

Part-based modelling of compound scenes from images

Anton van den Hengel^{12*} Chris Russell³ Anthony Dick¹ John Bastian¹
Daniel Pooley¹ Lachlan Fleming¹ Lourdes Agapito³

The University of Adelaide¹ The Australian Centre for Robotic Vision² University College London³

Anton.vandenHengel@adelaide.edu.au, crussell@cs.ucl.ac.uk

Abstract

We propose a method to recover the structure of a compound scene from multiple silhouettes. Structure is expressed as a collection of 3D primitives chosen from a pre-defined library, each with an associated pose. This has several advantages over a volume or mesh representation both for estimation and the utility of the recovered model. The main challenge in recovering such a model is the combinatorial number of possible arrangements of parts. We address this issue by exploiting the intrinsic structure and sparsity of the problem, and show that our method scales to scenes constructed from large libraries of parts.

1. Introduction

We propose a method to estimate the structure of compound scenes from a set of images. Such scenes are prevalent in our everyday environment, and in many cases our knowledge of their innate structure is essential to our understanding of them. They include man made objects such as buildings, furniture, and cars, but also natural objects such as trees and plants. Our goal is to find the simplest construction which explains the shape of the scene, using a given library of parts. Unlike most work on the recovery of shape from images, our method does not generate a point cloud, or a volume, but a structural explanation of way the scene depicted is constructed. In this sense it is aligned with the blocks-world approach [15], recently revisited by [7].

Much like the blocks-world approaches, our goal is to recover a semantic model of the structure of the scene. Instead of creating a simpler volumetric, or point cloud model of a scene, we wish to create a model which captures interdependencies between parts of a scene, and allows us to say “These are the wheels of the car so this is how it will move.” (fig. 1), or “This is how a wall might collapse in an accident, or a temple might collapse in an earthquake.” (c.f. sup. video).

*This work was in part funded by the Data to Decisions Cooperative Research Centre, Australia, and by the Australian Research Council.

In this work, we focus on Lego[®] block based reconstructions of both synthetic and natural scenes. We do this for two reasons: The use of Lego blocks emphasizes the tactile and interactive nature of our reconstructions, while the grid structure of Lego also provides a natural discrete parametrization of our pose space. It should be emphasized, however, that despite the grid structure of Lego, our method can be successfully applied to reconstructing natural structures that are not based on a grid.

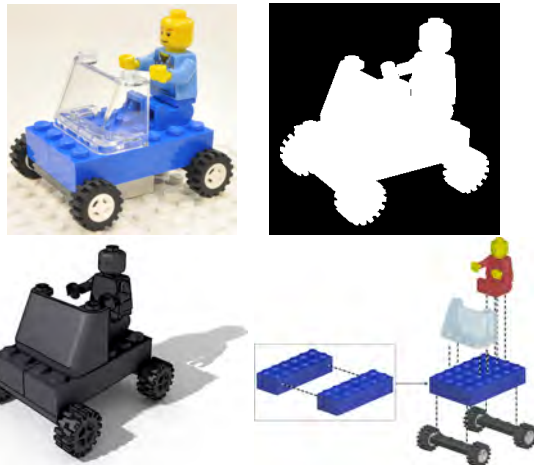


Figure 1. An illustration of the deconstruction process, from real world images, and silhouettes, to an estimate of the building blocks from which an object is constructed, and how they fit together.

The method we propose reasons in 3D about the structure of a scene on the basis of its appearance in an image set. This requires an initial set of building blocks from which an scene might be composed. As we are interested in structure, rather than appearance, these building blocks are defined uniquely by their shape and position. Our recovered structure estimate is the smallest set of building blocks required to reconstruct the scene in question.

By estimating structure, rather than shape, the method provides a semantic interpretation of the elements from which the scene is constructed. This means that the resulting structure estimate can be used to animate the scene,

decompose it, or augment it. For example, the estimated structure of a car identifies the doors and wheels, as is required to rig it in order to animate each component appropriately. This in turn allows the modeled object to be incorporated into a video game or video sequence without further work. Our method thus differs significantly from classical Structure from Motion methods; we are effectively deconstructing the scene into semantically meaningful constituent parts.

Representing a scene or object by 3D primitives also has advantages in shape estimation. The space of possible primitive arrangements is far smaller than the space of possible combinations of 3D vertices or voxels, resolving ambiguities and eliminating shapes that are impossible.¹ However, the solution space remains extremely large for complex scenes, and one of our main contributions in this paper is a way of formulating the primitives and optimizing their arrangement so that a global optimum, corresponding to the simplest configuration that explains the input silhouettes, can be found within this space.

Our notation follows standard conventions: we label scalars in non-bold italic typeface, with lower-case indicating an index and upper case its limit, thus $p \in \{1, \dots, P\}$; vectors are represented as bold lower case letters (eg. \mathbf{x}), and are column vectors; Matrices appear as bold upper case letters (eg. \mathbf{X}).



Figure 2. A selection of the template types used in recovering the structure of the Lego[®] models. The set of templates consists of each template type rendered in every plausible location and orientation.

1.1. Related Work

The idea of analyzing the composition of a visible object or scene in terms of a set of basis parts has a long history in computer vision, stretching back to the blocks-world interpretation of flat shaded polyhedral surfaces in the 1960s [15]. This was further developed to interpret collections of simple shapes [4], to reason about structure from physical constraints [3], and, recently, to inform the recovery of a qualitative scene reconstruction from a single image [7]. There are also a variety of methods which interpret images or point clouds using predefined families of surfaces, or implicit functions (see for example [20], and for

¹Fig. 10 shows a clear example of this where a high fidelity reconstruction is obtained using only a few primitives.

a reverse engineering application [5]). Parametrized sets of transformations have also been used to identify the building blocks of building facades [13] and 3D structures [14, 2].

In contrast, our approach poses structure recovery as a classification problem where evidence is sought within the image set for the existence of each of the possible building blocks that could make up the object. This approach is thus more closely related to work such as [21] which describes a method for object recognition using linear combinations of images. They recognize objects by reconstructing the location of features in a query image as a linear combination of those in images of database objects. However, the method aims only to recognize objects for which it has a pre-existing model.

Shakhnarovich et al. in [18] propose Parameter Sensitive Hashing (PSH) as a method for estimating human pose from a single silhouette. The method thus recovers pose from a projection, as we aim to do. A similar problem is tackled in [6] using a Bayesian human shape model. These methods aim to recover the pose of a single predefined structure, however, rather than constructing a model from a set of basis shapes or templates.

Our method also has similarities with work in non-rigid reconstruction including the piecewise methods such as [16, 17] that fit generic models to parts of a video using a segmentation approach. These methods typically require a very specific camera model and a continuous video sequence to work well, and aim to recover shape rather than structure.

Being silhouette-based, our method is also comparable to much of the work in the shape-from-silhouette area, and specifically space-carving [10, 2] and the significant work that followed. In contrast to these approaches we do not aim to recover a volumetric model consistent with a set of silhouettes, but rather are concerned with describing the structure of the object which explains the silhouettes in terms of the conjunction of a set of building blocks.

2. Estimating structure

We wish to estimate a simple physically plausible structure that is consistent with a set of binary images of silhouettes (which we denote, in their concatenated and vectorized form by \mathbf{y}). This physical structure is described in terms of the presence or absence of a set of templates, and is parametrized by α (a binary vector in which $\alpha_i = 1$ indicates the presences of template i in the structure).

Given a measurement vector \mathbf{y} of length S , and a collection of T silhouettes of templates, stacked into an $S \times T$ matrix $\mathbf{\Pi}$, we seek the most parsimonious combination of templates that is both physically plausible and close to the observed image data:

$$\arg \min_{\alpha \in \Omega} \delta(\mathbf{y}, \mathbf{\Pi}\alpha) + \lambda\tau(\alpha), \quad (1)$$

where Ω is the set of physically plausible models, $\delta(\mathbf{y}, \mathbf{\Pi}\alpha)$ is an error based on some measure of data fidelity, $\tau(\alpha)$ measures the complexity of the model, and λ controls the degree to which accuracy or parsimony are preferred.

2.1. The measurement vector

We measure the silhouette of the scene in every image, each of which has a corresponding projection matrix \mathbf{P}^p (where $p \in \{1, \dots, P\}$). We assume that these projection matrices are available a priori, or calculable from the image set. In the case of the experiments presented in Section 4, the \mathbf{P}^p were estimated from the image sets using standard camera calibration software [22],

The silhouette of the scene in image p is flattened into a vector \mathbf{y}^p . We then form the S dimensional binary vector \mathbf{y} by concatenating the \mathbf{y}^p for $p = 1, \dots, P$.

The advantage of silhouettes as a cue is that they only depend on shape. There is no reliance on texture to generate identifiable feature points, on an active sensor to generate a point cloud, or the interaction with a light source to generate usable shading variations. Thus, although information is lost in the converting images to silhouettes, this information does not need to be modeled in our reconstruction. Most importantly, however, the silhouettes of components of a model can be composited in order to form the silhouette of the whole, without requiring complex occlusion models.

2.2. The templates

Each *template* \mathbf{T}_t represents a particular 3D shape (or template type) placed in a specific location and orientation. These templates are the elements from which the structure estimate is constructed. In the case of the Lego example each template represents a particular type of block with a specific 3D position and orientation.



Figure 3. Generating a template vector π requires rendering a shape into each of the input images and calculating its silhouette in each.

Each template \mathbf{T}_t is rendered using each projection matrix \mathbf{P}^p to produce a corresponding synthetic image \mathbf{I}_t^p . The silhouettes of these images are concatenated into a vector π_t of length S . There are T such binary vectors π_t , and these make up the columns of the matrix $\mathbf{\Pi} \in \mathbb{Z}_2^{S \times T}$. The columns of matrix $\mathbf{\Pi}$ thus represent a basis of template silhouettes from which we aim to construct the true image silhouette according to Equation (1).

Note that we need only render each component individually, rather than in every possible combination. As such

the number of templates scales linearly with the number of components rather than combinatorially with the complexity of the scene being modeled.

3. Structure recovery as linear programming

Due to noise in the images, and the fact that no real set of templates is likely to perfectly explain the shape of all suitable scenes, the error term of (1) is always expected to be non-zero. A number of suitable error functions have been proposed in the literature, but our concern is whether a combination of template silhouettes are as close as possible to the silhouette of the original scene. A natural choice is to optimise

$$\arg \min_{\alpha \in \Omega} \|\mathbf{y} - \mathbf{\Pi}\alpha\|_1 + \lambda \|\alpha\|_1, \quad (2)$$

where λ is a scale factor controlling the degree to which the method focuses on reconstruction error or parsimony. It is worth noting here that we expect α to be extremely sparse, as only a very small subset of templates will be used in any single structure estimate. Note also that α remains a binary vector.

3.1. Compositing silhouettes

In practice, the silhouettes of the various template shapes in a structure estimate inevitably overlap. The fact that these silhouettes are composited linearly in (2) means that pixels in overlapping areas of the reconstructed silhouette $\mathbf{\Pi}\alpha$ are no longer necessarily less than or equal to 1. This means even in the absence of noise that $\|\tilde{\mathbf{y}} - \mathbf{\Pi}\tilde{\alpha}\|_1 \neq 0$ where $\tilde{\mathbf{y}}$ and $\tilde{\alpha}$ represent the true measurement and structure descriptor respectively.

This problem does not apply to those pixels falling outside the silhouette of the original images, however, as in this case $y = 0$ and any deviation from this value would indicate an erroneous estimate. We thus separate the elements of \mathbf{y} into two vectors, \mathbf{y}_0 and \mathbf{y}_1 corresponding to the 0 and 1 elements of \mathbf{y} respectively. We similarly define $\mathbf{\Pi}_0$ and $\mathbf{\Pi}_1$ as being composed of the columns of $\mathbf{\Pi}$ corresponding to the 0 and 1 elements of \mathbf{y} respectively. Noting that $\mathbf{y}_0 = \mathbf{0}$ we see that $\|\mathbf{\Pi}_0\alpha\|_0$ counts the pixels corresponding to a model α which fall outside the original image silhouette.

Having concentrated the composition problem on the elements of \mathbf{y}_1 we introduce a slack variable of the same length which we label ξ . This allows us to penalize image silhouette pixels that are not accounted for by α without penalizing those accounted for more than once. Combining this with equation (2) we get

$$\begin{aligned} \arg \min_{\alpha \in \Omega, \xi} & \|\mathbf{\Pi}_0\alpha\|_1 + \|\xi\|_1 + \lambda \|\alpha\|_1, \\ \text{s.t. } & \mathbf{y}_1 - \mathbf{\Pi}_1\alpha \leq \xi, \quad 0 \leq \xi \leq 1, \end{aligned} \quad (3)$$

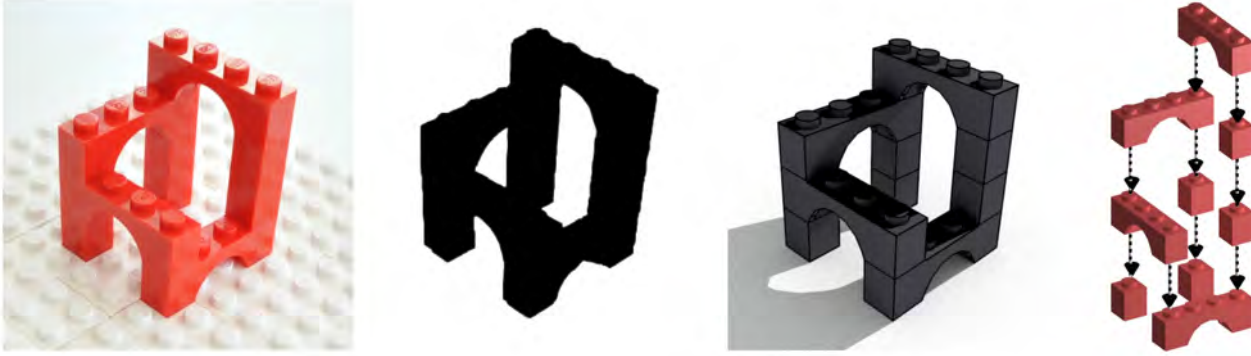


Figure 4. Lego Escher arches: estimating structure, rather than just shape, provides semantic information such as how scenes are constructed. In this example the structure estimate calculated on the basis of 4 silhouettes allows an analysis of the components of the scene and how they fit together.

Adding the variable ξ increases the state space for the optimization, but this does not have a significant impact on complexity as the appropriate value for ξ is easily identifiable.

3.2. Preventing self-intersection

To ensure that structure estimates are constrained to the set of physically plausible structures we want to guarantee that the templates selected do not intersect. This can be achieved by adding a constraint that

$$\mathbf{\Gamma}\alpha \leq \mathbf{1} \quad (4)$$

where each row of $\mathbf{\Gamma}$ represents a single mutual exclusion constraint. That is if \mathbf{T}_i intersects \mathbf{T}_j we add a row to $\mathbf{\Gamma}$ with elements i and j set to 1. The matrix $\mathbf{\Gamma}$ is thus in $\mathbb{Z}_2^{\Gamma \times T}$ where Γ is the number of mutual exclusion constraints. The matrix $\mathbf{\Gamma}$ is very sparse as each template intersects with a very small proportion of other templates. In practice Γ scales with $O(Tr/d) = O(T)$, where r is the maximal radius of a block and d the density of candidate blocks in the scene; or more informally, adding blocks around the edges does not increase the number of intersections for a block in the middle.

3.3. Physical Plausibility

We can ensure that the structure is physically consistent, and that each block is supported by another block directly below by adding the constraint:

$$\mathbf{\Upsilon}\alpha \leq \mathbf{0} \quad (5)$$

where each row of $\mathbf{\Upsilon}$ represents a single support requirement. Where template \mathbf{T}_i requires v units of support (or Lego terms ‘studs of support’) from the set $\{\mathbf{T}_k\}_{k \in \mathcal{K}}$ then we add a row to $\mathbf{\Upsilon}$ with element i set to v and elements $k \in \mathcal{K}$ set to $-v(k, i)$, the amount of support that \mathbf{T}_k offers to \mathbf{T}_i . The matrix $\mathbf{\Upsilon}$ formed in this manner is sparse, with

each row containing at most $O(r/d)$ elements (by a similar argument to 3.2). The matrix also contains fewer than T rows as the bottom templates are assumed to be supported by the ground

3.3.1 Dimensionality reduction by projection

The remaining vector \mathbf{y} is still large and consequently the problem in (3) is too large to solve efficiently. Given the sparsity we expect of α , however, we are able to project the problem to a lower dimensional space and solve it there.

In practice we apply dimension reduction by pre-multiplying the measurement and reconstruction by a matrix $\mathbf{\Phi} \in \mathbb{R}^{D \times S}$ where $D \ll S$ is chosen to ensure that the relative distances will be preserved by the projection with high probability. Note that the matrix $\mathbf{\Phi}$ is constructed once, and is common to all templates.

The Johnson-Lindenstrauss Lemma asserts [1] that a set of n points in any Euclidean space can be mapped to a Euclidean space of dimension $M = O(\epsilon^{-2} \log N)$ so that all distances are preserved up to a multiplicative factor between $(1 - \epsilon)$ and $(1 + \epsilon)$. A variety of such mappings based on pre-multiplication by a $M \times N$ matrix (where $M \ll N$) have been proposed, including that of Matoušek [12] which employs a sparse matrix with non-zero entries chosen randomly from $\{1, -1\}$.

To enforce the slack variable inequality in (3) we require that the projection matrix $\mathbf{\Phi}$ contains only non-negative elements. This means that the Johnson-Lindenstrauss Lemma does not strictly apply, and that the corresponding guarantees are not available to us. Testing in Section 4 allows us to evaluate the impact of the loss of theoretical guarantees, and to determine which forms of projection matrix are most suitable.

3.4. Optimization

Limiting the plausibility constraint from Sec. 2 to the self-intersection and support constraints detailed above allows us to write (3) as a mixed-integer optimization problem

$$\begin{aligned} \arg \min_{\alpha \in \mathbb{Z}_2^T, \xi} & \|\Pi_0 \alpha\|_1 + \|\xi\|_1 + \lambda \|\alpha\|_1, \\ \text{s.t.} & \Phi \mathbf{y}_1 - \Phi \Pi_1 \alpha \leq \xi, \quad \Gamma \alpha \leq \mathbf{1}, \quad \Upsilon \alpha \leq \mathbf{0}. \end{aligned} \quad (6)$$

In practice we use GUROBI [8] to find the optimum of the following equivalent problem

$$\begin{aligned} \arg \min_{\alpha \in \mathbb{Z}_2^T, \xi} & \sum \mu \alpha + \sum \xi + \lambda \sum \alpha, \\ \text{s.t.} & \nu - \Psi \alpha \leq \xi, \quad \Gamma \alpha \leq 1.5, \quad \Upsilon \alpha \leq 0, \quad \xi \geq 0. \end{aligned} \quad (7)$$

where $\mu = \Pi_0^T \mathbf{1}$ represents the column sums of Π_0 , $\Psi = \Phi \Pi_1$, and $\nu = \Phi^T \mathbf{1}$.

Algorithm 1 Algorithm overview

Require: Input images, Matrices $\Phi, \Psi, \Gamma, \Upsilon$
 Calculate target image silhouette, and vectorize
 Eliminate columns of Ψ falling outside image silhouette
 Solve Equation (7) for α

The time required to find the optimum of (6) depends largely upon T (the number of templates used) and D (the number of rows in Φ). Decreasing D decreases the probability that the optimum of (6) will correspond to that of the version of the problem where $\Phi = \mathbf{I}$. Culling unnecessary templates, however, has no impact upon the quality of the solution. The simplest culling is achieved by removing from consideration those templates with silhouettes extending significantly beyond that of the real scene. This typically achieves an order of magnitude reduction in the number of columns in Ψ in our testing.

4. Experimental testing

The models used for synthetic testing were randomly generated Lego constructs that satisfy the self-supporting constraints of Section 3.3. Unless otherwise specified, synthetic tests used 4 images (of 140×105 pixels) per scene, 20 blocks per scene, and each reported result represents an average over 50 tests. The default parameters to the method were $\lambda = 10^{-3}$, $D = 2 \times 10^3$, and $T = 500$. Note that the probability of randomly identifying the correct 20 blocks from a set of 500 possibilities is less than 10^{-53} ! To simulate the effect of calibration errors we add multiplicative noise to the synthetic camera position and rotation parameters, and report the standard deviation of the zero-mean Gaussian distribution from which the noise is sampled. Figure 5 provides an illustration of the magnitude of the noise

added (which is otherwise difficult to grasp). Note that, when 0.5 magnitude camera noise is present many of the block silhouettes differ from the truth by approximately half their width.



Figure 5. A synthetically generated test scene, and two images showing the magnitude of the noise added to the synthetic image formation process. The images were generated by adding together 10 silhouettes of the scene, each of which had noise of 0.3 or 0.5 (respectively) added to the camera parameters.

Figure 6 illustrates the impact that varying the number of input images used has upon the fraction of blocks correctly recovered. Note that the number of measurements (D) used is the same for all tests. Figure 7 shows the im-

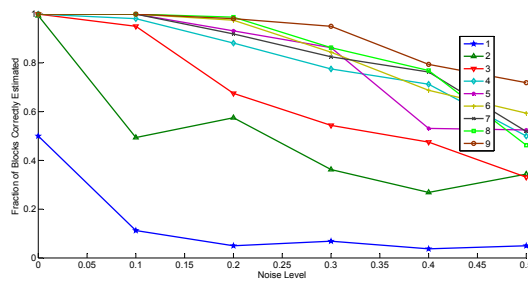


Figure 6. Fraction of blocks correctly recovered for different numbers of input images at differing noise levels.

part of varying the weighting parameter λ on the accuracy of the recovered structure estimates. Performance is close to optimal for a very wide range of values of λ .

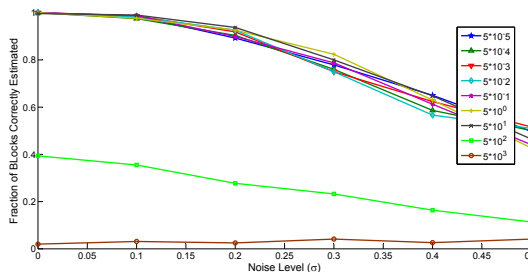


Figure 7. The fraction of blocks correctly recovered for values of λ between 5×10^{-5} and 5×10^3 .

Three forms of the matrix Φ which projects from silhouette space to measurement space were evaluated, two deterministic and one random. The first deterministic form is effectively the horizontal concatenation of identity matrices (note that the matrix Φ is not square) which we label *Diagonal*. This has the effect of allocating subsequent pixels in

Noise(σ)	0.0	0.1	0.2	0.3	0.4	0.5
Diagonal	0.978	0.933	0.683	0.530	0.315	0.228
Stepped	1.000	0.988	0.925	0.765	0.575	0.477
Rand(10^{-4})	0.998	0.850	0.610	0.475	0.377	0.267
Rand(10^{-3})	0.998	0.900	0.747	0.567	0.395	0.255
Rand(10^{-2})	0.980	0.807	0.592	0.490	0.347	0.322

Table 1. The fraction of true blocks estimated for two deterministic forms of the projection matrix Φ , and 3 randomly sampled projection matrices. The density of the randomly sampled matrices is reported in brackets.

the (vectorized) silhouette into subsequent elements of the vector \mathbf{y} . The second deterministic form is effectively the opposite of this, in that it groups subsequent pixels of the silhouette into the same element of \mathbf{y} . The matrix Φ in this case is zeros except for a continuous stepped diagonal row of ones. We label this form *Stepped*. We also evaluate a random projection matrix motivated by [12] where the elements of Φ are sampled independently from $\{0, 1\}$ such that the density of the 1’s may be specified. Table 1 details the results of these tests, and shows that the *Stepped* form of Φ performs significantly better than its competitors, particularly at higher noise levels. We also evaluated a modified form of the matrix proposed by Indyk and Motwani[9] where the elements of Φ are sampled independently from $\mathcal{N}(0, 1)$, a zero-mean Gaussian distribution with standard deviation 1, but with inferior results.

Figure 8 shows that, at least for models of 20 pieces visible in 4 images, increasing the number of measurements (that is D) beyond 1225 has little impact on performance. The fact that the method performs well for even relatively

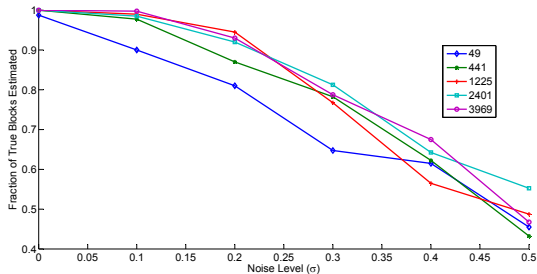


Figure 8. The fraction of blocks correctly recovered for varying numbers of measurements D . All tests were performed using the *Stepped* form of Φ .

small numbers of measurements D suggests the images contain significant redundancy, as would be expected. The fact that performance ceases to increase significantly for D beyond 2,401 suggests that there is little to be gained by larger numbers of measurements. This might be taken to indicate that, although we cannot avail ourselves of the guarantees provided by the Johnson-Lindenstrauss Lemma (as explained in Section 3.3.1), it seems that the solution to the projected problem is the same as that of the original prob-

lem with high probability. One potential avenue of exploration is whether learning a projection from the data might (see [19, 11] for example) provide a better approach.

Noise(σ)	0.0	0.1	0.2	0.3	0.4	0.5
$W = 10^{-1}$	1.000	0.610	0.442	0.407	0.310	0.240
$W = 1$	1.000	0.880	0.802	0.655	0.587	0.440
$W = 10^1$	1.000	0.857	0.572	0.430	0.307	0.272

Table 2. The fraction of true blocks recovered for various values of the weighting parameter W applied to the component of the cost function representing the background pixels.

By separating foreground and background elements of the silhouette in equation (3) each component can be weighed differently. It was believed that this might be advantageous in the presence of significant camera noise to reduce the weighting of background elements in order to allow blocks to more easily extend beyond the boundaries of the silhouette, for example. Table 2, however, shows this not to be the case, as even for high levels of camera noise re-weighting always decreases accuracy.

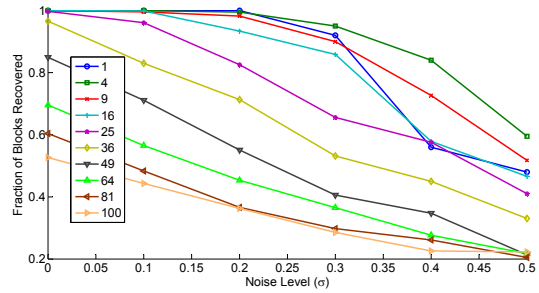


Figure 9. Fraction of true blocks correctly estimated for varying numbers of blocks per model, and at varying noise levels. Note that the zero noise results reflect the fact that the randomly chosen set of blocks that make up the test scene are often not the most parsimonious explanation of its silhouette.

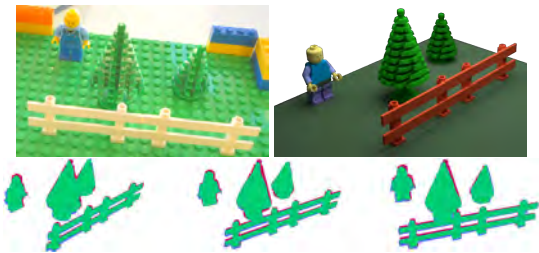


Figure 10. *Top left*: One of 8 input images. *Right*: Novel view of generated model. *Bottom*: A visualization of the reconstruction projected back into the first 3 input images. The projection of the reconstruction is shown in red, and the original silhouette in blue. Green indicates overlap. Nine template types were used in the test.

4.1. Real images tests

Lego objects. We initially report the performance of the method in recovering the set of blocks used to construct a

real Lego model. This is intended to demonstrate that the method works well in real scenes when the template types accurately reflect the components from which the scene is constructed.

Figures 1 and 4 relate the performance of the method in estimating structure from real images. In the real image testing the pictures were taken with a hand-held consumer SLR camera, and resized to 640×480 pixels. Forming the matrix Ψ required of the order of 4 minutes which includes rendering time. The Lego template types were generated using LeoCAD² which in turn uses the models from LDraw.org³. The templates represent different blocks at all possible positions and rotations within the bounds of the space within which the model may appear. To restrict the number of templates to a manageable number only those with silhouettes which overlap the real scene silhouettes are considered. For all real image tests $\lambda = 4 \times 10^{-4}$.

In Figure 4 4 images were used and 2 template types. The 1×1 block could appear in 196 positions horizontally and was modeled up to 4 layers high, resulting in 784 templates. The arch had 121 possible positions (horizontally) at each of 2 orientations, which at 4 levels generated 968 templates. The total problem thus had 1,752 templates, which was reduced to just over 500 after culling. A total of $D = 1,727$ measurements were used, and solving the system required approximately 3 minutes of processing time. The timing information is approximate as much of the process was parallelized and calculating equivalent serial timings necessarily involves estimation.

In the case of Figure 1 four 3D template types were used, the figure, the windscreen, the wheel pairs, and a 6×2 block. The number of templates after culling was 861. Rendering, segmenting and projecting the templates took approximately 6 minutes, and solving the system approximately 4 minutes.

General scenes. Figures 15 and 16 demonstrate the performance of the method on a set of general objects to illustrate its performance in the case where the template shapes do not accurately reflect the construction of the real scene. This is interesting because it illustrates the ability of the method to decompose a general scene into 3-dimensional semantic units. All of the tests on general scenes are based on images taken for other purposes, which is intended to illustrate its applicability in general circumstances.

In all cases the rendering of the silhouettes took around half an hour, and solving the optimization problem approximately the same amount of time. It is possible to solve much more complex models by eliminating the support constraint, but the resulting models are much less convincing, and interesting.

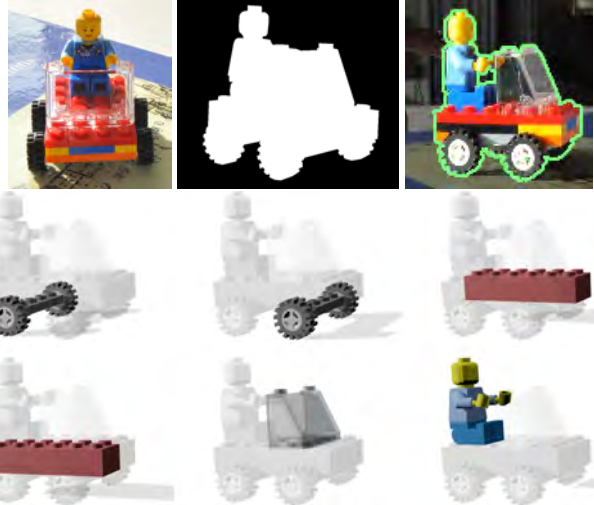


Figure 11. One of 12 input images for the second car test (cropped), one of the calculated silhouettes, the reprojection of the ground truth back into one of the images, and rendering of the estimated structure. Three times as many templates types as required were used in the test (shown in Figure 2), leading to over 3,500 templates. The result is stable for $\lambda \in [0, 2 \times 10^{-3}]$.



Figure 12. Overlaid ground-truth and reconstruction silhouettes from the ‘Temple of the Dioskouroi’ sequence from Middlebury. Red indicates common silhouette, while blue indicates missing parts of the ground-truth silhouette, and yellow parts of the rendered silhouette that do not match ground-truth. See reconstruction in Figure 13.



Figure 13. **Left, center:** Two views of the Lego ‘Temple of the Dioskouroi’ reconstruction corresponding to the silhouettes in Figure 12. **Right:** A reconstruction using coarser blocks

5. Conclusion

The method we have described recovers a structural explanation of the shape of a scene from a set of silhouettes. It is applicable in cases where structure can be described in terms of a set of building blocks, and where this set contains hundreds of thousands of elements. Experimental testing has shown robustness to noise, and the general object tests show it can recover plausible structures even when the object in question was not constructed from the set of building blocks that we wish to reconstruct it from.

The primary limitation of the method lies in its use of parsimony. It tends to produce models which accurately re-

²See <http://www.leocad.org/trac>

³See <http://www.ldraw.org/>

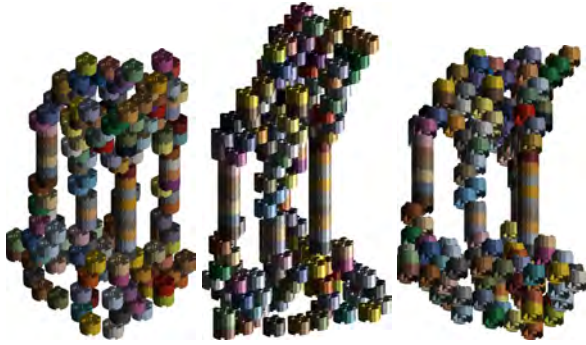


Figure 14. Three views of the ‘Temple of the Dioskouroi’ as constructed purely from cylindrical Lego blocks. Here the coarse resolution and limited composability of cylindrical blocks make stair-casing artifacts inevitable.



Figure 15. The two input images of the Archway sequence from [22], their silhouettes, and those of the reconstructed model, followed by two rendered images of the model.



Figure 16. The 2 images of the Eiffel Tower and their silhouettes. The first input image is from Reuters taken by Charles Platiau, the second from Wikipedia, taken by Stefan Krause. Each silhouette was used twice, on the basis of the symmetry of the tower.

construct the silhouettes of the input images, but which have holes in them when viewed from other angles. Developing an additional regularization term reflecting the assumption that none of the images are taken from a particularly special angle remains further work. We also aim in future to learn



Figure 17. Three views of the reconstruction achieved using the images from Figure 16. With so many blocks stacked vertically the support constraints are complex, and leads to a very long optimization process (over 2 hours on a 32 core machine).



Figure 18. Three views of the reconstruction achieved using the images from Figure 16, without support constraints. The optimization in this case took less than 2 minutes, but has many holes and unsupported blocks.



Figure 19. Two views of a model constructed from ‘The Dinosaur Sequence’ of Wolfgang Niem, University of Hannover.

a dictionary of template types from sufficient examples of pre-existing scenes.

One of the most interesting features of the method is that it optimizes over combinations of 3D building blocks, and to this extent is reasoning in 3D about the problem of understanding scenes from image sets. It might thus offer step towards reasoning about the relationships between general objects in images.

References

- [1] R. Arriaga and S. Vempala. An algorithmic theory of learning: robust concepts and random projection. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 616–623, 1999. 4
- [2] J. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. Dick. Interactive modelling for ar applications. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 199–205. IEEE, 2010. 2
- [3] M. Brand, P. Cooper, and L. Birnbaum. Seeing physics, or: Physics is for prediction. In *Proc. Physics-based modeling in computer vision*, 1995. 2
- [4] R. A. Brooks, R. Creiner, and T. O. Binford. The acronym model-based vision system. In *Proceedings of IJCAI*, pages 105–113, 1979. 2
- [5] A. W. Fitzgibbon, D. W. Eggert, and R. B. Fisher. High-level cad model acquisition from range images. *Computer-Aided Design*, 29:321–330, 1997. 2
- [6] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. In *ICCV*, pages 641–647 vol.1, 2003. 2
- [7] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 1, 2
- [8] I. Gurobi Optimization. Gurobi optimizer reference manual, 2014. 5
- [9] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. 6
- [10] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000. 2
- [11] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *International Conference on Machine Learning (ICML'13)*, Atlanta, USA, 2013. 6
- [12] J. Matoušek. On variants of the Johnson-Lindenstrauss lemma. *Random Struct. Algorithms*, 33:142–156, September 2008. 4, 6
- [13] P. Müller, G. Zeng, P. Wonka, and L. V. Gool. Image-based procedural modeling of facades. *ACM. Transactions on Graphics*, 26(3):85, July 2007. 2
- [14] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3):#43, 1–11, 2008. 2
- [15] L. Roberts. *Machine perception of 3D solids*. PhD thesis, Stanford, 1965. 1, 2
- [16] C. Russell, J. Fayad, and L. Agapito. Energy based multiple model fitting for non-rigid structure from motion. In *CVPR*, pages 3009–3016. IEEE, 2011. 2
- [17] C. Russell, R. Yu, and L. Agapito. Video pop-up: Monocular 3d reconstruction of dynamic scenes. In *ECCV 2014*, pages 583–598. Springer, 2014. 2
- [18] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–757 vol.2, oct. 2003. 2
- [19] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1562–1569. IEEE, 2013. 6
- [20] G. Taubin. Estimation of planar curves, surfaces, and non-planar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. PAMI.*, 13(11):1115–1138, Nov. 1991. 2
- [21] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. PAMI*, 13:992–1006, October 1991. 2
- [22] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. Torr. Videotrace: rapid interactive scene modelling from video. In *ACM Transactions on Graphics (TOG)*, volume 26, page 86. ACM, 2007. 3, 8