Spectral Methods for View-Based 3-D Object Recognition using Silhouettes

Diego Macrini[1], Ali Shokoufandeh[2], Sven Dickinson[3], Kaleem Siddiqi[4], and Steven Zucker[5]

[1] Department of Computer Science
University of Toronto
[2] Department of Mathematics and Computer Science
Drexel University
[3] Department of Computer Science
University of Toronto
[4] Centre for Intelligent Machines
School of Computer Science
McGill University
[5] Center for Computational Vision and Control
Yale University

Abstract. The shock graph is an emerging shape representation for object recognition, in which a 2-D silhouette is decomposed into a set of qualitative parts, captured in a directed acyclic graph. Although a number of approaches have been proposed for shock graph matching, these approaches do not address the equally important indexing problem. We extend our previous work in both shock graph matching and hierarchical structure indexing to propose the first unified framework for view-based 3-D object recognition using shock graphs. The heart of the framework is an improved spectral characterization of shock graph structure that not only drives a powerful indexing mechanism (to retrieve similar candidates from a large database), but also drives a matching algorithm that can accommodate noise and occlusion. We describe the components of our system and evaluate its performance using both unoccluded and occluded queries. The large set of recognition trials (over 25,000) from a large database (over 1400 views) represents one of the most ambitious shock graph-based recognition experiments conducted to date. This paper represents an expanded version of [12].

1 Introduction

There are two approaches to 3-D object recognition. One assumes a 3-D object-centered model, and attempts to match 2-D image features to viewpoint-invariant 3-D model features, e.g., [2, 11, 7]. Over the last decade, this approach has given way to a viewer-centered approach, where the 3-D model is replaced by a collection of 2-D views. These views can be represented in terms of segmented features, such as lines or regions, e.g., [22], or in terms of the photometric "appearance" of the object, e.g., [21, 13]. Although these latter, appearance-based recognition schemes have met with great success, it must be understood that they address the task of exemplar-based recognition. When faced with novel exemplars belonging to known classes, they simply do not scale up.

To achieve such categorical, or generic, object recognition requires a representation that is invariant to within-class shape deformations. One such powerful representation is offered by the shock graph [20], which represents the silhouette of an object in terms of a set of qualitatively defined parts, organized in a hierarchical, directed acyclic graph. Figure 1 illustrates an example of a two-dimensional shape, its shocks (singularities), and the resulting shock graph. In previous work, we introduced the first algorithm for matching two shock graphs, and showed that it could be used to recognize novel exemplars from known classes [19]. Since then, other approaches to shock graph matching have emerged, including [14] and [15]. However, earlier approaches, including our own, have not been extensively tested on noisy graphs, occluded scenes, or cluttered scenes.

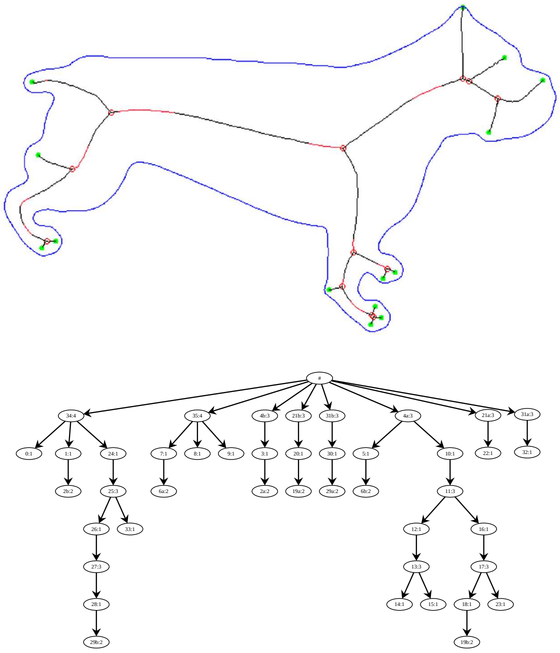


Fig. 1. A Two-Dimensional Shape and its Corresponding Shock Graph. The nodes represent groups of singularities (shocks) along with their geometric attributes. The edges are between adjacent shock groups and in a direction opposite to Blum's grassfire flow [19].

A shock graph representation of shape suggests the use of graph matching techniques for shape recognition. However, matching (graph or otherwise) is only half the problem. Without an effective *indexing* mechanism with which to narrow a large database down to a small number of candidates, recognition degenerates to matching a query to each model in the database. In the case of view-based object recognition, in which a large number of objects map to an even larger number of views, such a linear search is intractable. Unfortunately, very few researchers, in either the computer vision or graph algorithms communities, have addressed the important problem of graph indexing. How, then, can we exploit the power of the shock graph to perform view-based object recognition?

In recent work, we introduced a novel indexing method which maps the structure of a directed acyclic graph to a point in low-dimensional space [18]. This same mapping, in fact, was used as the basis for our shock graph matching algorithm [19]. Using standard, nearest-neighbor search methods, this compact, structural signature was used to retrieve structurally similar candidates from a database. The highest scoring candidates, in turn, were compared to the query using our matching algorithm, with the "closest" candidate used to "recognize" the object. Our experiments showed that the target ranked highly among the candidates, even in the presence of noise and occlusion.

Armed with a unified approach to the indexing and matching of graphs, we now turn to the problem of view-based object recognition using shock graphs. In fact, we are not the first to apply shock graphs to this problem. In recent work, Cyr and Kimia [3, 15] explore the important problem of how to partition the view sphere of a 3-D object using a collection of shock graphs. However, they do not address the shock graph indexing problem, resorting to a linear search of all views in the database in order to recognize an object. Even for small object databases, the number of views required per object renders this approach intractable. In this paper, we unify our shock graph indexing and matching techniques to yield a novel, effective method for view-based 3-D object recognition.

2 A Compact Encoding of Graph Structure

In [19], we introduced a transformation mapping the structure of a directed acyclic graph to a point in low-dimensional space. As mentioned earlier, this mapping was the heart of an algorithm for matching two *shock trees*, derivable from shock graphs in linear time. This same transformation later gave rise to an indexing mechanism, which used the low-dimensional, structural signature of a shock tree to select structurally similar candidates from a database of shock trees [18]. In this latter paper, we analyzed the stability of a tree's signature to certain restricted classes of perturbations.

In a recent paper on matching multi-scale image decompositions, we have strengthened this encoding from undirected, unique rooted trees to directed acyclic graphs, yielding a more powerful characterization of graph structure [17]. This new formulation has led to a broader stability analysis that accommodates *any* graph perturbation in terms of node addition and/or deletion. Furthermore, we extend our matching algorithm to deal with directed acyclic graphs rather than undirected, unique rooted trees. Due to space constraints, we will summarize our new encoding of graph structure; details of the new encoding, as well as an analysis of its stability can be found in [17].

To encode the structure of a DAG, we turn to the domain of eigenspaces of graphs, first noting that any graph can be represented as an antisymmetric $\{0, 1, -1\}$ adjacency matrix, with 1's (-1's) indicating a forward (backward) edge between adjacent nodes in the graph (and 0's on the diagonal). The eigenvalues of a graph's adjacency matrix encode important structural properties of the graph, and are stable under minor perturbations in structure. Our goal, therefore, is to map the eigenvalues of a DAG to a point in some low-dimensional space, providing a stable, compact encoding of structure.

Specifically, let T be a DAG whose maximum branching factor is $\Delta(T)$, and let the subgraphs of its root be $T_1, T_2, \ldots, T_{\delta(T)}$, as shown in Figure 2. For each subgraph, T_i, whose root degree is $\delta(T_i)$, we compute[6] the magnitudes of the eigenvalues of T_i's submatrix, sort them in decreasing order by absolute value, and let S_i be the sum of the $\delta(T_i) - 1$ largest absolute values. The sorted S_i's become the components of a $\Delta(T)$-dimensional vector assigned to the DAG's root. If the number of S_i's is less than $\Delta(T)$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each nonterminal node in the DAG, computed over the subgraph rooted at that node. We call each such vector a *topological signature vector*, or TSV. The details of this transformation, the motivation for each step, and an evaluation of its properties is given in [17].

3 Shock Graph Indexing

Given a query shape, represented by a shock graph, the goal of indexing is to efficiently retrieve, from a large database, similar shock graphs that might account for the query or some portion thereof (in the case of an occluded query or a query representing a cluttered scene). These *candidate* model graphs will then be compared directly with the query, i.e., *verified*, to determine which candidate model best accounts for the query. We therefore seek an effective index for shock graph recognition that possesses a number of important properties, including:

1. low dimensionality
2. captures both local and global structural properties
3. low ambiguity
4. stable to minor perturbations of graph structure
5. efficiently computed

Our topological signature vector, in fact, satisfies these five criteria. Its dimensionality is bounded by the graph's maximum branching factor, *not* the size of the graph (criteria 1); for shock graphs, the branching factor is typically low (< 5). TSV's for nodes high in the graph capture global structure while lower nodes capture local structure (criteria 2). The components of a node's vector are

[6] We use SVD to compute the magnitudes of the eigenvalues.

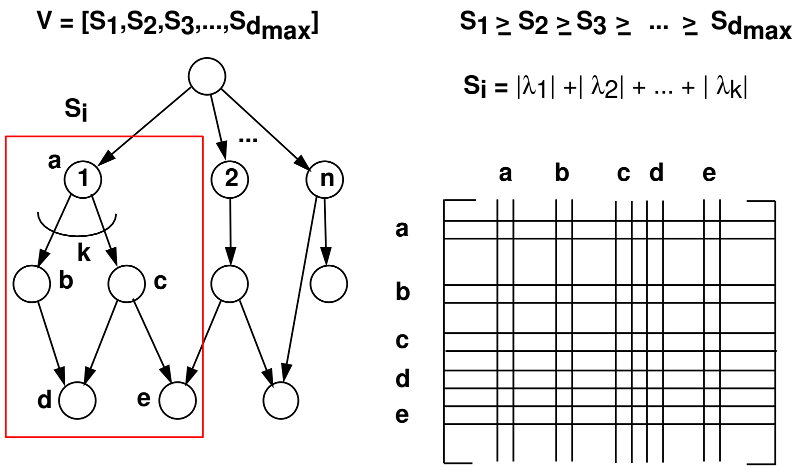


Fig. 2. Forming the Topological Signature Vector (TSV) for the root. For a given DAG rooted at a child (e.g., a) of the root, compute the sum of the magnitudes of the k largest eigenvalues (k is the out-degree of a) of the adjacency submatrix defining the DAG rooted at a. The sorted sums, one per child of the root, define the components of the TSV assigned to the root. The process can be repeated, defining a TSV for each non-leaf node in the DAG. The dimensionality of a shock graph's TSV's is equal to the maximum branching factor of the shock graph and not the size of the graph.

based on summing the largest eigenvalues of its subgraph's adjacency submatrix. Although our dimensionality-reducing summing operation has cost us some uniqueness, our partial sums still have very low ambiguity (criteria 3).[7] From our improved sensitivity analysis, described in [17], we have shown our index to be stable to minor perturbations of the DAG's structure (criterion 4). Moreover, as shown in [19], these sums can be computed even more efficiently (criterion 5) than the eigenvalues themselves. The vector labeling of all DAGs isomorphic to T not only has the same vector labeling, but spans the same subspace in $R^{\Delta(T)-1}$. Moreover, this extends to any DAG which has a subgraph isomorphic to a subgraph of T.

3.1 A Database for Model DAGs

Our spectral characterization of a DAG's structure suggests that a model DAG's structure can be represented as a vector in δ-dimensional space, where δ is an upper bound on the degree of any vertex of any image or model DAG. If we

[7] Moreover, if p is the probability that a query graph and a model graph have different structure but are isospectral, then the probability that the k vectors corresponding to the query graph's nodes are identical to the k vectors corresponding to the model graph's nodes is p^k. This suggests the use of a collection of indexes rather than a single index, as will be discussed later.

could assume that an image DAG represents a properly segmented, unoccluded object, then the TSV computed at the query DAG's root, could be compared with those topological signature vectors representing the roots of the model DAGs. The vector distance between the image DAG's root TSV and a model DAG's root TSV would be inversely proportional to the structural similarity of their respective DAGs, as finding two subgraphs with "close" eigenvalue sums represents an approximation to finding the largest subgraph isomorphism.

Unfortunately, this simple framework cannot support either cluttered scenes or large occlusion, both of which result in the addition or deletion of significant structure. In either case, altering the structure of the DAG will affect the TSV's computed at its nodes. The signatures corresponding to the roots of those subgraphs (DAGs) that survive the occlusion will not change. However, the signature of the root of a subgraph that has undergone any perturbation will change which, in turn, will affect the signatures of any of its ancestor nodes, including the root of the entire DAG. We therefore cannot rely on indexing solely with the root's signature. Instead, we will exploit the local subgraphs that survive the occlusion.

We can accommodate such perturbations through a local indexing framework analogous to that used in a number of geometric hashing methods, e.g., [9, 5]. Rather than storing a model DAG's root signature, we will store the signatures of *each* node in the model DAG, as shown in Figure 3. At each such point (node signature) in the database, we will associate a pointer to the object model containing that node as well as a pointer to the corresponding node in the model DAG (allowing access to node label information). Since a given model subgraph can be shared by other model DAGs, a given signature (or location in δ-dimensional space) will point to a list of (model object, model node) ordered pairs. At runtime, the signature at each node in the query DAG becomes a separate index, with each nearby candidate in the database "voting" for one or more (model object, model node) pairs. Nearby candidates are retrieved using a nearest neighbor retrieval method, described in [8].

3.2 Accumulating Local Evidence

Each node in the query DAG will generate a set of (model object, model node) votes. To collect these votes, we set up an accumulator with one bin per model object, as shown in Figure 4. Furthermore, we can weight the votes that we add to the accumulator according to two important factors. Given a query node and a model node (retrieved from the database),

1. we weight the vote according to the distance between their respective TSV's – the closer the signatures, the more weight the vote gets.
2. we weight the vote according to the complexity of its corresponding subgraph, allowing larger and more complex subgraphs (or "parts") to have higher weight. This can be easily accommodated within our eigenvalue framework, for the richer the structure, the larger its maximum eigenvalue:

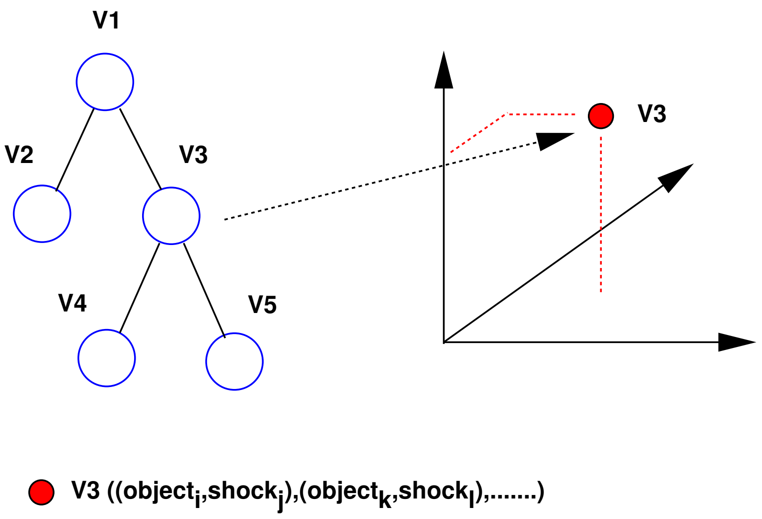


Fig. 3. Populating the Database. For every non-leaf node, n_i, in every model view shock graph, insert into a point database at the location defined by n_i's TSV the label of n_i and the object (model and view) that contains n_i. If multiple nodes collide at the same location, then maintain a list of objects (or "parts") that share the TSV.

Theorem 1 (Lovász and Pelikán [10]) *Among the graphs with n vertices, the star graph ($K_{1,n-1}$), has the largest eigenvalue ($\sqrt{n-1}$), while the path on n nodes (P_n) has the smallest eigenvalue ($2\cos\pi/(n+1)$).*

Since the size of the eigenvalues, and hence their sum, is proportional to both the branching factor (in some sense, richness or information content) as well as the number of nodes, the magnitude of the signature is used to weight the vote.

Before assembling the components of our weighting function, we must address one final issue. Namely, for a query point q and neighboring model point m, we would like to increase the weight of the vote for an object model M if m represents a larger proportion of M. Similarly, we would like to increase the weight of the vote for M if q represents a larger proportion of the query. Equivalently, we favor models that can cover a larger proportion of the image, while at the same time, we favor models which have a larger proportion covered by the query. These two goals are in direct competition, and their relative merit is a function of the task domain. Our weighting function can now be specified as:

$$W = \frac{(1-\omega)||q||}{T_q(1+||m-q||)} + \frac{\omega||m||}{T_m(1+||m-q||)} \quad (1)$$

where q is the TSV of the query DAG node, m the TSV of the model DAG node (that is sufficiently close), T_q and T_m are the sums of the TSV norms of the entire

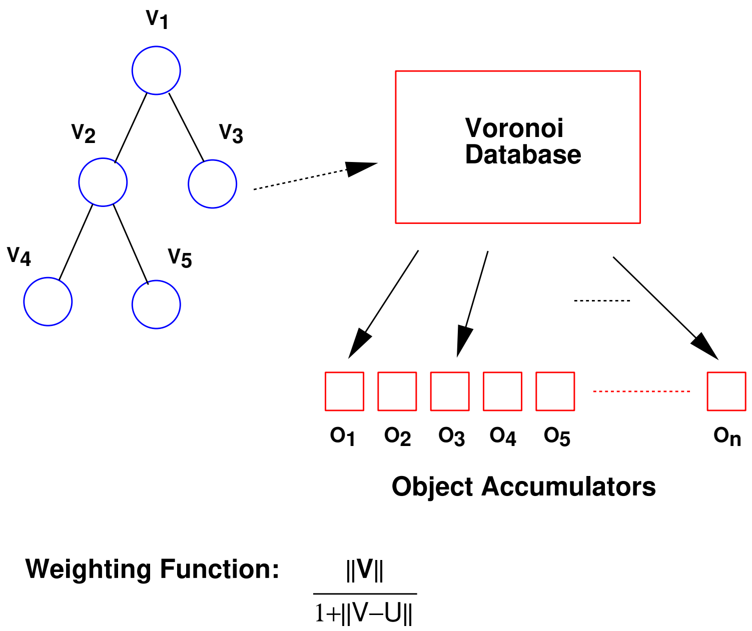


Fig. 4. Accumulating Evidence for Candidate Models. For each non-leaf node in the query DAG, find the nearest neighbors in the database. Each nearest neighbor defines a list of objects which contain that part (DAG). For each object whose nodel label (of the root of the DAG defining the TSV) matches that of the query, accumulate evidence for that model. In general, evidence is weighted proportionally to the size and complexity of the part and inversely proportionally to the distance between the query and neighbor.

query and model DAGs, respectively, and convexity parameter $\omega, 0 \leq \omega \leq 1$ is the weighting affecting the roles of the opposing goals described above. The first term favors models that cover a larger proportion of the image, while the second favors models with more nodes accounted for.

Once the evidence accumulation is complete, those models whose support is sufficiently high are selected as candidates for verification. The bins can, in effect, be organized in a heap, requiring a maximum of $O(\log k)$ operations to maintain the heap when evidence is added, where k is the number of non-zero object accumulators. Once the top-scoring models have been selected, they must be individually verified according to the matching algorithm described in the next section.

4 Shock Graph Matching

Our spectral characterization of graph structure forms the backbone of our indexing mechanism, as described in the previous section. Moreover, this *same* spectral characterization forms the backbone of our matching algorithm, thereby unifying the mechanisms of indexing and matching [16]. In previous work [19], we showed that a shock graph could be transformed into a unique rooted undirected *shock tree* in linear time. We introduced a novel algorithm for computing the distance between two shock trees (including correspondence) in the presence of noise and occlusion. As mentioned earlier, we have strengthened our indexing and matching framework to include directed acyclic graphs.

We will now briefly describe the algorithm; details can be found in [17]. Having already computed the TSV at each node in both the query graph as well as the model graph (at compile time), we can use this information to compute node correspondence. Specifically, we set up a bipartite graph spanning the nodes of the query graph and the nodes of the model graph. The edge weights in the bipartite graph will be a function of both the structural similarity of the directed acyclic subgraphs rooted at these nodes and the similarity of the nodes' contents. We then compute a maximum cardinality, maximum weight matching in the bipartite graph, leading to a selection of edges defining the final node correspondence.

This procedure, unfortunately, will not enforce the hierarchical constraints imposed by a shock graph, allowing inversions in the computed correspondence.[8] Instead, we take a greedy approach and select only the best edge in the bipartite solution to add to the correspondence set. We then recursively continue, computing the matching between the subgraphs rooted at these two nodes and adding its best edge to the solution set. The maximum weight maximum cardinality matching is based on an objective function that measures the quality of the correspondence between matched regions while penalizing for unmatched nodes in the image, the model, or both, depending on the a priori conditions on query generation.

Before stating the algorithm, let us define some of its components. Let $G = (V_1, E_1)$ and $H = (V_2, E_2)$ be the two DAGs to be matched, with $|V_1| = n_1$ and $|V_2| = n_2$. Define d to be the maximum degree of any vertex in G and H, i.e., $d = \max(\delta(G), \delta(H))$. For each vertex v, we define $\chi(v) \in R^{d-1}$ as the unique topological signature vector (TSV), introduced in Section 2.[9] The bipartite edge weighted graph $\mathcal{G}(V_1, V_2, E_{\mathcal{G}})$ is represented as a $n_1 \times n_2$ matrix $\Pi(G, H)$ whose (u, v)-th entry has the value:

$$\phi(u, v) = \mathcal{W}(u, v) \times e^{-(||\chi(u) - \chi(v)||)}, \tag{2}$$

where $\mathcal{W}(u, v)$ denotes the similarity between u and v, assuming that u and v are compatible in terms of their nodes (more on this later), and has the value zero otherwise. Using the scaling algorithm of Goemans, Gabow, and Williamson

[8] An inversion occurs when an ancestor/descendant in one graph is mapped to a descendant/ancestor in the other graph.

[9] Note that if the maximum out-degree of a node is d, then excluding the edge from the node's parent, the maximum number of children is $d-1$. Also note that if $\delta(v) < d$, then then the last $d - \delta(v)$ entries of χ are set to zero to ensure that all χ vectors have the same dimension.

```
procedure isomorphism(G,H)
    Φ(G,H) ← ∅ ;solution set
    d ← max(δ(G), δ(H)) ;TSV degree
    for u ∈ V_G { ;compute TSV at each node and unmark all nodes in G
        compute χ(u) ∈ R^{d-1} (see Section 2)
        unmark u
    }
    for v ∈ V_H { ;compute TSV at each node and unmark all nodes in H
        compute χ(v) ∈ R^{d-1} (see Section 2)
        unmark v
    }
    call match(root(G),root(H))
    return(cost(Φ(G,H))
end

procedure match(u,v)
    do
        {
        let G_u ← rooted unmarked subgraph of G at u
        let H_v ← rooted subgraph of H at v
        compute |V_{G_u}| × |V_{H_v}| weight matrix Π(G_u, H_v)
        M ← max cardinality, max weight bipartite matching
                in G(V_{G_u}, V_{H_v}) with weights from Π(G_u, H_v) (see [6])
        (u', v') ← max weight pair in M
        Φ(G,H) ← Φ(G,H) ∪ {(u', v')}
        call match(u',v')
        mark G'_u
        mark G'_v
        }
    while (G_u ≠ ∅ and H_v ≠ ∅)
```

Fig. 5. Algorithm for Matching Two Hierarchical Structures

[6], we can compute the maximum cardinality, maximum weight matching in $\mathcal{G}$, resulting in a list of node correspondences between G and H, called $\mathcal{M}_1$, that can be ranked in decreasing order of similarity. The precise algorithm, whose complexity is $O(n^3)$, is given in Figure 5; additional details, analysis, and examples are given in [17].

5 Experiments

We have systematically tested our integrated framework using both occluded and unoccluded queries. With over 27,000 trials and a database of over 1400 graphs, this represents one of the most comprehensive set of shock graph experiments to date. Our database consists of views computed from 3-D graphics models obtained from the public domain. Using a graphics modeling tool (3D Studio Max), each model is centered in a uniformly tessellated view sphere, and a silhouette is generated for each vertex in the tessellation. A shock graph is computed for each silhouette [4], and each node of the resulting graph is added to the model database, as described in Section 3. A sampling of the object views is shown in Figure 6.

In the first set of experiments, we evaluate the performance of the system on a set of unoccluded queries to an object view database. The database contains

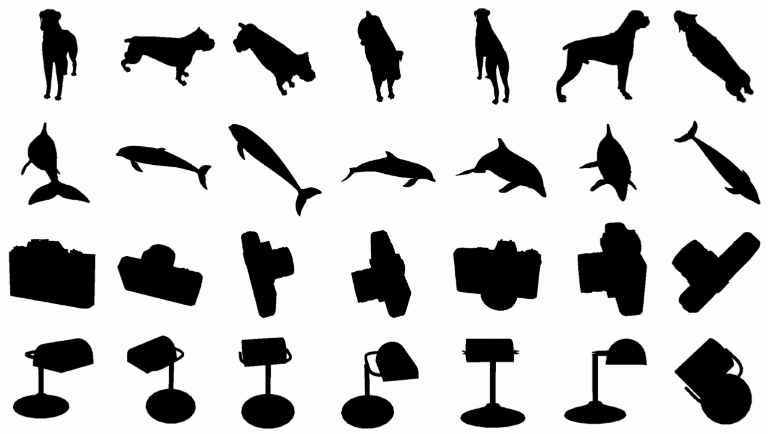


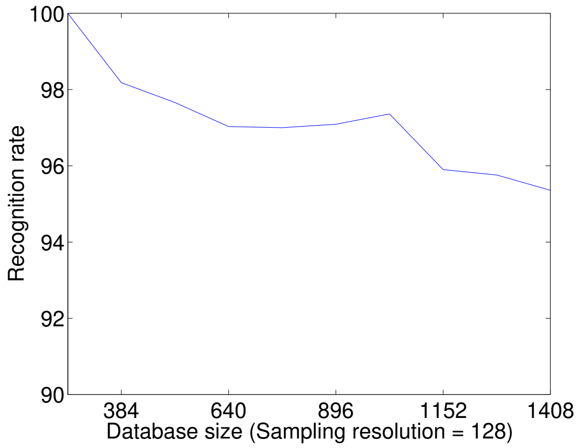
Fig. 6. Some example object views drawn from our database.

1408 views describing 11 objects (128 uniformly sampled views per object). We then remove each view from the database and use it as a query to the remaining views. For each node of the query DAG, the indexing module (see Section 3) will return all neighbors within a radius of 40% of the norm of (query) node. Evidence for models (containing a neighbor) is then accumulated, and the model bins are sorted. The indexer will return at most the highest scoring 50 candidates, but will return fewer if the sorted bins' contents drop suddenly. The candidates are matched to the query, using the matcher (see Section 4), and sorted according to similarity. If the query object (from which the query view was drawn) is the same as the model object from which the most similar candidate view is drawn, recognition is said to be successful, i.e., the object label is correct.[10]

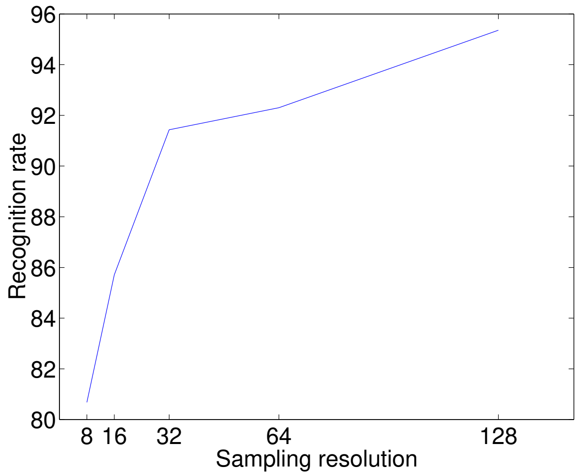
Figure 7(a) plots recognition performance as a function of increasing number of objects (with 128 views per new object), while Figure 7(b) fixes the number of objects (11) and plots recognition performance as a function of sampling resolution. Recognition performance is very high, with better than 90% success until sampling resolution drops below 32 views (over the entire view sphere) per object. This demonstrates both the efficacy of the recognition framework and the viewpoint invariance of the shock graph, respectively. The most complex component of the algorithm is the matcher. However, with a fixed number (50) of verifications per query, independent of database size, complexity therefore varies as a function of nearest neighbor search and bin sorting, both of which are sublinear in the number of database views.

In the final experiment, shown in Figure 7(c), we plot recognition performance as a function of degree of occlusion (for the entire database) for occluded queries. To generate an occluded query, we randomly choose a node in the query DAG and delete the subgraph rooted at that node, provided that the node

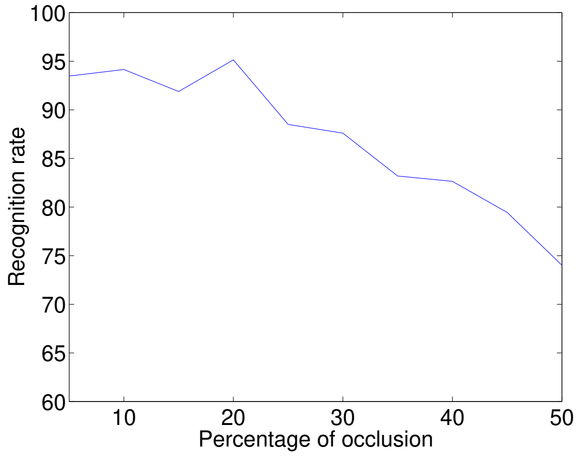
[10] Note that if multiple views (perhaps from different objects) are tied for "most similar", then each can be considered to be "most similar."



(a)



(b)



(c)

Fig. 7. Recognition Performance: (a) Recognition performance as a function of object database size; (b) Recognition performance as a function of sampling resolution; and (c) Recognition performance as a function of degree of occlusion.

"mass" of the graph does not drop by more than 50%. As can be seen from the plot, performance decreases gradually as a function of occluder size (or, more accurately, the amount of "missing data"), reflecting the framework's ability to recognize partially visible objects.

It should be noted that in the above experiments, erroneous matches may be due to either ambiguous views (views shared by different objects) or to queries representing "degenerate" views, in which the removed view acting as a query was the last view of its class and therefore not expected to match other views on the object.

6 Conclusions

We have presented a unified mechanism for shock graph indexing and matching, and have applied it to the problem of view-based 3-D object recognition. Our spectral-based indexing framework quickly and effectively selects a small number of candidates, *including the correct one*, from a large database of model views from which our spectral-based matcher computes an accurate distance measure. Our scaling experiments demonstrate the framework's ability to effectively deal with large numbers of views, while our occlusion experiments establish its robustness. Current work is focused on view-cell clustering and strengthening the indexer to include more geometric and node label information. In particular, it is known that nodes related to ligature are likely to be less stable and hence should be given less weight by both the indexer and the matcher [1].

7 Acknowledgements

The authors would like to acknowledge the programming support of Maxim Trokhimtchouk, Carlos Phillips, and Pavel Dimitrov. The authors would also like to express their thanks to Norio Katayama for the use of their SR-tree implementation. Finally, the authors would like to acknowledge the generous support of NSERC, FCAR, CFI, CITO, and NSF.

References

1. J. August, K. Siddiqi, and S. W. Zucker. Ligature instabilities in the perceptual organization of shape. *Computer Vision and Image Understanding*, 76(3):231–243, 1999.
2. R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.
3. M. Cyr and B. Kimia. 3d object recognition using shape similarity-based aspect graph. In *Proceedings, ICCV*, Vancouver, B.C., 2001.
4. P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head, SC, June 2000.

5. P. Flynn and A. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP:Image Understanding*, 55(2):119–129, March 1992.
6. H. Gabow, M. Goemans, and D. Williamson. An efficient approximate algorithm for survivable network design problems. *Proc. of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pages 57–74, 1993.
7. D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.
8. Norio Katayama and Shin'ichi Satoh. The sr-tree: an index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 369–380. ACM Press, 1997.
9. Y. Lamdan, J. Schwartz, and H. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578–589, October 1990.
10. L. Lovász and J. Pelicán. On the eigenvalues of a tree. *Periodica Math. Hung.*, 3:1082–1096, 1970.
11. D. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Norwell, MA, 1985.
12. D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. View-based 3-D object recognition using shock graphs. In *Proceedings, Internal Conference on Pattern Recognition*, Quebec City, August 2002.
13. H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24, 1995.
14. M. Pelillo, K. Siddiqi, and S. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, November 1999.
15. T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing shock graphs. In *Proceedings, ICCV*, Vancouver, B.C., 2001.
16. A. Shokoufandeh and S. Dickinson. A unified framework for indexing and matching hierarchical shape structures. In *Proceedings, 4th International Workshop on Visual Form*, Capri, Italy, May 28–30 2001.
17. A. Shokoufandeh, S. Dickinson, C. Jonsson, L. Bretzner, and T. Lindeberg. The representation and matching of qualitative shape at multiple scales. In *Proceedings, ECCV*, Copenhagen, May 2002.
18. A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral encoding of topological structure. In *Proceedings, IEEE CVPR*, pages 491–497, Fort Collins, CO, June 1999.
19. K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.
20. Kaleem Siddiqi and Benjamin B. Kimia. A shock grammar for recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 507–513, 1996.
21. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
22. S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1006, October 1991.