Predicting High Resolution Image Edges with a Generic, Adaptive, 3-D Vehicle Model

Matthew J. Leotta and Joseph L. Mundy
Brown University
Provdence, RI, USA
{mleotta,mundy}@lems.brown.edu

Abstract

In traffic surveillance applications a good prior model of vehicle shape and appearance is becoming increasingly more important for tracking, shape recovery, and recognition from video. The usefulness of 2-d vehicle models is limited to a fixed viewing direction; 3-d models are nearly always more suitable. Existing 3-d vehicle models are either generic but far too simple to utilize high resolution imagery, or far too complex and limited to specific vehicle instances. This paper presents a deformable vehicle model that spans these two extremes. The model is constructed with a multi-resolution approach to fit various image resolutions. At each resolution, a small number of parameters controls the deformation to accurately represent a wide variety of passenger vehicles. The parameters control both 3-d shape and appearance of parts that deform in the 2-d manifold of the vehicle surface. These parts are regions representing windows, headlights, taillights, etc. *The combination of part boundaries and surface occluding contours account for the most consistent edges observed in images of vehicles. It is shown that the model parameters can be recovered by fitting the deformable model to real images of vehicles.*

1. Introduction

Road vehicles are one of the most important subjects in computer vision applications today. They are surpassed only by human subjects. Three-dimensional (3-d) approaches to tracking, shape recovery, and recognition of vehicles are becoming increasingly important as cameras become ubiquitous and objects are observed from a variety of viewpoints. Prior models of vehicles are also extremely beneficial since image projection leaves many ambiguities about a scene. This paper describes such a 3-d prior model for the shape and appearance of vehicles (See Figure 1.) and uses it to recover vehicle shape and pose from real images.

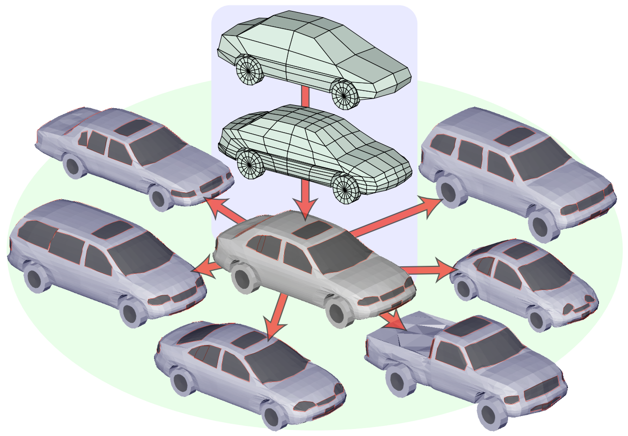
Figure 1. The multi-resolution deformable vehicle model with parts. The center column shows the mean vehicle shape for the first two mesh resolutions and shape with appearance for the third. The bottom ring depicts a variety of deformations in shape and appearance at the third resolution obtained by varying only 10 model parameters.

Over the past two decades, many authors [6, 7, 9, 10, 13, 16] have used a simple polyhedral vehicle model similar to that shown in Figure 2(a). This model contains only about a dozen faces and in some cases is deformable to adjust to the shape of different vehicles. A deformable model is used to obtain an estimate of the vehicle shape. The shape estimate can be used for recognition. It is usually assumed that the edges of the polyhedron are sufficient to predict intensity edges in an image. Clearly this assumption holds only in low resolution images.

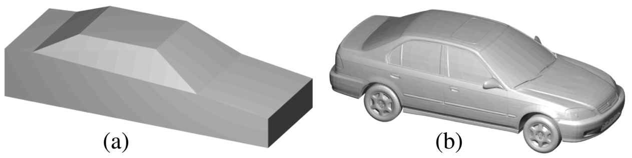
Figure 2. Meshes commonly used to model vehicles: (a) simple polyhedral with ∼12 faces, (b) CAD model with ∼80,000 faces.

In the same twenty year period that this simple model has been in use, resolution of video has increased dramatically. In other domains of model-based vision, such as human face [2] and body [1], the complexity of deformable 3-d models has grown accordingly. The complexity of deformable vehicle models has not kept pace.

To address the lack of detail in the simple polyhedral models, several authors [8, 12, 18] have opted for extremely high resolution CAD models (Figure 2(b)). These models have been designed for high quality rendering (*e.g.* for use in advertisements) and are poorly suited for use in most vision algorithms. Recognition algorithms must be run with each CAD model to find the one that best represents the data. It is intractable to test all CAD models, so a small number of CAD models from different vehicle classes are usually chosen as exemplars. Using multiple excessively complex exemplars is overkill, especially in the general case when the target vehicle is not one of the exemplars. A rigid mesh can only approximately represent a vehicle with different shape.

The goal of this paper is to bridge the gap between overly simple and overly complex. A 3-d deformable vehicle model (Figure 1) is presented that extends the generality and adaptability of the simple polyhedral model to the level of detail and accuracy of the CAD models. This model is fit to real images of vehicles by hypothesizing and matching image intensity edges. The fitting recovers the 3-d pose (rotation and translation) and vehicle shape. The proposed model is constructed at several mesh resolutions so that complexity can be adjusted for various image resolutions. In addition, a novel representation of appearance is proposed using deformable parts. These parts model regions of the vehicle surface (windows, headlights, *etc.*) whose boundaries likely generate intensity edges. The regions deform in a 2-d plane and are texture mapped to the 3-d vehicle surface. This approach frees the part geometry from being restricted to a fixed subset of the 3-d mesh edges. A combination of occluding contours and part boundaries projected into the image predict intensity edges.

The remainder of the paper proceeds as follows: Section 2 discusses additional related work. Section 3 shows how the model is represented and used to hypothesize image edges. Section 4 explains how the shape and pose parameters are estimated from images. Section 5 describes how the shape priors are learned from CAD models. Section 6 provides experimental results for fitting the model to images, and Section 7 draws conclusions and suggests future work.

2. Related Work

The method of fitting parameterized 3-d models to images falls into the framework put forth by Lowe [14]. Others have applied this technique to vehicles. Notably, Koller *et al.* [10] use the parameterized simple polyhedral model, and Ferryman *et al.* [6] extend this work by applying principal components analysis (PCA) to the parameter space. Projected polyhedron edges are aligned to detected image edges for fitting and tracking. Fitting accuracy is limited by the assumption that each visible polyhedron edge corresponds to an intensity edge produced by the vehicle.

On the opposite end of the spectrum, Guo *et al.* [8] use CAD models to match vehicle appearance over drastic viewpoint changes. Song and Nevatia [18] use CAD models to generate binary masks for vehicle detection. In recent work, Liebelt *et al.* [12] use CAD models to learn a sparse 3-d feature map for detecting vehicles and recovering approximate 3-d pose. The CAD models are used to render many training images from different viewpoints. These training images are processed to recover local feature points with descriptors. These features and descriptors are mapped back into 3-d and used to detect and localize vehicles in real images. This sparse model perfectly complements the deformable model in this paper. The sparse 3-d feature map provides an initial guess to the 3-d pose of vehicles. Initialization like this is needed for the gradient decent methods of this paper used to fit the deformable vehicle mesh to an image.

Leotta and Mundy [11] use vehicle models in the middle of the spectrum (hundreds of faces), but these are not parameterized. The models are constructed by hand using exemplar CAD models as guides.

3. Model Description

The vehicle model consists of several components: a 3-d mesh for vehicle shape, 2-d polygons for parts shape, a mapping of the 2-d parts onto the 3-d mesh, and a mapping from a low dimensional parameter space into high dimensional shape space. This section describes each of these components in detail and explains how they are used to hypothesize edges in an image.

3.1. Vehicle Mesh

The 3-d vehicle shape is represented by a polygon mesh with five components: one for the body and one for each of the four wheels. As shown in Figure 3(a), the mesh faces are primarily quadrilaterals with a few triangles. The complexity of the body mesh is only slightly greater than the polyhedral model in Figure 2(a). It features additional faces to represent front and rear bumpers and four wheel wells. This coarse 3-d mesh is called the template. As with the deformable simple polyhedral model, its vertices can be repositioned to capture the rough shape of sedans, minivans, sport utility vehicles (SUVs), pickup trucks, and various other passenger vehicles.

While the shape of the template mesh may change, its topology (*i.e.* number of faces and their connectivity) re-

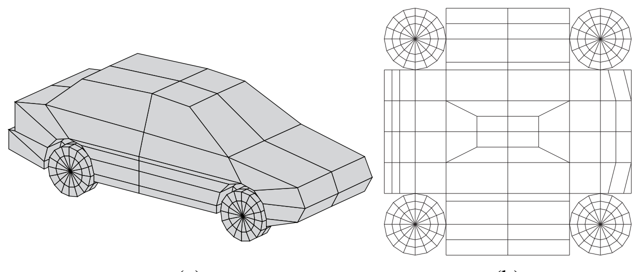
(a) (b)

Figure 3. (a) The template 3-d mesh and (b) the texture space mesh

mains the same. Keeping the same topology, the 3-d mesh can also deform into a plane (Figure 3(b)). Specifically, the template mesh is flattened into a unit square called the *texture space*. This concept comes from computer graphics where it is used to map intensities of texture images onto the surface of meshes for more realistic rendering. Here the texture map is used in a similar way to map 2-d geometry onto the vehicle surface. For simplicity, the texture map does not include the underside of the vehicle or inward sides of the wheels which are generally not observed. The connectivity of the remaining 2-d faces is the same as in 3-d. The texture coordinates of the template vertices are determined once and fixed. They are the same for all configurations of the corresponding 3-d vertices.

To represent vehicle shape at finer resolutions the template mesh is subdivided. Quadrilateral subdivision is applied by adding new vertices to the center of each edge and face and then splitting each face into quadrilaterals. This is the same process used in Catmull-Clark subdivision [4]; however, here the vertex repositioning equations are abandoned in favor of a data driven approach described in Section 5.1. Subdivision is also applied identically to the texture space mesh. New texture coordinates are added at their respective edge and face centroids while existing coordinates are not changed. In this paper only the vehicle body is represented at multiple subdivision levels. Subdivision is applied recursively as shown in Figure 4.

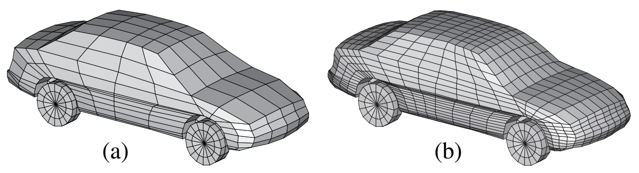
(a) (b)

Figure 4. The template mesh is subdivided to produce (a) the second resolution, and subdivided again for (b) the third resolution.

3.2. Vehicle Parts

Polygonal regions in the texture space describe the parts on the vehicle surface. Here *parts* refers to a collection of regions on the vehicle surface whose boundaries likely give rise to image edges. Parts are windows, headlights, taillights, grilles, hubcaps, and license plates. Each part is a 2-d polygon in the texture space with a fixed number of vertices that move freely to capture variations in shape. These parts are the 2-d analog of the 3-d mesh. Figure 5 shows an example of the mean parts in the texture space and after being mapped onto the mean vehicle surface. The notion of mean shape is clarified in the next section.

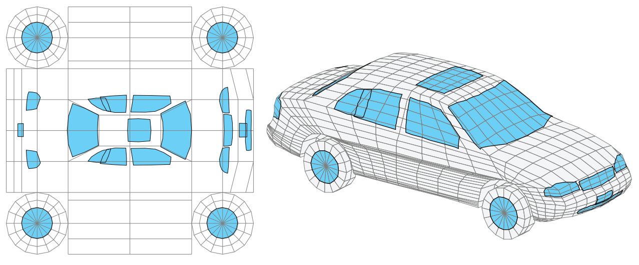
(a) (b)

Figure 5. Mean vehicle parts (a) in texture space and (b) mapped onto the mean vehicle surface.

A texture map is defined as an injective map from the deformable mesh surface to the texture space. The quadrilaterals at each mesh resolution are bisected into triangles to simplify texture mapping algorithms. Texture mapping is implemented using barycentric coordinates. Each 3-d surface point $\mathbf{p}$ is contained in one triangle i with vertices $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$. The point is uniquely represented as

$$\mathbf{p} = (1-u-v)\mathbf{a}_i + u\mathbf{b}_i + v\mathbf{c}_i, \qquad (1)$$

where (u, v) are the barycentric coordinates. Once u and v are found, (1) is applied again—this time in 2-d with $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$ being the 2-d vertices of corresponding triangle i in texture space. The resulting $\mathbf{p}$ is the texture mapped point. The texture mapping can be inverted as long as the texture space point lies within one of the mesh triangles.

It is not sufficient to map each part vertex into 3-d and connect by line segments. A straight line in texture space may cross several mesh faces, each oriented differently in 3-d. The line segment must be intersected with each mesh edge and each intersection point mapped into 3-d forming a chain of line segments.

3.3. Model Deformations

As mentioned earlier, the positions of both the 3-d mesh vertices and 2-d part vertices are adjusted to predict the shape and appearance of arbitrary vehicles. At the third resolution there are 1444 3-d vertices and 164 2-d vertices resulting in a 4660 dimensional shape space. A reduced parameter space is needed to make optimization practical. PCA provides the necessary reduction while simultaneously providing a prior distribution on the parameters.

Assume for now that the mesh and part vertices have been positioned to accurately represent each vehicle in a database of N vehicles. Section 5 will explain how these training data are learned. The average value each of the M parameters (4660 at the third resolution) is computed over the N training examples. The result is the mean mesh and mean parts shown in Figure 5. Next the means are subtracted from the training data which are stacked into a $M \times N$ matrix $\mathbf{D}$ of deviations from the mean. The singular value decomposition $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ provides the magnitudes ($\sigma_i = \Sigma_{ii}$ of diagonal matrix $\mathbf{\Sigma}$) and directions (columns of $\mathbf{U}$) of most significant variation. Parameters in the space of these principal components provide a linear combination of the N vehicles. One can interpolate between and extrapolate from the training data. To reduce the parameter space to m dimensions, the largest m singular values are retained and the rest are set to zero. By the properties of singular value decomposition, this m-d subspace captures the largest amount of the original variation. The $M \times m$ matrix $\mathbf{G}$ formed by the first m columns of $\mathbf{U}$ (assuming σ_i sorted largest to smallest) provides a mapping from a reduced parameter space point $\mathbf{p}$ to the full M-dimensional shape space point $\mathbf{q}$ via $\mathbf{q} = \mathbf{G}\mathbf{p}$.

The top m singular values can also be interpreted as the standard deviations of a multivariate normal distribution fit to the training data in the reduced parameter space. This normal distribution acts as a prior distribution on the shape and appearance of vehicles.

3.4. Generating Edge Hypotheses

Detected edges on images of vehicles arise from three main sources: 3-d surface geometry, albedo discontinuities, and specular reflections of the environment. Edges from the first two categories are predicted using the mesh and parts respectively. Edges from reflections are not consistent over varying viewpoints and are treated—along with background edges—as outliers.

Occluding contours of the 3-d mesh are used to predict edges from shape. If $\mathbf{n}_i$ is the normal vector of mesh face i and $\mathbf{v}$ is a vector in the viewing direction of the camera, then the occluding contour generator is defined as the set of all mesh edges with adjacent faces i and j such that $\mathbf{n}_i \cdot \mathbf{v}$ and $\mathbf{n}_j \cdot \mathbf{v}$ have different signs. Not all contour generator edges are visible in the image because of self occlusions. Hidden sections of the contour generator are removed by testing against a depth map rendered from the mesh. The green lines in Figure 6(c) show rendered occluding contours.

The vehicle parts are used to predict interior edges in a similar way. The part boundaries are mapped back from texture space onto the surface of the mesh and then projected into the image. As with the occluding contours, hidden lines are removed using a depth map (red lines in Figure 6(c)).

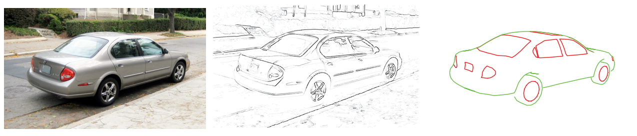

Figure 6. The edges from image (a) are shown in (b). Hypothesized edges from occluding contours and part boundaries are shown in (c) in green and red respectively.

4. Fitting to Images

Vehicles are difficult subjects for 3-d shape recovery from images. They are highly specular, have large textureless regions, and contain semi-transparent areas. Predicting image intensity at each pixel is intractable and would require a detailed model of the environment. Stable feature points are too sparse to fully constrain vehicle shape. Edges, on the other hand, are sufficiently dense and stable over changes in viewpoint and environment. Figure 6 indicates the population of edges that the deformable vehicle model can predict—both on the boundary and in the interior. It remains to explain how the model pose and shape are estimated from edges.

The complete parameter space now has either $m + 6$ dimensions (m shape, 3 translation, 3 rotation), or $m + 3$ dimensions (m shape, 2 translation, 1 rotation) when the vehicle is supported by a known ground plane. In either case, a Lie algebra representation is used for pose parameters. This is the same as in [5] except pose is relative to the vehicle instead of to the camera. The shape plus pose space is denoted m'-dimensional.

Given the mean vehicle shape and an initial guess for pose, the goal is to adjust the shape and pose parameters to minimize the distance between the hypothesized and actual image edges. For each hypothesized edge, a 1-d search in the normal direction finds all nearby edges. The perpendicular distance between each correspondence generates an error term. Following the framework of [14], the problem is formulated in terms the linear system:

$$\mathbf{J}\hat{\mathbf{x}} = \mathbf{e}, \tag{2}$$

where $\mathbf{J}$ is the Jacobian matrix and $\mathbf{e}$ is a vector of error terms. The solution, $\hat{\mathbf{x}}$, represents the incremental parameter updates needed to minimize the measured error. Each row i of $\mathbf{J}$ is the derivative of error function $e_i = E_i(\mathbf{x})$ which maps parameter space into a 1-d subspace of image coordinates for measurement i. Equation (2) is solved repeatedly using the iteratively reweighted least squares technique. The reader is referred to [14] and [5] for details on the fitting procedure, regularization with priors, and handling of outliers. Due to space limitations, only the computation of $\mathbf{J}$ is given here.

Computation of m'-d row vector $\mathbf{J}_i$ is easier when E_i is split into three composite functions

$$E_i(\mathbf{x}) = E_i^1(E_i^2(E_i^3(\mathbf{x}))). \qquad (3)$$

E_i^3 maps from m'-d parameter space into 3-d at the ith point, E_i^2 projects the 3-d point into 2-d image coordinates, and E_i^1 projects 2-d point into the edge normal direction. Given the $2 \times m'$ Jacobian $\mathbf{J}_i^2$ of $E_i^2 \circ E_i^3$, the final step is simply $\mathbf{J}_i = \mathbf{n}_i \mathbf{J}_i^2$, a linear projection with the 1×2 edge normal vector $\mathbf{n}_i$.

Computing $\mathbf{J}_i^2$ is more complicated because of perspective projection. Let $\mathbf{P}$ be the 3×4 homogenous projection matrix for image formation. The rows of the Jacobian of $\mathbf{P}$ evaluated at 3-d point $\mathbf{x}_i$ are

$$\mathbf{J}_k^{\mathbf{x}_i} = \frac{(\mathbf{m}_k^\top \mathbf{m}_3 - \mathbf{m}_3^\top \mathbf{m}_k)\mathbf{x}_i + \mathbf{m}_k^\top t_3 - \mathbf{m}_3^\top t_k}{(\mathbf{m}_3 \mathbf{x}_i + t_3)^2}, \qquad (4)$$

where $k = \{1, 2\}$, and $\mathbf{P} = [\mathbf{M}|\mathbf{t}]$ with $\mathbf{m}_k$ the row vectors of 3×3 submatrix $\mathbf{M}$ and t_k the elements of vector $\mathbf{t}$. Note that $\mathbf{J}^{\mathbf{x}_i}$ is independent of $\mathbf{x}_i$ if and only if $\mathbf{m}_3$ is the null vector (*i.e.* $\mathbf{P}$ is affine). Given the $3 \times m'$ Jacobian $\mathbf{J}_i^3$ of E_i^3, $\mathbf{J}_i^2 = \mathbf{J}^{\mathbf{x}_i} \mathbf{J}_i^3$.

Computation of $\mathbf{J}_i^3$ differs depending on whether point i is on an occluding contour or part boundary. If point i is fixed on vertex k of the 3-d mesh then $\mathbf{J}_i^3$ is simply the concatenation of $\mathbf{G}_k$ and the Jacobian of pose parameters. $\mathbf{G}_k$ is the $3 \times m$ submatrix of $\mathbf{G}$ (Section 3.3) corresponding to the 3-d mesh vertex k. The Jacobian of pose parameters is computed in the standard way for the SE(3) Lie Group [5]. Since occluding contour points are always on mesh edges, $\mathbf{J}_i^3$ is a linear interpolation of the $\mathbf{J}_k^3$ at the mesh edge endpoints. Part boundary points require barycentric linear interpolation of the three $\mathbf{J}_k^3$ at each corner of the triangle. However, this interpolation only accounts for the part motion due to 3-d shape. Another component in the tangent plane must be added to account for 2-d part shape deformation. The $2 \times m$ submatrix $\mathbf{G}_j$ corresponding to 2-d part vertex j provides this component in the texture space. Each column vector of $\mathbf{G}_j$ is mapped into 3-d using barycentric coordinates and added to $\mathbf{J}_i^3$.

Error terms from projections into multiple images can be minimized in a single linear system by stacking the rows of $\mathbf{J}$ and $\mathbf{e}$. If the cameras are calibrated with respected to each other, then a single set of vehicle pose parameters can be used. Figure 7 shows an example of hypothesized edges projected into multiple views before and after optimization.

5. Learning the Shape Priors

One key topic left to discuss is how to obtain the model deformations used in Section 3.3 to construct the PCA shape space. The template model must be deformed to accurately represent a database of N known vehicles. For the interpolated configurations to be meaningful, it is critical that all consistent local features be put into correspondence. To this end, the highly detailed CAD models are used as training data in a semi-automated algorithm for fitting both 3-d and 2-d vertices. Similar algorithms are applied separately to learning 3-d vehicle shape and to learning 2-d part shape. Each is addressed below.

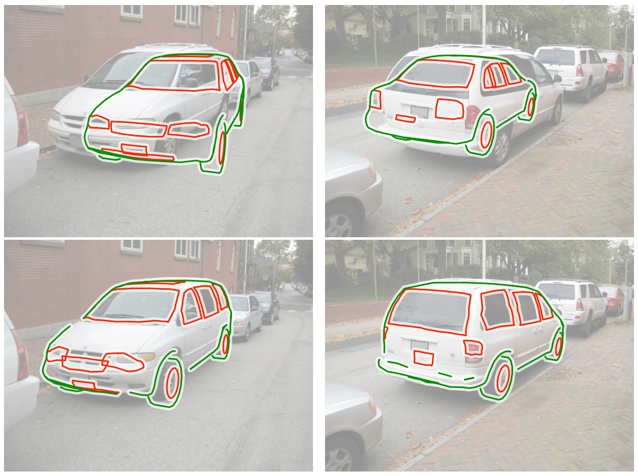
Figure 7. Two of six views of hypothesized model edges projected into images before (top) and after (bottom) fitting.

5.1. Learning Vehicle Shape

Learning starts with a CAD model as in Figure 8(a). The CAD model is stripped of protruding parts (side mirrors, spoilers, luggage racks, *etc.*) as these are not modeled. Wheels are also removed and are fit separately by hand. A simple cylindrical mesh is used with three parameters: width, outer diameter, and inner (hubcap) diameter. The remaining CAD model in Figure 8(b) is used for body fitting.

Finding correct correspondence between the template and CAD models is difficult due to large variations in shape across vehicle classes. There is often no well defined correspondence between a region in one class and another. To bootstrap the alignment, the coarsest scale template model is manually aligned to each CAD model to provide initialization (Figure 8(c)). The manual alignment does not cover the full degrees of freedom in the model. The initialized vertices are automatically pulled toward the CAD model surface like shrink wrap (Figure 8(d)).

Unlike the template mesh, the CAD models do not have an orderly structure. They contain many disconnected meshes of varying size and complexity. The meshes are frequently non-manifold, self-intersected, or contain small gaps. Disorderly meshes like these are commonly called "polygon soup". There is also a large variation in level of detail between different CAD models. To handle these problems the method Shen *et al.* [17] is employed to construct an implicit surface from polygon soup. The implicit surface is continuous and free of holes. It also provides a

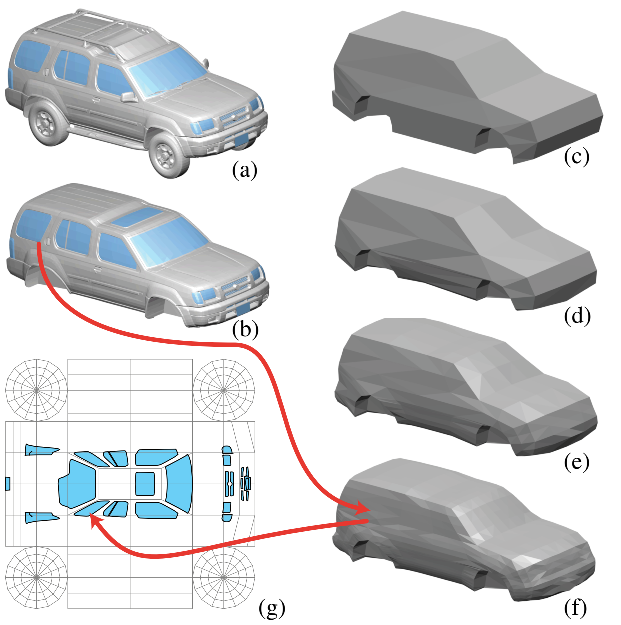

Figure 8. Fitting a mesh model and mapping parts: (a) the CAD model is (b) stripped of protuberances; (c) the template model is (d) fitted to the CAD body; (e,f) finer scales are derived by subdividing the template and refitting; (g) part boundaries are projected from the CAD model to the template and then mapped into texture space.

scale parameter, ϵ, which can be adjusted to smooth over excessive detail and better match the resolution of the template mesh.

To shrink-wrap the template to the CAD model, each template vertex follows the gradient of the implicit surface field function to the surface. Surface crossing are computing using numerical root finding—the same approach used for ray-tracing implicit surfaces [3]. A smoothing value of $\epsilon = 0.1$m is appropriate for the initial coarse mesh. Effectively, all CAD surface features of a size smaller than 0.1m are smoothed away. The shrink-wrapped template in Figure 8(d) provides initialization for the next finer resolution. The template is subdivided and ϵ is halved (doubling the resolution in each model). Figure 8(e) shows the result after fitting the finer mesh to the new implicit surface. This process may be applied repeatedly to produce finer scale models (Figure 8(f)). Two levels of subdivision provide enough detail for use in the experiments in this paper.

5.2. Learning Vehicle Parts

Once the 3-d shape has been learned, the 2-d deformable parts are learned on that surface. Luckily the parts are typically represented as disconnected meshes in the CAD models so that they can be assigned different material properties for rendering. Taking advantage of this fact, these parts are identified on the CAD models (shown in blue in Figure 8(a,b)) and their 3-d mesh boundaries are extracted. All boundary points are mapped into texture coordinates by a two step process as shown in Figure 8. First, a point on the CAD model is projected to the closest point on the shrink wrapped template mesh. Second, the point is mapped into texture space using barycentric coordinates.

Parts from each CAD model are projected into the texture space by way of the corresponding mesh deformations learned in Section 5.1. This collection of texture space contours forms the training data for learning deformable parts. Not all vehicle parts are found on all CAD models. For PCA, missing parts are replaced by the mean part shape.

The template parts are shown in Figure 9(a). Each template part is fit independently and automatically to the training data. Since the training parts are guaranteed to be closed and tend to be mostly smooth, implicit functions are not needed. Instead, shrink wrapping occurs by mapping template points directly to the closest point on the boundary of the training part using the dual space method of Ohtake and Belyaev [15].

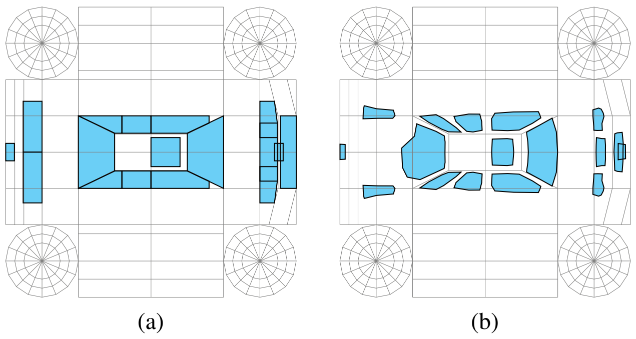

Figure 9. The initialized template parts (a) are fit to parts data in Figure 8(g) to produce the deformed template parts (b).

6. Experiments

The experiments presented here were devised to determine the optimal PCA subspace (*i.e.* select m) for fitting to images and to demonstrated the fitting accuracy and range of initial pose parameters that lead to a correct fit. In these experiments the model was trained on a database of $N = 79$ CAD models. The database includes 43 four-door sedans, 11 two-door sedans, 10 SUVs, 7 minivans, 6 pickup trucks, and 2 station wagons. The first 5 principal components account for 94% of the total variation and the first 10 account for 98%. A series of images was collected containing vehicles represented in the CAD model database. Images from multiple viewing directions were acquired for each vehicle and calibrated in a common world coordinate frame using standard techniques. For ground truth, CAD models were positioned in the coordinate frame by corresponding mesh vertices to image points. Fitting error is computed in 3-d as the root mean square (RMS) distance from body vertices on the fit model to the closest points on the shrink wrapped ground truth surface.

To select m, the model is fit to images of several vehicles while varying the dimensionality of the reduced parameter space. In these experiments, the pose parameters are held fixed at the true values, and shape is initialized with the ground truth mesh projected into parameter space. The results of shape optimization are shown in Figure 10. The error tends to decrease until the dimension reaches 5. After 5, the error levels off and in some cases increases due to overfitting. Hence, $m = 5$ is used in other experiments.

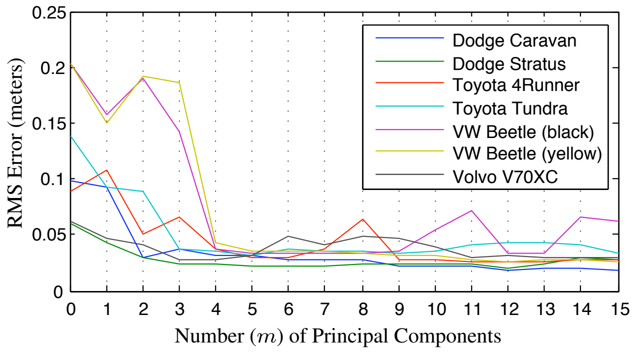

Figure 10. Results of fitting the model to various vehicle images and varying the number of principal components.

Figure 11 shows a typical fitting convergence result in the 2-d subspace of X translation and Z-axis rotation (the coordinate system aligns X with the driving direction and Z orthogonal to the ground). The plateau indicates that the algorithm converges to a similar low error solution for most initial poses near the true pose. When the initial pose is too far off, the number of edge mismatches is too high to overcome, and the algorithm converges to an incorrect optimum. In all cases, the initial shape parameters are those of the mean vehicle. A threshold of 0.1 meters on the RMS error defines the acceptable convergence region. This region is shown as the red, projected area in Figure 11. In the experiments below, the space of initial pose is explored along each of the six axes to estimate the approximate size of the convergence region. Table 1 gives the results of these experiments. The experiments show how error and convergence region size are affected by vehicle class, vehicle color, number of views, and number of pose parameters (*i.e.* use of a ground plane constraint). Figure 12 shows hypothesized edges for the converged results.

All experiments in Table 1 were run initially with 6 pose parameters. Those that failed (empty convergence region) are not shown in the table and were run again with 3 pose parameters. In each case, the ground plane constraint induced correct convergence. The black VW Beetle did not fail but had high error so results are shown with both 3 and 6 pose parameters (Figure 12 shows the bad result). Failure is most likely when using only one view, when the vehicle

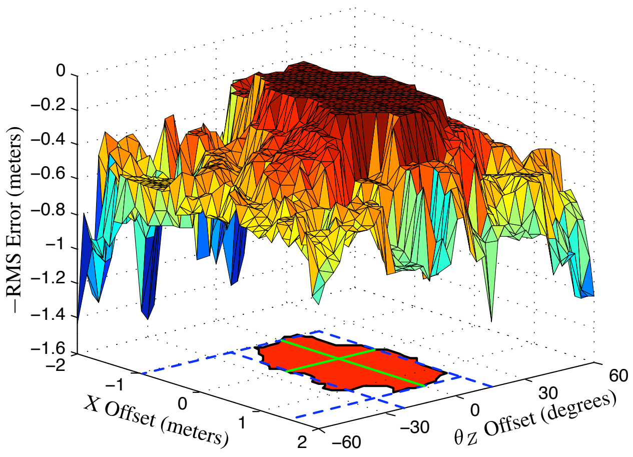

Figure 11. The region of convergence shown as a plateau in the function of (negative) RMS error. The plateau size is measured along the pose axes. This plot corresponds to row 1 of Table 1.

shape differs significantly from the mean, or when the vehicle color is very dark. The model is biased toward sedan shapes (68% of training vehicles are sedans) giving disadvantage to non-sedan shapes. This bias accounts for the spike at the tail of the Volvo wagon (there are only 2 wagons in the training data). The shape bias and dark color account for the very small convergence region of the black pickup. Convergence is improved with better shape initialization.

The images in these experiments had approximately 1-3 cm per pixel resolution on vehicles, similar to what is expected in high-definition vehicle surveillance applications. The third resolution of the vehicle mesh was used for all reported results. Experiments at other image and mesh resolutions showed similar results with a trade-off between fitting accuracy and convergence region size. This trade-off warrants future exploration of a pyramid approach to fitting.

7. Conclusion

The experiments indicate that the proposed vehicle model can be fit to edges in images of various vehicles. Existing methods for initialization should position a vehicle within the regions of convergence presented in Table 1. In [12], 3-d pose of detected toy vehicles is determined within 33.9 ± 21.74 mm and $10.7 \pm 5.2°$ (the toy is roughly a 1/10 scale model). Future work will evaluate methods like this for automatic initialization. Automatic initialization is likely to also provide some constraints to help initialize the shape and further expand the convergence regions. Future work will also address bias in the model by reweighing or providing more training data. Using the existing deformable vehicle can reduce manual work related to initializing parameter learning for new CAD model training data.

The proposed vehicle model has potential for many uses beyond those described in this paper. Obvious directions for

Vehicle	(class)	Color	NV	NS	NP	Error	X	Y	Z	$\theta_\mathbf{X}$	$\theta_\mathbf{Y}$	$\theta_\mathbf{Z}$
Dodge Stratus	(sedan)	white	7	5	6	0.0224	2.30	1.45	0.75	43	31	38
Dodge Stratus	(sedan)	white	4	5	6	0.0283	2.25	1.55	0.75	42	33	49
Dodge Stratus	(sedan)	white	2	5	6	0.0544	0.90	0.60	0.40	46	10	27
Dodge Stratus	(sedan)	white	1	5	3	0.0797	0.95	0.60	0.10	11	9	27
Dodge Caravan	(minivan)	white	6	5	6	0.0266	2.95	1.75	0.50	49	37	45
VW Beetle	(sedan)	yellow	4	5	6	0.0346	2.60	1.50	0.85	42	34	86
Toyota 4Runner	(SUV)	white	7	5	3	0.0356	2.50	1.90	NA	NA	NA	65
Volvo V70XC	(wagon)	blue	5	5	6	0.0493	2.15	1.60	0.65	38	25	46
VW Beetle	(sedan)	black	4	5	6	0.0903	2.05	1.30	0.70	52	25	52
VW Beetle	(sedan)	black	4	5	3	0.0340	2.10	1.20	NA	NA	NA	61
Toyota Tundra	(pickup)	black	6	5	3	0.0597	0.20	0.35	NA	NA	NA	4

Table 1. Experiments vary vehicle type, vehicle color, number of views (NV), number of shape parameters (NS), and number of pose parameters (NP). Results are RMS error (meters) and width of the convergence region in X, Y, Z (meters) and θ_X, θ_Y, θ_Z (degrees).

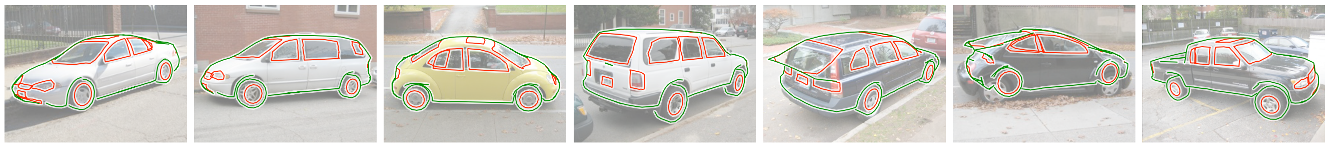

Figure 12. Edge hypotheses for converged results for the first instance of each vehicle in Table 1 shown from one of the views. Examples are loosely ordered in increasing order of difficulty (left to right).

future work include recognition from recovered parameters and tracking vehicles in video while estimating shape. The power of the model goes beyond edge prediction. Since parts are regions, a statistical model of intensity or color could be learned within the regions. Such a model could constrain the polarity of edge matches. Finally, since the vehicle model has a 3-d surface, it can also hypothesize effects of lighting such as cast shadows, surface shading, and specular highlights.

References

[1] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Trans. Gr.*, 24(3):408–416, 2005.

[2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of ACM SIGGRAPH 1999*, pages 187–194. ACM Press, 1999.

[3] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[4] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological surfaces. *Computer-Aided Design*, 10(6):350–355, 1978.

[5] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, Jul 2002.

[6] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker. A generic deformable model for vehicle recognition. In *Proceedings of the British Machine Vision Converence (BMVC)*, volume 1, pages 127–136, Surrey, UK, UK, 1995. BMVA Press.

[7] N. Ghosh and B. Bhanu. Incremental vehicle 3-d modeling from video. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 272–275, 2006.

[8] Y. Guo, C. Rao, S. Samarasekera, J. Kim, R. Kumar, and H. Sawhney. Matching vehicles under large pose transformations using approximate 3d models and piecewise mrf model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[9] Z. Kim and J. Malik. Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 524–531, October 2003.

[10] D. Koller, K. Daniilidis, and H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, June 1993.

[11] M. Leotta and J. Mundy. Learning background and shadow appearance with 3-d vehicle models. In *Proceedings of the British Machine Vision Converence (BMVC)*, volume 2, pages 649–658, September 2006.

[12] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[13] J. Lou, T. Tan, W. Hu, H. Yang, and S. Maybank. 3-d model-based vehicle tracking. *IEEE Trans. on Image Processing*, 14(10):1561–1569, October 2005.

[14] D. Lowe. Fitting parameterized three-dimensional models to images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(5):441–450, May 1991.

[15] Y. Ohtake and A. G. Belyaev. Dual/primal mesh optimization for polygonized implicit surfaces. In *Symposium on Solid modeling and applications (SMA)*, pages 171–178, New York, NY, USA, 2002. ACM.

[16] J. Schick and E. Dickmanns. Simultaneous estimation of 3d shape and motion of objects by computer vision. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 256–261, Oct 1991.

[17] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004*, pages 896–904. ACM Press, Aug. 2004.

[18] X. Song and R. Nevatia. A model-based vehicle segmentation method for tracking. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1124–1131, October 2005.