Learning Spatial Relationships between Objects

Benjamin Rosman* and Subramanian Ramamoorthy†

March 31, 2011

Abstract

Although a manipulator must interact with objects in terms of their full complexity, it is the qualitative structure of the objects in an environment and the relationships between them which define the composition of that environment, and allow for the construction of efficient plans to enable the completion of various elaborate tasks. This paper presents an algorithm which redescribes a scene in terms of a layered representation, from labeled point clouds of the objects in the scene. The representation includes a qualitative description of the structure of the objects, as well as the symbolic relationships between them. This is achieved by constructing contact point networks of the objects, which are topological representations of how each object is used in that particular scene, and are based on the regions of contact between objects. We demonstrate the performance of the algorithm, by presenting results from the algorithm tested on a database of stereo images. This shows a high percentage of correctly classified relationships, as well as the discovery of interesting topological features. This output provides a layered representation of a scene, giving symbolic meaning to the inter-object relationships useful for subsequent commonsense reasoning and decision making.

1 Introduction

As robots are becoming more capable of performing a wide range of tasks, and make the move from carefully engineered to open and unknown environments, the need for concise representation of a widely varying world is becoming more pertinent. These robots must deal with immense variability in the structure of the world, for example a manipulation robot may find objects in a wide variety of configurations. In order for a robot to be able to manipulate these objects, it needs to understand the qualitative structure of the relationships between them independent of other quantitative variation.

*B. Rosman (`B.S.Rosman@sms.ed.ac.uk`) is with the Institute of Perception, Action and Behaviour, in the School of Informatics, University of Edinburgh, Edinburgh, UK, and with the Mobile Intelligent Autonomous Systems (MIAS) group at the Council for Scientific and Industrial Research (CSIR), South Africa.

†S. Ramamoorthy (`s.ramamoorthy@ed.ac.uk`) is with the Institute of Perception, Action and Behaviour, in the School of Informatics, University of Edinburgh, Edinburgh, UK.

Furthermore, with the move into more natural environments, robots are more likely to encounter objects with which they have had minimal, if any, previous experience. An increase in the number of these poorly known objects in the vicinity of an active robot suggests that new strategies for exploring and interacting with these objects would be required. It is infeasible to enumeratively represent all aspects of some real-world scene if we want an agent to use that information subsequently in real-time decision making. Another situation where enumeration is difficult, is when manipulating articulated and flexible objects: complicated objects yielding seemingly disorganised point clouds, but with much qualitative structure.

As a result, we are particularly interested in the way in which different objects can be represented, in terms of their own structure as well as the way in which they relate to each other spatially in a scene. This would provide a redescription of the scene using a layered representation, including a qualitative level which provides interesting topological features of objects that could be used for disambiguating actions, such as that holes are candidates for inserting a finger to pick up an object, or that points of contact between two objects constrain the relative motion of those objects.

The issue we address in this paper is thus, given a three-dimensional representation of a scene consisting of several objects, using a point cloud as the raw scene description, redescribe the scene in terms of abstractions. We use a layered abstraction, consisting of a skeletonised description of the objects themselves, as well as a *symbolic* description of the spatial relationships between these objects. These relationships are important in manipulation and decision making, allowing for a simplification of task specifications, and allowing for vagueness and robustness in planning.

In this work we are less concerned with detailed object identification, but rather are interested in separating a scene into potential objects that can be manipulated, and examining the way in which these objects are used as structural components of the environment. In this way, a wooden box and an intricate overturned vase could be considered to be topologically equivalent, in that they are both single connected component structures on which other objects can be placed. This view of what constitutes an object is inspired by an emerging notions regarding topological invariance and qualitative structure in a wide variety of data sets, as in work by Carlsson et al. (Carlsson, 2009), and illustrates the level of description we are aiming towards with our representation scheme.

In order for a robot to efficiently manipulate objects in some environment, it needs to know something about how these objects relate to each other, e.g., what object is supported by what other object and constrained by what other objects – a class of qualitative relationships defined by specific features such as contact points. We seek to redescribe the scene in terms of these types of relationships. The most important of these are $\texttt{on}(\cdot,\cdot)$ and $\texttt{adjacent}(\cdot,\cdot)$, and while these are clearly a subset of the wide variety of qualitative relationships possible, we restrict our attention in this way. By learning and identifying these relationships in the surrounding environment, a robot can use these concepts in the context of motion synthesis, particularly for performing tasks which require the use of several objects.

We choose to work with point clouds here as they are gaining prominence as a representation of the world in a number of new mobile manipulation platforms. This is further driven by the move towards unified and generalised operating systems for robotic platforms, such as the open-source ROS[1], which comes complete with a Point Cloud Library (PCL). Point clouds provide a representation for the three-dimensional information that a robotic platform may extract from the world around it, in a number of ways ranging from the registration of stereoscopic cameras, to reconstructions from laser range scanners. As such, building a point cloud is a convenient[2] method for representing raw depth information about the world around a robot.

This paper describes our proposal for a layered description of a scene, derived by a novel algorithm for extracting the structure of objects and then classifying the spatial relationships between the point clouds of these pre-segmented objects. Our method creates a contact point network for each object as an abstraction of the object into a graph-like skeleton which identifies points where the object either touches other objects, or comes close to touching them. This structure is a potentially useful abstraction for manipulation, as it identifies regions where the object has an interface with the external world. These may be useful candidate points for functional components and surfaces of the objects, and the edges between these points indicate constraints on the objects that could be factored into motion synthesis and planning. More concretely, these edges provide seeding regions, around which the search for grasp points can begin. In a sense, these skeletons can be considered as being loosely related to the concept of affordances (Gibson, 1986), to the extent that contact points and intra/inter-object holes indicate constraints and possibilities for manipulation motion synthesis. These constraints thus allow for the disambiguation between families of motion strategies at a global level for a particular environment, and an agent may make use of this knowledge to provide a viable first pass at synthesising behaviours.

This paper is organised as follows: background information on this problem of extracting structure and learning spatial relationships is given in Section 2. Our algorithm to construct these representations from point cloud data is described in Section 3, with experimental results in Section 4. A discussion of these results appears in Section 5, and a summary of the conclusions, contributions and future work is in Section 6.

2 Background

The question of how to learn the spatial relationships between different objects is one which has received relatively little attention in the literature. This is an important question, as it seeks to provide structural information about some environment. This is necessary, as what is often difficult about motion planning is disambiguating between the global structure of candidate plans, and so finding good initial guesses at the appropriate course of action for some new scenario. Many researchers have examined the problem of identifying objects in a scene (Jain and Dorai, 2000; Leibe and Schiele, 2004; Desai et al., 2009; Alexe et al., 2010), but instead of focusing on object identification, we wish to extract coarser types of functional information which may aid a robot manipulating in that environment, by describing the structure and constraints of the scene.

[1]Robot Open-Source – www.ros.org

[2]Low-cost sensors such as Microsoft's Kinect for the Xbox 360

Galleguillos et al. (2008) examine the relative spatial arrangements of patches in two-dimensional images, and use this to disambiguate object labels. These relative spatial arrangements are defined in terms of bounding boxes around patches, as well as their relative centroid locations. This provides an abstraction of a complex scene, in a similar manner to what we desire in determining abstractions which capture the relationships between irregularly-shaped point clouds corresponding to regions of objects in a three-dimensional environment. This work is thus interesting, but we seek a version for extracting manipulation specific information from a scene. A similar approach, based on bounding boxes, is successfully used by Dee et al. (2009) to learn relationships between parts of frames in videos which correspond to regions experiencing similar motion. The relative spatial relationships between regions of an image are often used to improve interpretation and recognition in the scene (Galleguillos and Belongie, 2010), working on the assumption that certain objects are more likely to be found in the context of, or in particular positions relative to, certain other objects.

An intermediate step between logical plans and low-level control is the reasoning about large-scale structure of motion plans such as in Hauser and Latombe (2010). These approaches require the ability to appropriately seed, at the level of topology, the search process through strategy space.

Relational predicates are also an important component of first-order logic, which allows for reasoning about entities and the relationships between them. As a result, first-order logic is a language of choice for many planning problem descriptions and algorithms (Ghallab et al., 2004), as the goal of planning is the discovery of plans of action which can cause certain relationships between various objects in the domain of the problem to become true, remain true, or cease to be true.

For instance, the relationships between different regions in space can be defined and reasoned about using an interval logic known as region connection calculus (RCC), similarly to the way in which temporal logics reason about time and events (Randell et al., 1992). This provides an axiomatised description of the ways in which two regions can relate, such as 'overlaps', 'contains' and 'equals'. This theory provides no mechanisms for extracting these relationships from visual data, but provides a language for reasoning about them at a higher level.

Spatial relationships between objects in the physical (or a simulated) world are thus useful elements for agents in reasoning about planning tasks in worlds with multiple objects, for example stacking blocks to build towers (Pasula et al., 2007). The focus of such work is typically on methods for making inferences about the relationships that hold in the world, rather than deducing these relationships themselves. As a result, a hard-coded approach is often taken, where rules for recognising relationships can be designed by the system developers. An example of the usefulness of the relationships of parts of a domain in solving a problem can be seen in the robotic towel folding of Maitin-Shepard et al. (2010). By finding the key structural features of towels, the corners, as well as the relationships between them (hand-coded by the designers), a robot was able to successfully fold a towel. We are interested in extracting this type of structure and relationship in more general scenes.

A more general example of how such rules could be hard-coded would be: "if there are points in object A above points in object B, with a vertical displacement below some ϵ, then A is on B". These rules are specific to each particular problem, and rely on the fact that these systems often use simple geometric objects such as blocks. As a result, these rules easily break down for the kinds of general objects which one would expect a personal robot operating in a home environment to encounter. There is thus a need for more flexible methods for detecting relationships between objects.

Relationships between pairs of tracked vehicles moving on a road have been considered previously by Galata et al. (2002). The relationships captured between vehicles are similar to the types of relationships we seek, but we focus somewhat more on achieving a robotic manipulation centric representation. Additionally a crucial component of our layered representation is that we abstract some notion of the internal structure of the objects which may not be relevant in the application considered in Galata et al. (2002).

One notable approach to addressing the related problem, of relationships between regions and elements of a scene, is that of Waltz (1975). This work examines the broad problem of reconstructing a three-dimensional description of a scene, given a line drawing of the scene. A scene in this case is restricted to a set of planar-faced objects, described by the edges of each plane, and shaded areas to indicate shadows. The crux of this method is in labeling the line segments which are object edges. The edge labels indicate whether the edge describes features such as the edge of a shadow, a concave bend or a convex bend. The labels are assigned based on the types of junctions at either end of the line segments, as well as local information such as shadows and region orientation. Knowledge of the types of edges between two different objects in a scene, allows the system to deduce when one object is in front of, behind or supporting another. This work was limited in the strong requirements placed on the input data: that it be line drawings of planar-faced objects. We instead seek an approach which can handle point cloud data, of any objects.

Planar faces can be easily described by line drawings, but more complicated objects require a richer representation. The idea of representing an object as a graph, being some skeletal abstraction of the object, has found interesting uses in describing the topology and other symbolic properties of individual objects (Pinz et al., 2008). This is a sparse representation of what may be otherwise complicated objects. An example of this by Katz and Brock (2008) provides a method for discovering articulation points in objects, based on which parts of the object maintain a constant separation while a robot is moving that object. In this way, nodes of the graph correspond to distinctive features of the object, and edges exist between two features if the distance between those features has never exceeded some threshold.

We propose to build up from low-level point cloud data as acquired by sensing devices, into a layered representation for redescribing the scene to aid in reasoning and planning through the use of graph-like abstractions of objects. Instead of basing these graphs on features of an individual object, they arise from the way in which the object structurally forms part of a scene or environment. Much previous work has been dedicating to detecting affordances of objects of various sorts, such as Saxena et al. (2008) showing how good grasp points can be identified on an object, Rusu et al. (2009) describing curvature and other geometric properties, and Barck-Holst et al. (2009) showing how such grasp affordances can be learnt from an ontological reasoning system. Works such as these often focus on statistically finding useful local features. The next level up would be to look at how the scene is composed of these local features. To this end, we consider the relationship patterns between sets of objects, with the aim of learning high-level concepts relating to the topological structure of an entire scene. This provides a platform from which inter-object spatial relationships are inferred, giving a high-level representation which may be useful for motion synthesis by bringing down the search space.

3 Algorithm

3.1 Overview

Our algorithm builds a layered representation of a scene, by extracting spatial relationships which exist between objects, as well as a topological description of those objects, in order to redescribe that scene in terms of its structural properties. This description is based on easily observed features of the scene, such as object candidates and contact points between the objects. The assumptions made by the algorithm are described in Section 3.2. The object candidates used by the algorithm are derived from segmented point cloud data of the scene, and a discussion of a simple method for object segmentation is provided in Section 3.3.

Contact points are another visual feature used for extracting the structural representation of the scene. To define these, the algorithm relies on the concept of geometric separability between two different objects, and in particular identifies regions where the margin of separation between the two objects is small. These regions are most likely candidates for points of contact between the objects. The reason that geometric separability is important, rather than say the relative positions of the centres of mass of the objects, can be seen in Figure 1. In this case, the centres of mass of the two objects are far apart, both horizontally and vertically, and based on this property a relationship such as $\texttt{on}(\cdot, \cdot)$ could not be easily established. Algorithms such as support vector machines (SVMs) can compute this geometric separability reasonably efficiently, and so we use them as a tool in this sense, although other randomised algorithms may perform the same task. The SVM is thus trained to identify elements in the point cloud which are critical for defining the object separation. These regions are known as contact points, and can be connected in a graph to give a contact point network. The details of extracting these networks are discussed in Section 3.4.

Relationships between objects are then identified by examining the displacements between the contact points of two different objects. This is elaborated in Section 3.5. Although this algorithm operates on only pairs of objects at a time, the extension to sets of objects is given in Section 3.6.

An illustrative example of the process is shown in Figure 1, with sample input and output of the algorithm.

3.2 Assumptions

A point cloud $P_M = \{p_i\} = \{(x_i, y_i, z_i, r_i, g_i, b_i)\}$, $i = 1 \dots M$, consists of M points, where each point i has three dimensions of spatial information (x_i, y_i, z_i), as well as three dimensions of colour information (r_i, g_i, b_i). We assume this

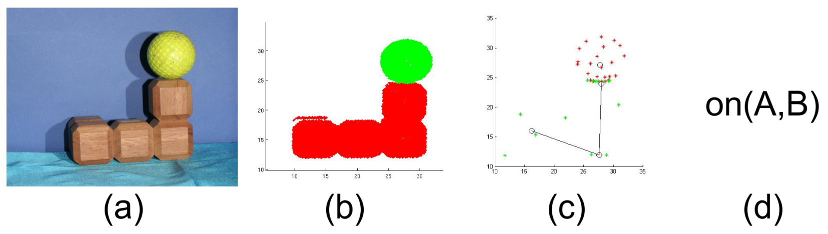

Figure 1: The relationship learning process. (a) The image captured by the left camera in a stereo camera pair. (b) The full point cloud, segmented into the two objects, after background subtraction. (c) The support vectors with the contact point networks of the two objects. The CPN of the ball is a single node, and the CPN for the L-shaped block is a three-node network. Note: in our experiments we used a user-defined threshold to discard contact points with less than 7 support vectors. This was not used in this image, to illustrate the concept of the network, but if it had, only the topmost contact point in the block would have remained. (d) The inferred relationship. A and B are arbitrary symbolic labels given to the two point clouds corresponding to the two objects.

point cloud has been captured by some device, such as stereo imaging or range scanning, but are indifferent to the hardware used for this purpose. What is required is that the pose of the imaging system is known. This work assumes the camera uses the same orientation as the scene, providing a known direction of "up". If this orientation is not known, a separate routine may be required to estimate this.

We assume that the background has already been segmented from the data, leaving a point cloud $P_N \subseteq P_M$, with $N \leq M$. P_N is thus the subset of P_M, where every point belongs to one of the objects in the scene.

Finally, we require that the various objects in the scene be segmented. This is the process of assigning a single class identity c_i to each point in the cloud, depending on the object to which that point belongs. This has been the subject of much previous work, using a number of different techniques based on various properties of the point clouds (e.g. Jiang et al. (2000)), and so will not be discussed here in depth. However, as it is an important part of the preprocessing of our algorithm, one simple procedure is given in Section 3.3. This procedure determines class identity as a function of point colour: $c_i = f(r_i, g_i, b_i)$, and although this is an unsophisticated procedure, it is sufficient for our current purposes.

A potential difficulty would arise if the objects are incorrectly segmented. However, as noted in Section 5, it may be reasonable for segmentation to fail in this way, as object identities in a static image may not even be correctly determined by a human, without the ability to perturb the scene. Even with this problem though, the algorithm returns a potential interpretation of the scene that could actually be useful, for example to seed a motion plan.

The colour information is not required by our algorithm, and so discarded. Hence the input to our algorithm is the combination of spatial information and object identities of each point in the cloud: $\tilde{P}_N = \{(x_i, y_i, z_i, c_i)\}$, for $i = 1 \dots N$. The final assumption we make is that this scene contains only two objects, i.e. $\forall i, c_i \in \{0, 1\}$. This simplifies the demonstrations in the paper, but is not a strong assumption. This assumption is relaxed in Section 3.6, by considering objects in the scene in a pairwise manner.

3.3 Object Segmentation

Presented here is one simple method for object segmentation, based on colour information in the point cloud. This is possible if the objects in the scene are of different colours. This strong assumption could be weakened by using a different segmentation method, relying on factors such as discontinuities in the curvature of surfaces fit to the point clouds, or assumptions based on the connectivity of regions of an object.

We normalise the RGB values of each point to reduce the effects of specularities, discretise the RGB space using a three-dimensional colour histogram, and then cluster the bins using the k-means algorithm. Setting $k = 3$ will generate three clusters, where the largest two clusters correspond to the two objects being segmented, and the third cluster absorbs remaining pixels from the background and shadows which may not have been correctly segmented out by the background subtraction process. Note that a larger value of k could be used if the number of objects was unknown, and any clusters of size below some threshold could be discarded as noise. Misclassified points are cleaned up by means of the k-nearest neighbours algorithm.

This process assigns each pixel p_i in P_N an identity $c_i = \{0, 1\}$, as belonging to one of the two clusters each corresponding to an object, resulting in the segmented and labeled point cloud $\tilde{P}_N$.

3.4 Extracting a Contact Point Network

The abstracted representation of the objects in a scene is useful as a concise description of the scene, and for reasoning about the structure of the environment. Furthermore, this representation also aids in classifying the relationships between the objects, as shown in Section 3.5. This representation is known as a contact point network for each object. We now describe the process for constructing these networks.

Given the labeled point cloud $\tilde{P}_N = \{p_i\} = \{(x_i, y_i, z_i, c_i)\}$, divide this into two distinct point clouds representing the two objects depending on the class identities c_i, such that object $\mathcal{O}^j = \{p_i | c_i = j\}$ with $j = \{0, 1\}$, giving that $\tilde{P}_N = \mathcal{O}^0 \cup \mathcal{O}^1$. We do not require a completely clean segmentation of the point clouds into the two objects, and the boundary may be noisy and slightly misaligned from the actual object boundaries, provided the noise does not qualitatively change the relationship between the objects.

The first step is to identify the contact regions between the two objects. This is done by training a support vector machine (SVM) to classify the two (already separated) objects, by defining a decision boundary between the classes which maximises the margin, being the smallest distance between any of the training points and the decision boundary (Bishop, 2006).

A conventional use of an SVM as a classifier might use features of the scene to classify whether or not a given test scene represents a particular predicate. Instead, we are using the property that an SVM also efficiently computes the geometric separator between two point sets, by classifying points as belonging to one of these two sets, which in this case are objects. The support vectors are thus in our case the features which are extracted from the data by the SVM, giving the contact points as markers of the geometric separators.

The training data provided is the combined dataset $\mathcal{O}^0 \cup \mathcal{O}^1$, which is labeled by object identities. As the data to be classified is in the form of two objects, it can be assumed that they are likely to form two regions (although each object may actually consist of multiple disconnected regions if it is obscured), with some unknown relationship between them. For this reason, we assume the objects are non-linearly separable, and so use a radial basis function (RBF) as the kernel in the SVM. An RBF is used as it is a nonlinear kernel with a localised response, and so is well suited to real-world objects, as the bulk of their mass is situated in closed regions with nonlinear boundaries.

The support vectors define the boundaries of the two objects. Let the support vectors for object $\mathcal{O}^i$ be $\mathbf{v}^i$. They are dense at any points where the two objects touch (or come near to touching) as these are regions where the boundary definition requires a high level of precision. Support vectors are also sparsely scattered around the silhouette of the object, as the contours of the decision boundary are defined with low precision in these regions. This is particularly true when the objects have a highly nonlinear boundary between them. As a result, regions of the object with a high concentration of support vectors are regions of the object at which the object is potentially in contact with another object. Furthermore, if the objects are far away from each other in some direction (relative to their sizes), very few if any support vectors are identified.

Clustering the support vectors $\mathbf{v}^i$ within an object provides a cluster at each (potential) point of contact between $\mathcal{O}^i$ and another object, with additional clusters of outlier points from the boundary of $\mathcal{O}^i$. The centroids of these clusters are known as the set of contact points $\{\chi_k^i\}$, $k = 1 \ldots K$ of the object. In practice, clusters are discarded if the number of support vectors in the cluster is below some user-defined threshold (a threshold of 7 was used in our experiments, but this parameter is somewhat dependent on the resolution of the data used).

The *contact point network* of an object $\mathcal{O}^i$ is defined as a graph CPN^i, where the nodes are the K contact points $\{\chi_k^i\}$, $k = 1 \ldots K$, and the edges $e(\chi_m^i, \chi_n^i)$, $m \neq n$ are the edges of the minimum weighted spanning tree (MWST) covering $\{\chi_k^i\}$, with the weights given by the Euclidean distances between the nodes.

An object's contact point network provides an abstracted representation of the object as a skeletal structure, based on the regions of the object where it comes into contact with another object in the scene. Although the contact points themselves convey information on the relative positioning of objects in a scene, the edges of the graph are useful firstly for identifying regions of an object which could be searched for grasp points, and secondly provide more structure to the representation. These graphs are useful for maintaining coherency of an individual object, as well as for determining when one composition of objects is a substructure of another. This is discussed further in Section 5. As a result, any of a number of algorithms could be used to construct these graphs, provided they were consistent across scenes.

3.5 Learning Relationships

Having extracted a contact point network from a pair of objects, the spatial relationship between these two objects can then be established. This relationship describes, in symbolic terms, the physical positioning of the one object relative to the other in the scene. We may wish to infer that "A is *on* B", or that "C is *under* as well as *adjacent* to D".

Given the contact point networks for two objects, CPN^i and CPN^j, consider all pairs of contact points (χ^i_m, χ^j_n). Let the displacement between these contact points be $d_{mn} = \chi^i_m - \chi^j_n$. This gives the oriented displacements of the two regions of the objects corresponding to those contact points. By definition, if the objects touch along some interface, then both objects must have a representative contact point for that region. Those contact points will then have a small separation displacement d_{mn}, with the distance and orientation defining the spatial relationship between the objects. The problem of classifying this spatial relationship has then been reduced to the problem of classifying $d_{mn} \in \mathcal{R}^3$. Although these displacement vectors may not be sufficient to capture every spatial relationship that exists in an environment as a result of, for example, large occlusions, the use of these contact points and the displacements between them provides us with a useful representation for abstracting a scene.

Taking a supervised learning approach to this classification requires a set of training data, that is, a set of point cloud images of pairs of objects, where the relationships between the objects have been provided. This input data is thus a set of triples, consisting of object pairs and relations, $\{(\mathcal{O}^i, \mathcal{O}^j, r^{ij})\}$, where r^{ij} is a symbolic label for a relationship that exists between the two objects. By the same procedure described above, the separation displacements can be extracted from the training images, and then labeled by the provided relationships. Call this set of labeled displacement vectors from the training data $\{(d_{ij}, r^{ij})\} = \mathcal{D}$.

To then classify the new separation displacement d_{query} a method such as k-nearest neighbours can be used to assign a label to d_{query} based on the most commonly occurring labels of the closest training points to d_{query} from $\mathcal{D}$. These labels are selected as the result of a voting scheme from the labels of the nearest neighbours. The variance of the labels provided by the nearest neighbours can then be used as a measure of confidence: the system is more confident of a solution if every neighbour proposes the same particular label, than if only 50% of them proposed that label. This acts as a quality measure of the output of the system in the form of predicted relationships.

Consider a training image with a pair of objects, $\mathcal{O}^0$ and $\mathcal{O}^1$, where CPN^0 has M^0 contact points and CPN^1 has M^1 contact points. There will be a total of M^0M^1 separation displacements for the object pair, $\{d_{mn}\}$, $m = 1 \dots M^0$, $n = 1 \dots M^1$. Each of these will receive the same relationship label r^{01}, as this label is provided at the level of objects, rather than regions. There is however probably only a single separation displacement which conveys this information accurately describing the spatial relationship. As a result, $M^0M^1 - 1$ spurious separation displacements will also carry the label r^{01}, even though they do not describe that particular relationship. These could enable the erroneous classification of a new object pair through d_{query}.

To overcome this issue, consider a threshold ω. A labeled separation distance d_{mn} extracted from the training data would only be added to $\mathcal{D}$ if $|d_{mn}| \leq \omega$. This would prevent the incorporation of displacements which are far from each other in the scene, although this threshold is dependent on the data. An alternative method for overcoming this problem is to weight the contributions of the k-nearest neighbours in classifying a label for d_{query}, by the inverse of the distance from d_{query}. This separation threshold is a context dependent, and therefore tunable, parameter.

One conjecture, yet to be examined in more detail, is that as an alternative to the classification approach, relations can be learned in an unsupervised manner. A similar approach of the unsupervised learning of relationships from data is seen in work such as that of Galata et al. (2002). Using this approach, labels are not provided for the training data. Instead, a set of separation displacements $\mathcal{D}$ are clustered, using an algorithm such as x-means (Pelleg and Moore, 2000) to determine the number of clusters as well. This approach does not suffer from the problem of spurious labeled data points. Each cluster is then interpreted as a unique binary relationship between two objects. These relationships can be assigned arbitrary names, as rel^k, $k = 1 \dots K$ for K relations, such that $\text{rel}^k : \mathcal{O} \times \mathcal{O} \to \{\mathtt{T}, \mathtt{F}\}$.

A concise description of the entire spatial relationships and abstraction extraction algorithm is given in Algorithm 1.

Algorithm 1: The spatial relationships and contact point network extraction algorithm

```
Input: A 3D point cloud with pre-segmented background $P$, and the
       labeled training data $\mathcal{D}$
Output: A contact point network for both objects in $P$, $\text{CPN}^0$ and
        $\text{CPN}^1$, as well as a symbolic relationship rel between the
        objects
begin
    $[\mathcal{O}^0, \mathcal{O}^1] \longleftarrow$ segment-objects($P$)
    svm $\longleftarrow$ train-svm($\mathcal{O}^0, \mathcal{O}^1$)
    $[\mathbf{v}^0, \mathbf{v}^1] \longleftarrow$ extract-support-vectors(svm)
    for each object $k$ do
        $\boldsymbol{\chi}^k \longleftarrow$ cluster($\mathbf{v}^k$)
        $\mathbf{e}^k \longleftarrow$ minimum-weighted-spanning-tree($\boldsymbol{\chi}^k$)
        $\text{CPN}^k \longleftarrow (\boldsymbol{\chi}^k, \mathbf{e}^k)$
    end
    rel $\longleftarrow \emptyset$
    for $(\chi^i, \chi^j)$, $\chi^i \in CPN^0$, $\chi^j \in CPN^1$ do
        $d_{ij} \longleftarrow \chi^i - \chi^j$
        $\mathbf{d} \longleftarrow$ nearest-neighbours($d_{ij}, \mathcal{D}$)
        $r \longleftarrow$ voted-common-labels($\mathbf{d}$)
        rel $\longleftarrow$ rel $\cup\, r$
    end
    return ($CPN^0$, $CPN^1$, rel)
end
```

3.6 Extensions to Multiple Objects

The procedure outlined in the previous section describes ways in which objects can be abstracted into their contact point network skeletons, and further that the spatial relationships between the objects can be determined from these networks. This was all done by considering only a pair of objects at a time. Considering larger groups of objects is a straightforward extension.

For a set of objects in a scene, the algorithm is extended to generate contact point networks by considering all pairs of objects in the scene. The question is then how to merge the networks generated for a particular object when considered together with other objects. The solution is to collect the set of all contact points from every different network representation of a single object, and then build a new network from these points.

Let $\mathrm{CPN}^{i|j}$ denote a contact point network of object $\mathcal{O}^i$ extracted by the algorithm when considered together with object $\mathcal{O}^j$. Now $\mathrm{CPN}^{i|j} \neq \mathrm{CPN}^{i|k}$, for $\mathcal{O}^j \neq \mathcal{O}^k$. This is because the two other objects cannot interface with $\mathcal{O}^i$ in the exact same way, and hence give rise to different skeleton representations of $\mathcal{O}^i$.

Merge $\mathrm{CPN}^{i|j}$ and $\mathrm{CPN}^{i|k}$ to give $\mathrm{CPN}^{i|(j\cup k)}$ by combining the two sets of contact points. Formally, let $\mathrm{CPN}^{i|j}$ have $M^{i|j}$ contact points $\{\chi_m^{i|j}\}$, $m = 1 \dots M^{i|j}$ and $\mathrm{CPN}^{i|k}$ have $M^{i|k}$ contact points $\{\chi_n^{i|k}\}$, $n = 1 \dots M^{i|k}$. When merged, $\mathrm{CPN}^{i|(j\cup k)}$ then has $M^{i|(j\cup k)} = M^{i|j} + M^{i|k}$ contact points $\{\chi_p^{i|(j\cup k)}\}$, $p = 1 \dots M^{i|(j\cup k)}$. The edges of this merged contact point network are created by constructing a minimum weighted spanning tree over the combined set of nodes, with the weights of the edges in the tree determined by Euclidean distances between the nodes.

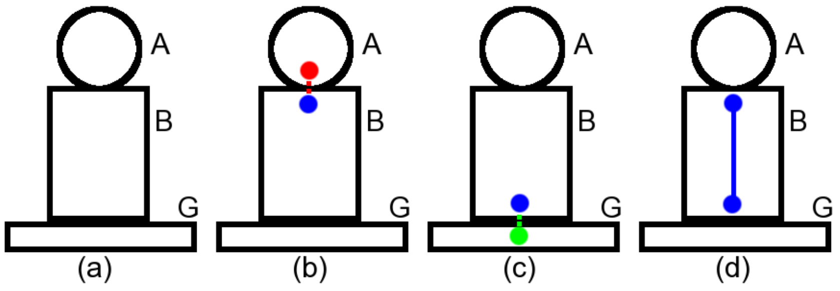

Figure 2: A simple illustrative example of combining contact point networks. Filled circles represent contact points. Solid lines are parts of the skeleton of an object, and dashed lines show relationships between two objects. (a) The base configuration of the scene. (b) Processing of objects A and B. (c) Processing of objects B and G. (d) Combining the information extracted from these two steps.

A simple illustrative example of combining two contact point networks for a single object is shown in Figure 2. This demonstrates how a scene consisting of three objects will be processed by the algorithm. The scene is described as: a ball A rests on a block B, which in turn rests on the ground G. Processing objects A and B gives one contact point in each object, which is enough to infer $\texttt{on}(A, B)$. Similarly, processing objects B and G gives one contact point in each object, which is enough to infer $\texttt{on}(B, G)$. Combining the information extracted from these two steps, gives a symbolic description of the scene as $\{\texttt{on}(A, B), \texttt{on}(B, G)\}$, as well as a two-node topological description of B. This represents how the object B is used in this particular scene, rather than describing any distinctive properties of B itself.

4 Experiments

4.1 Classifying Relationships

In order to validate our algorithm, we present the following demonstration: the algorithm is run on the segmented point clouds generated from a set of stereo images captured in our lab. Firstly, we show that given a set of training data, the relationships between two new objects can be correctly classified by our algorithm. Secondly, we show that the contact point networks extracted from the objects convey interesting topological structure, which may aid manipulation. Finally, we demonstrate the effect of merging contact point networks, as described in Section 3.6.

For our experiments we used the following hardware configuration to acquire our images: a stereo capture rig was constructed of two 8.0 MPixel Canon EOS 350D cameras, calibrated and using the maximum level of magnification supported by the standard EF-S lens (0.28m closest focus distance). Dense registered range and colour data was collected using a Dimensional Imaging system. Intensity images have a spatial resolution of 3456×2304 pixels, inter-pixel spacing is 25pixel/mm. RMS depth error is around 0.03mm. Although the resolution used in these images was high, the results appeared fairly robust when randomly discarding up to 70% of data points.

For the first experiment, we gathered a set of 128 stereo images, each consisting of a pair of objects. The objects varied greatly in shape, colour and texture, with the selected objects including a golf ball, blocks of various dimensions, a stone, a cloth, a piece of aluminium foil, and several plastic and furry childrens' toys. The range of objects used demonstrate that the algorithm is not particular to planar surfaces or convex volumes. In each case a blue material backdrop and table cover was used. This allowed for automated mask generation for background segmentation by the Dimensional Imaging software, as a part of the process of registering the stereo image pair from the two cameras and converting the output into a three-dimensional point cloud. A selection of the images which were photographed and used is shown in Figure 3.

This experiment aimed at examining two example relations: $\texttt{on}(\cdot, \cdot)$ and $\texttt{adjacent}(\cdot, \cdot)$. The adjacent relation is considered in this case to mean that to objects are visually next to each other in the static scene, i.e. an object A is either to the left or the right of another object B, and so $\texttt{adjacent}(A, B) = \texttt{left-of}(A, B) \cup \texttt{right-of}(A, B)$. For each image $\mathcal{I}^j$ with objects $\mathcal{O}^{j0}$ and $\mathcal{O}^{j1}$, a label $r^j \in \{0, 1\}^2$ was prepared, where:

- $r^j(0) = 1$ if ($\mathcal{O}^{j0}$ on $\mathcal{O}^{j1}$)
- $r^j(0) = 0$ if ($\mathcal{O}^{j0}$ not on $\mathcal{O}^{j1}$)
- $r^j(1) = 1$ if ($\mathcal{O}^{j0}$ adjacent to $\mathcal{O}^{j1}$)
- $r^j(1) = 0$ if ($\mathcal{O}^{j0}$ not adjacent to $\mathcal{O}^{j1}$)

Representing the relations as a vector of labels allows for cases where one object is both on and adjacent to the other object (possibly at different places). Using this label representation scheme, there are thus four ways in which one object can relate to another: on, adjacent, on and adjacent, none. In practice, all of the images we used in our experiments had one of these four relationships

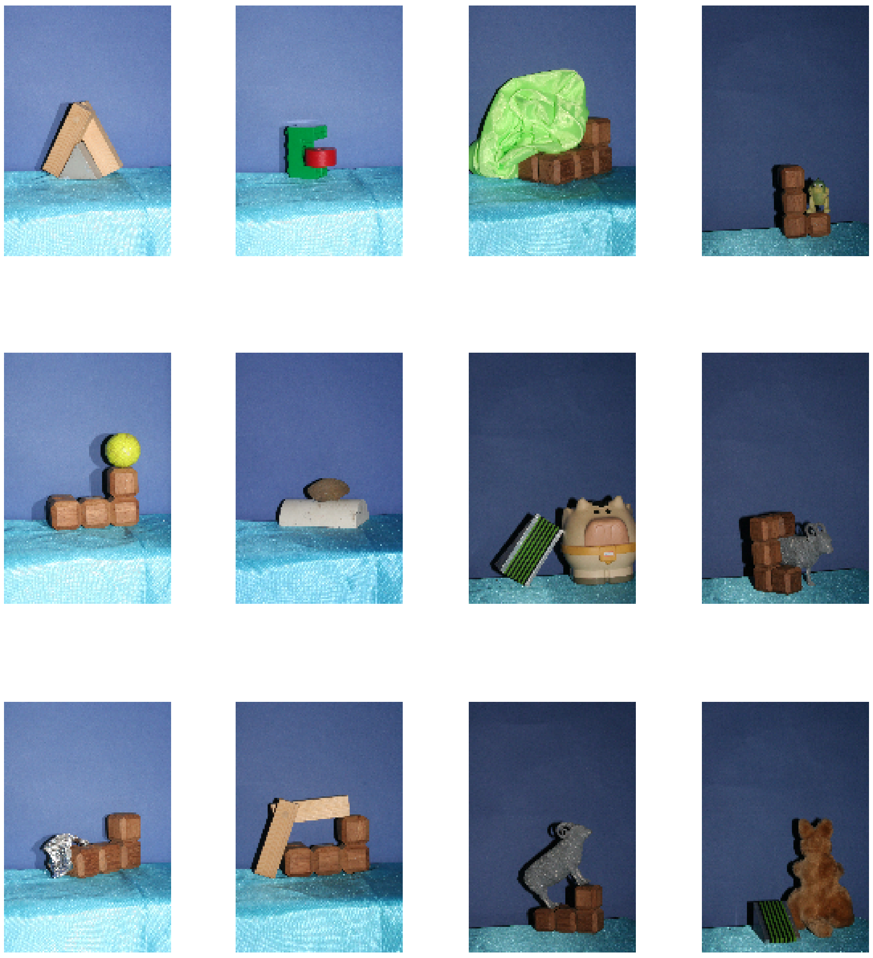

Figure 3: A subset of 12 of the object configurations used. A total of 128 such images were used in our experiments. In all cases the images shown are those which were photographed by the left camera of the stereo system. Some of the images are darker with less contrast as these photographs were taken in a second session with different lighting conditions.

existing between the two objects. Note that for any image where the distance between the two objects is great, there would be no contact points in either object, and as a result no displacement vectors would be generated.

The implementation of the algorithm was in MATLAB, using the external *SVM-light* implementation of Joachims (1999). In practice, this implementation was sufficient to process an entire image in well under a second on a dualcore desktop computer.

Each image in the dataset was run through the algorithm, and the separation displacements d_{mn} were extracted. A plot of these displacements is shown in Figure 4. These carry the labels assigned to the spatial relationships between the objects in the images. Several features are clearly visible in this data. Firstly, there is a defined cluster describing the $\texttt{on}(\cdot, \cdot)$ relationship, centered at $(0, -0.5)$.

Secondly, there is a broad band of points above $y = -0.1$ which corresponds to the `adjacent`$(\cdot, \cdot)$ relation. Note that this incorporates both `left-of`$(\cdot, \cdot)$ and `right-of`$(\cdot, \cdot)$, represented respectively by those displacements with a negative or a positive horizontal component. Object pairs which exhibit both `on` and `adjacent` at different points should contribute displacement vectors to both classes.

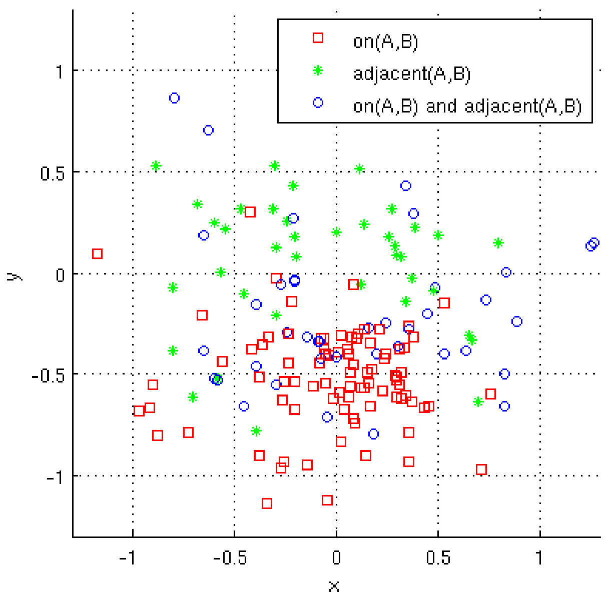

Figure 4: A plot of the x-y projection of the 3D separation displacements d_{mn} for the 128 images of object configurations used as training data. Each point corresponds to a vector starting at the origin and ending at that point, and is labeled according to the relationship exhibited in the corresponding image, being `on`$(\cdot, \cdot)$, `adjacent`$(\cdot, \cdot)$ or `on-and-adjacent`$(\cdot, \cdot)$.

We test the classification ability of this method by using each image of the training set as a testing image, in a leave-one-out cross-validation test. The procedure used was as follows: for each image $\mathcal{I}^j$, we remove that image from the dataset. Using the remaining training data, we classify the relationships between the objects in $\mathcal{I}^j$ by considering the nearest neighbours (the number of neighbours considered in our experiments was 4, but this parameter could be changed) to the separation displacements from that image. For the cases where there are two relationships between the objects in the image, indicated by the presence of multiple separation displacements, we take the union of the predicted values. So if one separation displacement is classified as `on` and another as `adjacent`, the description of the relationship between the objects is (`on` and `adjacent`). Note that the label $r^j \in \{0, 1\}^2$ in each case. As each test image came from the set of training images, we compare the predicted results to the true results. This is shown in Figure 5.

As may be expected, the method suffered most at the interface between the two clusters. This is indicative of the fact that humans often struggle to cleanly

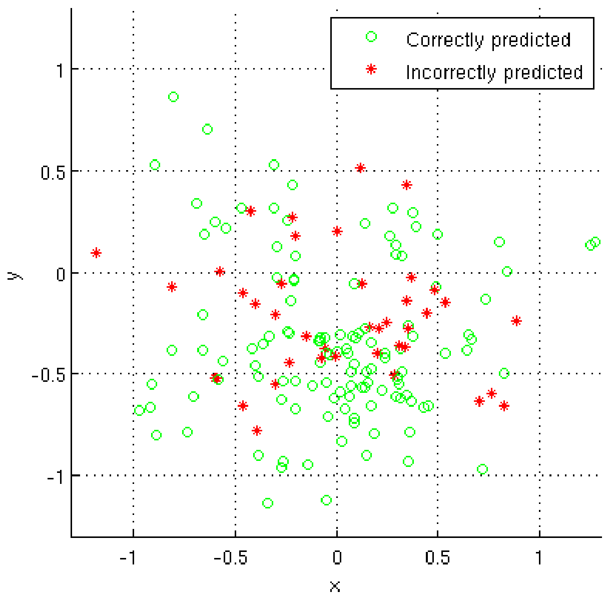

Figure 5: A comparison of the true and predicted object relationships

differentiate between these relations. For example, two objects leaning against each other could be described as either on or adjacent. The confusion matrix of the predicted and true labels is:

		Actual			
		on	adjacent	both	none
Predicted	on	64	6	13	0
	adjacent	2	20	5	0
	both	4	5	12	0
	none	0	1	0	0

As can be seen, the cases with which the algorithm had the most difficulties were those when both relationships were present. These were often misclassified as being `on`. We can explain this problem by considering two cases for when one object may be both on and adjacent to another. The first is if these two relationships occur at disjoint regions in the object. This will give rise to a pair of separation displacements, one at the interface described as `on` and the other at the `adjacent` regions. On the other hand, one region in an object may be both on and adjacent to the same region in another object. In this case, the separation displacement would not be classified as either relationship, but rather an intermediate one.

The input data set of 128 images was then randomly divided into a training set of 95 training images, and 33 testing images, corresponding to 116 training relationships and 49 testing relationships respectively, as each object pair may contact multiple contact points. Again using k-nearest neighbours, with $k = 4$, it was found that 40 of the 49 test relationships had their `on` component correctly determined, and 39 of the 49 test relationships had their `adjacent` component predicted correctly. Using the naïve confidence measure of the percentage of the nearest neighbours with labels agreeing with the predicted values, 28 and 18 of the respective predicted relationships agreed with 75% of their nearest neighbours from the training set.

4.2 Contact Point Networks

In addition to classifying the relationships between objects, the algorithm abstracts the pair of objects into two contact point networks. The advantage of this is that these networks are representations of the objects in terms of how they interact with other objects in the scene. These skeletons allow an agent interacting with the objects to uncover the topological structure present in the individual objects, as well as the entire scene, which could then be used by the agent to guide further interaction.

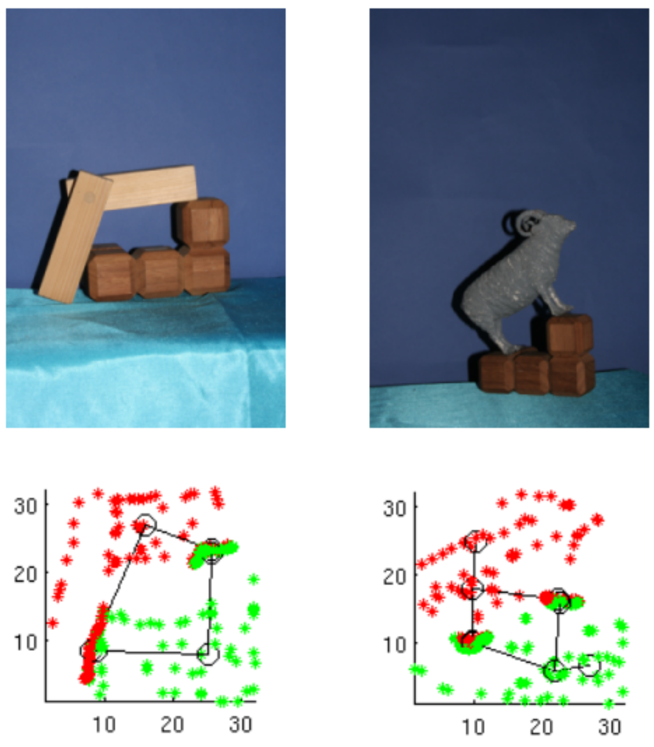

Figure 6: Using contact point networks to discover topological structure in two scenes. The algorithmic output is shown below each scene, where the points are the support vectors for each object, and the open circles are the contact points.

As an example of this principle, consider the two scenes in Figure 6. Both scenes consist of an object resting on an L-shaped wooden block, but these objects (two wooden blocks connected by a rotary joint, and a plastic toy goat) are very different. However, in both scenes a hole is formed as a result of the interaction between the two objects. This is clearly visible in the contact point networks of the objects in the scene. In both cases, the algorithm has discovered a topological hole in the environment. This suggests that if, for example, an agent has a motion strategy to pick up the top object from underneath for the first scene, it could use a similar motion strategy as a candidate for performing this action in the second scene.

This demonstrates the ability of the algorithm to abstract complicated objects and shapes down into the fundamental topological structure which arises from the way in which these objects are used in this particular scene. Having extracted these structures, there is a large body of literature which addresses the topologies of graphs (see for example Cook and Holder (2007); Carlsson (2009)), for operations such as detecting commonly occurring sub-structure.

4.3 Multiple Objects

Section 3.6 describes a method for extending the algorithm to cases where there are more than two objects in the scene, which is to be expected from any real environment. This extension involves examining the objects in a pairwise manner, and then joining the contact point networks of the same object, provided by its interfaces to various other objects in the scene.

Two examples of scene with multiple objects are shown in Figure 7. The first instance shows three objects stacked above each other, with the result that instead of each skeleton being a single point, these two points are merged into one skeleton of two nodes for the central object. The second instance actually represents a failed segmentation, where two objects are incorrectly segmented into three. The same skeleton structure is observed in these two instances, showing that each object in the first scene has an equivalent object in the second scene, in terms of the structural use of that object in the scene. Note that even with the failed segmentation, a viable interpretation of the structure of the scene is extracted.

Clearly, in cases such as those in Figure 7, the objects as well as the contact points that are detected by the algorithm are only candidates, and may not actually correctly reflect the scene's structure. These could perhaps be disambiguated by motions, such as moving components of the scene and observing which objects move consistently together. Nonetheless, the structural abstraction in both cases mirrors the essence of the scene.

5 Discussion

The algorithm presented in Section 3 generates two different representations of a scene. Firstly, the spatial relationships between the different objects in the scene are extracted as symbolic predicates, and secondly the topological structure of the objects in the scene is constructed, as determined by these inter-object interactions.

The symbolic relationships between objects define the structure of an environment, and as a result, learning these relationships and being able to identify them in a static scene has important ramifications for planning and learning

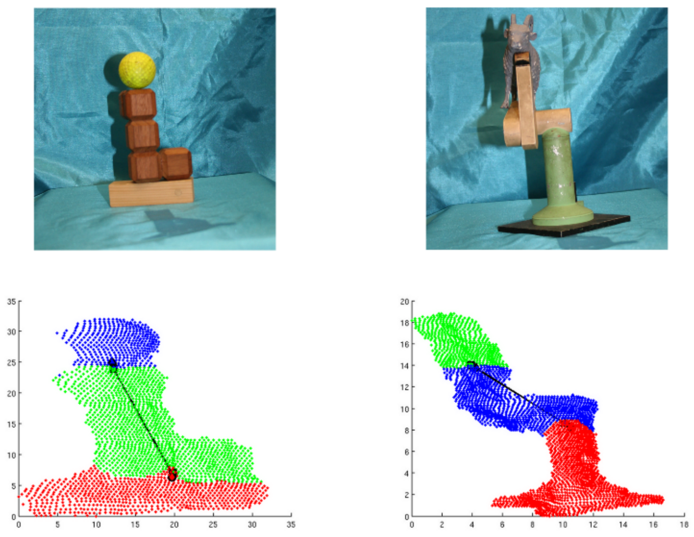
Figure 7: Examples of processing more than two objects in a scene. The point clouds are shown, with the skeletons of the objects.

the effects of actions, as done in the work of Pasula et al. (2007). All of this builds towards a better understanding of the overall structure and behaviour of the components of an environment, as these relationships provide a means for describing changes in the relative positioning of objects.

Similarly, these contact point networks are important for understanding the capabilities of an individual object, or set of objects, and provide insight into the topological structure of the environment. This enables reasoning and policy reuse at a coarser, more abstract level, which is important in practice. This is in contrast to the approach of first identifying an object from a database of known objects, as an agent may not need to know exactly what an object is, in order to use it.

This algorithm generates a layered representation of a scene, shown in Figure 8. At the lowest level is the point cloud, which consists of the most information, and is the direct output of the perception system of the robot acting in that environment. This level is useful for predicting collisions between the robot and object, and other low level control functions.

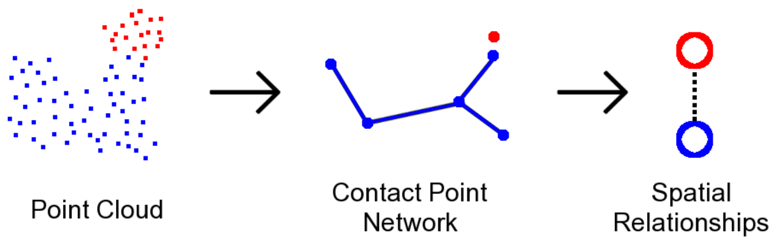

Figure 8: An illustration of the layered scene representation

The next level is the contact point network, which provides a manipulation robot with a set of candidate points and edges for interacting with the objects, as well as knowledge of similarities in structure of different parts of the scene. Finally, the relational predicates provide the robot with symbolic knowledge of the inter-object relationships, which can be easily used in plan formulation.

There are many useful mechanisms for reasoning about the relationships between objects and parts of objects, such as the spatial calculus of Randell et al. (1992). Logics such as this do not describe how the elementary concepts would arise from data, and our work attempts to bridge that divide, by providing a semantic interpretation of a scene in symbolic terms. Our algorithm provides a mechanism for identifying some of these relationships in a scene, and thus serves to ground this spatial logic calculus, thereby allowing the calculus to be utilised by a robot.

Reasoning can also be done at the level of the topological structures observed in the contact point networks in a scene. An example of this can be seen in the work of Calabar and Santos (2011). Our algorithm again bridges the gap, and would allow for the real-world versions of the spatial puzzles solved in this work, by identifying holes and other structures in the topology of the puzzles. We are not only interested in determining valid grasp points (Saxena et al., 2008), but are interested in the structure of a scene as a whole.

In addition to each layer providing an agent with different means for reasoning about a scene, another advantage to this layered representation is an increased robustness to segmentation failures. The most likely cause of failure of this algorithm is a result of its dependence on a relatively clean segmentation of the objects in the scene. The method is robust to small noise-related errors in the segmentation, but is prone to difficulties if, say, two parts of the same object are classified as being different objects, or conversely if two separate objects are classified as being the same object. However, as shown in Section 4.3, a plausible explanation for the structure of the scene will still be generated. In fact, given that any objects may be fixed together in any scene by glue or some other medium, only the incorporation of actions to perturb the scene such as poking and prodding would lead to a guaranteed correct object segmentation.

For the first of these cases, the segmentation of one object into multiple parts, the algorithm will still return the correct relationship between those parts, even if they do not correspond to whole objects. For a static scene, this is still a plausible description of the scene, even if not the most likely. A simple example of this is shown in Figure 9, where even though an object is incorrectly segmented, the relationships between these regions are still correctly identified. Using a similar methodology to the interactive perception proposed by Katz and Brock (2008), a robot manipulating in that environment may discover that even when subjected to various forces, the relationships between these parts remain constant. Using this observation, the parts could be conceptually merged into a single object. This remains the subject of future work.

Similarly, if two objects have not been separated and are instead considered as a single object, this is also a possible description of a single scene. When these objects are acted on, the relationship between them is likely to change, and thus the second object could be detected. In fact, it often happens that two objects only become distinguishable to a human under action, such as two adjacent papers on a desk, or a stick-insect in a tree.

This algorithm is somewhat related to the idea of bootstrap learning, as

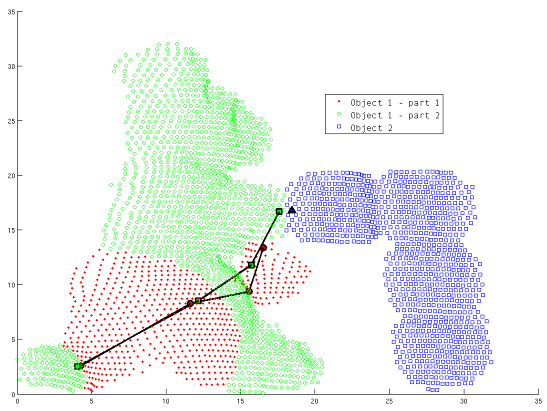

Figure 9: Relationships between incorrectly segmented objects. The object on the left was incorrectly subdivided into two different objects.

expounded by Kuipers et al. (2006). This paradigm aims at developing a set of methods whereby agents can learn commonsense knowledge of the world, from its own observations and interactions with that world. There has already been much success in agents learning reliable primitive actions and sensor interpretations, as well as recognising places and objects. Our algorithm provides that same agent with a mechanism for learning about the different spatial relationships between those objects.

As an example of this, consider Figure 10. This shows two different scenes, each consisting of three objects stacked on each other. While stacking in the first scene involves each object having only one point of contact with the object below it, the second scene has two contact points between the topmost object and the one below it. The abstract skeletal structure of the first scene in the form of its contact point network can be seen here to be a subgraph of the skeleton of the second scene.

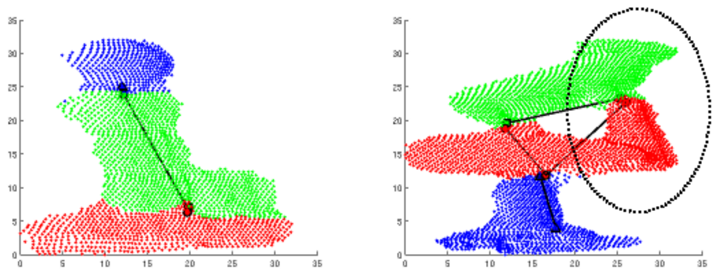

Figure 10: Example of one structure as a substructure contained in another. Without the inclusion of the objects under the dotted ellipse, the structure of the second image is topologically equivalent to that of the first. The skeleton of the first image is thus a subgraph of the second.

As a result of the first scene being a subgraph of the second, an agent operating in the space of these scenes can seed its behaviour in the case of the second scene with the behaviours it used in the first scene, e.g. grasp points. Furthermore, the additional components of this new structure over the previous one afford the agent new regions on the pile of objects to manipulate through exploratory actions.

6 Conclusion

In order for a robot to be able to manipulate objects in a meaningful way within an environment containing spatially interacting objects, it must possess knowledge of how the different objects in that environment are used in that environment, as well as how they relate to one another. The set of spatial relationships between objects is the glue which holds a scene together, and allows for differentiation between a set of items, and a structured environment.

We propose an algorithm for learning a layered representation of an environment, including a structural abstraction as well as these spatial relationships, and being able to classify them in either a supervised or an unsupervised manner. The algorithm finds regions of contact between pairs of objects, known as contact points, by locating areas where there is a small margin of geometric separability between the objects. Constructing a graph from these contact points gives a contact point network which is a topological description of the role of the object in the scene. These contact point networks provide the agent with the ability to abstract objects into simpler topological structures. Inter-object relationships are classified based on the separation between contact points from the skeletons of two different objects.

The networks and the relationships provide a layered representation of a scene consisting of multiple objects. This representation facilitates reasoning about the objects at different levels, through the use of planning mechanisms, as well as searching for common structure in different environments to seed behaviours and thus significantly reduce the search space for motion synthesis. This could enable a robot to perform comprehensive tasks in a manipulation environment.

Acknowledgements: The authors would like to gratefully acknowledge Lily Li Xiang, Steven McDonagh and David Braude for their help in calibrating and using the equipment used for capturing the image data used in the experiments. We also wish to thank the three anonymous reviewers for their very insightful and thorough comments and suggestions.

This work has taken place in the Robust Autonomy and Decisions group within the Institute of Perception, Action and Behaviour, School of Informatics. Research of the RAD Group is supported in part by grants from the UK Engineering and Physical Sciences Research Council (EP/H012338/1) and the European Commision (TOMSY).

References

Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? *International Conference on Computer Vision and Pattern Recognition*, pages 73–80.

Barck-Holst, C., Ralph, M., Holmar, F., and Kragic, D. (2009). Learning grasping affordance using probabilistic and ontological approaches. *International Conference on Advanced Robotics.*

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Calabar, P. and Santos, P. E. (2011). Formalising the fisherman's folly puzzle. *Artificial Intelligence*, 175(1):346–377.

Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308.

Cook, D. J. and Holder, L. B. (2007). *Mining Graph Data.* John Wiley and Sons.

Dee, H. M., Hogg, D. C., and Cohn, A. G. (2009). Scene modelling and classification using learned spatial relations. *COSIT-09, Lecture Notes in Computer Science*, (5756):295–311.

Desai, C., Ramanan, D., and Fowlkes, C. (2009). Discriminative models for multi-class object layout. *International Conference on Computer Vision*, pages 229–236.

Galata, A., Cohn, A. G., Magee, D. R., and Hogg, D. C. (2002). Modeling interaction using learnt qualitative spatio-temporal relations and variable length markov models. *European Conference on Artificial Intelligence*, pages 741–746.

Galleguillos, C. and Belongie, S. (2010). Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722.

Galleguillos, C., Rabinovich, A., and Belongie, S. (2008). Object categorization using co-occurence, location and appearance. *International Conference on Computer Vision and Pattern Recognition.*

Ghallab, M., Nau, D. S., and Traverso, P. (2004). *Automated Planning: Theory and Practice.* Morgan Kaufmann.

Gibson, J. J. (1986). *The Ecological Approach to Visual Perception.* Lawrence Erlbaum Associates, Inc., 2nd edition.

Hauser, K. and Latombe, J.-C. (2010). Multi-modal motion planning in non-expansive spaces. *International Journal of Robotics Research*, 29(7):897–915.

Jain, A. K. and Dorai, C. (2000). 3d object recognition: Representation and matching. *Statistics and Computing*, 10(2):167–182.

Jiang, X., Bowyer, K., Morioka, Y., Hiura, S., Sato, K., Inokuchi, S., Bock, M., Guerra, C., Loke, R. E., and du Buf, J. M. H. (2000). Some further results of experimental comparison of range image segmentation algorithms. *International Conference on Pattern Recognition*, 4:877–881.

Joachims, T. (1999). *Making Large-Scale SVM Learning Practical*, chapter 11. Advances in Kernel Methods - Support Vector Learning. MIT Press.

Katz, D. and Brock, O. (2008). Manipulating articulated objects with interactive perception. *International Conference on Robotics and Automation*, pages 272–277.

Kuipers, B. J., Beeson, P., Modayil, J., and Provost, J. (2006). Bootstrap learning of foundational representations. *Connection Science*, 18(2):145–158.

Leibe, B. and Schiele, B. (2004). Scale-invariant object categorization using a scale-adaptive mean-shift search. *Lecture Notes in Computer Science*, 3175:145–153.

Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. *International Conference on Robotics and Automation*, pages 2308–2315.

Pasula, H. M., Zettlemoyer, L. S., and Kaelbling, L. P. (2007). Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29(1):309–352.

Pelleg, D. and Moore, A. W. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. *International Conference on Machine Learning*, pages 727–734.

Pinz, A., Bischof, H., Kropatsch, W., Schweighofer, G., Haxhimusa, Y., Opelt, A., and Ion, A. (2008). Representations for cognitive vision: A review of appearance-based, spatio-temporal, and graph-based approaches. *Electronic letters on computer vision and image analysis*, 7(2):35–61.

Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. *International Conference on Knowledge Representation and Reasoning*, pages 165–176.

Rusu, R. B., Holzbach, A., Diankov, R., Bradski, G., and Beetz, M. (2009). Perception for mobile manipulation and grasping using active stereo. *Humanoids*, pages 632–638.

Saxena, A., Driemeyer, J., and Ng, A. Y. (2008). Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(2):157–173.

Waltz, D. L. (1975). *Understanding Line Drawings of Scenes with Shadows*, pages 19–92. The Psychology of Computer Vision. McGraw-Hill.