int I_x I_y dxdy + v\int\int I_y^2 dxdy + \int\int I_y I_t dxdy = 0$$

This is the final equation to be solved and can be implemented using the OpenCV image processing library.

Appendix B - Shannon's entropy

B.1 Introduction

In this chapter it is going to discuss the Shannon's entropy, which is used as an average measure of information contained in an image.

B.2 Calculating Shannon's entropy

Shannon's entropy represents the average amount of information contained in a random variable, which is defined by the equation:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

Where X is a discrete random variable with n outcomes and $p(x_i)$ is the probability mass function of outcome xi. A probability mass function is used to calculate the probability that a discrete random variable is exactly equal to some value. Base of the logarithm b is considered as 10. For a gray-scaled image, n represents the value 256 and the range of the random variable is from 0 to 255.

When the outcome of the Shannon's entropy is low, the difference of gray levels in the input image is low and considered to be an obstacle.

Appendix C - Emulating Optical Flow

C.1 Introduction

This chapter contains the details of the developed Optical Flow agent, emulated inside the android emulator.

C.2 Implementation of the Optical Flow agent

Figure C.1 represents a screen-shot of the optical flow agent implemented using the Android emulator. Lines represent optical flow vectors.

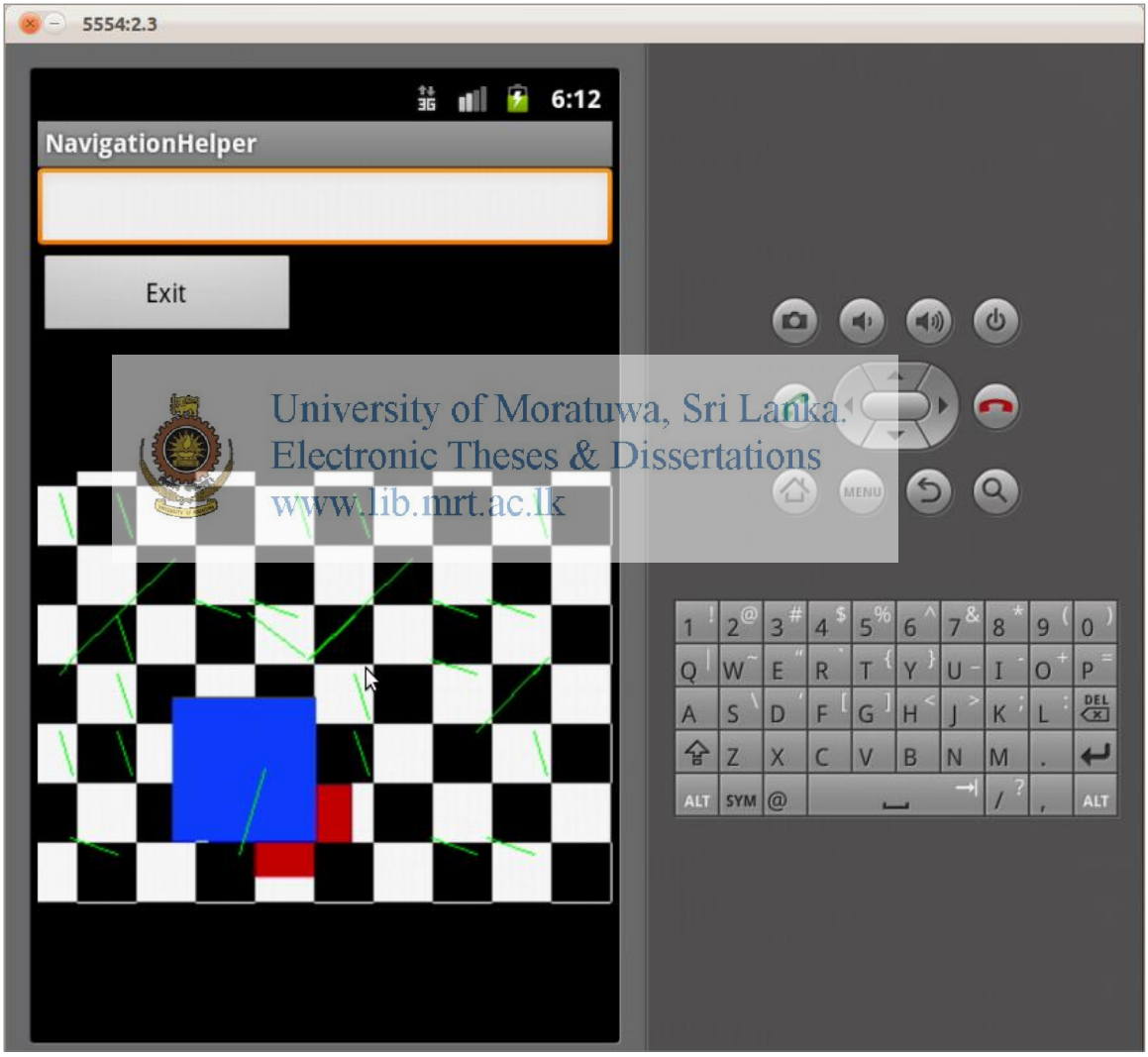

Figure C.1 - Implementation of the optical Flow agent

Appendix D - Sample obstacle detection scenarios

D.1 Introduction

This chapter contains some additional obstacle detection scenarios, apart from the scenarios discussed in the evaluation chapter.

D.2 Sample scenarios

Figure D.1 represents a scenario where an obstacle is moved towards the camera and Figure D.2 represents a scenario where the camera is moved towards an obstacle. The system was able to detect the obstacle and produce a warning in both scenarios.

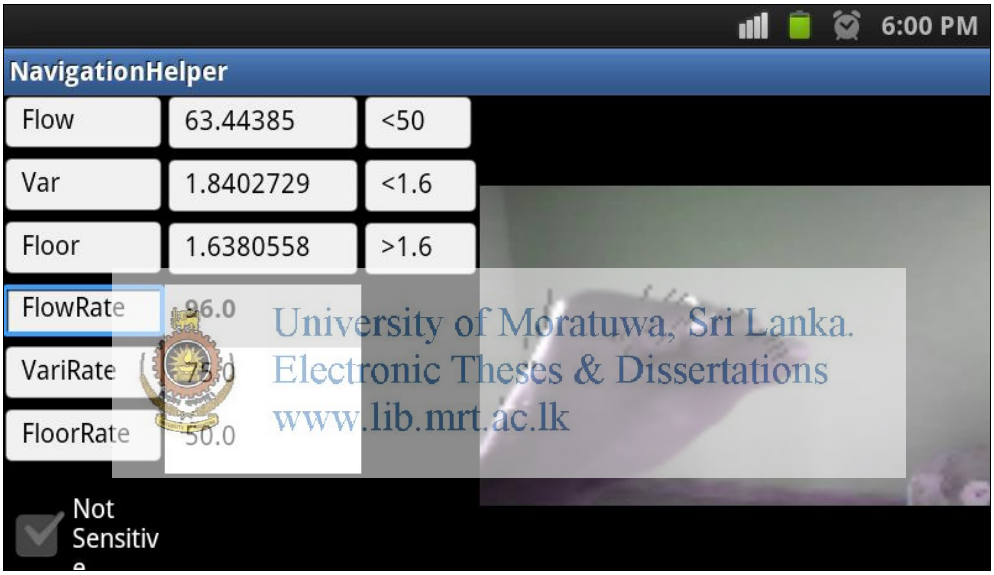

Figure D.1 – Moving an obstacle (a human hand) towards the camera

Figure D.2 – Moving the camera towards an obstacle