Real Time Motion Processing for Autonomous Navigation

J. Kolodko and L. Vlacic

Abstract: An overview of our approach to autonomous navigation is presented showing how motion information can be integrated into existing navigation schemes. Particular attention is given to our short range motion estimation scheme which utilises a number of unique assumptions regarding the nature of the visual environment allowing a direct fusion of visual and range information. Graduated non-convexity is used to solve the resulting non-convex minimisation problem. Experimental results show the advantages of our fusion technique.

Keywords: Machine vision, navigation systems, sensor fusion.

1. INTRODUCTION

For humans, navigating in a complex, dynamic environment is second nature however, engineers are yet to design an autonomous vehicle that can reliably complete this task in an unstructured environment. Our aim is to bring this goal one step closer to reality by designing a practical system that can operate in dynamic environments similar to public roads. In this presentation, robots are the only moving obstacles. We intend to achieve our goal by using motion as a low level, primitive quantity rather than by using high level features such as image tokens.

Our approach has several key features: A multi-stage architecture to solve the overall motion processing problem and its mapping to hardware. A core set of assumptions regarding the environment. A robust fusion of range and visual information to give preliminary motion information. A navigation approach designed to complement existing methodologies. Offline test data generation for algorithm validation.

The first section of this paper introduces each of these features. Subsequent sections give a more detailed description of our short range motion estimation system, directions for future work, and conclusions.

1.1. Multistage motion processing architecture

Fig. 1 illustrates our proposed multistage motion processing architecture. Preprocessed visual information and range data are passed to a robust, incremental motion estimation algorithm, which is mathematically simple. The algorithm minimises the processing required at each frame (leading to simpler hardware implementation) while still providing an estimate of the motion and boundaries of dynamic objects. The optical flow constraint equation and the fundamental equations of motion are used in combination to fuse visual and range data, eliminating depth ambiguity.

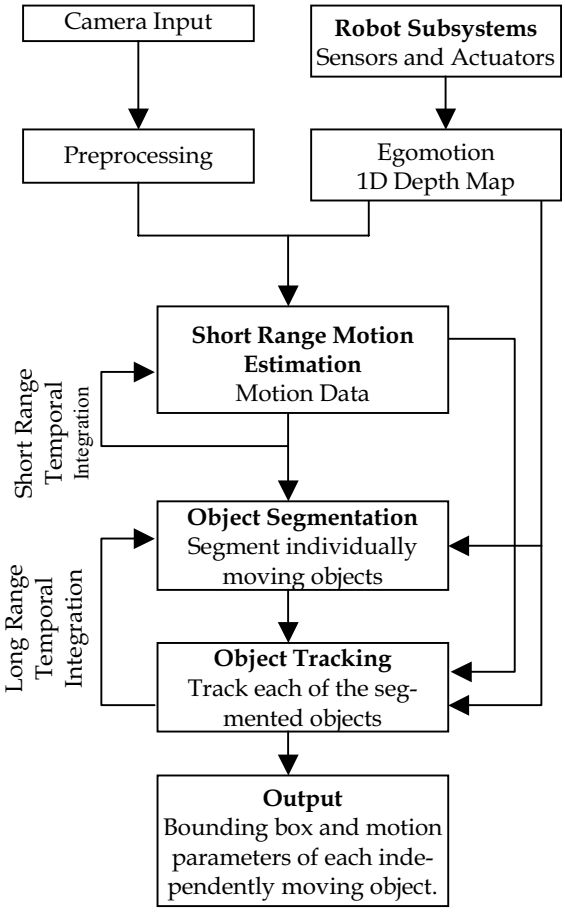

Fig. 1. Block diagram of motion processing system.

Manuscript received February 28, 2002; revised February 28, 2002; accepted June 24, 2002. Our thanks go to the Fraunhofer Autonomous Intelligent Systems (AiS) group for their generous donation of the Signal Master Platform and Fuga Camera used in this work.

Julian Kolodko is with the Intelligent Control Systems Laboratory, Griffith Univesrity, Australia. (e-mail: j.kolodko@sct.gu.edu.au).

Ljubo Vlacic is with the Intelligent Control Systems Laboratory, Griffith University, Australia. (e-mail: l.vlacic@mailbox.gu.edu.au).

Dynamic scale space gives highest priority to the nearest object (the one with which collision is most likely) while preventing temporal aliasing. A blend of regularisation and regression improves noise robustness and gives a one dimensional motion estimate.

The resulting motion estimate is passed to a segmentation routine that fuses a range of information in order to give a final estimate of object boundaries. Predicted position feedback is used to incrementally improve results. Finally, the resulting tracking information (position and velocity) for each object is passed to the vehicles navigation systems.

For motion information to be useful in a dynamic world, it must be extracted quickly relative to the veloities of objects in the environment so that navigation decisions are made with up-to-date information. However, motion estimation is highly processor intensive due to the massive amount of information contained in video data; thus, any solution to this problem requires significant processing power in addition to clever algorithms. To this end, we have chosen the Signal Master Platform, coupled with a Gatesmaster add-on board [8], providing us with a combination of Sharc Digital Signal Processor (DSP) [9], 486 and Virtex Field Programmable Gate Array (FPGA) processing element and a wealth of interface options. Video data comes from a compact Fuga 15D camera that does not require a frame grabber – rather it is accessed directly through a digital interface much like a simple RAM.

Fig. 2 illustrates how our processing architecture is mapped onto hardware resources. Motion processing, along with the requisite memory management functionality and interfacing occurs in FPGA. For preliminary testing purposes, this output is fed to a PC for visualisation, though, in the final system, motion processing results are passed to the DSP for final segmentation and tracking. The output from the DSP is a set of position and velocity estimates that can be passed to the robot, providing an extra sensory mode for navigation planning.

1.2. Core assumptions

We make a number of assumptions in order to simplify motion processing. Our system utilises a single dimensional range scanner, generating a single range measurement for each image column hence we must assume that this measurement is correct throughout the column (i.e. there is a single planar object visible in that column). To ensure the validity of this assumption, we use 512*32 pixel images. These narrow images greatly reduce the likelihood that two obstacles (or planes) are visible in any given column and reduces the computational workload. Our work also assumes flat illumination (minimising photometric problems, a reasonable assumption in our laboratory test environment), and rigid ground plane motion.

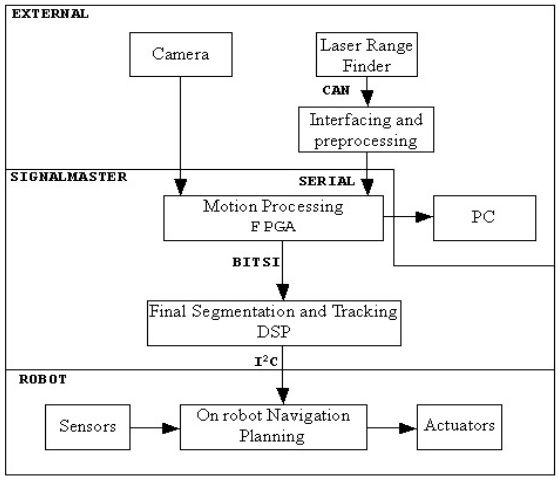

Fig. 2. Mapping algorithms to hardware.

1.3. Navigation approach

Autonomous Navigation schemes are generally comprised of two integrated stages [1, 5]. Given the robots current location, goal location, and the structure of the environment, a path plan is created. Short term, or reactive planning (obstacle avoidance, lane keeping etc.) ensures that the path plan is successfully executed. This can be an iterative scheme: the path plan may be re-evaluated if reactive planning shows the current plan is unfeasible.

We incorporate a dynamic obstacle reasoning (DOR) stage into an existing navigation scheme as follows. Using a combination of motion data from our motion sensor and object location from other sensors DOR creates a map in three-dimensional trajectory-velocity-time space. Using this map, the path with highest available velocity for the longest period of time is selected provided this path is consistent with (a) the overall path plan and (b) a set of "road rules." These road-rules are required to prevent deadlock in multi-robot environments and ensure efficient use of the "road."

Our robotic test beds are configured with a simple path-planning scheme; the goal is simply to reach a position relative to the starting position with no apriori knowledge of the environment.

DOR will enforce the rules of the (Australian) road in the sense that one should pass to the left of static obstacles. This planning is by no means an optimal strategy in terms of time, distance or resources; however it is realistic in terms of existing road infrastructure. On the road, overtaking is a special case of obstacle avoidance since we pass to the right of a slower vehicle. However, to maintain uniformity, our DOR-equipped robot will overtake an obstacle on obstacles left hand site.

1.4. Testing

We plan to test algorithms in an offline simulation

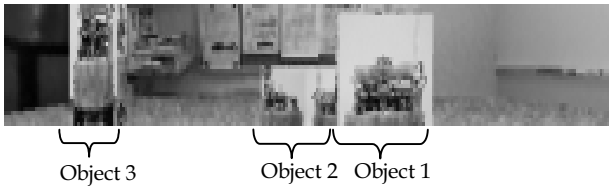

Fig. 3. Visual scene, our database.

before proceeding to hardware implementation. Doing this allows algorithms to be validated and optimized without the peripheral problems associated with hardware implementation.

For simplicity, we use MATLAB, whose matrix based variables are ideally suited to image processing applications. Algorithms are implemented, and then tested by using a database of visual and range data generated by our simulation package. The simulation package used to generate these scenarios is based on Polyray [10], a freeware ray tracing program. The simulation package allows the user to specify a simple virtual environment including the intrinsic and extrinsic camera parameters, camera motion, range finder parameters and motion, as well as the shape, texture and motion of environmental objects and lighting. The result is a set of video, range and "ground truth" motion information. Scenarios in our database are designed to be visually similar to our laboratory environment. Fig. 3 show a typical scenario.

2. ROBUST DATA FUSION

In this section, we discuss our short range motion estimation scheme in greater detail. The goal of this scheme is to create an initial estimate of the boundaries of moving objects. We achieve this goal by combining both visual data and range data in a robust statistical framework.

Our approach to motion estimation is based on the optical flow constraint equation (OFCE) [4] and is broadly similar to Blacks' [2] approach; however, we utilize the assumptions presented in Section 1.2 to reduce the computational load. Furthermore, this algorithm fuses range information and visual data into a single objective function rather than creating separate object functions for range and visual data, respectively, as in [7].

We begin with the OFCE (eq 1) where I_x, I_y and I_t are image derivatives and $\boldsymbol{u},\boldsymbol{v}$ are the horizontal and vertical components of image velocity, respectively (bold italics indicate parameters being estimated).

$$\boldsymbol{u}\mathrm{I}_x + \boldsymbol{v}\mathrm{I}_y + \mathrm{I}_t = 0. \tag{1}$$

Apparent vertical motion is of little interest in our work since all objects in the environment are constrained to move on a ground plane. In this situation apparent vertical motion can only occur when objects approach or recede from the camera, or when an uneven ground plane causes "noise motion." For the purposes of this work we assume $\boldsymbol{v}$ is zero.

We now apply the assumption that there is a single object in every image column. Rather than compute $\boldsymbol{u}$ at every pixel, we use an entire column as the region of support for our motion estimate. This approach is valid if a column contains a single object, but it is possible for more than a single object to be visible in any given column, so we need to weaken this assumption. Consider aligning the camera so that the point of contact between objects and the ground is below the area visible to the camera. This alignment guarantees that objects will begin from the bottom of the image though it does not guarantee that the object will reach the upper boundary of the image. To minimise the effect of secondary objects near the top of the image, we weight data at the bottom of the image more highly than data at the top, creating the weakened assumption that we seek. Combining this information with the fundamental equations from [6] and perspective projection gives the estimation problem

$$\min_{Tx} \sum_{rows \in col} W_{row}\rho\left(\frac{f\psi}{Z_{col}}T_{x_{col}}I_{x_{row,col}} + I_{T_{row,col}}\right). \tag{2}$$

Here W_{row} is the weighting function and f is the camera focal length, Z_{col} is the depth at the current column and ψ is a constant that converts our motion estimate from meters/sec on the camera surface to pixels/frame. Traditionally the estimator function ρ is quadratic, leading to a least squares problem, but we use the Lorentzian [2] for increased robustness. As a result of substituting the fundamental equations into (1) we are no longer solving for apparent horizontal image velocity $\boldsymbol{u}$ but for apparent left to right velocity in physical space $\boldsymbol{Tx}$.

Because we expect neighbouring velocity estimates to be alike, we add a smoothness term to form the final estimation problem

$$\min_{Tx} \sum_{cols \in image}\left\{\sum_{rows \in col}\lambda W_{row}\rho\left(E_{D_{row,col}}\right) + \sum_{g \in clique}\rho\left(T_{x_{col}} - T_{x_g}\right)\right\}, \tag{3}$$

where E_D is the bracketed term from (2), λ is the relative weight of the data term and *clique* is the local neighbourhood about the current column. In general, this is a non-convex minimisation [3] making it difficult to solve. Our solution uses graduated non-convexity [3] and successive over-relaxation due to their suitability for hardware implementation.

Robust estimation functions (ρ) have a parameter used to adjust sensitivity to outliers (data that does not fit the current model), a parameter which effectively adjusts the algorithms' sensitivity to object boundaries. We observe that object boundaries usually correspond to range discontinuities; hence, it would make sense to modulate the sensitivity parame-

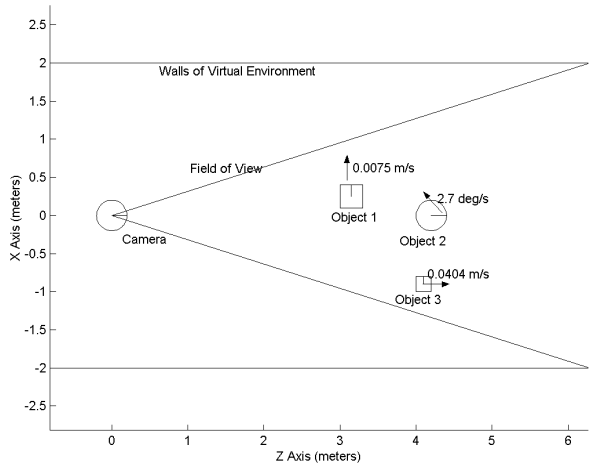

Fig. 4. Scenario 1.

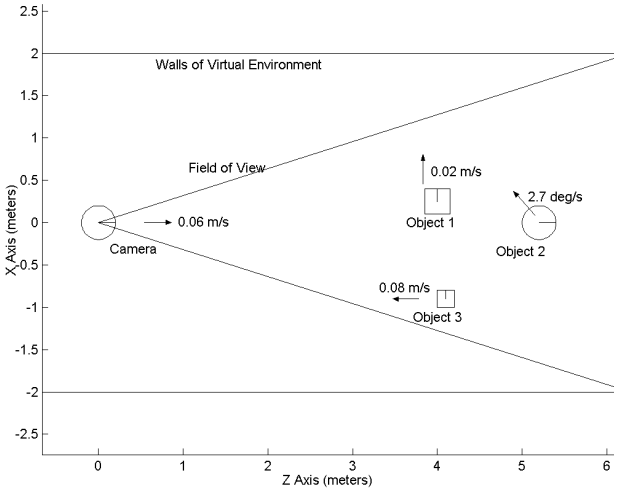

Fig. 7. Scenario 2.

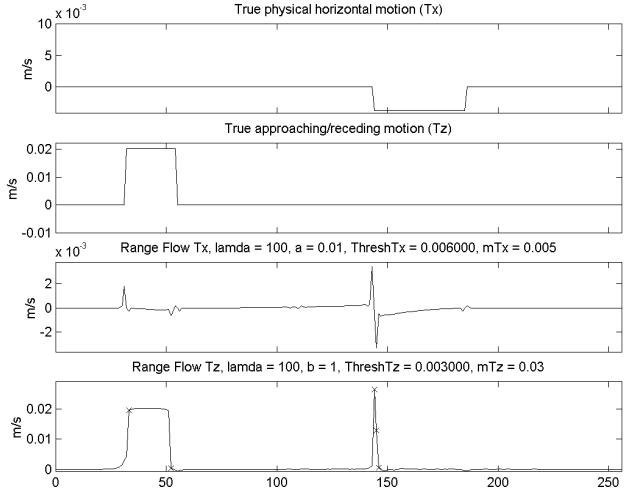

Fig. 5. Range flow.

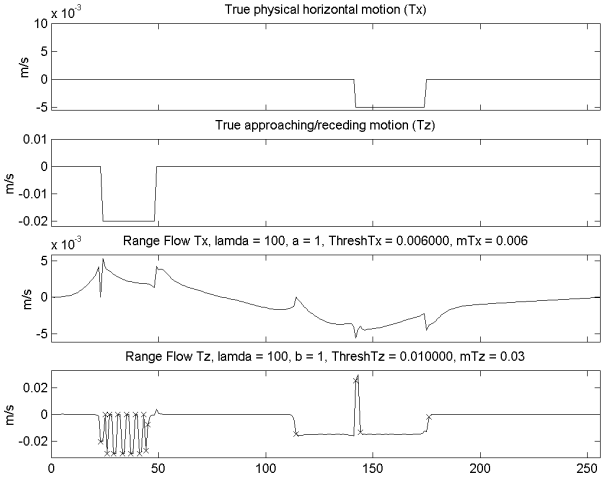

Fig. 8. Range flow.

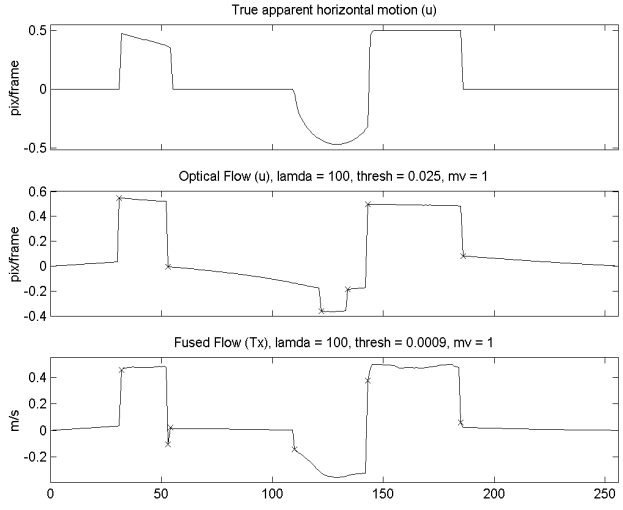

Fig. 6. Optical and fused flow

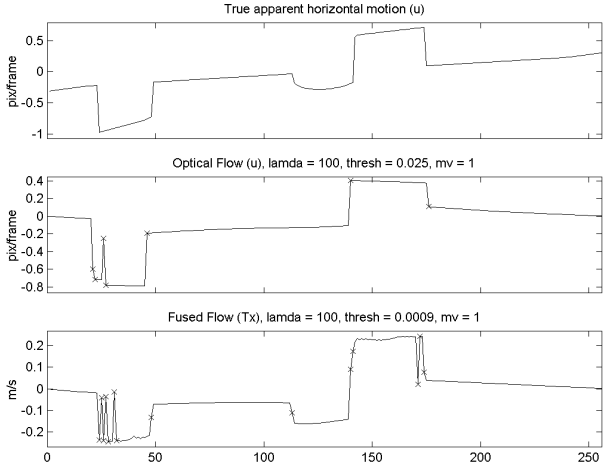

Fig. 9. Optical and fused flow.

ter so that motion discontinuities are more likely to be formed at range discontinuities than elsewhere. One could also filter the detected edges based on the presence of range discontinuities. Our system performs both operations.

3. EXPERIMENTAL RESULTS

To show that our fused approach to motion estimation produces improved results, we compare it to a purely optical approach (similar to [2]) and a purely range-based approach based on the range flow constraint equation [7]. Each algorithm is implemented using the same statistical framework and similar parameter values, where practical. In each case 200 iterations are performed. The test scenarios are designed to avoid temporal aliasing (i.e. motion is less than 1 pixel per frame). Our first test scenario (Fig. 4) has a stationary camera and three moving objects – Object 1 is translating parallel to the camera, Object 2 is rotating and Object 3 is moving perpendicular to the camera.

Fig. 5 shows the results of using only range data. In this situation, we are able retrieve two components of velocity, parallel (Tx) and perpendicular (Tz) to the camera's focal plane. The upper two plots show the theoretically correct result while the lower two plots show the result of our processing. We can see that Tz is approximately correct, except for glitches caused by derivatives. Crosses in these plots indicate locations where the algorithm has detected an object edge. Tx is more problematic and shows the limitations of a purely range-based approach. The problem is again the derivatives, which are rather uniform due to the smooth nature of our objects. Because of this, only the edges of the third object are detected. Visual approaches also require a derivative that changes from point to point, but "flat" derivatives are more common with range data because relatively flat surfaces are abundant. Finally, notice that the second object, which is rotating, was missed all together. This is a correct result: the algorithm only detects objects whose range changes.

Fig. 6 shows the advantage of using visual information. The upper plot indicates the ground truth optical flow. The second graph shows the result for pure visual data. This result matches well with the expected result (top Fig. 5) though it is clear that there are some problems around object 2. Data fusion overcomes this problem (bottom plot), locating the edge of the rotating object and giving a more accurate velocity profile over all. As mentioned above, our application does not require the detection of rotating objects but raw visual information can not distinguish between rotational and translation motion so the object is detected.

Our second scenario is similar to the first (see Fig. 7) – the primary difference is that the camera is now moving.

In this case, the results using range data are very poor (Fig. 8) again due to the nature of the range derivatives used in the computation. The central graph in Fig. 9 shows that using purely visual information has considerable advantage over the use of range data alone. However the lower graph shows again that the fused approach is better able to discern object boundaries, though it is susceptible to over segmentation. Neither of the visual based approaches was able to detect the subtle shift in apparent velocity caused by the cameras forward motion.

4. FUTURE WORK

The algorithms discussed above form one part of our overall system. Immediate work includes implementing an incremental approach for these algorithms so that fewer iterations are required per frame. Such an approach will facilitate hardware implementation. Further work is necessary to find the optimal parameter tuning and the source of the over-segmentation in Fig. 9, which appear to be related to the structure of the algorithm and to parameter tuning, in particular.

5. CONCLUSIONS

In this paper, we presented the overall framework of our navigation scheme and gave a theoretical and an experimental background for our novel form of data fusion. This algorithm fuses range and visual data to provide a one-dimensional estimate of motion. In our application, a single dimensional estimate was sufficient since all objects were constrained to move on a ground plane. The advantage of this approach is its enhanced ability to locate the edges of apparently moving objects, which leads to more accurate motion estimates overall.

REFERENCE

[1] R. C. Arkin, "Towards the unification of navigational planning and reactive control," *Working Notes of the AAAI Spring Symposium on Robot Navigation*, Stanford University, pp 1-5, March 28-30, 1989.

[2] M. J. Black, *Robust Incremental Optical Flow*, Ph. D. Thesis, Yale, 1992.

[3] A. Blake, A. Zisserman, *Visual Reconstruction*, MIT Press, 1987. ISBN 0-262-02271-0.

[4] Horn and Schunk, *Determining Optical Flow*, MIT Artificial Intelligence Lab, AI Memo 572, April 1980.

[5] C. Laugier, T. Fraichard, *Decisional Architectures for Motion Autonomy*, Chapter 11, Intelligent Vehicle Technologies: Theory and Applications, L. Vlacic, M. Parent, F. Harashima (eds), Butterworth Heinemann, 2001.

[6] A. Mitiche, *Computational Analysis of Visual Motion*, New York, Plenum Press, ISBN: 0-306-44786-X, 1994.

[7] H. Spies, B. Jahne, J. L. Barron, "Dense range flow from depth and intensity data," *International*

Conference on Pattern Recognition, pp. 131-134, 2000.
[8] http://www.signal-lsp.com/.
.
[9] http://www.analog.com/technology/dsp/Sharc/ index.html
[10] http://pages.infinit.net/gollum/polyray/poly1. Html.

Julian Kolodko completed both his B. Eng. in Microelectronic Engineering and B. InfTech with first class honours at Griffith University, Australia. He is currently comple-ting his Ph.D. work designing and implementing a sensor specifically designed for detecting and estimating motion for autonomous navigation purposes. His research interests range from Digital Signal Processing to autonomous vehicle technologies and the principals of multi-robot cooperation.

Ljubo Vlacic received the Grad diploma in Engineering, MPhil and Ph.D. degrees in electrical engineering (control) from the University of Sarajevo in 1973, 1976 and 1986 respectively.

Currently, he is an Associate Professor and Foundation Director of the Intelligent Control Systems Laboratory, Griffith University, Brisbane Australia. His research interest and contributions span the areas of control systems, decision theory, intelligent control, artificial intelligence and computer & systems engineering; and the application of these methodologies to intelligent vehicles and transport systems, mechatronics, intelligent robotics, industrial automation and knowledge management.

He is a fellow of the Institution of Electrical Engineers (FIEE), a fellow of the Institution of Engineers Australia (FIEAust) and a Senior member IEEE (SMIEEE).