https://doi.org/10.2352/ISSN.2470-1173.2017.17.COIMG-452

3D Shape Recovery From Real Images Using a Symmetry Prior

Vijai Jayadevan[1], Aaron Michaux[1], Edward Delp[1] and Zygmunt Pizlo[2];
[1]**Department of Electrical and Computer Engineering, Purdue University; West Lafayette, IN;**
[2]**Department of Psychological Sciences, Purdue University; West Lafayette, IN**

Abstract

3D shape recovery from a single camera image is an ill-posed inverse problem which must be solved by using a priori constraints (a.k.a priors). We use symmetry and planarity constraints to recover 3D shapes from a single view. In many ways, symmetry and planarity represent the simplicity of a 3D object, and by applying these constraints we attempt to reconstruct a simple 3D shape that can explain the 2D image. Once we assume that the object to be reconstructed is symmetric, all that is left to do is: (i) estimate the plane of symmetry, and (ii) establish the symmetry correspondence between the various parts of the object. The edge map of the image of an object serves as a representation of its 2D shape; establishing symmetry correspondence means identifying pairs of symmetric curves in the edge map. In this work, we assume that the vanishing point, which establishes the symmetry plane up to a scale factor, is known. In addition, we also assume that the focal length and the direction of gravity are known. We extract long smooth curves from the edge map by solving the shortest (least-cost) path problem, where the cost function penalizes large interpolations and large turning angles. We then find the optimal curve matches that minimize the number of planes required to approximate the final 3D reconstruction. This optimization problem is framed as a binary integer program and solved using the Gurobi solver [1].

Introduction

Recovering the 3D shape of an object from a single view is an ill-posed inverse problem and solving it requires the use of informative priors [2]. Symmetry and planarity constraints are used in this work, as they have already been identified as constraints used by the human visual system [3, 4]. Moreover, symmetry is ubiquitous in nature, as well as in man-made objects, and furthermore, many real world objects are piece-wise planar. The focus of this work is to reconstruct shape: a spatially global property of an object. Symmetry and planarity in many ways capture the essence of shape for many objects. With the abundance of images on the internet, an algorithm that can reconstruct shape from a single image will have many applications. Although there are many kinds of symmetry, in this work we always use the term symmetry to refer to bilateral (mirror) symmetry.

Recovering shapes from a single view has advantages compared to algorithms based on binocular disparity. Establishing symmetry correspondence (i.e., identifying which two pixels in the image are projections of 3D symmetric points) leads to more accurate reconstructions in comparison to establishing binocular correspondence, and then using binocular disparity for reconstruction. Reconstructions based on binocular disparity loose accuracy quickly as the distance between the object and camera increases. However, reconstructions based on symmetry are more robust to changes in distance. Moreover, 20 years ago Zabrodsky and Weinshall [5] showed that using a symmetry prior can substantially improve the accuracy of reconstructions from multiple views.

Symmetry has been used in the past for 3D reconstruction of objects and scenes; however, some of these methods [6, 7, 8] require extensive user intervention, like manually establishing symmetry correspondences. Methods described in [9, 10] concentrate on dense 3D reconstruction of scenes, rather than shape reconstruction from curves. Sinha et al. [11] considered the symmetric curve matching problem; however, their dynamic programming algorithm only works with restricted views of the 3D object. Xue et al. [12] used symmetry to obtain depth maps; however, they used synthetic images that have planar surfaces bound by straight lines. In our work, we are interested in estimating a 3D shape representation of the object in the scene. Though we use a planarity constraint, we are also interested in obtaining shape representations for objects with approximately planar surfaces. We accomplish this by using a planarity measure that counts the number of planes required to approximate the object. For instance, we can approximate the furniture shown in Figure 1 with four planes. Three of these planes are shown in Figure 1, and the fourth plane, not shown, is opposite to the orange plane. This concept is made more clear in the following sections. In the next section we provide an overview of the algorithm, and the following sections describe each step in detail. This is followed by results and conclusions.

Overview

We begin by defining the vanishing point which characterizes 3D mirror-symmetry. A vanishing point is a point on the image plane where the perspective projection of 3D parallel lines intersect (see Figure 2(a)). Therefore, a vanishing point represents a particular direction in the 3D world. Let a symmetry line be defined as a line connecting two 3D points that are symmetric with respect to a symmetry plane. Figure 2(a) shows a symmetric chair. The red lines are the projections of symmetry lines. By definition, all the symmetry lines for the object shown in Figure 2(a) are perpendicular to the plane of symmetry, and hence represent the same 3D direction. Therefore, there is a vanishing point in the image corresponding to that direction, let us call it V_P. If we imagine a 3D line with the same direction as the symmetry lines, and passing through the center of perspective projection, this line will intersect the image plane at V_P. Therefore, if we know the vanishing point we also know the direction of the normal of the symmetry plane. Two pixels, p_i and p_j, are said to symmetrically correspond if they are the projections of two 3D symmetric points. If the symmetry correspondence of all the pixels representing the object is known, and if the normal of the symmetry plane is known, then the object can be accurately reconstructed in three dimensions up to a scale factor. See [13, 14, 15] for a detailed explanation. The edge map of the image of an object is often a reasonable representation of its 2D shape, and so it can be used to establish symmetry correspondence by identifying pairs of points on the edge map that are projections of 3D symmetric points. Points that symmetrically correspond in an image must be co-linear with the vanishing point. However, as shown in Figure 2(b), more than two edge pixels can be co-linear with the vanishing point, and therefore, we need a way to discriminate correct and incorrect correspondences. It is advantageous to work with smooth curves (if such curves can be extracted), because it reduces the complexity of the problem (the number of curves is often much less than the number of edge pixels). Additionally, curves have shape which can be utilized. Therefore, in this framework, we establish symmetry correspondence for pairs of 2D curves, instead of just working with pairs of 2D points.

The first step to solving symmetry correspondence is estimating the position of the vanishing point in the camera image. Though there are several methods available in the literature for estimating vanishing points from monocular images [16, 17, 18], we use the estimates obtained by Michaux and Pizlo [19], because it is more reliable as it uses binocular information. Note that if one can identify higher order features like long curves or corners, it is also possible to partially solve symmetry correspondence without estimating the vanishing point first. Establishing correspondence for one or more pairs of features will lead to the vanishing point, which can then be used to solve correspondence for the remaining points and parts of the image. Once the vanishing point is known, the next step is extracting long meaningful curves, where the word meaningful implies that the curve would make sense to a human observer. We have evidence that the human visual system extracts long curves by solving the shortest (least-cost) path problem in the image [20]. We incorporate this method in our algorithm. Specifically, we minimize the cost of a path, where the cost is a weighted combination of the interpolations and turning angles. From now on, we use the term correspondence to denote a pair of 2D curves that symmetrically correspond to each other in the 3D representation. The next step is identifying some candidate correspondences and candidate planes (planes that could be used to approximate the 3D shape of the object). Though we could start off by assuming that any long curve extracted could correspond to any other curve, we use a few criteria to reject some unlikely correspondences from the list of all possible correspondences, resulting in what we refer to as candidate correspondences. Once we have the candidate correspondences, we evaluate which correspondences lead to a 3D shape recovery that can be approximated by a minimum number of planes. This is achieved by converting the problem into a binary integer program and solving it using the Gurobi solver [1]. Each of these steps are explained in detail in the next sections.

Curve extraction

The first step in curve extraction is edge detection. As mentioned earlier, the image edge map serves as a representation of the 2D shape of the object. The canny operator is used with an adaptive threshold to form an edge map. Connected components in the edge map are then identified and are broken down, based on gradient orientation, to get short pieces of curve. The idea

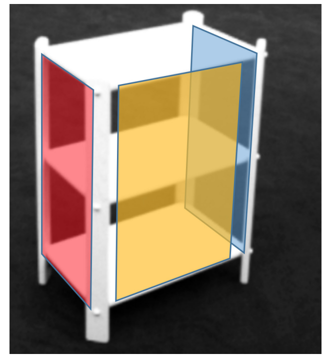
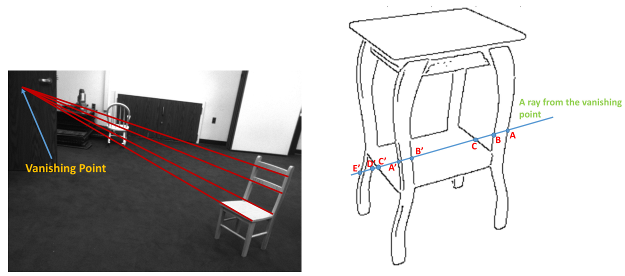


Figure 1. Planar approximation by using minimum number of planes.



(a) (b)

Figure 2. (a) Vanishing Point (b) Symmetry Correspondence Problem.

is to split the connected components at high curvature points, like junctions, to obtain short and smooth pieces of curves. Figure 3(a) shows short curve pieces obtained for the image of a piece of furniture. Longer curves are obtained by combining these short curve pieces. This is achieved by finding the shortest paths between all pairs of short pieces of curve with a cost function that penalizes spatial separation and large turning angles. To determine the turning angle and the spatial separation, the end points of the short curve pieces are first computed. The closest endpoints of two curves decides how the curves connect, which in turn decides the distance and the turning angle between them. For instance, consider curve combination a) in Figure 3(b). An approximate 145 ° turn is required to continue from the blue curve to the red curve. So what we are calculating is literally the turning angle. A point to note here is that when joining curves, rather than straight lines, the direction of the curve (used to calculate turning angle) is represented by a few pixels near the vicinity of the connecting end points. After the pairwise distances and turning angles are computed for all curve combinations, curve combinations with very high turning angles, or very large interpolated distances, are rejected. I.e., combining such curves is forbidden. The turning angle values and distances are then normalized separately, by subtracting the mean and dividing by the standard deviation. This can result in a negative cost for some curve combinations, so the absolute value of the minimum is added to avoid this. Turning angles are weighted one and a half times in comparison to distances. As shown in Figure 3(b), smooth curves are assigned lower costs. Although shortest paths between all pairs of short curve pieces are computed, we only use those whose cost is lower than a threshold in the next step. An example of such a low cost path is shown in

Figure 3(c). These long curves are then used to identify candidate correspondences and candidate planes.

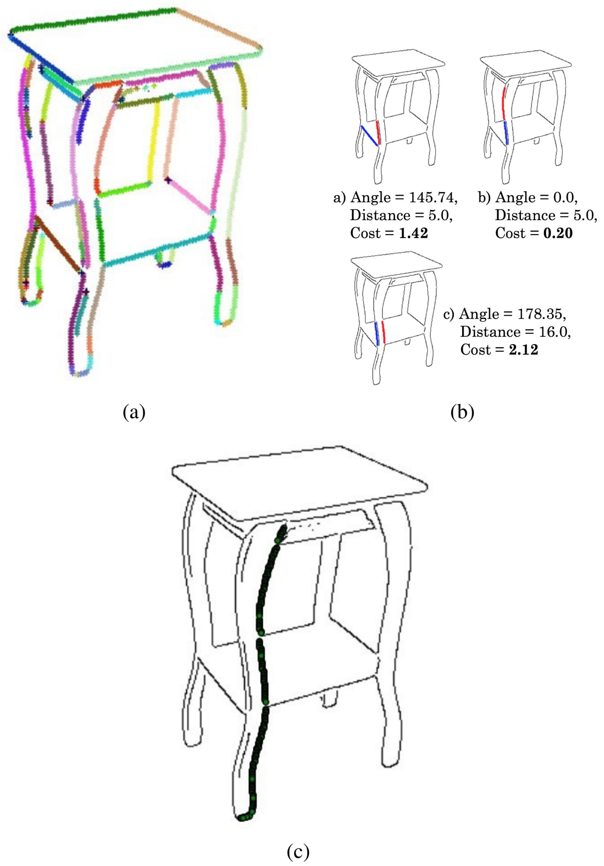


Figure 3. *(a) Different pieces of curves are represented by different colors. (b) Costs for combining short curves. (c) A low-cost long curve extracted by the shortest path algorithm.*

Identifying Candidate Correspondences and Planes

The idea, as mentioned earlier, is to view the correspondence problem as a curve matching problem. I.e., given a curve, say curve A, we would like to identify another curve from the set of extracted curves, which is the symmetrical counterpart of A. If curve B is the symmetric counterpart of curve A, then curve B is said to correspond to curve A. The set (A, B) is called a correspondence. We have a set of long curves at the end of the curve extraction step; however, we do not know the correspondences. The problem of identifying correspondences can be converted into a binary integer program (BIP). However, candidate correspondences must first be identified in order to formulate the problem as a BIP. Ideally, the candidate correspondences form a superset of which the correct correspondences are a subset. Let $s_1, s_2, \ldots, s_{N_c}$ represent N_c extracted long curves. Let S_A represent the set of all possible pairs of curves (correspondences), i.e., $S_A = \{(s_i, s_j) \mid i \neq j\}$. Most of the correspondences in this set are incorrect, and can be rejected based on criteria described later. The idea here is to select a set of correspondences, S_C, such that $S_C \subset S_A$, and ensure at the same time that the true correspondences are included in S_C. In order to accomplish this, we use two types of criteria. One type of criteria applies to curves in the 2D image, and the other applies to the 3D reconstruction of the curves.

2D and 3D Criteria

We use three 2D criteria when choosing candidate correspondences. The first two criteria deal with necessary conditions, and the third criterion is a heuristic. The first criterion used to judge whether a correspondence (s_i, s_j) should be a part of S_C, is to look at the overlap between s_i and s_j when viewed from the vanishing point. As shown in Figure 4(a), curves that truly correspond have a large overlap. Images of symmetric 3D curves have 100% overlap when viewed from the vanishing point, and therefore correspondences with low overlap can be safely rejected.

Shape dissimilarity between two 2D curves is another criterion for rejecting correspondences. As long as the 3D symmetric curves are approximately planar, their projected images have similar shape [21]. Psychophysical experiments show that when the 2D curves are arbitrarily different, then they are not perceived by observers as 3D or symmetrical [22]. The shape similarity is evaluated for polygonal approximations of the curves, where the polygonal approximations are obtained by sampling the curves using rays from the vanishing point (as shown in Figure 4(b)). Comparing the turning angles at each of the sampled points serves as a shape match metric, which can be used to decide whether a correspondence should be part of S_C. Images of two planar symmetric curves either always turn the same way, or always turn the opposite way at each sampled point [22]. For instance, the symmetric curves that are part of the object in Figure 4(a) turn the opposite way, while those in Figure 4(b) turn the same way. The signed turning angles are either subtracted or added, depending on whether the curves turn the same way or the opposite way. The ambiguity in whether turning angles are added or subtracted is resolved by counting the number of times that the curves turn in the same way, or the opposite way (at the sampled points), and then choosing the direction with the maximal count. This measure leads to a low shape cost for the images of planar symmetric curves, but not necessarily for non-symmetric curves.

For pairs of straight lines, the shape similarity criterion is ineffective, because lines always have zero turning angles, up to pixelation error in the image. In such cases their relative edge orientation can be used to choose candidate correspondences. I.e., corresponding straight lines usually have similar edge orientations. We use K-Means to cluster the edge orientations for the entire image into three clusters. We use three clusters because rectangular objects have three dominant directions. We then remove edge pixels that are more than one standard deviation away from their cluster center. This results in four clusters of edge pixels (as shown in Figure 4(c)). Three clusters correspond to the three cluster centers (shown in red, green and blue) and another set of unclustered edge pixels belonging to none of the clusters(shown in black). This is done to ensure that pixels with ambiguous edge orientations (i.e., their edge orientations cannot be assigned to any group with confidence) are not forced into being part of one cluster or another. When looking for candidate correspondences between two curves, we insist that of those pixels that are clustered, more than half of the pixel-wise correspondences belong to the same cluster.

There is one 3D criterion for choosing candidate correspondences: we assume that 3D curves are approximately planar. Therefore, the approximate planarity of a pair of curves can be used to decide whether or not to include or exclude a correspondence. To produce a planarity score, we first assume that the two curves correspond and reconstruct them in 3D. Planes are then fit using RANSAC. The goodness of the fit tells us if the 3D curves are approximately planar. Curve correspondences that produce highly non-planar 3D reconstructions can be rejected.

Identifying Candidate Planes

We derive a set of candidate planes, that are used to approximate the 3D object, from the set of candidate correspondences. First we reconstruct the 3D curve pairs from the candidate correspondences, and consider each curve pair in turn. These two 3D curves may be co-planar. In this case, we can fit a single plane to both curves and add it to the list of candidate planes. When curve pairs are not coplanar, we fit planes to both the 3D curves separately, and add them to the set of candidate planes.

As mentioned before, we assume that the approximate direction of gravity is known. It is reasonable to assume that most planes used to approximate a real world object are vertical [22], and this is useful for finding additional candidate planes. When the individual curves in a curve pair are 3D lines, then we also add the candidate vertical planes that pass through each 3D line. This is accomplished by minimizing the function: $\sum_{i=1}^{N}(a x_i + b y_i + c z_i - d)^2 + \alpha(a g_1 + b g_2 + c g_3)^2$, where $[x_i, y_i, z_i]$ represents the points on the 3D line, $[a, b, c, d]$ represents the plane we are seeking, $[g_1, g_2, g_3]$ represents direction of gravity, and α is a weight factor.

We now have a large set of candidate planes; however, it is very likely that many planes are redundant. Therefore, mean shift clustering is performed to reduce the number of planes. Once we have identified the candidate planes and candidate correspondences, we frame the problem of choosing the correct curve correspondences from S_C as a binary integer program as described in the next section.

Choosing the Correct Correspondences

Let $c_1, c_2, \ldots, c_N \in S_C$ be the set candidate correspondences, and $\pi_1, \pi_2, ..., \pi_M \in \Pi_C$ be the set candidate planes, identified in the previous steps. In the next step we identify, the symmetric correspondences, and the planes, used to approximate the 3D shape of the object. In other words, we choose a subset of the correspondences in S_C, and a subset of planes in Π_C, such that the resulting 3D reconstruction uses a minimal number of planes, while ensuring that a substantial portion of the object is still reconstructed. This problem can be formulated as a binary integer program (BIP).

The table on the left in Figure 5 shows the variables involved in the BIP. Each row represents curve correspondences, while each column represent candidate planes. Every variable $x^k{}_{ij}$ is a binary variable. Two things are implied when a variable $x^k{}_{ij}$ is set (i.e., when $x^k{}_{ij} = 1$). First, the curve correspondence i is chosen as part of the subset of true correspondences. Second, the 3D curve obtained from this reconstruction is assigned to plane j, which in turn means that plane j is chosen as one of the planes used to approximate the 3D object. As shown in Figure 5, each correspondence is repeated twice. This is because each correspon-

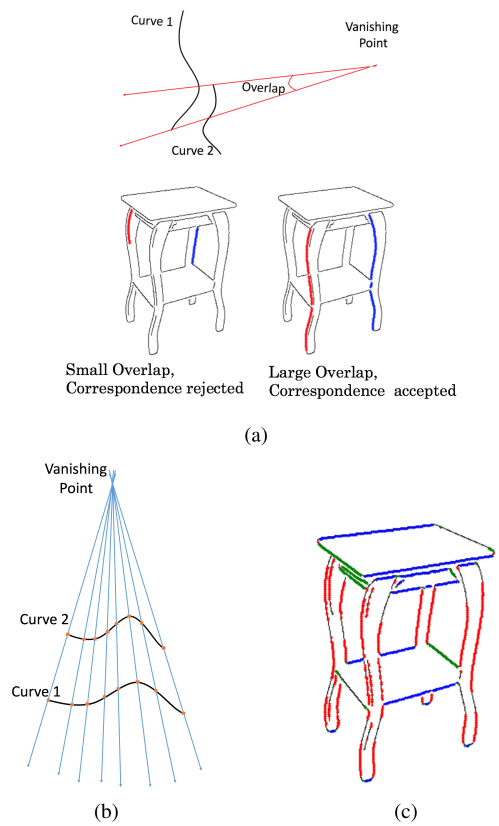


Figure 4. *(a) Overlap from Vanishing Point, (b) Polygonal approximation for the shape match metric, and (c) Clustered edge orientations.*

dence involves two 2D curves from which two 3D curves are reconstructed, and these two 3D curves need not be assigned to the same plane. That is, we must be able to assign a single correspondence to one or two separate planes, and hence, in the BIP, each correspondence is repeated twice. For example, let us say that the i^{th} correspondence, $c_i = (s_p, s_q)$, then $x^1{}_{im} = 1$ and $x^2{}_{in} = 1$ means that the 3D curve resulting from the reconstruction of 2D curve s_p, was assigned to plane π_m, and the 3D curve resulting from the reconstruction of the 2D curve s_q was assigned to plane π_n.

The table on the right in Figure 5 shows the cost associated with setting each variable $x^k{}_{ij}$. I.e., $w^k{}_{ij}$ represents the cost of choosing correspondence i, and assigning (associating) the k^{th} 3D curve resulting from the correspondence i to plane j. Here, $k \in \{1, 2\}$, and refers to either the first or second reconstructed 3D curve from correspondence i. The weight, or cost $w^k{}_{ij}$, consists of two terms, and is given by, $w^k{}_{ij} = exp(d^k{}_{ij}) + \beta\delta_i$. The first term is the exponential of the mean distance $d^k{}_{ij}$ of the k^{th} 3D curve (reconstructed based on the correspondence i) to plane j. The second term depends on the shape match δ_i (Figure 4(b)) between the two 2D curves involved in the correspondence, and is measured as the average difference in turning angles at the sampled points. Correspondences, whose reconstructed 3D curves are far from the plane, or whose shape match is poor, are removed. The remaining values are normalized, and then a weighted combination (represented by β) of the two terms is taken. The problem of picking the right correspondences and planes can now be framed as a constrained BIP as shown below.

$$\begin{aligned}
&\textit{minimize} \quad \boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{\mu}\boldsymbol{y}\\
&\text{subject to}\\
&\boldsymbol{x^T l} \geq \left(\frac{p}{100}\right) L && (C_1)\\
&\boldsymbol{Ux} \leq \boldsymbol{1_N} && (C_2)\\
&\boldsymbol{Rx} = \boldsymbol{0_N} && (C_3)\\
&\begin{bmatrix}\boldsymbol{E} & -\boldsymbol{I_M}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix} \leq \boldsymbol{0_M} && (C_4)\\
&\sum_{k=1}^{M} y_i \ \geq\ 2 && (C_5)\\
&x^k{}_{ij} \in \{0,1\}\ \forall\ i,j,k && (C_6)\\
&y_i \in \{0,1\}\ \forall\ i && (C_7)
\end{aligned}$$

where,

$\boldsymbol{x} = (x^1{}_{11}, x^1{}_{12}, .., x^1{}_{1M}, x^1{}_{21}, x^1{}_{22}, ..., x^1{}_{2M}, ..., x^1{}_{NM}, x^2{}_{11}, x^2{}_{12}, .., x^2{}_{1M}, x^2{}_{21}, x^2{}_{22}, ..., x^2{}_{2M}, ..., x^2{}_{NM})$

$\boldsymbol{w} = (w^1{}_{11}, w^1{}_{12}, .., w^1{}_{1M}, w^1{}_{21}, w^1{}_{22}, ..., w^1{}_{2M}, ..., w^1{}_{NM}, w^2{}_{11}, w^2{}_{12}, .., w^2{}_{1M}, w^2{}_{21}, w^2{}_{22}, ..., w^2{}_{2M}, ..., w^2{}_{NM}) \in \mathbb{R}^{MN}$

$\boldsymbol{y} = (y_1, y_2, ..., y_M)$, $\boldsymbol{l} = (l_1, l_2, ..., l_{MN}) \in \mathbb{R}^{MN}$

$\boldsymbol{U} \in \mathbb{R}^{N\times(NM)}$, $\boldsymbol{R} \in \mathbb{R}^{N\times(NM)}$, $\boldsymbol{E} \in \mathbb{R}^{M\times(NM)}$

$\boldsymbol{I_M}$ is the identity matrix of order M

$\boldsymbol{1_N} = (\underbrace{1,1,...,1}_{N\text{ elements}})^T$, $\boldsymbol{0_N} = (\underbrace{0,0,...,0}_{N\text{ elements}})^T$, $\boldsymbol{0_M} = (\underbrace{0,0,...,0}_{M\text{ elements}})^T$

$\boldsymbol{\mu} \in \mathbb{R}^M$, $\mu_i \in (0,\infty)\ \forall\ i$

The elements of vector $\boldsymbol{y}$ are binary variables which indicate whether a plane is selected or rejected. I.e., if $y_i = 1$ then plane i is selected as a plane that is used to approximate the 3D object. Constraint C_4 is devised to ensure that the elements of $\boldsymbol{y}$ are indicator variables for including or excluding planes. As mentioned before in Figure 5, the columns represent planes. Hence, if any of the variables $(x^k{}_{ij})$ are set in a column, say j, then constraint C_4 ensures that plane j is chosen and y_j is set. This can be achieved by insisting that:

$$\sum_{k=1}^{2}\sum_{i=1}^{N}\frac{x^k{}_{ij}}{2N} \leq y_j$$

The variable μ represents the cost for each included plane, thus biasing the solution towards fewer planes. Another important constraint that needs to be imposed is that if one of the variables in row i is set, then one variable in row $i+N$ has to be set too. This is because we cannot assign one 3D curve (from correspondence i) to a plane, and not assign the other 3D curve (from correspondence i) to a plane. This can be achieved by imposing the following constraint for row i:

$$\sum_{j=1}^{M} x^1{}_{ij} = \sum_{j=1}^{M} x^2{}_{(i+N)j}$$

Constraint C_3 imposes this condition for all the correspondences (half the rows). Hence matrix $\boldsymbol{R}$, used to represent this set of constraints, has N rows.

The trivial solution – setting $\boldsymbol{x}$ and $\boldsymbol{y}$ to the zero vectors – is not interesting, since there is no reconstructed shape. Constraint C_1 ensures that the trivial solution is not selected. Element $l_{(iM+j)}$ of vector $\boldsymbol{l}$ represents the length of the curves involved in correspondence i. Hence, the dot product $\boldsymbol{x^T l}$ gives the total length of all the curves that are part of the selected correspondences. Constraint C_1 ensures that this length is at least $p\%$ of L, where L is the total length of all the distinct curve pieces in the image.

Constraints C_2 ensure that the correspondences are unique. The idea here is that every edge pixel has a unique symmetric counterpart in the 3D object, and this constraint has to be explicitly imposed in our formulation of the problem. For instance, in Figure 6, correspondences (a) and (b) could be chosen at the same time, but correspondences (a) and (c) cannot, because of the angular overlap from vanishing point. For each correspondence, we can identify other correspondences that have a substantial angular overlap from the vanishing point. This information is used to add a constraint that only one among those with substantial overlap is selected. Constraints C_2 is a matrix where each row represents this constraint for a given correspondence.

Since, we do not expect a single plane to approximate any object, we ensure that at least 2 planes are selected by enforcing constraint C_5. We also add a preference for planes approximately parallel or perpendicular to the symmetry plane, by decreasing the weight, μ_i, of such planes to eighty percent of that of other planes.

We refer to straight line edges in the edge map that approximately pass through the vanishing point as self-symmetric lines. For example, in Figure 4(c), the blue lines are self-symmetric. They are referred to as self-symmetric because, the symmetric counterparts of points on such lines lie on the same line. Since we frame the problem as a curve matching problem, and since we do not expect any other curve/line to be the symmetric counterpart of a self-symmetric line, we remove these lines from the edge map prior to all processing. These lines can be fit later to the 3D reconstruction that was obtained from optimizing the BIP. The 3D orientation of these lines is the same as the normal of the symmetry plane. The neighborhood information from the 2D image is used to determine the exact position (and extent) of these lines in 3D. A small neighborhood of pixels in the 2D image, around the endpoints of the self-symmetric line, is considered. After the BIP optimization is complete, the 3D position of some of these neighborhood pixels may be available depending on whether the optimization process was able to find matches for them. If 3D positions are available for pixels in both neighborhoods (corresponding to both end-points of the self-symmetric line), then we consider all possible lines that go between points in one neighborhood with points in the other neighborhood. We then pick the 3D line that is most aligned with the normal of the symmetry plane, to obtain the 3D line corresponding to the self-symmetric 2D line.

As mentioned before, the BIP is solved using the Gurobi solver [1]. In order to account for occlusions in the image, we ask the algorithm to reconstruct only 70% of the pixels in the edge map. Specifically, the value of p in constraint C_1 is set to 70. If a large part of the object is occluded, this value will be too high, and the problem is infeasible. The value of p is initialized to 70, and is automatically decremented if the Gurobi solver detects that

Variables

	Plane 1 π_1	Plane 2 π_2	....	Plane M π_M
Correspondence 1 c_1	x^{1}_{11}	x^{1}_{12}	....	x^{1}_{1M}
Correspondence 2 c_2	x^{1}_{21}	x^{1}_{22}	....	x^{1}_{2M}
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	x^{1}_{N1}	x^{1}_{N2}	....	x^{1}_{NM}
Correspondence 1 c_1	x^{2}_{11}	x^{2}_{12}	....	x^{2}_{1M}
Correspondence 2 c_2	x^{2}_{21}	x^{2}_{22}	....	x^{2}_{2M}
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	x^{2}_{N1}	x^{2}_{N2}	....	x^{2}_{NM}

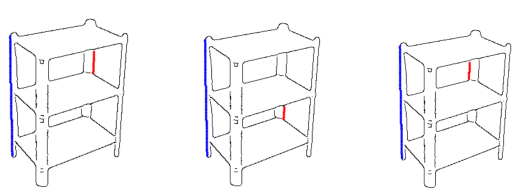
Cost

	Plane 1 π_1	Plane 2 π_2	....	Plane M π_M
Correspondence 1 c_1	w^{1}_{11}	w^{1}_{12}	....	w^{1}_{1M}
Correspondence 2 c_2	w^{1}_{21}	w^{1}_{22}	....	w^{1}_{2M}
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	w^{1}_{N1}	w^{1}_{N2}	....	w^{1}_{NM}
Correspondence 1 c_1	w^{2}_{11}	w^{2}_{12}	....	w^{2}_{1M}
Correspondence 2 c_2	w^{2}_{21}	w^{2}_{22}	....	w^{2}_{2M}
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	w^{2}_{N1}	w^{2}_{N2}	....	w^{2}_{NM}

Figure 5. *BIP formulation.*

the problem is infeasible. In practice it does not take more than a couple of attempts to find a feasible value of p.

Though a large number of correspondences obtained by the optimization process are correct, it was observed that a few mistakes were made. One of the reasons for the loss of accuracy comes from the clustering of candidate planes. In order to obtain better plane estimates, once the optimization process converges and a solution to the problem is obtained, the planes associated with the correspondences chosen by the optimization process are identified. Keep in mind that the candidate planes were obtained from candidate correspondences in the first place, and hence for each chosen correspondence, the planes it added/contributed can be identified. (I.e., the planes before clustering.) The optimization process is then rerun with these planes as candidate planes. When the process is rerun, we consider if the new set of candidate planes requires clustering. It may not, because we expect the new set of candidate planes to be much smaller than the initial set of candidate planes. Even when the number of planes is large enough to require clustering, the accuracy of the clustered planes should be much better. In practice, rerunning the optimization process with the new set of planes corrects some of the errors made in the first run.



(a) (b) (c)

Figure 6. *Correspondences (a) and (b) could be chosen simultaneously, but correspondences (a) and (c) cannot, because of the angular overlap from vanishing point.*

Results

Figures 7 and 8 show some of the results obtained. Some of these images were taken by us using a Point Grey Bumblebee2® stereo camera, and about half were obtained from the internet. For images taken with the Bumblebee2®, we estimated the vanishing point and the direction of gravity using the algorithm described in [19]. The image from the left camera was then used by our algorithm as input, along with the estimates for the vanishing point and the direction of gravity. The focal length and the principal point were read off of the camera's firmware. For the internet images we tried to obtain the estimates of camera calibration parameters and vanishing points using existing algorithms [16, 17, 18]. These estimates were not reliable. In fact, under the Manhattan world assumption [23], used by most of these algorithms, two of the three vanishing points correspond to the direction of gravity, and the normal of the symmetry plane. But it is difficult to apply to Manhattan world assumption to objects like E, K, and L, because not all the required vanishing points are salient in the images. This is perhaps the reason why the automatic vanishing point estimating algorithms find these images difficult to handle. Therefore, for such images from the internet, two vanishing points (representing orthogonal directions in 3D) were estimated by hand. Since, these vanishing points represent orthogonal directions in the 3D space, the focal length can be obtained if the principal point is assumed – typically the center of the image. The solution to $x_{V1}\,x_{V2} + y_{V1}\,y_{V2} + f^2 = 0$ gives the focal length, where (x_{V1}, y_{V1}) and (x_{V2}, y_{V2}) are the two estimated vanishing points, and f is the focal length. Though estimating the direction of gravity can be challenging, our algorithm only needs a crude estimate, and hence this is not a problem in general.

The results show that the algorithm was able to obtain a reasonably good representation of the 3D shape of the objects. The 3D shape representation is accurate, in most cases, if we take into consideration that the algorithm was not designed to take care of occlusions in the image. Shape is a spatially global property and so is symmetry and planarity. Therefore, enforcing these constraints should lead to good shape recovery for objects where such regularities (at least approximately) exist. As an example, for object L, the hind legs are occluded, but the algorithm reconstructs it from a wrong correspondence, and it is consistent with a good shape representation because of the regularities imposed on the 3D reconstruction. Similarly, the correspondences obtained for the top part of the hind legs (represented by red, brown and dark

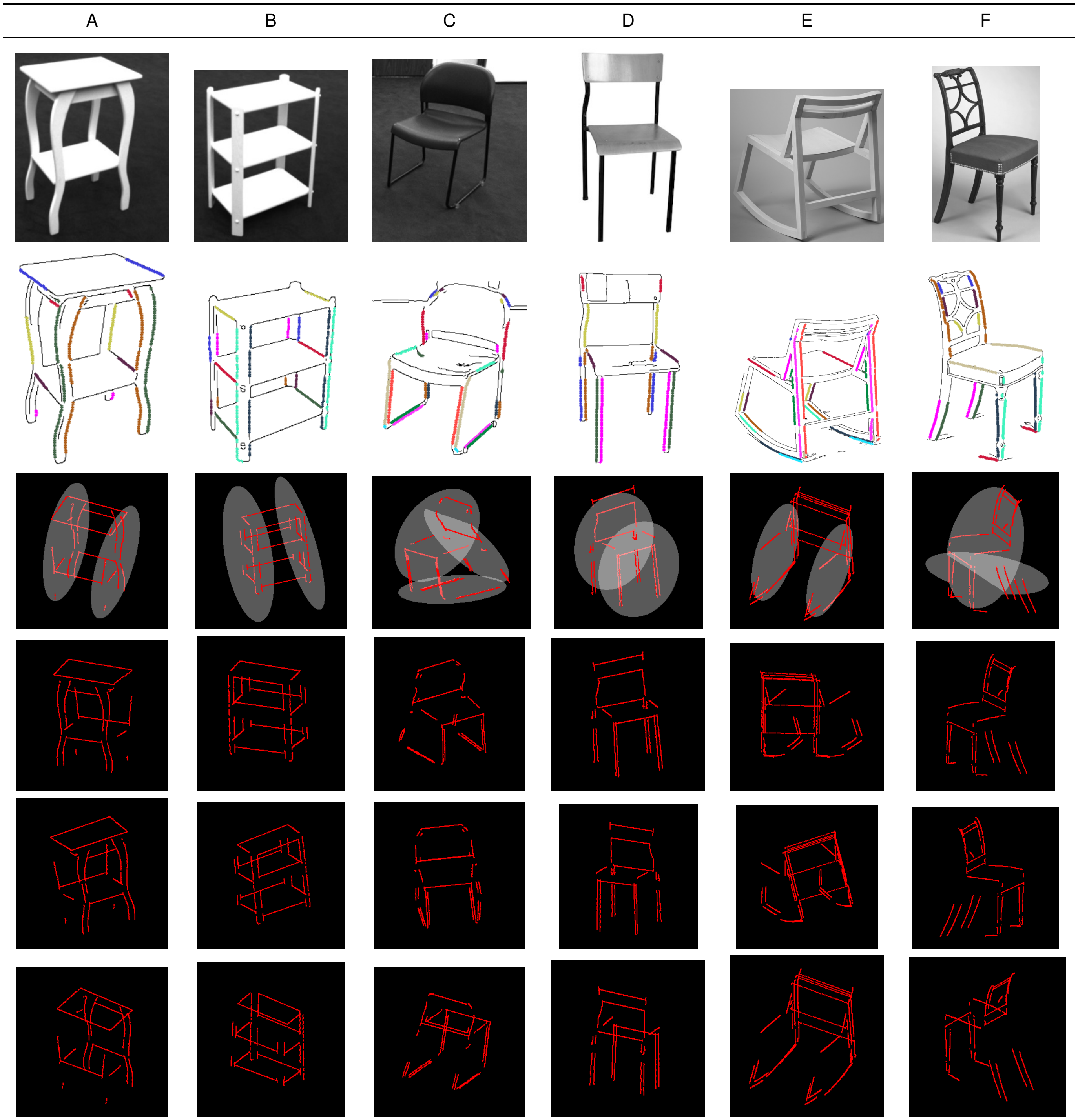


Figure 7. Results for objects A-F: Original Image is shown in row one, row two shows the symmetric correspondences detected with corresponding curves shown in same color, the planes selected are shown in row three, and rows four through six show three different views of the reconstructed object.

blue lines) for object L are not accurate, but the shape representation is still good.

One of the problems we faced while performing reconstructions, is that when 2D curves that are very close to each other are allowed to correspond, it often leads to bad reconstructions, as shown in Figure 9. To deal with this issue, we use a distance threshold to prevent close 2D curves from corresponding. The distance is measured as the median distance between corresponding points on the two 2D curves. To decide on the threshold, we first note that the shape of an object is defined mostly by curves

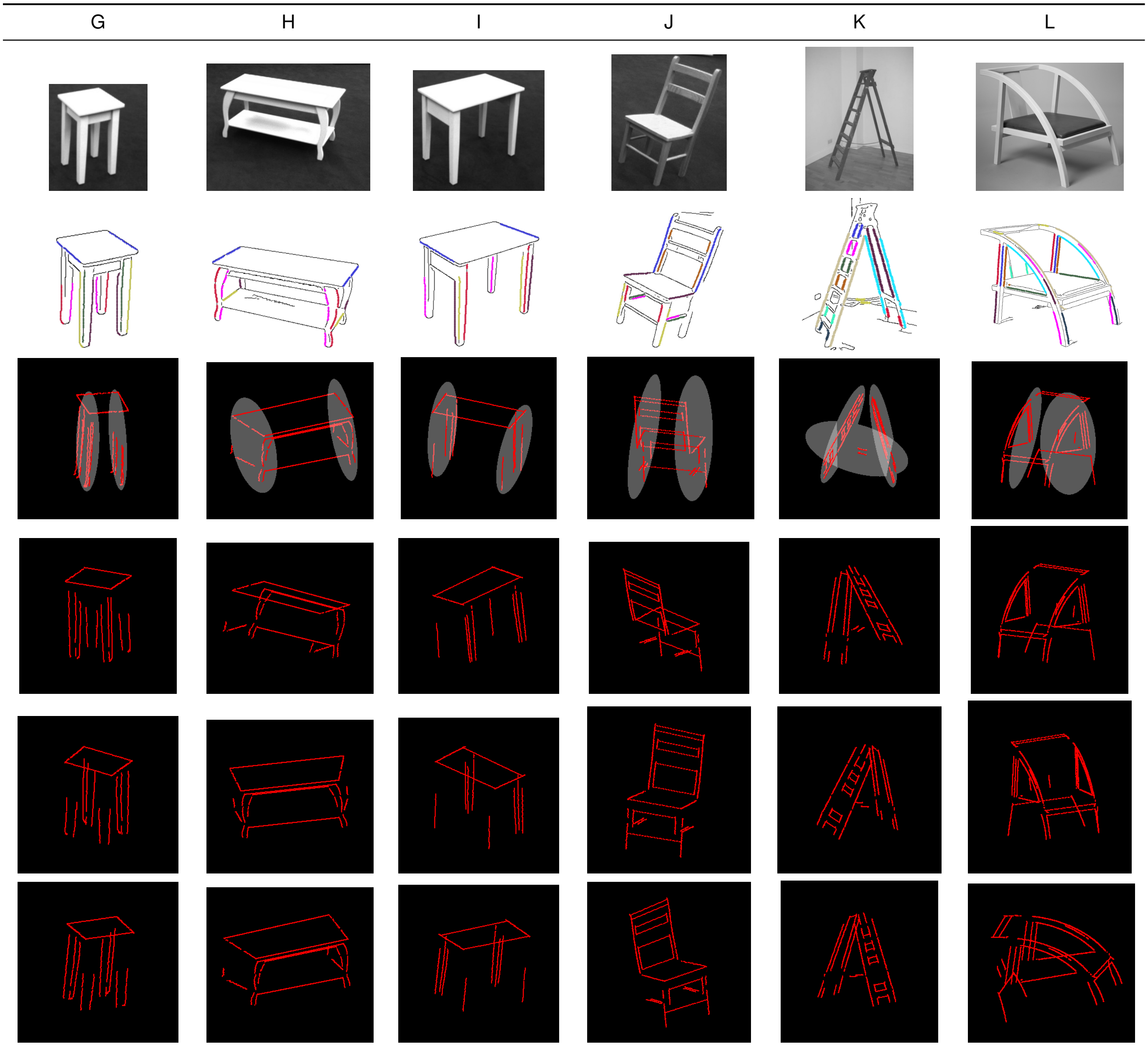
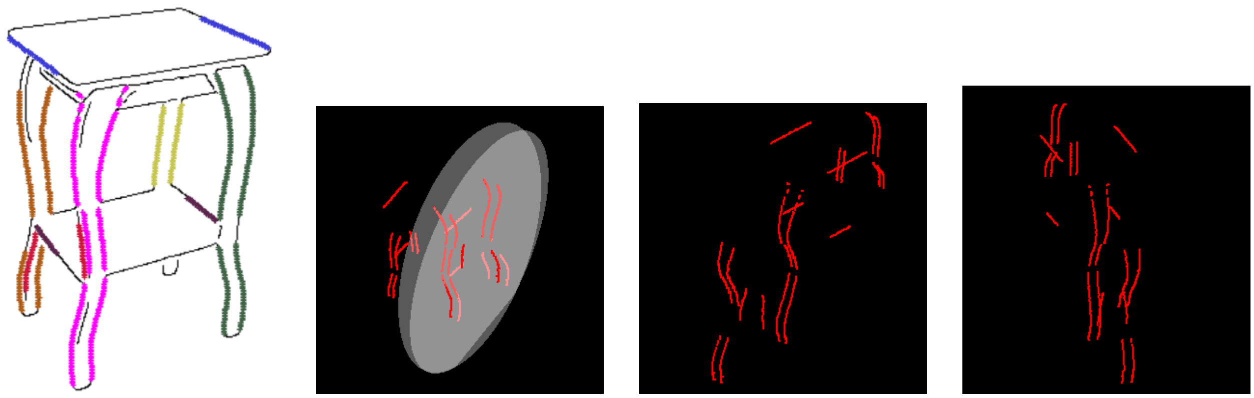


Figure 8. Results for objects G-L: Original Image is shown in row one, row two shows the symmetric correspondences detected with corresponding curves shown in same color, the planes selected are shown in row three, and rows four through six show three different views of the reconstructed object.

close to the 3D convex hull of the object. Hence, reconstructing 3D curves close to the hull is more important. Self-symmetric lines are used to dynamically decide this threshold. For non-degenerate views, the length of self-symmetric lines is a good estimate of the distance between curves that actually correspond. We choose a value of $0.4 l_{ss}$ as the distance threshold, where l_{ss} is the length of the longest self-symmetric line. This value works for all images except for object K, for which it had to be set to $0.2 * l_{ss}$. Using such a threshold means that some of the internal details of the shape of the object may not be reconstructed, as seen with object F. But this method can still capture the most important aspects of the 3D shape. A better solution to this problem is to view 3D shapes as composed of 3D parts, and correspondences should be found between the images of parts rather than the images of curves. The object in Figure 9 can be thought of as being composed of six parts: four legs, and two flat surfaces parallel to the ground. Parts either have symmetric counterparts, or they are self-symmetric. (In Figure 9, the legs have symmetric counterparts, and the flat surfaces are self-symmetric.) Dealing with parts, if they can be reliably detected, would simplify the problem and make it more robust. Moreover, some work [24, 25] towards part detection is already available and can be used.



(a) (b) (c) (d)

Figure 9. *(a) Wrong correspondences resulting from allowing to curves very close to each other to correspond, (b) the planes picked by the algorithm, (c) and (d) different views of the reconstruction.*

The algorithm runtime averages about 15 secs on a 2.8 GHz Intel® Core i7 quad core processor with 16 GB RAM. The code is written in Python. Though BIP is an NP-Hard problem, the number of variables involved in the BIP is usually not more than 2000. The Gurobi solver can easily handle such problems and usually converges within a second or two. Setting up the integer program accounts for the bulk of the runtime. There is ample scope for performance improvements, if it is a priority.

Conclusion

We have designed an algorithm that can effectively reconstruct 3D shapes from single camera images by employing symmetry and planarity priors. It demonstrates how these priors can be used to convert an ill-posed problem into a well-posed one. The results demonstrate the effectiveness of the idea of viewing the reconstruction problem as a constrained curve matching problem. By shape, some sort of regularity is implied, and by using regularities like symmetry and planarity we have successfully reconstructed simple shapes that can explain the given images.

References

[1] I. Gurobi Optimization, "Gurobi optimizer reference manual," (2016).

[2] T. Poggio and C. Koch, "Ill-posed problems in early vision: from computational theory to analogue networks," *Proceedings of the Royal Society of London B: Biological Sciences* **226**(1244), 303–323 (1985).

[3] M. W. Chan, A. K. Stevenson, Y. Li, and Z. Pizlo, "Binocular shape constancy from novel views: the role of a priori constraints," *Perception & psychophysics* **68**(7), 1124–1139 (2006).

[4] Y. Li and Z. Pizlo, "Depth cues versus the simplicity principle in 3d shape perception," *Topics in cognitive science* **3**(4), 667–685 (2011).

[5] H. Zabrodsky and D. Weinshall, "Using bilateral symmetry to improve 3d reconstruction from image sequences," *Computer vision and image understanding* **67**(1), 48–57 (1997).

[6] A. R. François, G. G. Medioni, and R. Waupotitsch, "Mirror symmetry? 2-view stereo geometry," *Image and Vision Computing* **21**(2), 137–143 (2003).

[7] H. Mitsumoto, S. Tamura, K. Okazaki, N. Kajimi, and Y. Fukui, "3-d reconstruction using mirror images based on a plane symmetry recovering method," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(9), 941–946 (1992).

[8] N. Jiang, P. Tan, and L.-F. Cheong, "Symmetric architecture modeling with a single image," in *ACM Transactions on Graphics (TOG)*, **28**(5), 113, ACM (2009).

[9] K. Köser, C. Zach, and M. Pollefeys, "Dense 3d reconstruction of symmetric scenes from a single image," in *Joint Pattern Recognition Symposium*, 266–275, Springer (2011).

[10] C. Wu, J.-M. Frahm, and M. Pollefeys, "Repetition-based dense single-view reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 3113–3120, IEEE (2011).

[11] S. N. Sinha, K. Ramnath, and R. Szeliski, "Detecting and reconstructing 3d mirror symmetric objects," in *Computer Vision–ECCV 2012*, 586–600, Springer (2012).

[12] T. Xue, J. Liu, and X. Tang, "Symmetric piecewise planar object reconstruction from a single image," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2577–2584, IEEE (2011).

[13] T. Sawada, Y. Li, and Z. Pizlo, "Any pair of 2D curves is consistent with a 3D symmetric interpretation," *Symmetry* **3**(2), 365–388 (2011).

[14] Y. Li, T. Sawada, L. J. Latecki, R. M. Steinman, and Z. Pizlo, "A tutorial explaining a machine vision model that emulates human performance when it recovers natural 3d scenes from 2d images," *Journal of Mathematical Psychology* **56**(4), 217 – 231 (2012).

[15] A. Michaux, V. Jayadevan, E. Delp, and Z. Pizlo, "Figure-ground organization based on three-dimensional symmetry," *Journal of Electronic Imaging* **25**(6), 061606–061606 (2016).

[16] J. P. Tardif, "Non-iterative approach for fast and accurate vanishing point detection," in *2009 IEEE 12th International Conference on Computer Vision*, 1250–1257 (2009).

[17] B. Li, K. Peng, X. Ying, and H. Zha, *Simultaneous Vanishing Point Detection and Camera Calibration from Single Images*, 151–160. Springer Berlin Heidelberg, Berlin, Hei-

delberg (2010).

[18] Y. Xu, S. Oh, and A. Hoogs, "A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 1376–1383 (2013).

[19] A. Michaux and Z. Pizlo, "Two correspondence problems easier than one," in *Computational and Mathematical Models in Vision (MODVIS)*, (2015).

[20] T. Kwon, K. Agrawal, Y. Li, , and Z. Pizlo, "Spatially-global integration of closed, fragmented contours by finding the shortest-path in a log-polar representation.," *Vision Research* (In press).

[21] T. Sawada, T. Li, and Z. Pizlo, "Detecting 3-d mirror symmetry in a 2-d camera image for 3-d shape recovery," *Proceedings of the IEEE* **102**(10), 1588–1606 (2014).

[22] Z. Pizlo, Y. Li, T. Sawada, and R. Steinman, *Making a machine that sees like us*, Oxford University Press, New York, NY (2014).

[23] J. M. Coughlan and A. L. Yuille, "Manhattan world: Orientation and outlier detection by bayesian inference," *Neural Computation* **15**(5), 1063–1088 (2003).

[24] T. Lee, S. Fidler, A. Levinshtein, C. Sminchisescu, and S. Dickinson, "A framework for symmetric part detection in cluttered scenes," *Symmetry* **7**(3), 1333–1351 (2015).

[25] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," *International journal of computer vision* **104**(2), 117–134 (2013).

Author Biography

Vijai Jayadevan received his bachelor's degree from Cochin University, India in 2008 and his master's degree from The University of Arizona in 2013, both in Electrical and Computer Engineering (ECE). He is currently working toward his PhD in ECE at Purdue University and is being advised by Prof. Edward Delp from the ECE Department and Prof. Zygmunt Pizlo from the Department of Psychological Sciences. His research interests include computer vision, computer graphics, machine learning, and signal processing.