Obstacle Avoidance in Cluttered Environments using Optic Flow

Matthew A. Garratt
University of New South Wales, Australia
m.garratt@adfa.edu.au

Allen Cheung
University of Queensland, Australia
a.cheung@uq.edu.au

Abstract

Autonomous navigation of a flying vehicle in a cluttered environment has been simulated to demonstrate novel yet simple techniques for avoiding obstacles using optic flow and image loom signals whilst successfully navigating a vehicle towards a goal. The simulation makes use of simulated flow fields calculated on cameras looking downwards, forwards, rearwards and sidewards. This computation could be carried out using a single processor networked with multiple remote cameras or using separate optic flow sensors oriented in each direction.

1 Introduction

A plethora of MAVs have been produced in recent time that can be safely operated in populated areas owing to their small size and low kinetic energy. Such craft can be flown into a building, through a sewer, down a busy street etc. The applications of MAVs to military operations are many and varied including surveillance inside buildings, search and rescue, hostage situation monitoring, chemical agent detection and bomb search. The problem of navigating a flying robot through a cluttered terrain is of significant interest, since conventional techniques such as GPS technology tend to fail in urban canyon environments and will not work indoors. Remote piloted vehicles also become problematic when line-of-sight is lost and telemetry systems start to experience frequent dropouts. Due to payload restrictions for MAVs, the only practical sensor to achieve collision avoidance on this scale is a visual sensor. Alternative technologies such as laser range-finding, radar and high-grade inertial systems are simply impractical for small UAVs owing to the physical barriers to miniaturisation that exist. Extra advantages of visual guidance are that it is passive, small and low cost. In our prototype systems, components for a guidance payload could be purchased for as little as US$500 and such a system would weigh less than 50g.

2 Related Work

Motivation for this project is partly derived from observations from biology. Insects with tiny brains less than one tenth of a milligram and tens of thousands of times less complex than a human brain can take off, fly, navigate and land using vision. Because of their small size, autonomy systems for small autonomous vehicles also need to be high-bandwidth and precise, yet also low-weight and low power. Many believe (e.g.[Iida, 2001; Green et al., 2003; Thakoor, et al., 2003; Srinivasan et al., 2004]) that these seemingly contradictory requirements can be met through biomimetic sensing, particularly, through the use of visual sensors, rather than by conventional techniques. Experiments in biology [Srinivasan et al., 1989] have led researchers to the conclusion that flying creatures use image motion or *optic flow* to determine range to objects and hence avoid obstacles in flight.

To date, optic flow has been used in two main ways for controlling flight vehicles. Firstly, given ground speed from a non-visual sensor such as GPS, optic flow can be used to estimate range. This has been used for the problem of terrain following [Garratt and Chahl, 2008], centering in a natural canyon [Griffiths et al., 2006] and in conjunction with stereo vision for guiding a UAV through an urban canyon [Hrabar et al., 2005]. The second way that optic flow has been used is to estimate velocity given a measure of range from another sensor such as stereo vision or a laser rangefinder [Garratt and Chahl, 2007].

An excellent contribution to the solution of urban navigation using vision is [Beyeler et al., 2007], in which a helicopter simulation is combined with graphics generation and optic flow processing to demonstrate obstacle avoidance in an urban environment. In their simulation, the helicopter is steered in a direction to balance the optic flow on left and right sides of the field of view of a forward looking camera. When optic loom exceeds a threshold, the helicopter executes a 180° turn. In Beyeler et al's work, the helicopter is assumed to only travel for-

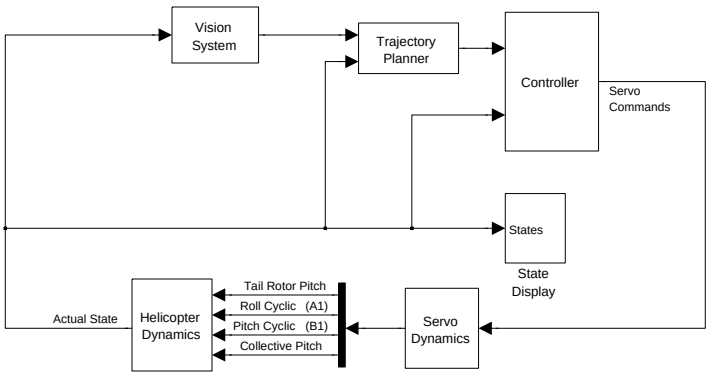

Figure 1: SIMULINK Simulation of Closed Loop Helicopter

wards so that side ways flight is not permitted. The helicopter also does not work towards a goal, only executing a random flight path and the simulation requires that the urban environment is surrounded by a perimeter wall to prevent the helicopter from flying outside of the virtual environment. In our scheme we proceed towards a goal and we do not make assumptions about the motion of the helicopter, relying instead on the realistic helicopter dynamics, so that lateral motion occurs. To achieve this, we use optic flow from multiple cameras to avoid obstacles on all sides.

3 Simulation Details

A closed loop micro helicopter simulation has been developed which includes a graphics engine to simulate the view from multiple cameras onboard the UAV. OpenGL software is used to rapidly draw the simulated views from a downwards, upwards and two sideways looking cameras. The images are processed using code implemented in MATLAB to produce an optic flow field consisting of a grid of evenly spaced flow vectors. Figure 1 shows the top level view of our visual guidance simulator implemented in Simulink. The helicopter dynamics block is based on a small electrically propelled autonomous helicopter used by the group and has been validated thoroughly against real flight test data [Garratt, 2007]. The vision sub-systems contain blocks for generating the simulated view from each camera. The image stream from each simulated view is fed into optic flow processing blocks which calculate a vector flow field for each image frame from each camera.

3.1 Helicopter Dynamics Model

Mathematical models for small-scale helicopters have been extensively documented in the literature [Hashimoto et al., 2000; Gavrilets et al., 2001; Civita et al., 2002]. The general approach is to combine a linearised rotor aerodynamic model with the non-linear rigid body equations. Sometimes a non-linear thrust and rotor inflow model is incorporated, usually involving an iterative scheme to simultaneously solve for the thrust and the induced velocity through the rotor. The non-linear fuselage and tailplane aerodynamics forces are only usually added when forward flight is considered. An underlying theme in simulating small helicopters has been the realisation that owing to the complexity of the helicopter, minimum complexity simulations tends to be most practical, so most researchers have chosen to use the minimum level of detail required to capture the key system dynamics. This approach was championed by Heffley who published a report for NASA in 1988 on a minimum complexity helicopter model [Heffley, 1988] that is regularly cited in the field. Heffley maintains that adding higher order effects such as the off-axis moments due to blade flapping does not automatically lead to a better match with flight data. SIMULINK® was chosen as a development environment for convenience as the graphical drag and drop building block approach speeds up the development cycle and allows the simulation to be quickly adapted to other helicopters. Balancing simulation fidelity against practicality, a simulation has been created that is capable of simulating the following effects:

- Exact non-linear rigid body equations of motion;
- Wind gusts and turbulence;
- First order main rotor flapping dynamics;
- Hover, rearwards, sideways and forward flight;
- Dynamic effects of the Bell-Hiller stabiliser bar;
- Fuselage and tailplane aerodynamic forces;
- Approximate servo dynamics; and
- Sensor lags, filtering, offsets and noise.

The simulation does not model interaction of rotor downwash with the fuselage and rotor lead-lag mechanisms. Higher order flapping modes are only really of interest when studying vibration and aeroelastic problems [Leishman, 2006] so they have also been ignored. Only low rates of descent are allowed on the helicopter so that momentum theory can still be applied. This seems reasonable noting that all of the simulation scenarios relate to flight close to the ground so that high descent rates cannot be achieved. A detailed explanation and validation of the helicopter dynamics simulations can be found in [Garratt, 2007].

3.2 Graphics and Image Processing

The openGL code is written in C++ and compiled to provide a dynamic link library executable (.dll) file that will run under the Windows operating system. The .dll

library routines are called from inside the Simulink program. The routines are called every 0.02 seconds of simulation time to simulate a camera speed of 50 frames per second. The .dll function calls are passed input variables corresponding to the UAV position and attitude calculated in the simulation. The openGL executable program outputs the images as a 400×400 pixel 24-bit color image. The color image is converted into an array of 8-bit intensity values by averaging the intensities from the red, green and blue color channels. The images are displayed as the simulation proceeds. In future work, the openGL code has the potential to allow simulation of more complicated environments such as trees, shadows, rocks, multiple light sources and grass. Figure 2 shows the views from multiple cameras when the helicopter is flying inside a simulated city block environment.

For this work, we calculate the optic flow vector flow field using the Image Interpolation Algorithm, abbreviated I^2A. The I^2A is quick to execute and robust to noise. A full derivation of the I^2A can be found in [Srinivasan, 1994]. To achieve the optic flow calculation, a Simulink delay is used to store the previous frame at each sample time. The optic flow block in Simulink is implemented as compiled C code. A graphical menu, allows the parameters of the optic flow calculation to be changed. For example, the patch size in pixels can be changed and the spacing and number of flow vectors can be changed.

Simulink blocks for calculating average optic flow in the image and also image loom have been constructed. Image *loom* corresponds to image expansion or contraction due to motion perpendicular to the image plane of the camera. The loom value is extracted by subtracting the sum of the flow vectors on the left-hand side of the image from the sum of the flow vectors on the right-hand side of the image. The loom calculation can be better understood by considering the equations which relate image plane velocity (Q_u and Q_v) and vehicle motion which are given in [Corke, 2004] and reproduced as equation 1.

$$\begin{bmatrix} Q_u \\ Q_v \end{bmatrix} = \begin{bmatrix} \frac{F_x}{Z} & 0 & -\frac{U}{Z} & -\frac{UV}{F_x} & -\frac{F_x^2+U^2}{F_x} & -V \\ 0 & \frac{F_y}{Z} & -\frac{V}{Z} & -\frac{F_y^2+U^2}{F_y} & -\frac{UV}{F_y} & U \end{bmatrix} \times \begin{bmatrix} \dot{X} & \dot{Y} & \dot{Z} & p & q & r \end{bmatrix}^T \quad (1)$$

where F_x and F_y are the horizontal and vertical focal lengths of the camera respectively (equal in the simulation); U and V are the horizontal and vertical pixel locations of features in the field of view; $\dot{X}$, $\dot{Y}$ and $\dot{Z}$ are the sideways, vertical and normal velocities of the camera; and p, q and r are rotation rates about the camera axes.

Consider the horizontal component of the optic flow field defined by $Q_u(x, y)$ at a location (U, V) in the image plane. If optic flow is calculated using a symmetrical and even distribution of patches and summed over the left and right sides of the image separately and then the sums subtracted, many of the terms in equation 1 are cancelled out owing to the symmetry. The resulting relationship is:

$$\sum_{left} Q_u - \sum_{right} Q_u = \frac{\dot{Z}}{Z}\left[\sum_{left} U - \sum_{right} U\right] \quad (2)$$

The sums on the right hand side of equation 2 are simply the sums of the spatial coordinates of the centers of the patches used for optic flow computation and are trivially calculated. The loom term $L = \dot{Z}/Z$ represents the time to contact and can be calculated explicitly from the measured optic flow field using equation 3 which is derived from equation 2 after noting that the distances to the centers of the patches on left and right sides are the same.

$$L = \frac{\dot{Z}}{Z} = \left[\sum_{left} Q_u - \sum_{right} Q_u\right] \times \left[2\sum_{right} U\right]^{-1} \quad (3)$$

4 Simple Obstacle Avoidance

A simple scheme for controlling the flight of a helicopter in a cluttered environment was tested first. This scheme is sufficient to demonstrate the ability of a vehicle with small rotorcraft dynamics to avoid walls in a rectangular room 8m×16m using vision. The scheme makes use of five sensors which are oriented downwards, sideways (left and right) and fore and aft. (Note: the 5 sensors could be combined into one panoramic sensor looking with a 180° field of view with appropriate optics). To speed up execution of the simulation, optic flow vectors for sideways, front and rear images were only calculated in a horizontal row of 8 patches. In real-flight, many more vectors would be calculated, improving the accuracy of the results significantly. Optic flow on 8×8 patches was calculated for the ground camera video stream.

A simple sensor fusion loop is implemented that fuses a measurements from a simulated height sensor with inertial data and the optic flow information from the downwards looking camera (see [Garratt and Chahl, 2007] for details). The resulting lateral and directional velocity estimates are used in a PID based feedback loop to control the lateral and longitudinal motion of the helicopter. Speed is maintained at 1m/s in forward and sideways directions to create a diagonal flight path where the sense of the forward and sideways velocity components is switched in response to obstacles detected in

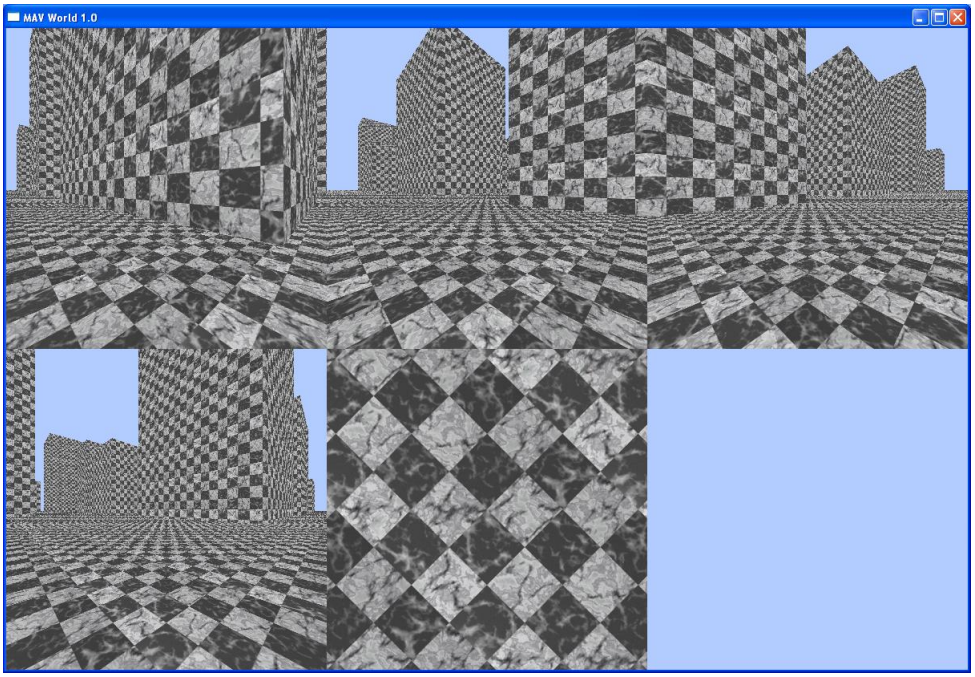

Figure 2: Example of imagery produced by the openGL software. The following camera views can be seen: left view (top left), right view (top middle), front view (bottom left), downwards view (bottom middle) and rear view (far right).

the flight path. The heading of the helicopter was kept constant throughout using a PID control loop to control yaw. The height of the helicopter above ground was also set constant at 1m using a PID control loop.

Owing to the diagonal flight direction, both front and side cameras tend to observe flow from translation. The exception to this is when the direction of flight is directly towards an obstacle. In this case, the presence and relative range of an obstacle is determined from the expansion of the flow vectors (loom).

If the range calculated from flow (or loom) is less than some threshold on the side, the helicopter lateral velocity component is reversed to take the vehicle away from the obstacle. Similarly if the range calculated from flow (or loom) is less than some threshold on the front or back, then the helicopter reverses its longitudinal direction away from the obstacle.

The resulting path of the helicopter is a zig-zag through a cluttered environment which results in an exploration of the environment. Figure 3 shows the results of simulations for the helicopter inside the room when the helicopter is commanded to maintain a heading which is (a) initially aligned with the surrounding walls; and (b) offset by 22.5° from the longer walls. In case (a), the loom signal may not be necessary as the diagonal motion of the helicopter tends to stimulate flow from translation on all cameras at once. As the helicopter nears a wall, an increase in optic flow and loom can be seen for any cameras which image the wall. The combination of loom and optic flow from translations is sufficient to prevent the helicopter from striking the walls.

4.1 Goal Directed Behaviour

The work on simulation of navigation was extended to include bias towards a particular goal. The approximate positions of the goal and the current vehicle position is assumed to be known from a means such as GPS. (Alternatively, optic flow odometry could be used to navigate towards a point at a set distance and bearing from the starting position. The basic principle of optic flow odometry is to integrate the optic flow from the sensor over time to compute distance travelled.) In the latest scheme, velocity commands are sent to the lower level horizontal controller. The velocity is sensed by a downwards looking optic flow sensor that measures sideways and longitudinal drift.

In this means of control, we again simulate the use of five cameras looking left, right, forwards, rearwards and down. Each camera has a field of view of 90 degrees and the resolution of each image is 400×400 pixels. This field of view provides complete coverage of the surrounding environment, so that obstacle avoidance can be achieved when flying in any direction. Optic flow is calculated on the image from each camera using a grid of 16×16

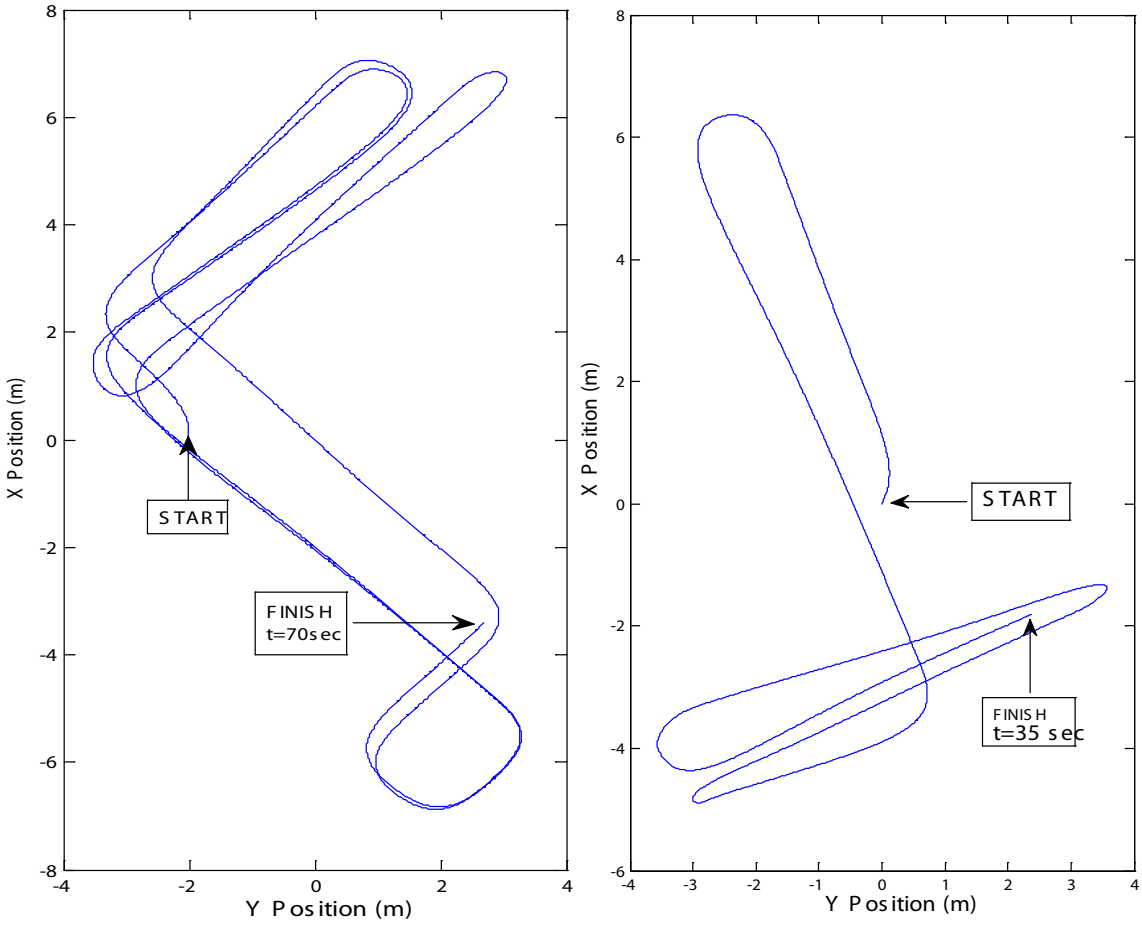

Figure 3: Trajectory of helicopter in horizontal plane within an 8m×16m arena for (a) heading aligned with longer walls and (b) offset by 22.5° from the longer walls.

flow vectors and then the average of the horizontal flow component from each vector is calculated to provide one optic flow scalar values from each camera. Each flow vector is calculated using a 64×64 pixel patch.

The velocity commands are based on empirical formulae which blend the need to avoid obstacles with the desire to head towards the goal. The longitudinal velocity command (V_x^{cmd}) consists of the sum of:

- A gain k_1 multiplied by the difference in average horizontal optic flows (Q_A and Q_F) between the aft and front views, divided by the lateral velocity (V_y) of the vehicle. As low lateral velocity results in a very noisy signal, the lateral velocity is saturated to a minimum velocity of 0.2m/s.
- A gain k_2 multiplied by the difference in average loom between the aft and front views, divided by the longitudinal velocity (V_x).
- The longitudinal component of the desired velocity towards the target (V_x^{tgt}). The desired velocity is simply a fixed velocity of 2 m/s in a direction directly towards the target. In some cases, a slightly curved or spiral path can be used by following a vector which is pointed slightly away from the target. This helps prevent the helicopter from getting trapped in a limit cycle trajectory where it moves backwards and forwards repeatedly over the same path. Although, unlikely in practice, this could theoretically occur in the simulation if an obstacle was blocking the desired trajectory and the surrounding obstacles were arranged perfectly symmetrically.

The lateral velocity command (V_x^{cmd}) is structured in the same way except that the flow in left (Q_L,L_L) and right (Q_R,L_R) views is differenced instead. The equations for the command velocities are as below:

$$\begin{aligned} V_x^{cmd} &= V_x^{tgt} + k_0 \frac{(L_A - L_F)}{V_x} + k_1 \frac{(Q_A - Q_F)}{V_x} \\ V_y^{cmd} &= V_y^{tgt} + k_0 \frac{(L_L - L_R)}{V_y} + k_1 \frac{(Q_L - Q_R)}{V_y} \end{aligned} \quad (4)$$

In certain situations, the equations above could lead to low velocities so that flow in all directions becomes limited. If this were permitted to continue, it might be possible for the UAV to gradually drift in to an obstacle. To eliminate the risk of this happening, the velocity commands in equation 4 are normalised to provide a vector magnitude of 2m/s whilst keeping the same ratio between lateral and longitudinal velocities. The helicopter therefore is commanded to maintain a 2m/s velocity in a direction determined to fly towards the target whilst moving away from obstacles. The gains k_0 and k_1 can be adjusted to regulate how far from obstacles the UAV should fly. If the gains selected are too small, the UAV will fly into obstacles. If the gains are made too high the commands to the velocity controller can change too rapidly, placing undue strain on the servos. For the experiments herein, we found suitable values of k_0 and k_1 to be 4.0 and 5.0 respectively.

For testing, an urban environment was created in an area of 300×300m, consisting of random sized buildings arranged approximately in a grid pattern. The views from the 5 cameras are shown in figure 2. Outside the building array was open space i.e. no boundary. The helicopter started at a position bottom right of the building (x=-130m, y=80m) and was given a target location of (x=110m, -110m). Height, horizontal velocity and heading were controlled using the same scheme described for the simple obstacle avoidance strategy.

Figure 4 shows the trajectory where the heading of the helicopter was maintained parallel to the X-direction. Figure 5 shows the trajectory where the heading of the helicopter was at 22.5° to the right of the X-direction. Some simulations were attempted using a heading of 45°, but this was subject to occasional infringements on the building owing to a small blind spot region on the boundary of the four views. Each camera has a field of view of 90 degrees but optic flow is not calculated across the edge of the images as the reference shift pixels would fall off the side of the image. As a result, the effective field of view of each camera was only about 70°. This problem could be prevented using more views or by overlapping the field of view of each camera.

In both cases, the helicopter is able to maintain clearance with the buildings. At times the helicopter does come close to the buildings, but does not cross the wall boundaries. The helicopter tends to 'hug' the walls as it moves. Figure 6 shows the actual and commanded velocities given to the low-level velocity controller. The helicopter responds quickly to the commanded velocities but provides a filtering effect when the commanded velocities fluctuate rapidly. Some pre-filtering of the commanded velocities may improve the performance. Figure 7 shows the optic flow and loom from each camera during the simulation.

5 Conclusion

Simulations of autonomous navigation of a flying vehicle in a cluttered environment have been progressed. The simulations demonstrate novel yet simple techniques for avoiding obstacles using optic flow and image loom signals and can successfully be used to navigate a vehicle towards a goal. The simulation makes use of simulated flow fields calculated on cameras looking downwards, forwards, rearwards and sidewards.

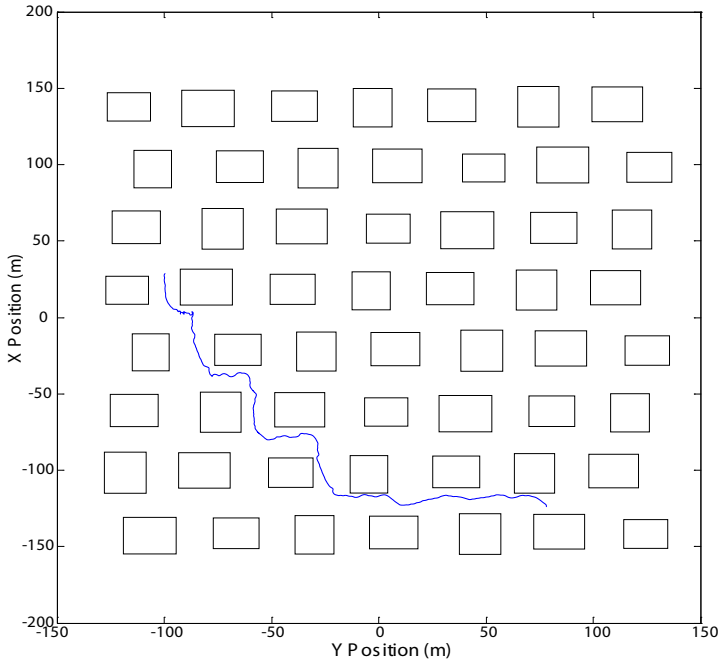

Figure 4: Plan view of helicopter trajectory in virtual city with heading parallel to city street. Rectangles represent buildings.

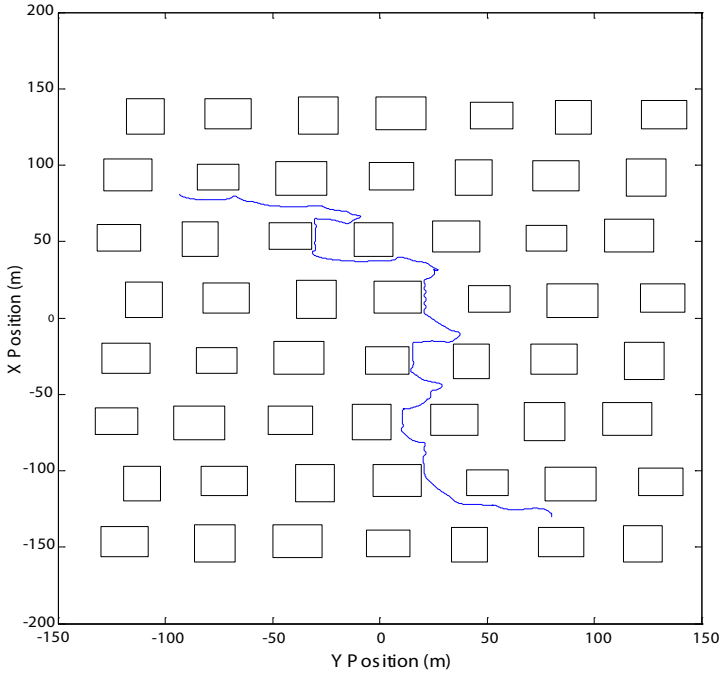

Figure 5: Plan view of helicopter trajectory with heading 22.5° to the street.

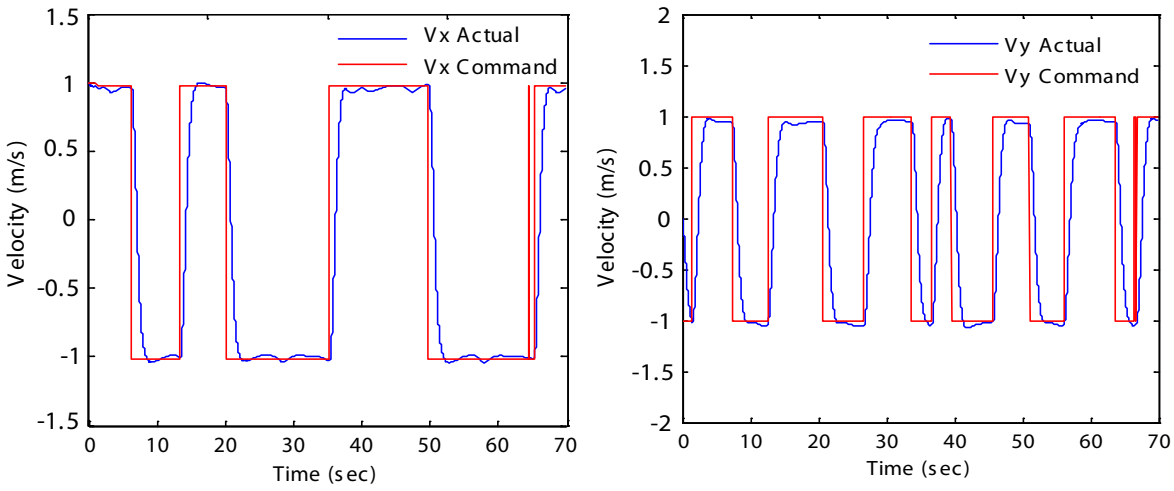

Figure 6: Commanded and actual velocities for trajectory with heading parallel to the urban streets

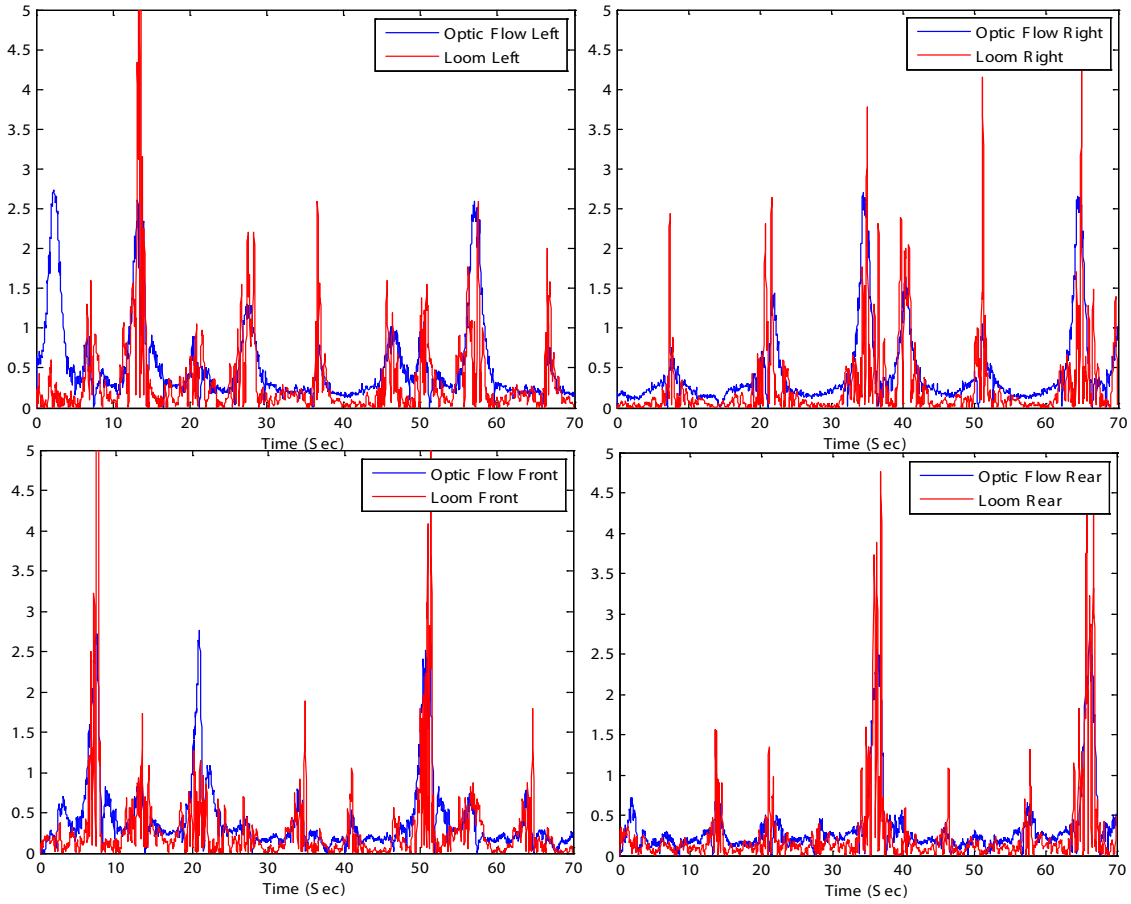

Figure 7: Optic flow and loom results for trajectory with heading parallel to the urban streets

Acknowledgments

This work has been partly supported by seed funding from the US Army International Technology Center Pacific (ITC-PAC).

References

[Barrows, 2002] G.L. Barrows. Future visual microsensors for mini/micro-UAV applications. In *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications*, pages 498–506, 2002.

[Beyeler et al., 2007] A.Beyeler, J.Zufferey, and D.Floreano. 3D Vision-based Navigation for Indoor Microflyers. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation Roma, Italy*, pages 1336–1341.

[Chahl, 2004] J.S.Chahl. Terrain following for UAVs using optical flow. In *Proceedings of Unmanned Systems North America 2004*, Anaheim Ca.

[Civita et al., 2002] M.La Civita, W.C.Messner, and T.Kanade. Modelling of small-scale helicopters with integrated first-principles and system-identification techniques. In *Proceedings of the American Helicopter Society 58th Annual forum*, Montreal, Canada, 11-13 June 2002.

[Corke, 2004] P.Corke. An Inertial and Visual Sensing System for a Small Autonomous Helicopter Journal of Robotic Systems 21(2):43–51, 2004.

[Garratt, 2007] M.A.Garratt. *Biologically Inspired Vision and Control for an Autonomous Helicopter*. PhD thesis, Australian National University, 2007.

[Garratt and Chahl, 2007] M.A.Garratt and J.S.Chahl. An optic flow damped hover controller for an autonomous helicopter. In *Proceedings of the 22nd International UAV Systems Conference*, Bristol, UK.

[Garratt and Chahl, 2008] M.A.Garratt and J.S.Chahl. Vision-Based Terrain Following for an Unmanned Rotorcraft Journal of Field Robotics 25(4-5):284–301, 10 Apr 2008.

[Gavrilets et al., 2001] V.Gavrilets, B.Mettler, and E.Feron. Nonlinear model for a small-size acrobatic helicopter. In *AIAA Guidance, Navigation, and Control Conference*, 6-9 August 2001.

[Green et al., 2003] W.Green, P.Oh, K.Sevcik, and G.Barrows. Autonomous landing for indoor flying robots using optic flow. In *Proceedings of the 2003 ASME International Mechanical Engineering Congress*, Washington, USA.

[Griffiths et al., 2006] S.Griffiths, J.Saunders, A.Curtis, B.Barber, T.McLain, and R.Beard. Maximizing miniature aerial vehicles: Obstacle and terrain avoidance for MAVs. *IEEE Robotics and Automation Magazine*, pages 34–43.

[Hashimoto et al., 2000] S.Hashimoto, T.Ogawa, S.Adachi, A.Tan, and G.Miyamori. System identification experiments on a large-scale unmanned helicopter for autonomous flight. In *Proceedings of the 2000 IEEE International Conference on Control Applications*, pages 850–855, Anchorage, Alaska, 25–27 September 2000.

[Heffley, 1988] R.K.Heffley. *Minimum-Complexity Helicopter Simulation Math Model*, USAAVSCOM Technical Report 87-A-7 edition, April 1988. Prepared for US Army Research and Technology Activity under contract NAS2-11665.

[Hrabar et al., 2005] S.Hrabar, G.Sukhatme, P.Corke, P.Usher, and J.Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3309–3316.

[Iida, 2001] F.Iida. *Goal-Directed Navigation of an Autonomous Flying Robot using Biologically Inspired Cheap Vision*, In *Proceedings of the 32nd International Symposium on Robotics* 19-21 April 2001.

[Leishman, 2006] J.G.Leishman. *Principles of Helicopter Aerodynamics*, volume 2nd Edition. Cambridge University Press, New York, 2006.

[Srinivasan, 1994] M.Srinivasan. An image interpolation technique for the computation of optic flow and egomotion. *Biological Cybernetics*, 71:401–415.

[Srinivasan et al., 1989] M.Srinivasan, M.Lehrer, S.Zhang, and G.Horridge. How honeybees measure their distance from objects of unknown size. *Journal of Comparative Physiology*, A(165):605–613.

[Srinivasan et al., 2004] M.V.Srinivasan, S.W.Zhang, J.S.Chahl, G.Stange and M.A.Garratt. *An Overview of Insect Inspired Guidance for application in ground and airborne platforms , Proceedings of Inst Mech Engineers Part G: Journal of Aerospace Engineering*, 218:375–388, 2004.

[Thakoor, et al., 2003] S.Thakoor, N.Cabrol, N.Lay, and J.S.Chahl. *The Benefits and Applications of Bioinspired Flight Capabilities.* Journal of Robotic Systems 2003, pages 687–706.

[Zufferey et al., 2006] J.Zufferey, A.Klaptocz, A.Beyeler, J.Nicoud, and D.Floreano. A 10-gram Microflyer for Vision-based Indoor Navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.*