Recognition and Location by Parallel Pose Clustering

W.J. Austin and A.M. Wallace
Dept. of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh EH14 4AS

Abstract

This paper describes the recognition and location of 3D object models from depth-based data using a parallel pose clustering algorithm. We describe a leader-based partitional clustering algorithm and demonstrate a successful parallel implementation of this. Results are presented for a variety of synthetic and real scene data. We also consider how the basic approach can be extended to recognise objects from a single 2D intensity image by perspective inversion. Our eventual aim is to combine such a dual data approach within a single parallel system.

1 Introduction

Object recognition and location can be achieved by a variety of algorithmic approaches [1], including generalised Hough transformation [2] or pose clustering [3] which have been suggested as suitable candidates for parallel implementation. In these approaches, a candidate pose can be computed from any minimal subset of matched model and scene features, for example surfaces or space curves in a depth image are compared with a model surface representation. If all such subsets are considered, these can be stored in a transformational or pose space. Clusters or accumulations of points in this space represent probable pose and object identification.

For parallel implementation, Hough transformation has an obvious appeal [4]. Since the image and parameter space are pre-defined and rigid it is possible to employ data parallelism either over image points or parameter bins, though points that form a cluster in parameter space will in general be spread out in image space, and *vice versa*. This can cause problems in parallel implementations: if the parameter space is split among the processors, matched features are broadcast to all processors and duplication of effort occurs; if the feature space is distributed, subsequent redistribution of the parameter space for peak detection may outweigh the parallel advantage [5].

However, there are more fundamental problems associated with the use of generalised Hough transformation for pose determination in 3D space. Dependent on the representation of the pose matrix, there may be at least five or six translational and rotational parameters. Where the dimensionality of the parameter space is

BMVC 1995 doi:10.5244/C.9.66

high, storage restrictions limit the resolution that can be achieved. Fixed parameter space quantisation can lead to difficulties if an interesting feature straddles a cell boundary. Such problems result in diffuse peaks that can be difficult to detect and may fall below the background noise level. In general, the parameter space is sparsely filled and various authors have proposed hierarchical algorithms that exploit this by concentrating only on 'significant' regions of the parameter space, e.g. [6]. Another way to reduce the space complexity of the HT is to restrict the number of parameters that are initially accumulated.

Pose clustering avoids problems related to the parameter space, but may be less suitable for parallel implementation, at least at first sight. In this alternative, the translational space is not defined explicitly as a quantised accumulator array. Instead the exact values of candidate poses are retained and grouped together according to some distance measure. This avoids quantisation problems and only requires sufficient storage for the pose estimates and their associated cluster labels.

Data clustering algorithms can be split into two main classes, hierarchical and partitional [7]. In the former, clusters are represented in a hierarchical data structure and we move from single atomic clusters towards a single data cluster by merging clusters (or in the opposite direction by splitting) with the aim of detecting the substructure that best represents the data. In the latter a single partition of the data is sought and the number of clusters, K, is fixed in advance. Clusters may be formed by minimising a *global* measure such as square error or a *local* measure such as proximity. A common approach is to form K initial clusters and iteratively refine these by moving data vectors between clusters such that the global error or cluster radius is reduced. The effectiveness of this approach is clearly influenced by the size of K. If K does not match the data, the clusters that are derived may be meaningless.

In this paper, we show how a parallel pose clustering algorithm can be implemented successfully and demonstrate the efficiency of the algorithm on both synthetic and real depth data. However, we have an additional motivation for the use of pose clustering, that we wish to employ the *same* approach to determine pose from a perspective intensity image as well. This will not be a separate process; scene features will be extracted from both intensity and depth data concurrently and used in an optimal fashion to locate and identify the object.

2 A Pose Clustering Approach

Our pose clustering algorithm examines point-vector (pv) pairs that represent the orientation and location of surfaces in the model and scene. Four surface types can be represented in the model: planes, cylinders, cones and spheres. Since an orientation vector cannot be reliably extracted for spherical surfaces, only the first three are used in the clustering process.

The parallel clustering algorithm is based on a partitional approach that uses *local* proximity measures. While in clustering it is common to use a single proximity measure, it is often difficult to produce a proximity measure that is 'fair' to all components of the data vector. The pose clustering problem can be split naturally into rotational and translational components giving rise to two proximity measures that can be varied simultaneously in the clustering process. There are 5 basic steps

in the clustering process: pose estimation, initial cluster determination, cluster centre determination, iterative cluster refinement, and hypothesis extraction.

2.1 Pose Estimation

Pose estimates are determined by considering pairs of matched model and scene pv pairs. Given a pair of model orientation vectors, (v_1, v_2) and a matched pair of scene orientation vectors, (s_1, s_2), the axis and angle of rotation (A_R and θ_R) are determined by the following equations [8]

$$A_R = (v_1 - s_1) \otimes (v_2 - s_2) \tag{1}$$

and

$$\theta_R = \arccos \frac{(A_R \otimes v_1) \cdot (A_R \otimes s_1)}{|A_R \otimes v_1||A_R \otimes s_1|}. \tag{2}$$

Once the rotational component of the pose is known the translational component, T, is determined using

$$p_s = p_m R + T \tag{3}$$

where p_s is a scene location point, p_m is the matched model location point and R is the rotation matrix formed using A_R and θ_R.

To generate match hypotheses, the pose estimates determined by comparison of pairs of matched model and scene pv pairs are clustered. To consider all possible combinations of model and scene pairs is computationally expensive so an 'inverted index' into the model base is used to reduce the number of plausible matches. Given the angle between the scene orientation vectors and the distance between the location points, the indexing function returns a list of model pv pairs that are compatible with each scene pv pair. The indexing function uses angle, distance and type constraints. Each pair of 'matched' model and scene pv pairs can result in up to 4 different solutions of equations 1 to 3 due to uncertainty in the order of the vectors in the equations.

2.2 Initial Cluster Determination

The initial clusters are determined using a 'leader' clustering process [9]. Normally, K data vectors are selected as cluster leaders and the remaining data are assigned to the closest leader. This basic process has been modified to allow variable K. Only one leader is allocated initially and new leaders are created each time the current pose estimate is found to be 'incompatible' with an existing vector. Compatibility is determined by comparing the rotational and translational components of the current pose estimate with the existing leaders until agreement is found or a new leader is created.

2.3 Centre Determination and Axis Resolution

Once the initial clusters are known, the centre of the clusters (i.e. the translation and rotation that best fit the clustered data) can be determined by least-squares techniques. Kanatani [8] shows how a correlation matrix, M_k, can be used to find

the rotation that best fits a set of matched model and scene vectors. This matrix is represented by the equation

$$M_k = \sum_{i=1}^{nv} (v_{mi} v_{si}^T) \tag{4}$$

where nv is the number matched vectors associated with a cluster and v_{mi} and v_{si} are matched model and scene vectors respectively. Kanatani shows that singular value decomposition (SVD) of M_k can be used to determine the rotation matrix and therefore the axis and angle of rotation. Having determined a least squares solution for rotation, the uncertainty that exists in equation 1 can be resolved and approximately half of the initial clusters can be deleted before refinement begins.

2.4 Iterative Cluster Refinement

During cluster refinement each pose estimate is compared with all cluster centres; this is carried out on the translational component, rotational compatibility within the cluster is maintained. When a pose estimate is found to be closer to a different cluster center, this is noted and, once all comparisons have taken place, inter-cluster moves are implemented and the affected cluster centres are recalculated. This continues until no further moves are necessary or the maximum number of iterations is exceeded.

2.5 Hypothesis Extraction

The final stage of the clustering process extracts a list of unique matched model and scene features that are associated with each cluster. Model surfaces that match more than one scene patch in any given cluster can be detected at this point. The resulting hypothesis can then be passed to a verification process for further examination if required.

3 Parallelism in Pose Clustering

Applying each of the stages described in sections 2.1 to 2.5 in turn results in a serial clustering algorithm. However, parallelism can be identified at several different levels of this process. Pose estimation is independent for each matched model and scene vector pair. Similarly, the SVD-based calculation of rotation for each of the initial clusters can be carried out in parallel. During refinement,pose estimates can be compared with the cluster centres in parallel and, if there are sufficient changes, the re-calculation of centres can also be performed in parallel.

The algorithm is implemented in occam 2 on a Meiko computing surface. To achieve good parallel performance in this environment it is necessary to reduce data movement as much as possible. This is achieved by using only minimal coordination data at a central farmer process. Tasks are farmed out to the worker processes but only minimal results are returned, the bulk of the data is retained in the worker network: only the correlation matrix contributions and leader/centre values are passed over the communications links during clustering. Figures 1 and

2 illustrate the main tasks carried out by the farmer and worker processors during clustering, communication steps are marked **Out** or **In** to indicate transmission or receipt of data respectively.

Each worker receives a proportion of the 'matched' model and scene data and generates pose estimates from these. Leader clustering is performed locally when the pose estimates are generated and the local contribution to each cluster's correlation matrix is accumulated. When the local leaders are complete the leader pose and correlation matrix for each significant cluster is returned to the farmer process where individual clusters are merged if their leader poses agree. 'Significance' is determined by a simple threshold on cluster size, experiments have shown that the combination of variable K and rotational and translational tolerances result in many 1 or 2 element clusters that can safely be ignored. The size threshold reduces the communications overhead at the end of the leader clustering phase.

The centres of the merged clusters are determined by farming out the SVD-based calculation to the workers. Once the centres are known and axis resolution has been performed the centre information of the valid clusters is broadcast to the workers and refinement begins.

Parallelism in the refinement stage is similar to the iterative cluster refinement process described by Barlos and Bourbakis in [10]. During each iteration the workers compare the valid pose estimates that are stored locally with the cluster centres. When a pose estimate is moved between clusters this is reported to the farmer where recalculation of centres at the end of each iteration is coordinated. Work is approximately balanced as each worker receives the same number of model and scene pairs and should retain roughly the same number of valid estimates as any other worker. Fine tuning of the load balance would require the movement of pose estimates between processors and the cost of this is likely to outweigh the possible benefits. Currently recalculation of the centre values is carried out at the farmer process.

When refinement is complete each worker determines its own contribution to the clusters and transmits this to the farmer where the data is combined and presented to the user.

4 Experimental Results

We are interested in using the parallel pose clustering process in a multi-level, multi-data source vision system with high, middle and low level vision processes and processing routes for depth, intensity and fused depth/intensity data. To do this we are interested in how modification of the control parameters (size threshold, rotational tolerance (`Atol`), translational tolerance (`TranTol`) and the index tolerances) affect the performance of the system.

Here we consider the effect of the number of worker processes and the size of `TranTol` (`Atol` and the index function tolerances were set to values reflecting the accuracy of the input data, the cluster size threshold was set to 5). `TranTol` was varied between 5mm and 50mm and 5 different worker grids were employed. For each data set, three separate tests were carried out. In the first, noise free scene data was generated directly from the model. In the second and third, random perturbations were added to the scene data; in both cases the orientation vectors

```
Generate compatible model and scene feature pairs
Transmit subset of pairs to each worker Out 1
Receive local leader data from workers & merge In 2
Transmit k-matrix tasks to workers (for SVD) Out 3
Gather centres and resolve axis In 4
Transmit centres to workers (* Init refine *) Out 5
nmoves = 1, iter = 0
WHILE (iter < MaxIter) AND (nmoves > 0) DO
  Gather move lists from workers In 6
  Modify k-matrices and calculate new centres
  Transmit changes to worker processes Out 7
  iter = iter + 1
END WHILE
Gather and merge match lists from workers. In 8
```

Figure 1: Farmer processing during parallel pose clustering.

```
Receive compatible model-scene feature pairs In 1
FOR each matched pair DO
  Estimate pose
  Add pose to leader clusters and accumulate K-matrix
END FOR
Transmit leaders and k-matrices to farmer Out 2
Receive k-matrices from farmer and perform SVD  In 3
Transmit centres to farmer Out 4
Receive complete centre list from farmer In 5
refining = TRUE
WHILE refining DO
  Determine inter-cluster moves
  Transmit move list to farmer Out 6
  Receive centre updated from farmer In 7
  IF (new centres = 0) THEN
    refining = FALSE
  END IF
END WHILE
Gather match info and send to farmer. Out 8
```

Figure 2: Worker processing during parallel pose clustering.

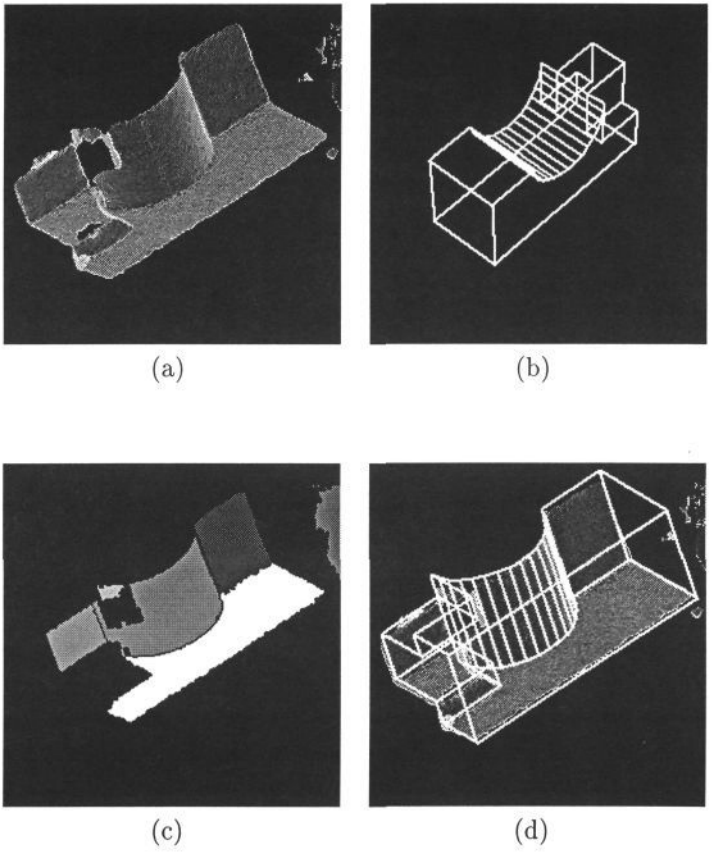

Figure 3: An example of pose clustering applied to real scene data (a) to match model (b) using segmentation (c). The resulting pose estimate is shown in (d).

were perturbed within a 2 degree cone of the true value, the location points were perturbed by random vectors of maximum size 2.5mm and 5mm respectively.

Four sets of simulated input data were considered, in the first the model and scene data were derived from the same CAD object model. The scene data was generated by transforming the object model and the same number of vectors appeared in the model and scene. In the second data set the same object model was used but the transformed scene data was restricted to 12 pv pairs. For the third test data set 20 model point-vector pairs were randomly generated and the scene data derived by transforming this model. The final data set also contained 20 randomly generated model pv pairs but the orientation vectors were aligned along 4 main axes to simulate orientation vectors in a typical object model. An example of clustering applied to real data is given in figure 3.

The experiments showed that varying `TranTol` within the range 5mm to 50 mm had little effect on speedup. Figure 4 shows the speedup characteristics that were obtained with a translational tolerance of 5mm for test data 1 and 4 respectively (speedup is measured relative to a serial version of the program that runs on a single transputer). For the model-derived scene data, the effects of noise are

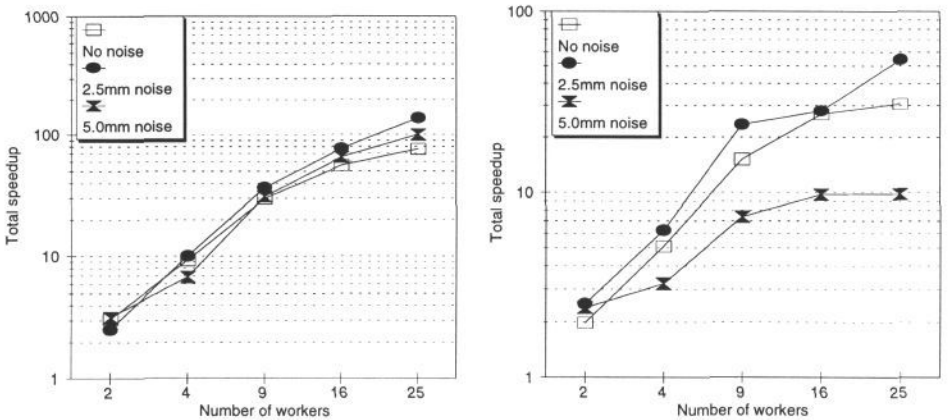

Figure 4: Speedup for test data 1 (a) and 4 (b) with `TranTol` = 5mm

minimal. In test data 4 the addition of 5mm noise vectors to the location vector results in a fall in speedup. This occurs because the translational tolerance is close to the level of the added noise and inter-cluster movement during refinement is possible. Increasing the translational tolerance to 10mm results in fewer initial clusters and no similar fall in speedup is observed.

For test data 1, the best cluster contained all of the expected model-scene matches. When the translational tolerance exceeded the minimum inter-surface separation in the model, spurious matches were also introduced. For the test data 2 (not illustrated), only 9 of the 12 expected surfaces appeared in the best cluster indicating that the size threshold should be reduced to avoid loss of data in the leader clustering process. If the threshold is too high, relevant local clusters can be missed. This is especially true as the number of worker processes increases (and the number of pose estimates held by each worker decreases).

For randomly generated data set 3 (not illustrated), a single best cluster containing all of the expected matches was generated in the absence of noise. When perturbation is introduced, the main cluster can become split as the number of workers increases. This occurs when the selected leaders are close to the edge of the cluster and could prove problematic if the splitting results in clusters that fall below the threshold size (though this was not observed in the test data). Examination of the split clusters show that a single cluster can be recovered by merging, either following initial centre determination or once refinement is complete. Similar splitting of the main peak as the number of clusters increased was observed for test data 4. The introduction of 4 principal orientation directions in this data also led to a higher number of clusters being generated. This can be attributed to the increased probability that a scene pv pair will be compatible with more than one model pair that occurs when the orientation vectors are aligned. These additional clusters were all close to the threshold and can be distinguished from the main cluster.

4.1 Super-linear Speedup

The speedup characteristics in figure 4 show that super-linear speedup, i.e. speedup in excess of the number of worker processes, has been achieved in all but one of the cases. This occurs because the parallel version reduces the *total* amount of computation that is required in the centre determination and refinement phases. In the parallel version the cluster size threshold is applied at the worker processes rather than on the complete pose space, since each worker examines fewer pose estimates than the serial process this results in greater 'pruning' of the pose space. This means that fewer least squares calculations are required during centre determination. During refinement there are fewer clusters and the number of retained pose estimates is smaller than in the serial version so the number of comparisons is reduced. This reduction in total computational effort enhances the speedup characteristics of the parallel algorithm and leads to the observed super-linear speedup.

The size threshold in the parallel program could be reduced in proportion to the number of workers to increase the total amount of calculation carried out by the parallel program. While this would allow a more direct comparison with the serial program, the experiments show that the results of the parallel program are not significantly worse than those of the serial program and that the parallel advantage obtained by keeping the same threshold in both cases should be retained.

5 Further Work

As mentioned earlier, the parallel clustering process will be employed in a multi-level, multi-data system. Before a control process can be developed for this system, it will be necessary to perform further tests to determine the effect of each of the control parameters (index tolerances on distance and orientation, threshold size, and rotational and translational tolerances on leader clustering) on overall algorithm performance.

The clustering process can be used to cluster and match intensity-based data by changing the pose estimation process to deal with perspective inversion while the remainder of the clustering process remains largely unchanged. Parallel algorithms for 3D model-based matching from perspective images exist already [11]. The same direct solution can be used to solve 3-point and 4-point perspective problems allowing a number of difference scene primitives to be handled, for example trihedral junctions and line-arc combinations by 3-point perspective inversion and line-line combinations by 4-point perspective inversion.

While it is possible to envisage clustering depth and intensity pose estimates in the same process, the main difficulty in matching intensity-based features is the lack of constraining information that results from the perspective imaging process. This is expected to result in higher data volumes during intensity-based clustering and it may be impractical to cluster both sources of pose data in the same process. Instead depth-based processing may be used to provide constraining information to the intensity-based process.

6 Summary

We have described a pose clustering algorithm that can be used to recognise and locate 3D object models from depth data. A parallel version of this algorithm has been implemented and shown to exhibit super-linear speedup in some cases. By modifying the pose estimation process we expect to be able to recognise 3D objects from a single perspective image, either using the same clustering process as the depth data or in a separate clustering process that cooperates with a depth-based clustering process.

7 Acknowledgments

This work was supported by ESPRC grant number GR/J07884.

References

[1] A.M. Wallace. A comparison of approaches to high level image interpretation. *Pattern Recognition*, 21(3):241–249, 1988.

[2] D.H. Ballard. Generalising the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(1):111–122, 1981.

[3] G. Stockman. Object recognition and localisation via pose clustering. *Computer Vision, Graphics and Image Processing*, 40:361–387, 1987.

[4] V.F. Leavers. Which Hough Transform? *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(2):250–264, 1993.

[5] W.J. Austin, A.M. Wallace, and V. Fraitot. Parallel Algorithms for Plane Detection using an Adaptive Hough Transform. *Image and Vision Computing*, 9:371–384, 1991.

[6] H. Li, M.A. Lavin, and R.J. Le Master. Fast Hough Transform: A Hierarchical Approach. *Computer Vision, Graphics and Image Processing*, 36:139–161, 1986.

[7] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[8] K. Kanatani. *Geometric Computation for Machine Vision.* Oxford Science Publications, 1993.

[9] J.A. Hartigan. *Clustering Algorithms*, chapter 3. Wiley, 1975.

[10] F. Barlos and N. Bourbakis. A Parallel Image-clustering Algorithm on the 'HERMES' Multiprocessor structure. *Engineering Applications of Artificial Intelligence*, 5(4):299–307, 1992.

[11] A.M Wallace, G.J. Michaelson, P. McAndrew, K.G. Waugh, and W.J. Austin. Dynamic control and prototyping of parallel algorithms for intermediate and high level parallel vision algorithms. *IEEE Computer*, 25(2):43–53, 1992.