International Journal of Computer Vision 62(3), 221–247, 2005

Shape-From-Silhouette Across Time Part I: Theory and Algorithms

KONG-MAN (GERMAN) CHEUNG, SIMON BAKER AND TAKEO KANADE
The Robotics Institute, Carnegie Mellon University
german+@cs.cmu.edu
simonb@cs.cmu.edu
tk@cs.cmu.edu

Received October 13, 2003; Revised May 27, 2004; Accepted May 27, 2004
First online version published in November, 2004

Abstract. Shape-From-Silhouette (SFS) is a shape reconstruction method which constructs a 3D shape estimate of an object using silhouette images of the object. The output of a SFS algorithm is known as the Visual Hull (VH). Traditionally SFS is either performed on static objects, or separately at each time instant in the case of videos of moving objects. In this paper we develop a theory of performing SFS across time: estimating the shape of a dynamic object (with unknown motion) by combining all of the silhouette images of the object over time. We first introduce a one dimensional element called a Bounding Edge to represent the Visual Hull. We then show that aligning two Visual Hulls using just their silhouettes is in general ambiguous and derive the geometric constraints (in terms of Bounding Edges) that govern the alignment. To break the alignment ambiguity, we combine stereo information with silhouette information and derive a Temporal SFS algorithm which consists of two steps: (1) estimate the motion of the objects over time (Visual Hull Alignment) and (2) combine the silhouette information using the estimated motion (Visual Hull Refinement). The algorithm is first developed for rigid objects and then extended to articulated objects. In the Part II of this paper we apply our temporal SFS algorithm to two human-related applications: (1) the acquisition of detailed human kinematic models and (2) marker-less motion tracking.

Keywords: 3D reconstruction, Shape-From-Silhouette, Visual Hull, across time, stereo, temporal alignment, alignment ambiguity, visibility

1. Introduction

As its name implies Shape-From-Silhouette (SFS) is a method of estimating the shape of an object from its silhouette images. The idea of using silhouettes for 3D shape reconstruction was first introduced by Baumgart in 1974. In his PhD thesis (Baumgart, 1974), Baumgart estimated the 3D shapes of a baby doll and a toy horse from four silhouette images. Since then, different variations of the Shape-From-Silhouette paradigm have been proposed. For example, Aggarwal et al. (Martin and Aggarwal, 1983; Kim and Aggarwal, 1986) used volumetric descriptions to represent the reconstructed shape. Potmesil (1987), Noborio et al. (1988) and Ahuja and Veenstra (1989) all suggested using an octree data structure to speed up SFS. Shanmukh and Pujari derived the optimal positions and directions to take silhouette images for 3D shape reconstruction in Shanmukh and Pujari (1991). Szeliski built a non-invasive 3D digitizer using a turntable and a single camera with Shape-From-Silhouette as the reconstruction method (Szeliski, 1993). In summary, SFS has become a popular 3D reconstruction method for *static objects*.

The term Visual Hull (VH) has been used in a general sense by researchers for over a decade to denote the shape estimated using the Shape-From-Silhouette principle: the intersection of the visual cones formed by the silhouettes and camera centers. The term

was first coined in 1991 by Laurentini (1991) who also published a series of subsequent papers studying the theoretical aspects of Visual Hulls of 3D polyhedral (Laurentini, 1994, 1995) and curved objects (Laurentini, 1999).

Estimating shape using SFS has many advantages. First of all, silhouettes are readily and easily obtainable, especially in indoor environment where the cameras are static and there are few moving shadows. The implementation of most SFS methods is also relatively straightforward, especially when compared to other shape estimation methods such as multi-baseline stereo (Okutomi and Kanade, 1993) or space carving (Kutulakos and Seitz, 2000). Moreover, the inherently conservative property (see Section 2.3) of the shape estimated using SFS is particularly useful in applications such as obstacle avoidance in robot manipulation and visibility analysis in navigation. These advantages have prompted a large number of researchers to apply SFS to solve other computer vision and graphics problems. Examples include human related applications such as virtual human digitization (Moezzi et al., 1997), body shape estimation (Kakadiaris and Metaxas, 1998), motion tracking/capture (Delamarre and Faugeras, 1999; Bottino and Laurentini, 2000) and image-based rendering (Buehler et al., 1999).

On the other hand, SFS suffers from the limitation that the shape estimated by SFS (the VH) can be a very coarse approximation when there are only a few silhouette images, especially for complex objects such as the dinosaur/bananas example shown in Fig. 1(a). Figures 1(b)–(d) show respectively the (colored) voxel models of the dinosaur/bananas built using 6, 36 and 66 silhouette images. As can be seen, the shape model built using only 6 silhouette images is very coarse, while much better shape estimates are obtained using 36 or 66 silhouettes.

Better shape estimates can only be obtained using SFS if the number of distinct silhouette images is increased. The most common way to do so is the "across space" approach. By across space, we mean increasing the number of physical cameras used. This approach, though simple, may not be feasible in many practical situations due to financial or physical limitations. In this paper we introduce and develop another approach: the "across time" approach. The across time approach increases the number of effective silhouette images by capturing a number of silhouettes from each camera over time (while the object is moving) and then combining all the silhouettes (after compensating for the motion of the object) to reconstruct a refined Visual Hull of the object.

The remainder of this paper is organized as follows. In Section 2 a brief review of SFS and the traditional ways of representing and constructing Visual Hulls are presented. In Section 3 we introduce a new Visual Hull representation called the Bounding Edge representation and derive an important property of the Bounding Edges called the Second Fundamental Property of Visual Hulls (2nd FPVH). In Section 4 we show that aligning two Visual Hulls using only the silhouettes is inherently ambiguous and derive the geometric constraints which govern the alignment. We show how photometric information (in the form of color images) can be used to break the alignment and develop a temporal SFS algorithm for a rigid object as follows. We first combine the 2nd FPVH with multi-camera stereo to extract 3D points called Colored Surface Points (CSPs) on the surface of the object. Using an idea similar to the 2D image alignment problem as in Szeliski (1994), we then align the 3D CSPs with the 2D silhouette and color images to estimate the 6 DOF motion between two Visual Hulls. The visibility issue is also discussed in Section 4. In Section 5 we extend our temporal SFS algorithm to

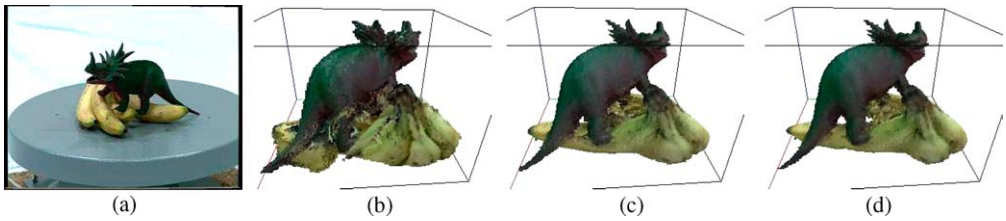

Figure 1. (a) An image of a toy dinosaur and a bunch of bananas. (b) A 3D colored voxel model reconstructed using 6 silhouette. Some details such as the legs and the horns of the dinosaur are missing. (c) A model reconstructed using 36 silhouette images. A much better shape estimate is obtained. (d) A model reconstructed using 66 silhouette images. An even better shape estimate is obtained.

articulated objects using the Expectation-Maximization (EM) formulation (Dempster et al., 1977) and imposing spatial coherency and temporal consistency. Both synthetic and real experimental results are shown at the end of Sections 4 and 5. We conclude in Section 6 with a brief discussion. In the Part II of this paper we apply our temporal SFS algorithm to two human-related applications: (1) the acquisition of detailed human kinematic models and (2) marker-less motion tracking.

2. Background

In this section we give a brief review of Shape-From-Silhouette (SFS). We first define the SFS problem scenario and present two equivalent definitions of the Visual Hull (VH). We proceed to describe two common ways of representing and constructing VHs.

2.1. Problem Scenario and Notation

Suppose there are K cameras positioned around a 3D object O. Let $\{S_j^k; k = 1; \ldots, K\}$ be the set of silhouette images of the object O obtained from the K cameras at time t_j. An example scenario is depicted in Fig. 2 with a head-shaped object surrounded by four cameras at time t_1. It is assumed that the cameras are calibrated with $\Pi^k() : I\!R^3 \rightarrow I\!R^2$ and C^k being the perspective projection function and the center of camera k respectively. In other words $p = \Pi^k(P)$ are the 2D image coordinates of a 3D point P in the kth image. As an extension of this notation, $\Pi^k(A)$ represents

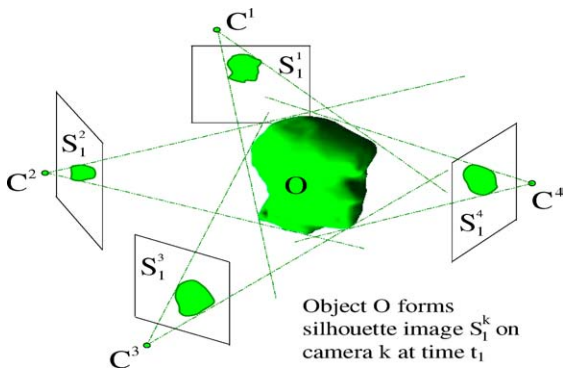

Figure 2. The Shape-From-Silhouette problem scenario: a head-shaped object O is surrounded by four cameras at time t_1. The silhouette images and camera centers are represented by S_j^k and C^k respectively.

the projection of a volume A onto the image plane of camera k. Assume we have a set of K silhouette images $\{S_j^k\}$ and projection functions $\{\Pi^k\}$. A volume A is said to *exactly explain* $\{S_j^k\}$ if and only if its projection onto the kth image plane coincides exactly with the silhouette image S_j^k for all $k \in \{1, \ldots, K\}$, i.e. $\Pi^k(A) = S_j^k$. If there exists at least one non-empty volume which explains the silhouette images exactly, we say the set of silhouette images is consistent, otherwise we call it inconsistent.

2.2. Definitions of the Visual Hull

Here we present two different ways to define the Visual Hull (Cheung, 2003). Although these two definitions are seemingly different, they are in fact equivalent to each other. See Cheung (2003) for a proof.

Visual Hull Definition I (Intersecting Visual Cones). The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the intersection of the K visual cones, each formed by projecting the silhouette image S_j^k into the 3D space through the camera center C^k.

This first definition, which is the most commonly used one in the SFS literature, defines the Visual Hull as the intersection of the visual cones formed by the camera centers and the silhouettes. Though this definition provides a direct way of computing the Visual Hull from the silhouettes (see Section 2.4.1), it lacks information and intuition about the object (which forms the silhouettes). We therefore also use a second definition (Laurentini, 1991):

Visual Hull Definition II (Maximally Exactly Explains). The Visual Hull H_j with respect to a set of consistent silhouette images $\{S_j^k\}$ is defined to be the largest possible volume which *exactly explains* $\{S_j^k\}$ for all $k = 1, \ldots, K$.

Generally for a consistent set of silhouette images $\{S_j^k\}$, there are an infinite number of volumes (including the object O itself) that *exactly explain* the silhouettes. Definition II defines the Visual Hull H_j as the largest one among these volumes. Though abstract, this definition implicitly expresses a property of Visual Hull: the Visual Hull provides an upper bound on the object which forms the silhouettes. To emphasize the importance of this property, we state it as the first fundamental property of Visual Hulls.

2.3. First Fundamental Property of Visual Hulls

First Fundamental Property of Visual Hulls (1st FPVH). The object O that formed the silhouette set S_j^k lies completely inside the Visual Hull H_j constructed from S_j^k.

The 1st FPVH is important as it gives us useful information on the object O in applications such as robotic navigation or obstacle avoidance. The upper bound given by the Visual Hull gets tighter if we increase the number of distinct silhouette images. Asymptotically if we have an infinite number of every possible silhouette images of a *convex* object, the Visual Hull is exactly equal to the object. If the object is not convex, the Visual Hull may or may not be equal to the object.

2.4. Representation and Construction

2.4.1. 2D Surface Based Representation. For a consistent set of silhouette images, the Visual Hull can be (according to Definition I) constructed by intersecting the visual cones directly. By doing so, the Visual Hull is represented by 2D surface patches obtained from intersecting the surfaces of the visual cones. Although simple and obvious in 2D, this direct intersection representation is difficult to use for general 3D objects. Recently Buehler et al. (1999, 2001) and Matusik et al. (2000) proposed an approximate way to compute the Visual Hull directly using the visual cone intersection method by approximating the object as having polyhedral shape. Since polyhedral objects produce polygonal silhouette images, their Visual Hulls consist of planar surface patches. However, for a general 3D object, its Visual Hull consists of curved and irregular surface patches which are difficult to represent using simple geometric primitives and are computational expensive and numerically unstable to compute.

2.4.2. 3D Volume Based Representation. Since it is difficult to intersect the surfaces of the visual cones of general 3D objects, other more effective ways have been proposed to construct Visual Hulls. The approach which is used by most researchers (Potmesil, 1987; Noborio et al., 1988; Ahuja and Veenstra, 1989; Szeliski, 1993) is volume based construction. Voxel-based SFS uses the same principle of visual cone intersection. However, the Visual Hull is represented by 3D volume elements ("voxels") rather than 2D surface patches. The space of interest is divided into discrete voxels which are then classified into two categories: *inside* and *outside*. The union of all the *inside* voxels is an approximation of the Visual Hull. For a voxel to be classified as *inside*, its projection on each and every one of the K image planes has to be inside or partially overlap the corresponding silhouette image. If the projection of the voxel is totally outside any of the silhouette images, it is classified as *outside*. One of the disadvantages of using discrete voxels to represent Visual Hulls is that the voxel-based VH can be significantly larger than the actual VH (see Cheung, 2003 for details).

3. A 1D VH Representation: Bounding Edge

In Section 2 we described two common ways to represent Visual Hulls: two-dimensional surface patches and three-dimensional discrete voxels. In this section, we propose a new representation for Visual Hulls using a one-dimensional element called a Bounding Edge (BE).

3.1. Definition of Bounding Edge

Consider a set of K silhouette images $\{S_j^k\}$ at a given time instant t_j. Let u_j^i be a point on the boundary of the silhouette image S_j^k. By projecting u_j^i into 3D space through the camera center C^k, we get a ray r_j^i. A Bounding Edge E_j^i is defined to be the part of r_j^i such that the projection of E_j^i onto the lth image plane lies completely inside the silhouette S_j^l for all $l \in \{1, \ldots, K\}$. Mathematically the condition can be expressed as

$$E_j^i \subset r_j^i \quad \text{and} \quad \Pi^l\big(E_j^i\big) \subset S_j^l \quad \forall\, l \in \{1, \ldots, K\}. \tag{1}$$

Figure 3(a) illustrates the definition of a Bounding Edge at t_1. A Bounding Edge can be computed by first projecting the ray r_j^i onto the $K-1$ silhouette images $S_j^l, l = 1, \ldots, K; l \neq k$, and then re-projecting the segments which overlap with S_j^l back into 3D space. The Bounding Edge is the intersection of the reprojected segments. Note that the Bounding Edge E_j^i is not necessarily a continuous line. It may consist of several segments if any of the silhouette images are not convex. Hereafter, a Bounding Edge E_j^i is denoted by a set of *ordered* 3D vertex pairs as follows:

$$E_j^i = \big\{\big(SV_j^i(m), FV_j^i(m)\big); \quad m = 1, \ldots, M_j^i\big\}, \tag{2}$$

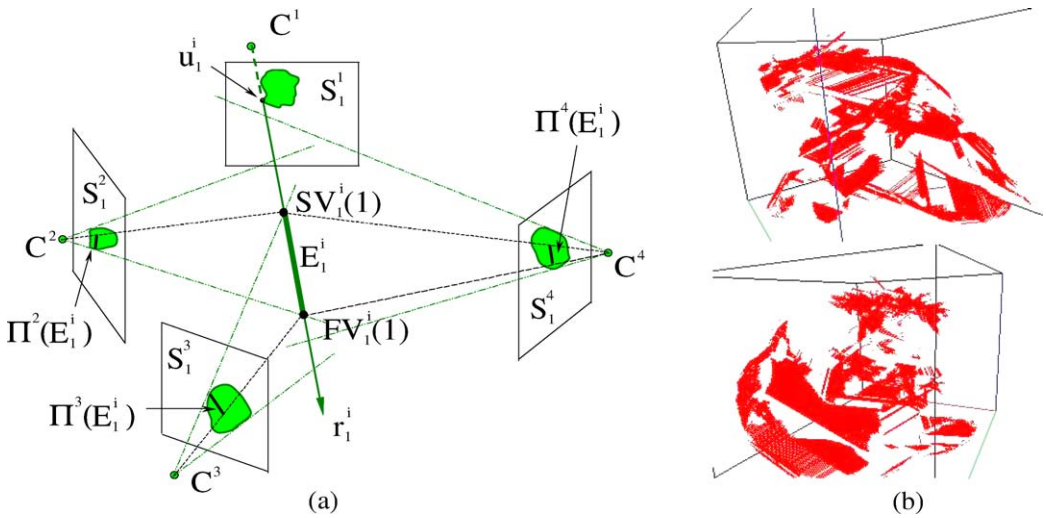

Figure 3. (a) The Bounding Edge E_1^i is obtained by first projecting the ray r_1^i onto S_1^2, S_1^3, S_1^4 and then re-projecting the segments overlapping with the silhouettes back into 3D space. E_1^i is the intersection of the reprojected segments. (b) Two different views of the Bounding Edge representation of the Visual Hull of the dinosaur/bananas object shown in Fig. 1.

where $SV_j^i(m)$ and $FV_j^i(m)$ represent the start vertex and finish vertex of the mth segment of the Bounding Edge respectively and M_j^i is the number of segments that E_j^i is comprised of. By sampling points on the boundaries of all the silhouette images $\{S_j^k; k = 1, \ldots, K\}$, we can construct a list of L_j Bounding Edges that represents the Visual Hull H_j. Figure 3(b) illustrates the Bounding Edge representation of the VH of the dinosaur/bananas object shown in Fig. 1(a).

3.2. *Second Fundamental Property of Visual Hulls*

The most important property of the Bounding Edge representation is that its definition captures one aspect of Shape-From-Silhouette very naturally. To be precise, we state this property as

Second Fundamental Properties of Visual Hulls (2nd FPVH). Each Bounding Edge of the Visual Hull touches the object (that formed the silhouette images) at at least one point.

The 2nd FPVH allows us to use Bounding Edges to represent one important aspect of the shape information of the object that can be extracted from a set of silhouette images. Although being an important property, the 2nd FPVH is often overlooked by researchers who usually focus on the 1st FPVH. In the next chapter, we will show how the 2nd FPVH can be combined with stereo to locate points on the surface of the object. A comparison of the advantages and disadvantages of the three VH representations (surfaces, voxels and Bounding Edges) can be found in Cheung (2003).

3.3. *Related Work*

In their image-based Visual Hull rendering work (Buehler et al., 1999; Matusik et al., 2000; Matusik, 2001), Matusik et al. proposed a ray-casting algorithm to render objects using silhouette images. Their way of intersecting the casting rays with the silhouette images is similar to the way our Bounding Edges are constructed. However, there are two fundamental differences between their approach and the definition of Bounding Edge. First, our Bounding Edges are originated only from points on the boundary of the silhouette image while their casting rays can originate from anywhere, including any point inside the silhouette. Second, their casting rays do not embed the important 2nd FPVH as Bounding Edges do. In a separate paper (Buehler et al., 2001), Matusik et al. also proposed a fast way to build polyhedral Visual Hulls. They based their idea on visual cone intersection but simplified the representation and computation by approximating the actual silhouette as polygons (i.e. any curved part of the silhouette is approximated by straight

lines) which is equivalent to approximating the 3D object as polyhedral shape. Due to this approximation, their results are not the exact surface-based representation discussed in Section 2.4.1 except for true polyhedral objects. Nevertheless their idea of calculating silhouette edge bins can be applied to speed up the construction of Bounding Edges. Lazebnik et al. (2001) independently proposed a new way of representing Visual Hulls. The edge of the "Visual Hull mesh" in their work is theoretically equivalent to the definition of a Bounding Edge. However, they compute their edges after locating frontier and triple points whereas we compute Bounding Edges directly from the silhouette images.

4. SFS Across Time: Rigid Objects

In this section we propose an algorithm for Shape-From-Silhouette across time for rigid objects. A number of silhouettes from each camera are captured as the object moves across time and then used to construct a refined VH. For example, for a system with K cameras and J frames, the effective number of cameras would be increased to JK. This is equivalent to adding an additional $(J-1)K$ physical cameras to the system.

There are two tasks to constructing Visual Hulls across time: (1) estimating the motion of the object between successive time instants and (2) combining the silhouette images at different time instants to get a refined shape of the object. In this section, we assume the object of interest is rigid, but the motion of the object between frames is totally arbitrary and unknown. In Section 5 we will extend the algorithm to articulated objects. We refer to the task of computing the rigid transformation as Visual Hull Alignment and the task of combining the silhouette images across time as Visual Hull Refinement.

4.1. Visual Hull Alignment: Theory

To combine silhouette images across time, the motion of the object between frames is required. For static objects, the problem may be simplified by putting the object on a *precisely calibrated* turn-table so that the motion is known in advance (Szeliski, 1993). However for dynamic objects whose movement we do not have control or knowledge of, we have to estimate the unknown motion before we can combine the silhouette images across time. To be more precise, we state the Visual Hull Alignment Problem as:

Visual Hull Alignment from Silhouette Images

Suppose we are given two sets of consistent silhouette images $\{S_j^k; k = 1; \ldots, K; j = 1, 2\}$ of a rigid object O from K cameras at two different time instants t_1 and t_2. Denote the Visual Hulls for these silhouette sets by H_j, $j = 1, 2$. Without loss of generality, assume the first set of images $\{S_1^k\}$ are taken when the object is at position and orientation of $(\boldsymbol{I}, \underline{0})$ while the second image set $\{S_2^k\}$ is taken when the object is at $(\boldsymbol{R}, \boldsymbol{t})$. The problem of Visual Hull alignment is to find $(\boldsymbol{R}, \boldsymbol{t})$ such that there exists an object O which exactly explains the silhouettes at both times t_j and the relative position and orientation of O is related by $(\boldsymbol{R}, \boldsymbol{t})$ from t_1 to t_2. Moreover, we say that the two Visual Hulls H_1 and H_2 are *aligned consistently with transformation* $(\boldsymbol{R}, \boldsymbol{t})$ if and only if we can find an object O such that H_1 is the Visual Hull of O at orientation and position $(\boldsymbol{I}, \underline{0})$ and H_2 is the Visual Hull of O at orientation and position $(\boldsymbol{R}, \boldsymbol{t})$.

4.1.1. Visual Hull Alignment Ambiguity. Since it is assumed that the two sets of silhouette images are consistent and come from the same object, there always exists at least one set of object O and motion $(\boldsymbol{R}, \boldsymbol{t})$ (the true solution) that exactly explains both sets of silhouette images. We now show that aligning two Visual Hulls using only the silhouette information is inherently ambiguous. This means that in general the solution is not unique and there exists more than one set of $(\boldsymbol{R}, \boldsymbol{t})$ which satisfies the alignment criterion. A 2D example is shown in Fig. 4. In the figure, both (a) and (b) have the same silhouette image sets (and hence the same Visual Hulls) at times t_1 and t_2. However, in (a), the silhouettes are formed by a curved object with a pure translation between t_1 and t_2, while in (b), the silhouettes are created by a polygonal object with both a rotation (200 degrees) and a translation between t_1 and t_2.

4.1.2. Geometric Constraints for Aligning 2D Visual Hulls. The motion ambiguity in Visual Hull alignment is a direct result of the indeterminacy in the shape of the object. Although the alignment solution is not unique, there are constraints on the motion and the shape of the object for a consistent alignment. In this section we discuss the geometrical constraints for aligning two 2D Visual Hulls and in the next section extend them to 3D.

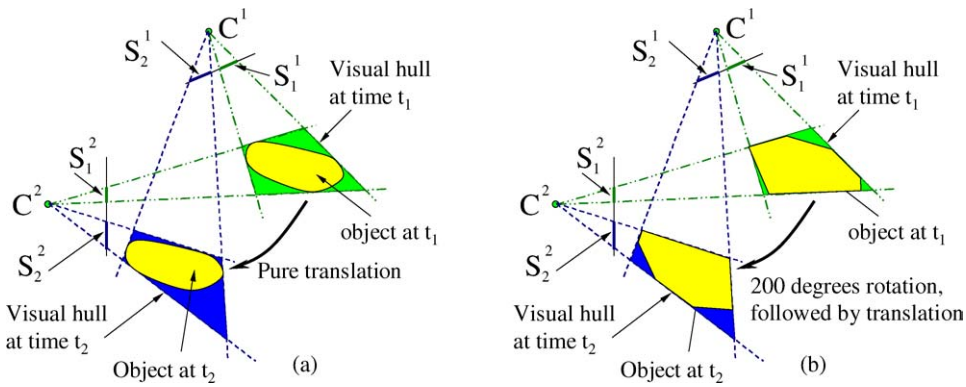

Figure 4. A 2D example showing the ambiguity of aligning Visual Hulls. Both cases (a) and (b) have the same silhouettes at times t_1 and t_2 but they are formed from two different objects with different motions.

To state the constraints for aligning two 2D polygonal Visual Hulls H_j, $j = 1, 2$ of a 2D object O, let E_j^i be the edges of H_j, $T_{(\boldsymbol{R},\boldsymbol{t})}(A)$ be the entity after applying transformation of $(\boldsymbol{R}, \boldsymbol{t})$ to A and $T^{-1}_{(\boldsymbol{R},\boldsymbol{t})}()$ denotes the inverse transformation. Now using the 2D version of the 2nd FPVH (see Cheung, 2003 for details), the geometric constraints are expressed in the following Lemma[1]:

Lemma 1. *Given two 2D Visual Hulls H_1 and H_2, the necessary and sufficient condition for them to be aligned consistently with transformation $(\boldsymbol{R}, \boldsymbol{t})$ is given as follows: No edge of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$ lies completely outside H_2 and no edge of H_2 lies completely outside $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$.*

Figure 5(a) and (b) shows examples of two 2D Visual Hulls of the same object. In (c), the alignment is consistent and all edges from both Visual Hulls satisfy Lemma 1. In (d), the alignment is inconsistent and the edges E_1^1, E_1^4, E_1^5, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^1)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^2)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^7)$ all violate Lemma 1. Lemma 1 provides a good way to test if the alignment of two 2D VHs is consistent or not.

To illustrate how these constraints can be used in practice, two synthetic 2D Visual Hulls (polygons) each with four edges (Fig. 6) were generated and Lemma 1 was used to search for the space of all consistent alignments. In 2D there are only three degrees of freedom (two in translation and one in rotation). The space of consistent alignments is shown in Fig. 6. There are two unconnected subsets of the solution space, clustered around two different rotation angles.

In order to extend Lemma 1 to 3D, consider the following variant of Lemma 1 for 2D objects:

Lemma 2. *$(\boldsymbol{R}, \boldsymbol{t})$ is a consistent alignment of two 2D Visual Hulls H_1 and H_2, constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$ if and only if the following condition is satisfied: for each edge E_1^i of $T_{(\boldsymbol{R},\boldsymbol{t})}(H_1)$, there exists at least one point P on E_1^i such that the projection of P onto the kth image lies inside or on the boundary of the silhouette S_2^k for all $k = 1, \ldots, K$.*

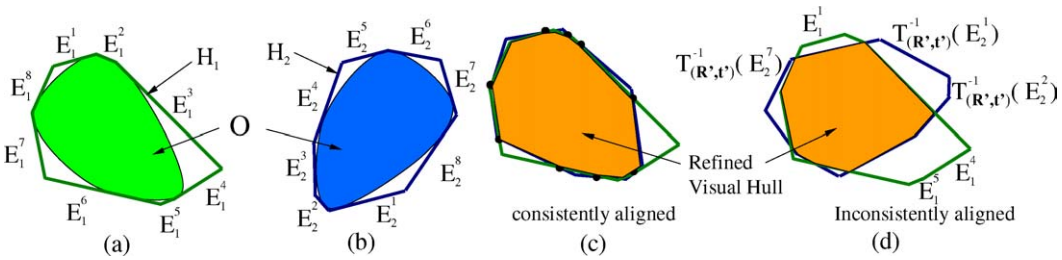

Figure 5. (a) and (b) Two Visual Hulls of the same object at different positions and orientations. (c) All edges satisfy Lemma 1 when the alignment $(\boldsymbol{R}, \boldsymbol{t})$ is consistent, (d) Edges E_1^1, E_1^4, E_1^5, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^1)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^2)$, $T^{-1}_{(\boldsymbol{R}',\boldsymbol{t}')}(E_2^7)$ all violate Lemma 1 and so the Visual Hulls are not aligned consistently.

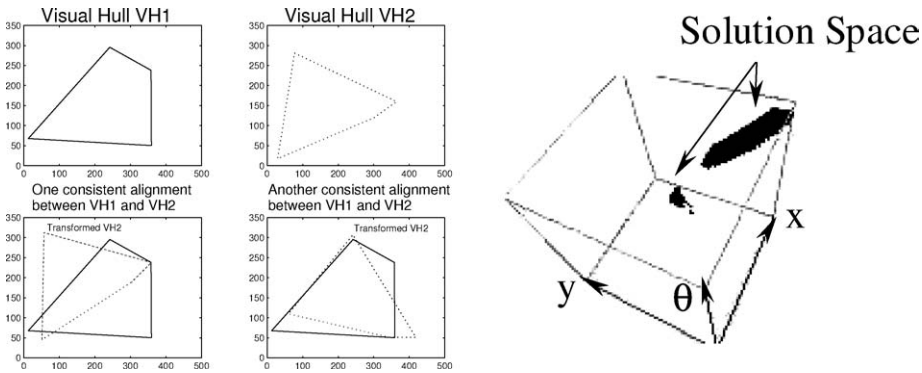

Figure 6. Two synthetic 2D Visual Hulls (each with four edges) and the space of consistent alignments.

Lemma 2 expresses the constraints in terms of the silhouette images rather than the Visual Hull. For 2D objects, there is no significant difference between using Lemma 1 or Lemma 2 to specify the alignment constraints because all 2D Visual Hulls can be represented by a polygon with a finite number of edges. For 3D objects, however, the 3D version of Lemma 1 is not very practical because it is difficult to represent a 3D Visual Hull exactly and completely (see Cheung, 2003). By expressing the geometrical constraints in terms of the silhouette images (Lemma 2) instead of the Visual Hull itself (Lemma 1), the need for an exact and complete Visual Hull representation can be avoided. In the next section, we extend Lemma 2 to 3D *convex* objects.

4.1.3. Geometric Constraints for Aligning 3D Visual Hulls. The geometric constraints for aligning two *convex* 3D VHs are expressed in the following lemma:

Lemma 3. *For two convex 3D Visual Hulls H_1 and H_2 constructed from silhouette sets $\{S_j^k\}$; $j = 1, 2$, the necessary and sufficient condition for a transformation $(\boldsymbol{R}, \boldsymbol{t})$ to be a consistent alignment between H_1 and H_2 is as follows: for any Bounding Edge E_1^i constructed from the silhouette image set $\{S_1^k\}$, there exists at least one point P on E_1^i such that the projection of the point $T_{(\boldsymbol{R},\boldsymbol{t})}(P)$ onto the kth image lies inside or on the silhouette S_2^k for all $k = 1, \ldots, K$. Similarly, for any Bounding Edge E_2^i constructed from $\{S_2^k\}$, there exists at least one point P on E_2^i such that the projection of the point $T^{-1}_{(\boldsymbol{R},\boldsymbol{t})}(P)$ on the kth image lies inside or on the silhouette S_1^k.*

The condition in Lemma 3 is still necessary, but not sufficient, if either one or both of the two Visual Hulls are non-convex. A counter example can be found in Cheung (2003). For general 3D objects, Lemma 3 is useful to reject inconsistent alignments between two Visual Hulls but cannot be used to prove if an alignment is consistent. Theoretically we can prove if an alignment is consistent as follows. First transform the Visual Hulls using the alignment transformation and compute the intersection of the two Visual Hulls. The resultant Visual Hull is then rendered with respect to all the cameras at both times and compared with the two original sets of silhouette images. If the new Visual Hull exactly explains all the original silhouette images, then the alignment is consistent. In practice, however, this idea is computationally very expensive and is inappropriate as an algorithm to compute the correct alignment between two 3D Visual Hulls. In Section 4.2.3, we will show how the hard geometric constraints stated in Lemma 3 can be approximated by soft constraints and combined with photometric consistency to align 3D Visual Hulls.

4.2. Resolving the Alignment Ambiguity

Since aligning Visual Hulls using silhouette images alone is ambiguous (see Section 4.1.1), additional information is required in order to find the correct alignment. In this section we show how to resolve the alignment ambiguity using color information (Cheung et al., 2003). First we combine the 2nd FPVH (introduced in Section 3) with stereo to extract a set of 3D points (which we call Colored Surface Points) on the surface of the object at each time instant. The two sets of 3D Colored Surface Points are then used to align the Visual Hulls through the 2D color images. We assume that besides the set of silhouette images $\{S_j^k\}$, the set of original color images (which the silhouette images

were derived from) are also given and represented by $\{I_j^k\}$.

4.2.1. Colored Surface Points (CSPs). Although the Second Fundamental Property of Visual Hull tells us that each Bounding Edge touches the object at at least one point, it does not provide a way to find this point. Here we propose a simple (one-dimensional) search based on the stereo principle to locate this touching point. If we assume the object is Lambertian and all the cameras are color balanced, then any point on the surface of the object should have the same projected color in all of the color images. In other words, for any point on the surface of the object, its projected color variance across the *visible* cameras should be zero. Hence on a Bounding Edge, the point which touches the object should have zero projected color variance. This property provides a good criterion for locating the touching points. Hereafter we call these touching points as the *Colored Surface Points (CSP)*.

To express the idea mathematically, consider a Bounding Edge E_j^i from the jth Visual Hull. Since we denoted the Bounding Edge E_j^i by a set of *ordered* 3D vertex pairs $\{(SV_j^i(m), FV_j^i(m))\}$ (Eq. (2)), we can parameterize a point $W_j^i(m, w)$ on E_j^i by two parameters m and w, where $m \in \{1, \ldots, M_j^i\}$ and $0 \le w \le 1$ with

$$W_j^i(m, w) = SV_j^i(m) + w * \big(FV_j^i(m) - SV_j^i(m)\big). \tag{3}$$

Let $\boldsymbol{c}_j^k(P)$ be the projected color of a 3D point P on the kth color image at time t_j. The projected color mean $\mu_j^i(m, w)$ and variance $\sigma_j^i(m, w)$ of the point $W_j^i(m, w)$ are given as

$$\begin{aligned} \mu_j^i(m, w) &= \frac{1}{n_j^i} \sum_k \boldsymbol{c}_j^k\big(W_j^i(m, w)\big); \\ \sigma_j^i(m, w) &= \frac{1}{n_j^i} \sum_k \big[\boldsymbol{c}_j^k\big(W_j^i(m, w)\big) - \mu_j^i(m, w)\big]^2. \end{aligned} \tag{4}$$

The projected color $\boldsymbol{c}_j^k(W_j^i(m, w))$ from camera k is used in calculating the mean and variance *only if* $W_j^i(m, w)$ is *visible* in that camera and n_j^i denotes the number of the visible cameras for point W_j^i. The question of how to conservatively determine the visibility of a 3D point with respect to a camera using only the silhouette images will be addressed shortly in Section 4.3. Figure 7(a) illustrates the idea of locating the touching point by searching along the Bounding Edge.

In practice, due to noise and inaccuracies in color balancing, instead of searching for the point which has zero projected color variance, we locate the point with the minimum variance. In other words, we set the Colored Surface Point of the object on E_j^i to be $W_j^i(\tilde{m}, \tilde{w})$ where $\tilde{m}$ and $\tilde{w}$ minimizes $\sigma_j^i(m, w)$ for $0 \le w \le 1; m \in \{1, \ldots, M_j^i\}$. This can be done by sampling discretely and uniformly over the 1D parameter space of w along each segment of the Bounding Edge and search for the point with the minimum variance. Note that by choosing the point with the

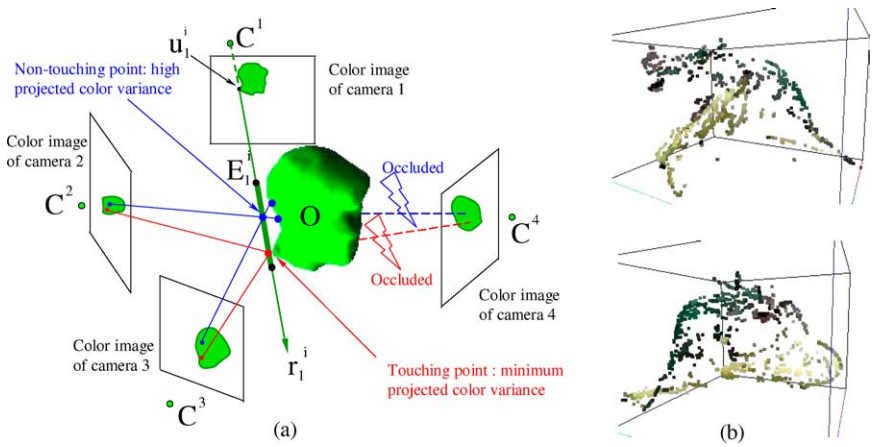

Figure 7. (a) Locating the touching point (Colored Surface Point) by searching along the Bounding Edge for the point with the minimum projected color variance. (b) Two sets of CSPs for the dinosaur/bananas example (see Fig. 1) obtained at two time instants with different positions and orientations. Note that the CSPs are sparsely sampled and there is no point-to-point correspondence between the two sets of CSPs.

minimum variance, the problem of tweaking parameters or thresholds of any kind is avoided. The need to adjust parameters or thresholds is always a problem in other shape reconstruction methods such as space carving (Kutulakos and Seitz, 2000) or multi-baseline stereo (Okutomi and Kanade, 1993). Space carving relies heavily on a color variance threshold to remove non-object voxels and stereo matching results are sensitive to the search window size. In our case, knowing that each Bounding Edge touches the object at at least one point (2nd FPVH) is the key piece of information that allows us to avoid any thresholds. In fact locating CSPs is a special case of the problem of matching points on pairs of epipolar lines as discussed in Szeliski and Golland (1998) and Irani et al. (2002). In Szeliski and Golland (1998) and Irani et al. (2002), points are matched on "general" epipolar lines on which there may or may not be a matching point so a threshold and an independent decision is needed for each point. To locate CSPs, points are matched on "special" epipolar lines which guarantee to have at least one matching point so no threshold is required.

Since we use local texture information to extract CSPs, for texture-less surface there is ambiguity in determining the correct positions of the CSPs. Unfortunately it is a common problem to a lot of 3D reconstruction methods which depend on texture and there is no easy solution to it. However, since CSPs are restricted to lie on the Bounding Edge, in practice if the positions of the CSPs are incorrectly estimated in the texture-less region, the deviations are usually small and have insignificant effects on our alignment algorithm to be discussed below. See Section 4.5 for experimental validation and further discussion.

Hereafter, for simplicity we drop the notation dependence of m, $\tilde{w}$, and denote (with a slight abuse of notation) the CSPs $W_j^i(\tilde{m}, \tilde{w})$ by W_j^i and its color $\mu_j^i(\tilde{m}, \tilde{w})$ by μ_j^i.

4.2.2. Alignment by Color Consistency. Suppose we have located two sets of Colored Surface Points at two different time instants t_1 and t_2. For example, Fig. 7(b) shows two sets of CSPs for the dinosaur/bananas (see Fig. 1) obtained at two time instants at two different positions and orientations. Since the sets of CSPs lie on their corresponding (rigid) Visual Hulls H_1 and H_2, the problem of aligning H_1 and H_2 is equivalent to aligning the two sets of CSPs. The question now is how can we align the two sets of CSPs. Before answering this question, we have to point out two very important facts about CSPs. First, the CSPs at each time instant are points on the occluding contours. This means that CSPs are only sparsely sampled points on the surface of the object (as opposed to the 3D data points acquired from laser range devices which produce densely sampled surface points on the object). The point sparsity prohibits us from using well established 3D point alignment methods such as the Iterative Closet Point (ICP) method (Besl and McKay, 1992; Zhang, 1994; Rusinkiewicz and Levoy, 2001). Secondly the only property common of the two sets of CSPs is that they all lie on the surface of the object. There is *no* point-to-point correspondence between any two sets of CSPs obtained at different time instants. Because of this, alignment methods which are used in the structure-from-motion literature (Tomasi and Kanade, 1992; Poelman and Kanade, 1992; Quan and Kanade, 1996) cannot be used to align the CSPs.

To solve the CSP alignment problem, we use an idea similar to that used to solve the 2D image registration problem in Szeliski (1994) (related idea has been proposed to register 3D laser range data with camera images in Wheeler (1996) and Kurazume et al. (2002)). In our case, instead of registering a 2D image with another 2D image, we align 2D images ($\{I_2^k\}$) at time t_2 with a "3D image" (the Colored Surface Points $\{W_1^i\}$) at time t_1 through the projection functions $\{\Pi^k\}$. The error measure used is the sum of squares of the color differences between the Colored Surface Points at time t_1 and their projected colors from the color images at time t_2 and vice versa. Mathematically, let $\{I_j^k, S_j^k, W_j^i, \mu_j^i; i = 1, \ldots, L_j; k = 1, \ldots, K; j = 1, 2\}$ be the two sets of data. To find the most color consistent alignment $(\boldsymbol{R}, \boldsymbol{t})$, consider the color error function $e = \sum_{i=1}^{L_2} e_{1,2}^i + \sum_{i=1}^{L_1} e_{2,1}^i$ where

$$\begin{aligned} e_{1,2}^i &= \sum_k e_{1,2}^{i,k} = \sum_k \left[\boldsymbol{c}_1^k\left(\boldsymbol{R}^T\left(W_2^i - \boldsymbol{t}\right)\right) - \mu_2^i\right]^2; \\ e_{2,1}^i &= \sum_k e_{2,1}^{i,k} = \sum_k \left[\boldsymbol{c}_2^k\left(\boldsymbol{R}W_1^i + \boldsymbol{t}\right) - \mu_1^i\right]^2. \end{aligned} \tag{5}$$

Here $e_{2,1}^{i,k}$ represents the difference between the mean color μ_1^i of the Colored Surface Point W_1^i at time t_1 and its projected color $\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t})$ in camera k at time t_2. Note that at time t_2, the new position of W_1^i due to the motion of the object is $\boldsymbol{R}W_1^i + \boldsymbol{t}$. Likewise, $e_{1,2}^{i,k}$ is the difference between the mean color μ_2^i of W_2^i and its projected color $\boldsymbol{c}_1^k(\boldsymbol{R}^T(W_2^i - \boldsymbol{t})$ in camera k at time t_1. From now on, we refer to the error of aligning 3D points with the 2D images forward in time (e.g. 3D points at t_1 and 2D images at t_2) as the forward error. Similarly the error of aligning 3D points with the 2D

images backward in time (e.g. 3D points at t_2 and 2D images at t_1) is referred to as the backward error. In the current example, $e^i_{2,1}$ is the forward error while $e^i_{1,2}$ is the backward error. Just as when locating the CSPs on the Bounding Edge in Eq. (4), the summations in Eq. (5) include the projected color of camera k *only if* the point of interest is visible in that camera. The process of Visual Hull alignment by color consistency is illustrated in Fig. 8.

If we parameterize $\boldsymbol{R}$ and $\boldsymbol{t}$ as $\boldsymbol{\Phi} = [\Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, \Phi_6]^T$, where Φ_1, Φ_2, Φ_3 are the Euler's angles of $\boldsymbol{R}$ and Φ_4, Φ_5, Φ_6 are the x, y, z components of $\boldsymbol{t}$, the minimization of Eq. (5) can be solved by a variant of the Levenberg-Marquardt (LM) algorithm (Dennis and Schnabel, 1983; Press et al., 1993):

1. With an initial estimate $\hat{\boldsymbol{\Phi}}$, calculate the Hessian matrix $\boldsymbol{H} = \{h_{mn}\}$ and the difference vector $\boldsymbol{d} = \{d_m\}$ with $m, n = 1, \ldots, 6$ as

$$h_{mn} = \sum_{i=1}^{L_2} \sum_k \frac{\partial e^{i,k}_{1,2}}{\partial [\boldsymbol{\Phi}]_m} \frac{\partial e^{i,k}_{1,2}}{\partial [\boldsymbol{\Phi}]_n} + \sum_{i=1}^{L_1} \sum_k \frac{\partial e^{i,k}_{2,1}}{\partial [\boldsymbol{\Phi}]_m} \frac{\partial e^{i,k}_{2,1}}{\partial [\boldsymbol{\Phi}]_n}, \tag{6}$$

$$d_m = -2\left[\sum_{i=1}^{L_2} \sum_k e^{i,k}_{1,2} \frac{\partial e^{i,k}_{1,2}}{\partial [\boldsymbol{\Phi}]_m} + \sum_{i=1}^{L_1} \sum_k e^{i,k}_{2,1} \frac{\partial e^{i,k}_{2,1}}{\partial [\boldsymbol{\Phi}]_m} \right]. \tag{7}$$

2. Update the parameter $\hat{\boldsymbol{\Phi}}$ by an amount $\delta\boldsymbol{\Phi} = (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1}\boldsymbol{d}$, where λ is a time-varying stabilization parameter.
3. Go back to 1. until the estimate of $\hat{\boldsymbol{\Phi}}$ converges.

Note that in calculating the Hessian matrix $\boldsymbol{H}$ and the difference vector $\boldsymbol{d}$ in Eqs. (6) and (7), the derivatives of the function $\boldsymbol{c}^k_j$ (which maps a 3D point to its projected color in the image I^k_j) with respect to $\boldsymbol{\Phi}$ are needed. Here we use a chain rule approach similar to that used for 2D image registration in Szeliski (1994). The derivatives are approximated by multiplying the image gradient (computed locally) with the camera projection matrix and the Jacobian of the transformed 3D point with respect to the transformation parameters $\boldsymbol{\Phi}$.

For objects with large motion between frames, we initialize the algorithm by approximating the two sets of CSPs at t_1 and t_2 each by an ellipsoidal shell. The initial estimate of the translation vector $\boldsymbol{t}$ is then set as the relative positions of the centers of the two ellipsoids. Similarly the initial guess for the rotation matrix $\boldsymbol{R}$ is set as the relative orientation of the two ellipsoids. This simple initialization method works well for most objects when the rotation of the object is less than 90 degrees. For objects with small motion between frames, it is suffice to initialize the algorithm with zero translation and rotation.

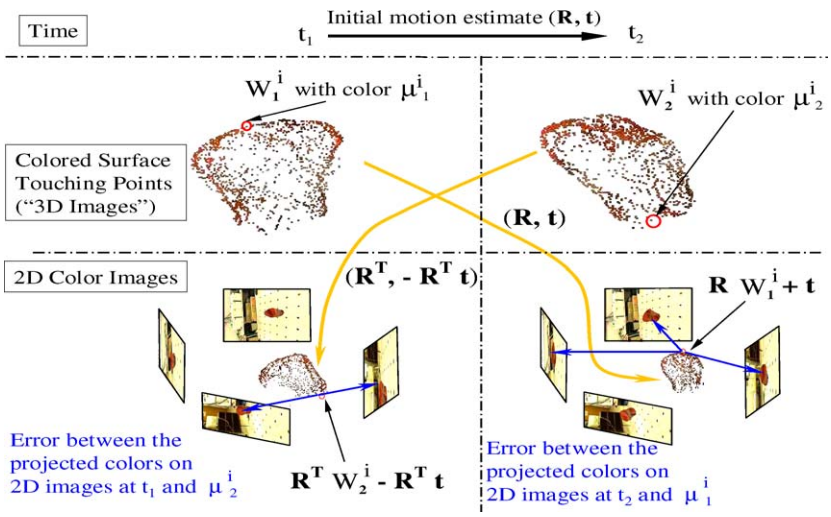

Figure 8. Visual Hull Alignment using color consistency. The error between the colors of the 3D surface points and their projected image colors is minimized.

4.2.3. Alignment by Color Consistency and Geometrical Constraints. Since the above formulation for aligning two sets of CSPs is inspired by the 2D image registration problem (Szeliski, 1994), the error measure in Eq. (5)) is based solely on color consistency (stereo). Though simple, this formulation does not take into account an important fact: the CSPs lie on the surface of Visual Hulls whose alignment is governed by the geometric constraints stated in Lemma 3. Here we show how the hard constraints of Lemma 3 can be converted into soft constraints and combined with color consistency to align the CSPs.

Recall that Lemma 3 states that if $(\boldsymbol{R}, \boldsymbol{t})$ is a consistent alignment, then for any Bounding Edge E_1^i, there exists at least one point P on E_1^i such that the projection of the transformed point $\boldsymbol{R}P + \boldsymbol{t}$ lies inside or on the boundary of all the silhouette images $\{S_2^k\}$ at time t_2 and vice versa. In fact P is the point where the object touches the Bounding Edge, which we have extracted as a CSP. Hence the constraint is equivalent to saying that all of the transformed CSPs at time t_1 must lie inside or on the boundary of the silhouette images $\{S_2^k\}$ and vice versa. In practice, due to noises and calibration errors, instead of applying this hard constraint directly to the optimization procedures, we incorporate it as a soft constraint by minimizing the distance between the projected CSP and the silhouettes as explained below.

Assume we have the same sets of data $\{I_j^k, S_j^k, W_j^i, \mu_j^i; i = 1, \ldots, L_j; k = 1, \ldots, K; j = 1, 2\}$ as before. Let $(\boldsymbol{R}, \boldsymbol{t})$ be an estimate of the rigid transformation. Consider first the calculation of the forward error. For a CSP W_1^i (with color μ_1^i) at time t_1, its 3D position at time t_2 would be $\boldsymbol{R}W_1^i + \boldsymbol{t}$. Consider two different cases of the projection of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ into the kth camera:

1. The projection lies inside the silhouette S_2^k. In this case, we use $[\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t}) - \mu_1^i]^2$ (the color difference) as the error measure, where as defined before, $\boldsymbol{c}_2^k(P)$ is the projected color of a 3D point P into the color image I_2^k. Otherwise, we set the color error to zero if the projection of P lies outside S_2^k. We call this error the forward photometric error.
2. The projection lies outside S_2^k. In this case, we use the distance of the projection from S_2^k, represented by $d_2^k(\boldsymbol{R}W_1^i + \boldsymbol{t})$ as an error measure. The distance is zero if the projection lies inside S_2^k. We call this error the forward geometric error.

Note that an approximation of the function d_j^k can be obtained by applying the distance transform to the silhouette image S_j^k (Jain, 1989). Summing over all cameras in which W_1^i is *visible*, the forward error measure of W_1^i with respect to $(\boldsymbol{R}, \boldsymbol{t})$ is given by

$$e_{2,1}^i = \sum_k \left\{\tau * d_2^k\left(\boldsymbol{R}W_1^i + \boldsymbol{t}\right) + \left[\boldsymbol{c}_2^k\left(\boldsymbol{R}W_1^i + \boldsymbol{t}\right) - \mu_1^i\right]^2\right\}, \tag{8}$$

where τ is a weighing constant. Equation (8) combines the color consistency constraint (stereo) with the geometric constraint (Shape-From-Silhouette) using the weighing constant τ. Similarly, the backward error measure of a CSP W_2^i at time t_2 is written as the sum of the backward photometric and geometric errors:

$$\begin{aligned} e_{1,2}^i = \sum_k \Big\{&\tau * d_1^k\left(\boldsymbol{R}^T\left(W_2^i - \boldsymbol{t}\right)\right) \\ &+ \left[\boldsymbol{c}_1^k\left(\boldsymbol{R}^T\left(W_2^i - \boldsymbol{t}\right)\right) - \mu_2^i\right]^2\Big\}. \end{aligned} \tag{9}$$

The problem of estimating $(\boldsymbol{R}, \boldsymbol{t})$ is now turned into the problem of minimizing the sum of the forward and backward error

$$\min_{R,t} e = \min_{R,t} \sum_{i=1}^{L_2} e_{1,2}^i + \sum_{i=1}^{L_1} e_{2,1}^i, \tag{10}$$

which can be solved using the same Iterative LM algorithm described in Section 4.2.2. Hereafter, we refer to this Visual Hull across time algorithm as the temporal SFS algorithm (for rigid objects) and summarize the steps as follows:

Temporal SFS Algorithm for Rigid Objects

1. Construct the Bounding Edges $\{E_j^i\}$ from the silhouette images $\{S_j^k\}$ at t_j where $j = 1, 2$.
2. Extract a set of Colored Surface Points $\{W_j^i, \mu_j^i\}$ at t_j from the list of Bounding Edges $\{E_j^i\}$ and the color images $\{I_j^i\}$.
3. Initialize the translation and rotation parameters by ellipsoid fitting.
4. Apply the Iterative LM algorithm (Section 4.2.2) to minimize the sum of the forward and backward errors in Eq. (10) with respect to the (6D) motion parameters until convergence is attained or for a fixed maximum number of iterations.

Note that in calculating the photometric error, setting the color error to zero if the projection of P lies

outside S_2^k may introduce instability in the optimization process due to the discontinuity of the photometric error at the boundary of the silhouettes. Although this instability problem did not happen in our experiments in Section 4.5, it can be avoided by setting the photometric error to transition smoothly to zero outside the silhouette boundary.

Ideally the weighing constant τ in Eqs. (8) and (9) should be set based on the relative accuracy between camera calibration and color balancing. However since such accuracy information is difficult to obtain, we instead determine τ experimentally. Using a synthetic data set (see Section 4.5.1) with ground-truth motion, we apply the above temporal SFS algorithm with different values of τ and choose the one which gives the best estimation results as compared to the ground-truth motion. Once the optimal τ is found, it is fixed and used for all the experiments discussed in Section 4.5 (and Part II of this paper). Although this experimental approach of determining τ may not be optimal, in practice it works well for a wide varieties of sequences.

4.3. Visibility

4.3.1. Determining Visibility for Locating CSPs. To locate the Colored Surface Points using Eq. (4), the visibility of the 3D point $W_j^i(m, w)$ with respect to all K cameras is required. Here, we present a way to determine the visibilities *conservatively* using only the silhouette images. Suppose we are given a 3D point P and a set of silhouette images $\{S_j^k\}$ with camera centers $\{C^k\}$ and projection functions $\{\Pi^k()\}$. The following lemma then holds:

Lemma 4. *Let $\Pi^l(P)$ and $\Pi^l(C^k)$ be the projections of the point P and the kth camera center C^k on the (infinite) image plane of camera l. If the 2D line segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ does not intersect the silhouette image S_j^l, then P is visible with respect to camera k at time t_j.*

Figure 9(a) gives examples where the points P_1, P_2 and P_3 are visible with respect to camera 2. The converse of Lemma 4 is *not* necessarily true: the visibility *cannot* be determined if the segment joining $\Pi^l(P)$ and $\Pi^l(C^k)$ intersects the silhouette S_j^l. One counter example is shown in Fig. 9(a). Both points P_1 and P_2 project to the same 2D point p on the image plane of camera 1 and the segment joining p and $\Pi^1(C^4)$ intersects with S_1^1. However, P_1 and P_2 have different visibilities with respect to camera 4 (P_2 is visible while P_1 is not). Note that special attention must be given to situations in which camera center C^k lies behind camera center C^l. In such cases, the correct line segment to be used in Lemma 4 is the outer line segment (passing through infinity) joining $\Pi^l(P)$ and $\Pi^l(C^k)$ rather than the direct segment. An example is given in Fig. 9(b).

Though conservative, there are two advantages of using Lemma 4 to determine visibility for locating CSPs. First, Lemma 4 uses information directly from the silhouette images, avoiding the need to estimate the shape of the object for the visibility test. Secondly, recall that to construct a Bounding Edge E_j^i, we start with the boundary point u_j^i of the kth silhouette. Hence all the points on E_j^i project to the same 2D point u_j^i on camera k which implies all points on the Bounding Edge E_j^i have the same set of conservative visible images. This property ensures that the color consistencies of points on the same Bounding Edge are calculated from the same set of images. Accuracy in searching for the touching point W_j^i is increased because the comparisons are made using the same images for all of the points on the same Bounding Edge.

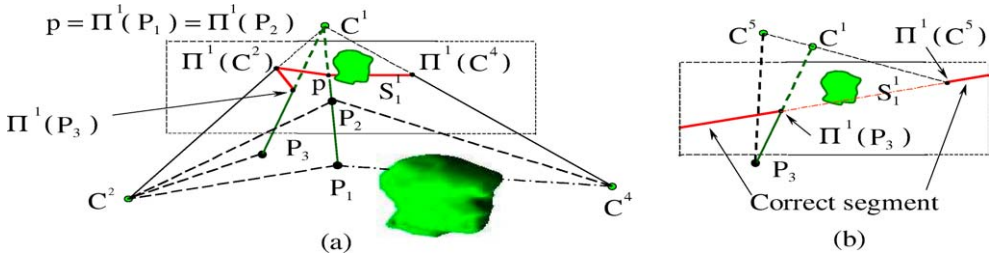

Figure 9. (a) Visibility of points with respect to cameras using Lemma 4. (b) An example where C^5 is behind C^1. The correct line to be used in Lemma 4 is the outer segment which passes through infinity instead of the direct segment.

4.3.2. Determining Visibility During Alignment. To perform the alignment using Eq. (10), we have to determine the visibility of the transformed 3D point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ with respect to the cameras at time t_2 (and vice versa the visibility for the transformed point $\boldsymbol{R}^T(W_2^i - \boldsymbol{t})$ with respect to the cameras at time t_1). Naively, we can just apply Lemma 4 to the transformed point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ directly. In practice, however, this "direct approach" does not work for the following reason. Since the CSP W_1^i lies on the surface of the object, the projection of the transformed point $\boldsymbol{R}W_1^i + \boldsymbol{t}$ should lie *inside* the silhouettes at time t_2, unless it happens to be on the occluding contour of the object again at t_2 such that its projection lies on the boundary of some of the silhouette images. Either way, this means that no matter where the camera centers are, the line joining the projection of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ and the camera centers almost always intersects the silhouettes. Hence, the visibility of the point W_1^i at t_2 will almost always be treated as indeterminable by Lemma 4 due to its over-conservative nature.

Here we suggest a "reverse approach" to deal with this problem. Instead of applying the transformation $(\boldsymbol{R}, \boldsymbol{t})$ to the point W_1^i, we apply the inverse transform $(\boldsymbol{R}^T, -\boldsymbol{R}^T\boldsymbol{t})$ to the camera centers and project the transformed camera centers into the one silhouette image (captured at t_1) where W_1^i is originated from as shown in Fig. 10. Lemma 4 is then applied to the boundary point u_1^i (which generates the Bounding Edge E_1^i that W_1^i lies on) and the projections of the transformed camera centers to determine the visibility. Since the object is rigid, the reverse approach generates the correct visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ with respect to the cameras at t_2 as the direct approach when $(\boldsymbol{R}, \boldsymbol{t})$ is the correct alignment.

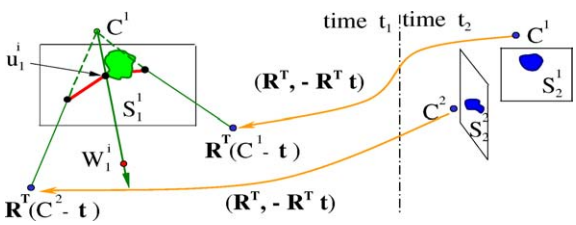

Figure 10. The "Reverse approach" of applying Lemma 4 to determine visibility of $\boldsymbol{R}W_1^i + \boldsymbol{t}$ with respect to $\{S_2^k\}$. The camera centers are inversely transformed by $(\boldsymbol{R}^T, -\boldsymbol{R}^T\boldsymbol{t})$ and then projected onto $\{S_1^k\}$. The visibility can then be determined by checking if the lines joining u_1^i and the projections of the transformed camera centers intersect with S_1^1 exactly as in Lemma 4.

4.4. Visual Hull Refinement

After estimating the alignment across time, the rigid motion $\{(\boldsymbol{R}_j, \boldsymbol{t}_j)\}$ is used to combine the J sets of silhouette images $\{S_j^k; k = 1, \ldots, K; j = 1, \ldots, J\}$ to get a tighter upper bound on the shape of the object. By fixing t_1 as the reference time, we combine $\{S_j^k\}; j = 2, \ldots J$ with $\{S_1^k\}$ by considering the former as "new" silhouette images captured by additional cameras placed at positions and orientations transformed by $(\boldsymbol{R}_j, \boldsymbol{t}_j)$. In other words, for the silhouette image S_j^k captured by camera k at time j, we use a new perspective projection function $\Pi_{j\to 1}^k$ derived from Π^k through the rigid transformation $(\boldsymbol{R}_j, \boldsymbol{t}_j)$. As a result, the effective number of cameras is increased from K to KJ.

4.5. Experimental Results

Two types of sequences are used to demonstrate the validity of our alignment and refinement algorithm. Firstly, a synthetic sequence is used to obtain a quantitative comparison of several aspects of the the algorithm. Two sets of experiments are run on the synthetic sequence. Experiment Set A compares the effectiveness of using (1) Colored Surface Points to align Visual Hulls with (2) voxel models created by Shape-From-Silhouette and (3) Space Carving (Kutulakos and Seitz, 2000). Experiment Set B studies how the alignment accuracy is affected by each component, color and geometry in the error measure in Eqs. (8) and (9). After we have tested our alignment algorithm on synthetic data, sequences of real objects are used in Section 4.5.2 for a qualitative evaluation on data with real noise, calibration errors and imperfectly color balanced cameras. Note that in all of the sequences discussed in this paper, the motion of the object is aligned with respect to the first frame of the sequence and we use the alignment results of frame $j-1$ to initialize the alignment of frame j.

4.5.1. Synthetic Data Set: Torso Sequence. A synthetic data set was created using a textured computer mesh model resembling the human torso. The model was moved under a known trajectory for twenty two frames. At each time instant, images of six cameras ($K = 6$) with known camera parameters were rendered using OpenGL. A total of 22 sets of color and silhouette images were generated. The textured mesh model and some input images

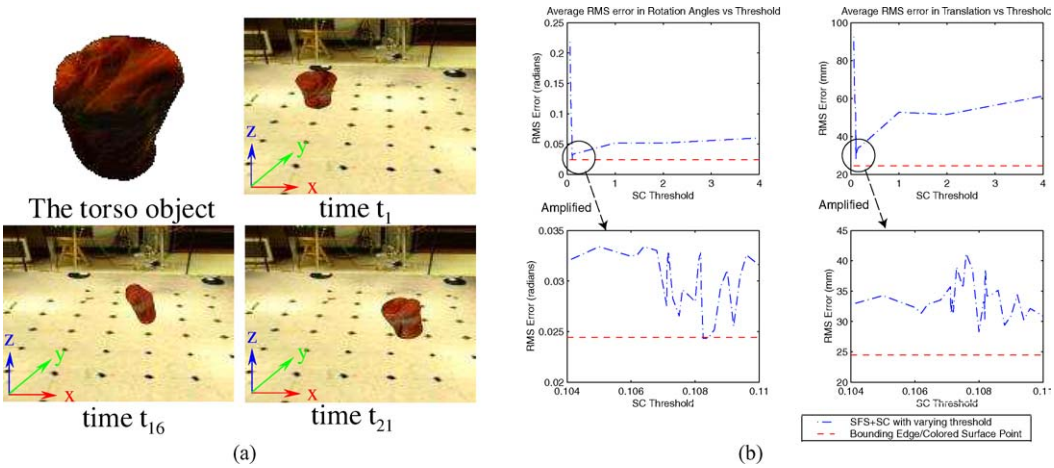

Figure 11. (a) The torso object and some of the input images of camera 1 of the synthetic torso sequence. (b) Graphs of the average RMS errors in rotation and translation against the threshold used in SC. The bottom half of the figure illustrates the amplified part of the graph near the optimal threshold value (0.108). Using Bounding Edges (the red dashed line) is always more accurate than using SC in alignment, even with the optimal threshold.

for camera 1 at a variety of frames are shown in Fig. 11(a).

Experiment Set A: BE/CSP versus SFS and SC

In Experiment Set A three algorithms were implemented to show the effectiveness of using Bounding Edges/Colored Surface Points to align Visual Hulls compared to using voxel models created by Shape-From-Silhouette (SFS) and Space Carving (SC) (Kutulakos and Seitz, 2000). Basically all the three algorithms use the same alignment procedure described in Section 4.2.2 but with input data (surface points) obtained from three different ways. In the first algorithm, BEs and CSPs are extracted and used as the input data for the alignment. In the second algorithm, a voxel model is built from the silhouette images using voxel-based SFS. Surface voxels are extracted and colored by back-projecting onto the color images. The centers of the colored surface voxels are then treated as input data points for alignment. In the third algorithm, a voxel model is first built using SFS (as in the second algorithm) and further refined by Space Carving (SC). The centers of the surface voxels (which are already colored by SC) are used as input data for the alignment. Note that in all of the above three algorithms, only the color error measure is used in the optimization equations.

To investigate the effect of the space carving threshold (which determines if a voxel is carved away or not) on alignment, we vary the threshold value from 0 to 4.0 to generate the input data (see the description of the second algorithm above) and compare the estimated motion parameters with the ground-truth values. Graphs of the average RMS errors in the rotation and translation parameters against the threshold are shown as the blue dotted-dashed lines in Fig. 11(b). When the threshold is too small, many correct voxels are carved away, resulting in a voxel model much smaller than the actual object. When the threshold is too large, extra incorrect voxels are not carved away, leaving a voxel model bigger than the actual object. In both cases, the wrong data points extracted from the incorrect voxel models cause errors in the alignment process. The optimal threshold value is found to be around 0.108 and the graph is amplified in the vicinity of this value in the bottom part of Fig. 11(b). As a comparison, the average RMS errors for the rotation and translation parameters obtained from using BEs and CSPs is drawn as the horizontal red dashed line. With the optimal SC threshold, the performance of using SFS+SC voxel models is comparable but less accurate than that of using Bounding Edges and Colored Surface Points. The results of the estimation of the Y-axis rotation angle and the X-component of translation at each frame using the SFS+SC input data with the optimal threshold are plotted as thick blue dotted lines in Fig. 12(a) while the results of using the SFS surface centers as input data are plotted as magenta dotted-dashed lines. Also, the estimated parameters of using BEs/CSPs as input data are plotted as red dashed lines with asterisks, together with

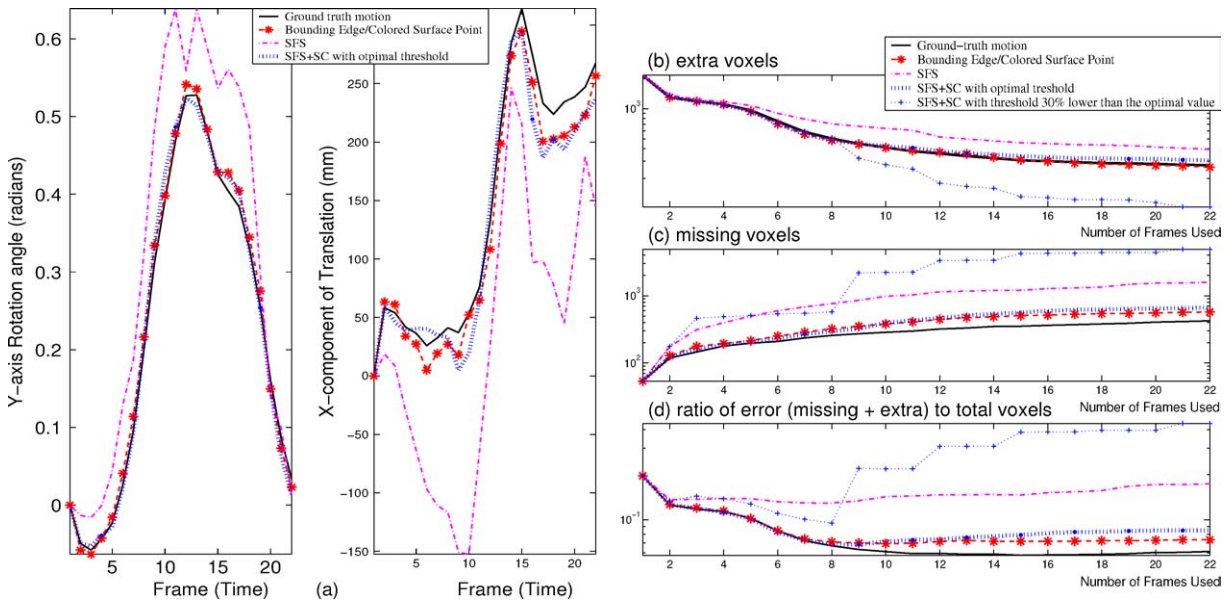

Figure 12. (a) Alignment results for the Y-axis rotation angle and the X-component of translation estimated at each frame (time) from Experiment Set A with different inputs: BEs/CSPs (red dashed lines with asterisks), SFS voxel models (magenta dotted-dashed lines), SFS+SC voxel models with the optimal threshold (blue thick dotted lines) and the ground-truth motion (solid black lines). Using BEs/CSPs is better than using either SFS or SFS+SC. (b), (c) and (d) Graphs of the refinement errors (missing and extra voxels) against the total number of frames used. Using BEs/CSPs has a lower error ratio than using either SFS or SFS+SC.

the ground-truth motion in solid black lines in the same figure. As can be seen, alignment using the SFS voxel model is much less accurate than using BEs/CSPs. SC with the optimal threshold performs well, but not quite as well as using BEs/CSPs. The results of all the motion (translation and rotation) parameters can be found in Cheung (2003).

To study the effect of alignment on refinement, the parameters estimated by the alignment algorithms were used to refine the shape of the torso model using the voxel-based SFS method as described in Section 4.4. The size of voxels used was 7.8 mm × 7.8 mm × 7.8 mm whereas the size of the original torso mesh model was approximately 542 mm × 286 mm × 498 mm. Since the mesh model cannot be used directly to compare with the refined voxel models, we converted the original mesh model into an *reference* voxel model and used it to quantify the refinement results. We are interested in two types of error voxels: (1) extra and (2) missing voxels. Due to the conservative nature of SFS, any voxel model constructed with finite number of silhouette images will always have extra voxels as compared to the actual object (the *reference* voxel model in this case) and the number of extra voxels decreases with the number of images used. On the other hand, since the synthetic silhouettes are perfect, missing voxels are the results of (1) voxel decision problem around the boundary of the silhouettes (see Cheung, 2003 for details) and (2) misalignment of motion across frames. Since the effect of the boundary problem is the same for all of the algorithms, the number of missing voxels indicates how the misalignment affects the refinement.

The quantitative refinement results are plotted in Figs. 12(b) and (c) which show respectively the number of extra and missing voxels between the refined shapes and the *object* voxel models against the total number of frames used. Figure 12(d) illustrates the ratio of total incorrect (missing plus extra) to total voxels. In all of the refinement results, the number of extra voxels decreases as the number of frames used increases as discussed above because a tighter Visual Hull is obtained with an increase in the number of silhouette images. However, the number of missing voxels also increases as the number of frames used increases due to alignment errors which remove correct voxels during construction. From the figure it can be seen that the number of missing voxels is very large if the alignments are way off (e.g. the magenta dotted-dashed curve for the SFS voxel centers or the blue dotted curves with '+' markers for SFS+SC with threshold 30% lower

than the optimal value). The best refinement results are the ones using the motion parameters estimated using BEs/CSPs (the red dashed lines with asterisks in Figs. 12(b), (c) and (d)).

Experiment Set B: Effect of Error Measure on the Alignment Accuracy

Experiment Set B investigates the effect of using color consistency and the geometric constraints as error measure on the alignment accuracy. In the first algorithm, only the error from the geometrical constraints is used (i.e. the first term $d_2^k(\boldsymbol{R}W_1^i+\boldsymbol{t})$ in Eq. (8)). In the second algorithm, only the error caused by the color inconsistency is used (i.e. the second term $[\boldsymbol{c}_2^k(\boldsymbol{R}W_1^i+\boldsymbol{t})-\mu_1^i]^2$ in Eq. (8)). In the third algorithm, both errors are used. The results for the Y-axis rotation angle and the X-axis translation component are shown in Figs. 13(a). In the figure, the ground-truth motion values are drawn with solid black lines, the results obtained from using both geometric constraints and color consistency are drawn with magenta dotted lines with an inverted triangle, the results with only the geometric constraints are drawn with blue dashed-dotted lines with circle, and the results with only color consistency are drawn with red dashed lines with asterisks. As expected, the results of using both error components are the best, followed by the results using only the color consistency. The results obtained using only the geometric constraints are the worst of the three. As discussed in Section 4.1.1, aligning Visual Hulls using only geometric (silhouette) information is inherently ambiguous. This means that if color consistency (the second term of Eq. (8)) is not used, there may be more than one global minimum to Eq. (10) (see the 2D example in Fig. 6). Under such situations, optimizing Eq. (10) may converge to a global minimum other than the actual motion of the object. This explains why the results of using only the silhouette information are not as good as using only color information, or both the silhouette and color information.

The refinement results of Experiment Set B are plotted in Figs. 13(b)–(d) which illustrate respectively the extra and missing voxels and the ratio of total incorrect (missing plus extra) to total voxels against the total number of frames used for refinement. The results are the best with the motion parameters estimated using both the color consistency and the geometric constraints (the magenta dotted lines with inverted triangle). Again just using the color

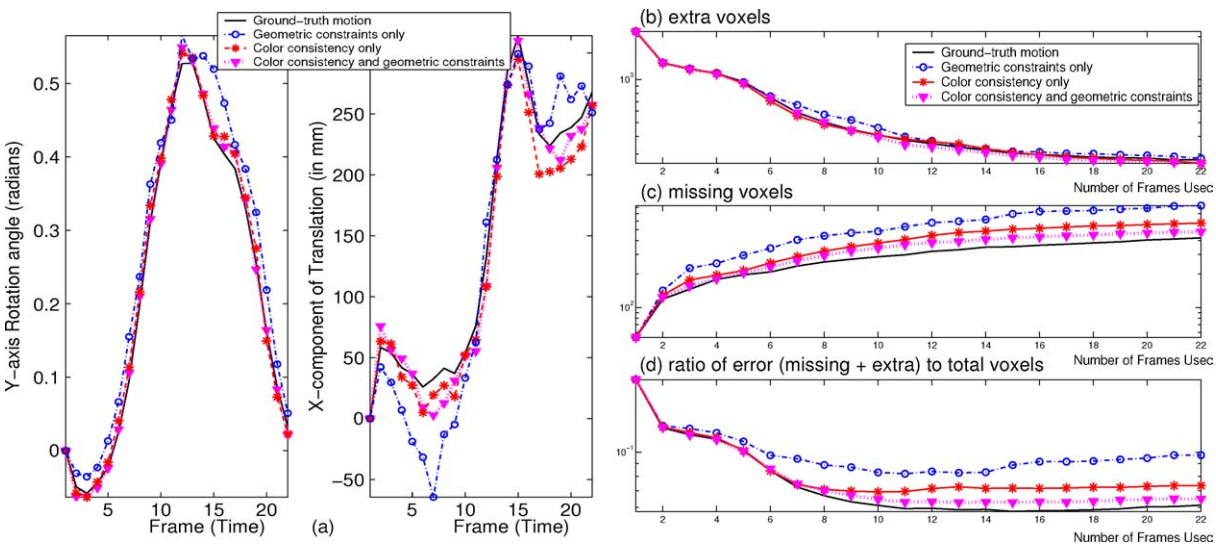

Figure 13. (a) Results of the Y-axis rotation angle and the X-component of translation estimated at each frame for Experiment Set B with different error measures: geometric constraints only (blue dashed-dotted lines with circle), color consistency only (red dashed lines with asterisks), both geometric constraints and color consistency (magenta dotted lines with inverted triangle). The solid black lines represents the ground-truth motion. The results obtained using both error components are the best followed by the results using only the color consistency. Due to the alignment ambiguity, the results using only the geometrical constraints are the worst of the three. (b), (c) and (d) The refinement errors (missing and extra voxels) against the total number of frames used. Using both the color consistency and the geometric constraints has lower error than just using either one of them.

consistency is better than just using geometric constraints. A video clip **Torso.mpg**[2] shows one of the six input image sequences, the unaligned and aligned Colored Surface Points and the temporal refinement/alignment results using BEs/CSPs computed with both the geometric and photometric error measures.

4.5.2. Real Data Sets: Toy Pooh and Dinosaur/ Bananas.

A. Pooh Sequence. The first test object is a toy (Pooh) with six calibrated cameras. The toy is placed on a table and moved to new but unknown positions and orientations manually in each frame. A total of fifteen frames are captured from each camera. The input images of camera 1 at several times are shown in Fig. 14(a). The CSPs extracted at time t_1 are shown in Fig. 14(b). Figures 14(c) and (d) show respectively the unaligned and aligned Colored Surface Points from all fifteen frames. It can be seen that since some part of the body of the toy is uniform in color, the positions of a few CSPs are not correctly estimated. However, since there are only a few of them and their deviations are small, the alignment is still very accurate. This shows the robustness of our alignment algorithm that as long as the number of incorrect CSPs are small, the algorithm works well. Refinement is done using the voxel-based SFS method. Figures 14(e)–(g) illustrate the refinement results at time instants t_1 (6 images), t_6 (36 images) and t_{15} (90 images). The improvement in shape is very significant from t_1 when 6 silhouette images are used to t_{15} when 90 silhouette images are used. The video clip **Pooh.mpg** shows some of the input sequences, the unaligned/aligned CSPs and the temporal refinement/alignment results for this sequence.

B. Dinosaur-Banana Sequence. The objects used in the second real data set are the toy dinosaur/bananas shown in Fig. 1(a). Six cameras are used and the dinosaur/bananas are placed on a turn-table with unknown rotation axis and rotation speed. Fifteen frames are captured and the alignment and refinement results are shown in Fig. 15. The video clip **Dinosaur-Banana.mpg** shows one of the six input image sequences, the unaligned/aligned Colored Surface Points and the temporal refinement/alignment results of the Dinosaur-Banana Sequence. Note that we have also applied the temporal SFS algorithm for rigid objects to sequences of a person standing rigidly on a turn-table. The results will be presented in Part II of this paper when we describe a system for building kinematic models of humans.

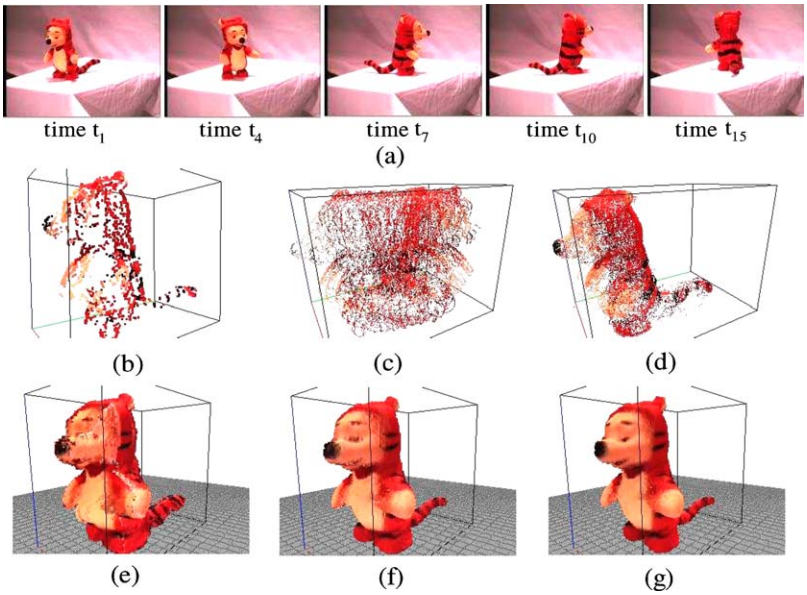

Figure 14. Pooh Data Set. (a) Some of the input images from camera 1. (b) Colored surface points at t_1. (c) Unaligned Colored Surface Points from all frames. (d) Aligned Colored Surface Points of all frames. (e) SFS model at t_1 (6 images used). (f) SFS refined shape at t_6 (36 images used). (g) SFS refined shape at t_{15} (90 images used). See **Pooh.mpg** for a movie illustrating these results.

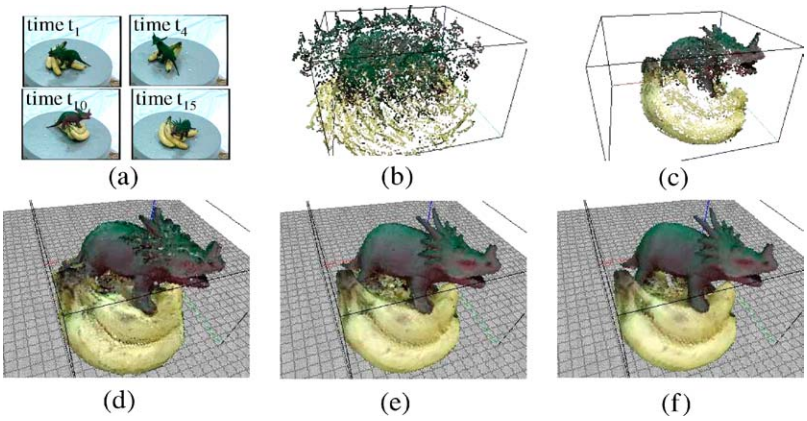

Figure 15. Dinosaur-Banana Sequence. (a) Example input images. (b) Unaligned Colored Surface Points from all frames. (c) Aligned Colored Surface Points from all frames. (d) SFS model at t_1 (6 images used). (e) SFS refined shape at t_6 (36 images used). (f) SFS refined shape at t_{15} (90 images used). There is significant shape improvement from (d) to (f). See **Dinosaur-Banana.mpg** for movie illustration.

4.5.3. Related Work. Despite the popularity of SFS as a shape reconstruction method at single time instant, little work has been done in extending it across time. The work most related to ours is by Mendonca et al. (2000) and Wong and Cipolla (2001a, 2001b) who study the problem of estimating structure and motion of a smooth object undergoing *circular motion* from silhouette profiles. They assume a single camera which is weakly calibrated (i.e. with known intrinsic but unknown extrinsic parameters). Either the camera (on a robotic arm) or the object (on a turntable) performs unknown circular motion while the silhouette images are taken. In Mendonca et al. (2001) symmetric properties of the surface of revolution swept by the rotating object are used to recover the revolution axis, leading to the estimation of homographies and full epipolar geometries between images using one-dimensional search. In Wong and Cipolla (2001b), they identity and estimate the *frontier points* (see Joshi et al., 1994 for the definition) on the silhouette boundary and use them to estimate the circular motion between images. Once the motion has been estimated, the object shape can be reconstructed using the classic SFS method.

Another group of researchers, lead by Joshi et al. (1994, 1995) and Vijayakumar et al. (1996) have also studied the problem of recovering the motion and shape of a *smooth curved* object from silhouette images. They define a local parabolic structure on the surface of the object and use that, together with epipolar geometry, to locate corresponding frontier points on three silhouette images. The motion between the images is then estimated using a two-step nonlinear minimization. In contrast to these algorithms, our approach has two advantages: (1) no shape assumptions are made about the object and (2) no assumptions are made about the motion (i.e. it does not have to be infinitesimal).

5. SFS Across Time: Articulated Objects

In this section we extend our temporal SFS algorithm to articulated objects. An object is articulated if it consists of a set of rigidly moving parts connected to each other at certain articulation points. A good example of an articulated object is the human body (if we approximate the body parts as rigid). Given CSPs of a moving articulated object, recovering the shape and motion requires two inter-related steps: (1) correctly segmenting the CSPs to each part of the object and (2) estimating the shape and motion of the individual parts. To solve this problem, we employ an idea similar to that used for multiple-layer motion estimation in Sawhney and Ayer (1996). The rigid parts of the articulated object are first modeled as separate and independent of each other. With this assumption, we iteratively (1) assign the extracted CSPs to different parts of the object based on their motions and (2) apply the rigid temporal SFS algorithm to align each part across time. Once the motions of the parts have been recovered, an articulation constraint is applied to estimate the joint positions. Note that this iterative approach can be categorized as belonging to the Expectation Maximization framework (Dempster et al., 1977). The whole algorithm is explained below using a two-part, one-joint articulated object.

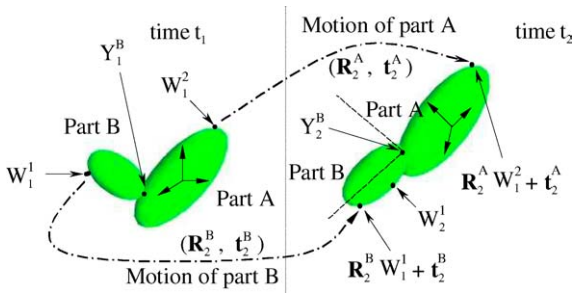

Figure 16. A two-part articulated object at two time instants t_1 and t_2.

5.1. Problem Scenario

Consider an unknown one-joint articulated object O which consists of two rigid parts A and B as shown in Fig. 16 at two time instants t_1 and t_2. Assume CSPs of the whole object have been extracted from the color and silhouette images of K cameras, denoted by $\{I_j^k, S_j^k, W_j^i, \mu_j^i; j = 1, 2\}$. Furthermore, treating A and B as two independently moving rigid objects allows us to represent the relative motion of A between t_1 and t_2 as $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A)$ and that of B as $(\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$. Now consider the following two complementary cases.

5.2. Alignment with Known Segmentation

Suppose we have segmented the CSPs at t_j into two groups belonging to part A and part B, represented by G_j^A and G_j^B respectively for both $j = 1, 2$. By applying the rigid object temporal SFS algorithm described in Section 4.2.3 (Eq. (10)) to A and B separately, estimates of the relative motions $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A)$, $(\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$ can be obtained.

5.3. Segmentation with Known Alignment

Assume we are given the relative motion $(\boldsymbol{R}_2^A, \boldsymbol{t}_2^A)$, $(\boldsymbol{R}_2^B, \boldsymbol{t}_2^B)$ of A and B from t_1 to t_2. For a CSP W_1^i at time t_1, consider the following two error measures:

$$e_{2,1}^{i,A} = \frac{1}{n_1^{i,A}} \sum_k \left\{ \tau * d_2^k\left(\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A\right) + \left[\boldsymbol{c}_2^k\left(\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A\right) - \mu_1^i\right]^2 \right\}, \quad (11)$$

$$e_{2,1}^{i,B} = \frac{1}{n_1^{i,B}} \sum_k \left\{ \tau * d_2^k\left(\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B\right) + \left[\boldsymbol{c}_2^k\left(\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B\right) - \mu_1^i\right]^2 \right\}. \quad (12)$$

Here $e_{2,1}^{i,A}$ is the error of W_1^i with respect to the color/silhouette images at t_2 if it belongs to part A. Similarly $e_{2,1}^{i,B}$ is the error if W_1^i lies on the surface of B. In these expressions the summations are over those cameras where the transformed point is *visible* and $n_1^{i,A}$ and $n_1^{i,B}$ represent the number of visible cameras for the transformed points $\boldsymbol{R}_2^A W_1^i + \boldsymbol{t}_2^A$ and $\boldsymbol{R}_2^B W_1^i + \boldsymbol{t}_2^B$ respectively. By comparing the two errors in Eqs. (11) and (12), a simple strategy to classify the point W_1^i is:

$$W_1^i \in \begin{cases} G_1^A & \text{if } e_{2,1}^{i,A} < \kappa * e_{2,1}^{i,B} \\ G_1^B & \text{if } e_{2,1}^{i,B} < \kappa * e_{2,1}^{i,A} \\ G_1^{\emptyset} & \text{otherwise} \end{cases}, \quad (13)$$

where $0 \le \kappa \le 1$ is a thresholding constant and $G_1^{\emptyset}$ contains all the CSPs which are classified as neither belonging to part A nor part B. Similarly, the CSPs at time t_2 can be classified using the errors $e_{1,2}^{i,A}$ and $e_{1,2}^{i,B}$. In practice, the above decision rule does not work very well on its own because of image/silhouette noise and camera calibration errors. Fortunately we can use spatial coherency and temporal consistency to improve the segmentation.

To use spatial coherency, the notion of a spatial neighborhood has to be defined. Since it is difficult to define a spatial neighborhood for the scattered CSPs in 3D space (see for example Fig. 7(b)), an alternate way is used. Recall (in Section 3.1) that each CSP W_1^i lies on a Bounding Edge which in turn corresponds to a boundary point u_1^i of the silhouette image S_1^k. We define two CSPs W_1^i and W_1^{i+1} as "neighbors" if their corresponding 2D boundary points u_1^i and u_1^{i+1} are neighboring pixels (in 8-connectivity sense) in the same silhouette image. This neighborhood definition allows us to easily apply spatial coherency to the CSPs. From Fig. 17(a) it can be seen that different parts of an articulated object *usually* project onto the silhouette image as continuous outlines. Inspired by this property, the following spatial coherency rule (SCR) is proposed.

Spatial Coherency Rule (SCR). If W_1^i is classified as belonging to part A by Eq. (13), it stays as belonging to part A if all of its m left and right immediate "neighbors" are also classified as belonging to part A by Eq. (13), otherwise it is reclassified as belonging to $G_1^{\emptyset}$, the group of CSPs that belongs to neither part A nor part B. The same procedure applies to part B.

Figure 17(a) shows how the SCR can be used to remove spurious segmentation errors. The second constraint we utilize to improve the segmentation results is

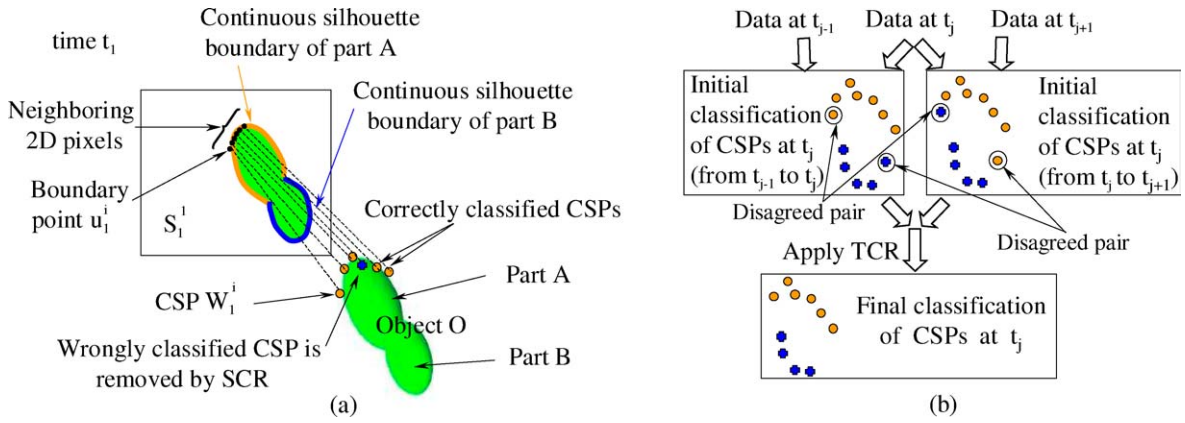

Figure 17. Spatial Coherency Rule removes spurious segmentation errors.

temporal consistency as illustrated in Fig. 17(b). Consider three successive frames captured at t_{j-1}, t_j and t_{j+1}. For a CSP W_j^i, it has two classifications due to the motion from t_{j-1} to t_j and the motion from t_j to t_{j+1}. Since W_j^i either belongs to part A or B, the temporal consistency rule (TCR) simply requires that the two classifications have to agree with each other:

Temporal Consistency Rule (TCR). If W_j^i has the same classification by SCR from t_{j-1} to t_j and from t_j to t_{j+1}, the classification is maintained, otherwise, it is reclassified as belonging to $G_j^{\emptyset}$, the group of CSPs that belongs to neither part A nor part B.

Note that SCR and TCR not only remove wrongly segmented points, but they also remove some of the correctly classified CSPs. Overall though they are effective because less but more accurate data is preferred to abundant but inaccurate data, especially in our case where the segmentation has a great effect on the motion estimation.

5.4. Initialization

As common to all iterative EM algorithms, initialization is always a problem (Sawhney and Ayer, 1996). Here we suggest two different approaches to start our algorithm. Both approaches are commonly used in the layer estimation literature (Sawhney and Ayer, 1996; Ke and Kanade, 2001). The first approach uses the fact that the 6 DOF motion of each part of the articulated object represents a single point in a six dimensional space. In other words, if we have a large set of estimated motions of all the parts of the object, we can apply a clustering algorithms to these estimates in the 6D space to separate the motion of each individual part. To get a set of estimated motions for all the parts, the following method can be used. The CSPs at each time instant are first divided into subgroups by cutting the corresponding silhouette boundaries into arbitrary segments. These subgroups of CSPs are then used to generate the motion estimates using the VH alignment algorithm, each time with a randomly chosen subgroup from each time instant. Since this approach requires the clustering of points in a 6D space, it performs best when the motions between different parts of the articulated object are relatively large so that the motion clusters are distinct from each other.

The second approach is applicable in situations where one part of the object is much larger than the other. Assume, say, part A is the dominant part. Since this assumption means that most of the CSPs of the object belong to A, the dominant motion $(\boldsymbol{R}^A, \boldsymbol{t}^A)$ of A can be approximated using all the CSPs. Once an approximation of $(\boldsymbol{R}^A, \boldsymbol{t}^A)$ is available, the CSPs are sorted in terms of their errors with respect to this dominant motion. An initial segmentation is then obtained by thresholding the sorted CSPs errors.

For a sequence of J frames, although we can initialize the segmentation of all frames together using one step, it is impractical especially when J is large. Instead we use a simpler approach and initialize the segmentation independently and separately using two (consecutive) frames at a time. Experimental results (see Section 5.7) show that this works well for different types of sequences.

5.5. *Summary: Iterative Algorithm*

Although we have described the algorithm above for an articulated object with two rigid parts, it can be generalized to apply to objects with N parts provided N is known. The following summarizes our iterative algorithm to estimate the shape and motion of parts A and B over J frames:

Iterative Temporal SFS Algorithm for Articulated Objects

1. Initialize the segmentation of the J sets of CSPs.
2. Iterate the following two steps until convergence (or for a fixed number of iterations):
   - 2a. Given the CSP segmentation $\{G_j^A, G_j^B\}$, recover the relative motions $(\boldsymbol{R}_j^A, \boldsymbol{t}_j^A)$ and $(\boldsymbol{R}_j^B, \boldsymbol{t}_j^B)$ of A and B over all frames $j = 2, \ldots, J$ using the rigid object temporal SFS algorithm described in Section 4.2.3.
   - 2b. Repartition the CSPs according to the estimated motions by applying Eq. (13), followed by the intra-frame SCR and then inter-frame TCR for all frame $j = 1, \ldots, J$.

5.6. *Joint Location Estimation*

After recovering the motions of parts A and B separately, the point of articulation between them is estimated. Suppose we represent the joint position at time t_1 as Y_1^B. Since Y_1^B lies on both A and B, it must satisfy the motion equation from t_1 to t_2 as $\boldsymbol{R}_2^A Y_1^B + \boldsymbol{t}_2^A = \boldsymbol{R}_2^B Y_1^B + \boldsymbol{t}_2^B$. Putting together similar equations for Y_1^B over J frames, we get

$$\begin{bmatrix} \boldsymbol{R}_2^A - \boldsymbol{R}_2^B \\ \vdots \\ \boldsymbol{R}_J^A - \boldsymbol{R}_J^B \end{bmatrix} Y_1^B = \begin{bmatrix} \boldsymbol{t}_2^B - \boldsymbol{t}_2^A \\ \vdots \\ \boldsymbol{t}_J^B - \boldsymbol{t}_J^A \end{bmatrix}. \tag{14}$$

The least squares solution of Eq. (14) can be computed using Singular Value Decomposition.

5.7. *Experimental Results*

5.7.1. Synthetic Data Set. We use an articulated mesh model of a virtual computer human body as the synthetic test subject. To generate a set of test sequences, the computer human model is programmed to only move one particular joint and the images of the movements are rendered using OpenGL. Since only one joint (and one body part) is moved at each time, we can consider the virtual human body as an one-link two part articulated object. A total of eight sets of data sequences (each set with 8 cameras) are generated, corresponding to the eight joints: left/right shoulder, elbow, hip and knee. For each of these synthetic sequences, we applied the articulated temporal SFS algorithm to recover the shape, motion and the joint location of the virtual human. Since the size of the whole body is much larger than a single part, the dominant motion initialization method is used. Figure 18 shows some input images from one of cameras and the segmentation/alignment/joint estimation results for the right elbow and right hip joints. As can be seen, our iterative segmentation/alignment algorithm performs well

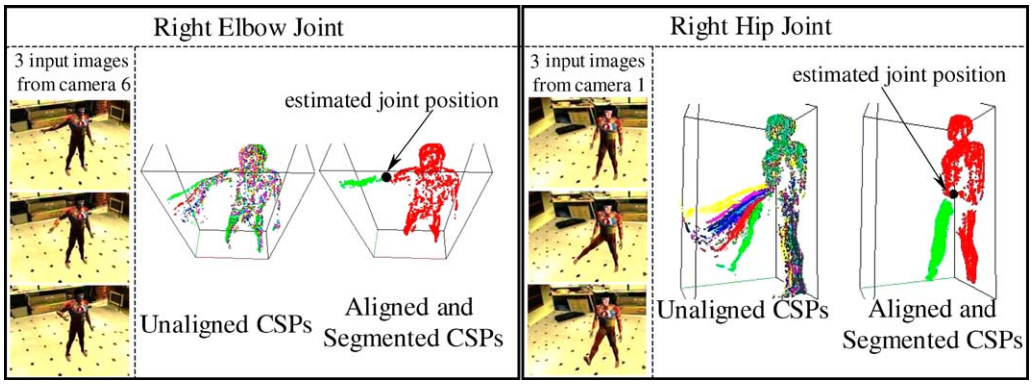

Figure 18. Input images and results for the right elbow and right hip joints of the synthetic virtual human. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere.

Table 1. The ground-truth and estimated positions of the eight body joints for the synthetic sequences. The absolute errors (averaging about 26 mm) is small compared to the actual size of the model (≈500 mm × 200 mm × 1750 mm).

Joints	Ground-truth (x, y, z) positions (in mm)	Estimated (x, y, z) positions (in mm)	Distance error (in mm)
Left Shoulder	(199.61, 66.06, 1404.75)	(203.40, 54.06, 1403.80)	12.62
Right Shoulder	(−200.34, 66.06, 1404.75)	(−206.09, 73.87, 1398.53)	11.52
Left Elbow	(411.75, −116.60, 1333.54)	(412.98, −119.61, 1323.23)	10.81
Right Elbow	(−407.00, 146.01, 1258.53)	(−398.89, 178.54, 1288.19)	44.76
Left Hip	(87.02, 43.32, 974.75)	(92.16, 40.46, 976.77)	6.22
Right Hip	(−91.65, 42.37, 979.51)	(−85.20, −2.13, 965.11)	47.21
Left Knee	(251.57, −438.03, 853.29)	(285.14, −432.44, 857.50)	34.29
Right Knee	(−143.90, −399.59, 723.32)	(−102.92, −393.13, 741.42)	45.27

and the joint positions are estimated accurately in both cases. Table 1 compares the ground-truth with the estimated joint positions for all the 8 synthetic sequences. The absolute distance errors between the ground-truth and the estimated joints locations are small (averaging about 26mm) when compared to the size of the human model (≈500 mm × 200 mm × 1750 mm). The input images, CSPs and the results for the left hip and knee joints are shown in the movie **Synthetic-joints-leftleg.mpg**.

5.7.2. Real Data Sets. Two different data sets with real objects were captured. The first real data set contains two separate, independently moving rigid objects while the second real data set investigates the performance of our articulated temporal SFS algorithm for the joint estimation for a real person.

A. Two Separately Moving Rigid Objects: Pooh-Dinosaur Sequence. The Pooh and dinosaur from Section 4.5.2 are used to test the performance of our iterative CSP segmentation/motion estimation algorithm on two separate and independently moving rigid objects. Eight calibrated cameras ($K = 8$) were used in this Pooh-Dinosaur sequence. Both toys are placed on the floor and individually moved to new but unknown positions and orientations manually in each frame. Fourteen frames were captured for each camera. Since the two objects are of comparable size but with large relative motion, we use the first initialization approach (clustering of motions) as described in Section 5.4 to initialize the alignment. Figure 19(a) shows some of the input images of camera 3. The segmentation/alignment results using our temporal SFS algorithm are illustrated in Figs. 19(b)–(f). Figure 19(b) shows the unaligned CSPs for all the 14 frames. Figure 19(c) shows the aligned and segmented CSPs. The figures demonstrate that our algorithm correctly segments the CSPs as belonging to each object. The alignments of both toys are also accurate except those of the dinosaur from frame 6 to frame 9 when the dinosaur rolled over. In those frames, our alignment algorithm failed as the rotation angles were too large (around 90 degrees). However, the alignment recovers after frame 9 when the dinosaur is upright again.

The shapes of the two toys were refined by SFS using the estimated motions in the same fashion as discussed in Section 4.4. Note that to refine the objects, there is no need to segment (which is difficult to do due to occlusion) the silhouettes as belonging to which object as long as the motions of the objects are significantly different from each other for at least one frame. The voxels that do not belong to the dinosaur, say, would be carved away by SFS over time as they do not follow the motion of the dinosaur. Figures 19(d)–(f) illustrate the SFS refined voxel models of both objects at t_1, t_5 and t_{13} respectively. Since the alignment data for the dinosaur from frame 6 to frame 9 are inaccurate, those frames were not used to refine the shape of the dinosaur. As can be seen, significant shape improvement is obtained from t_1 to t_{13}. The video clip **Pooh-Dinosaur.mpg** shows the input images from one of the eight cameras, the unaligned/aligned/segmented CSPs and the temporal refinement results.

B. Joints of Real Human. In the second set of real data, we used videos of a person (SubjectE) to qualitatively test the performance of our articulated object

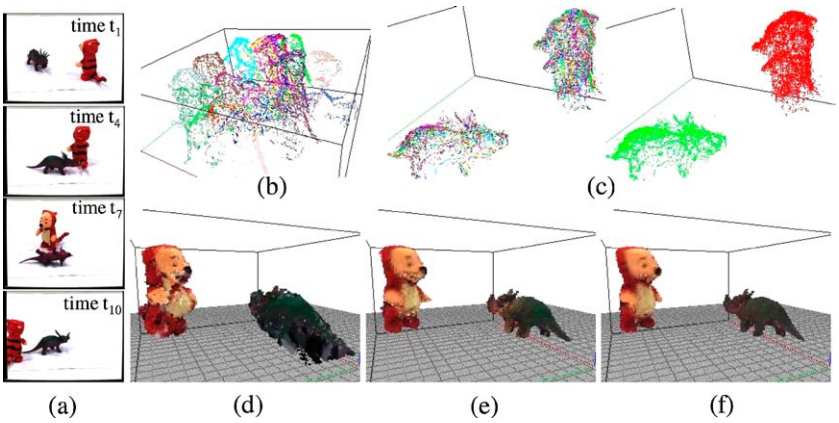

Figure 19. The Pooh-Dinosaur sequence. (a) Some of the input images from camera 3. (b) The unaligned CSPs from all frames. (c) The aligned and segmented CSPs. (d) SFS refined voxel models at t_1 (8 silhouette images are used). (e) SFS refined voxel models at t_5 (40 silhouette images are used). (f). SFS refined voxel models at t_{13} (104 silhouettes are used for the toy Pooh and 72 silhouette images are used for the dinosaur).

temporal SFS algorithm for joint location estimation. Eight sequences (each with 8 cameras) corresponding to the movement of the left/right shoulder, elbow, hip, knee joints of SubjectE were captured. In each sequence, SubjectE only moves one of her joints so that in that sequence her body can be considered as an one-joint, two part articulated object, exactly as the synthetic data set. Again, the dominant motion initialization method is used. Some of the input images and the results of segmentation/alignment/position estimation for two joints (left elbow and left hip) are shown in Fig. 20. As can be seen, the motion, the segmentation of the body parts, and the joint locations are all estimated correctly in both sequences. Some of the input images, the CSPs and the segmentation/estimation results of the right arm joints for SubjectE can be found in the movie clip **SubjectE-joints-rightarm.mpg**. Note that the joint estimation results for another two subjects SubjectG and SubjectS can be found in Part II of this paper when we discuss our human body kinematic modeling system.

5.7.3. Related Work. Though the work by Krahnstoever et al. (2001, 2003) uses only monocular images, their idea is very similar to ours in the sense that it is also based on the the layered motion segmentation/estimation formulation (Sawhney and Ayer, 1996). They first perform an EM-like segmentation/motion estimation of 2D regions on monocular images of the articulated object and then model

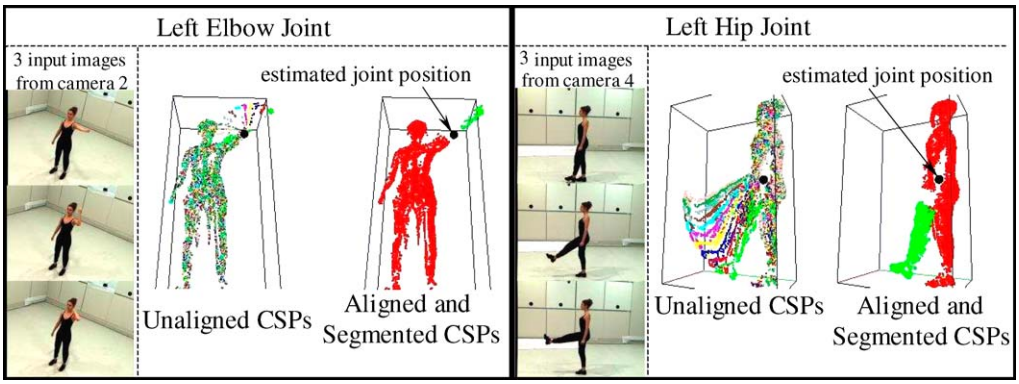

Figure 20. Input images and results for the left elbow and left hip joints of SubjectE. For each joint, the unaligned CSPs from different frames are drawn with different colors. The aligned and segmented CSPs are shown with two different colors to show the segmentation. The estimated articulation point (joint location) is indicated by the black sphere.

the articulated parts by 2D cardboard models. As common to other monocular methods, their approach does not handle occlusion and has difficulties estimating the motion of objects which do not contain rotation around an axis perpendicular to the image plane.

6. Conclusion

In this paper we have developed a theory of performing Shape-From-Silhouette across time for both rigid objects and articulated objects undergoing arbitrary and unknown motion. We first studied the ambiguity of aligning two Visual Hulls, and then proposed an algorithm using stereo to break the ambiguity. We first represented each Visual Hull using Bounding Edges. Colored Surface Points are then located on the Bounding Edges by comparing color consistencies. The Colored Surface Points are used to estimate the rigid motion of the object across time, using a 2D images/3D points alignment algorithm. Once the alignment has been computed, all of the images are considered as being captured at the same instant. The refined shape of the object can then be obtained by any reconstruction method such as SFS or Space Carving.

Our algorithm combines the advantages of both SFS and Stereo. A key principle behind SFS, expressed in the Second Fundamental Property of Visual Hulls, is naturally embedded in the definition of the Bounding Edges. The Bounding Edges incorporated, as a representation for the Visual Hull, a great deal of the accurate shape information that can be obtained from the silhouette images. To locate the touching surface points, multi-image stereo (color consistency among images) is used. Two major difficulties of doing stereo : visibility and search size are both handled naturally using the properties of the Bounding Edges. The ability to combine the advantages of both SFS and Stereo is the main reason why using Bounding Edges/Colored Surface Points gives better results in motion alignment than using voxel models obtained from SFS or SC (see Section 4.5.1). Another disadvantage of using voxel models and Space Carving is that each decision (voxel is carved away or not) is made *individually* for each voxel according to a criterion involving thresholds. On the contrary, in locating colored surface points on Bounding Edges, the decision (which point on the Bounding Edge touches the object) is made *cooperatively* (by finding the point with the highest color consistency) along all the points on the Bounding Edge, without the need of adjusting thresholds. In summary, the information contained in Bounding Edges/Colored Surface Points is more accurate than that contained in voxel models constructed from SC/SFS. In parameter estimation, few but more accurate data is always preferred over abundant but less inaccurate data, especially in applications such as alignment.

We also extended our Temporal SFS algorithm to (piecewise rigid) articulated objects and successfully applied it to solve the problems of segmenting CSPs and recovering the motions of two independently moving rigid objects and joint positions estimation for the human body. The advantage of our algorithm is that it solves the difficult problem of shape/motion/joint estimation by a two-step approach: first iteratively recover the shape (in terms of CSP) and the motion of the individual parts of the articulated object and then locate the joint using a simple motion constraint. The separation of the joint estimation and the motion estimation greatly reduces the complexity of the problem. Since our algorithm uses motion to segment the CSPs, it fails when the relative motion between the parts of the articulated objects is too small. Moreover, due to the EM formulation of the algorithm, the convergence of the algorithm depends on the initial estimates of the motion parameters. When the initial motion estimates are too far from the correct values, the algorithm may fall into a local minimum. Finally, although the algorithm can be generalized to apply to objects with N parts, in practice it does not work well when there are more than four parts due to the local minimum problem.

In Part II of this paper we will show how our Temporal SFS algorithms can be used to build a kinematic model of a person, consisting of detailed shape and precise joint information. The kinematic model is then used to perform vision-based (markerless) motion capture.

6.1. Future Work

While our temporal SFS algorithm can be used to recover the motion and shape of moving rigid and articulated objects, a lot of naturally occurring objects are non-rigid or deformable. A rational future direction is to extend our temporal SFS algorithms to deformable objects such as a piece of cloth or a crawling caterpillar. There are two major difficulties in extending temporal SFS to non-rigid objects. The first difficulty, which is common to other surface-point-based 3D shape/motion estimation methods (Aggarwal et al., 1994), is to assume suitable shape and motion models

for the object. The choice of the deformable model is critical and depends on the application. The second difficulty is caused by the fact that since our temporal SFS algorithm is not feature-based, the CSPs are not tracked over time and there is no point-to-point correspondence between two sets of CSPs extracted at different instants. Hence, it is unclear how the chosen deformable model can be applied to the CSPs across time. Despite these difficulties, the possibility of extending temporal SFS to non-rigid objects is worth studying as it would help solve important non-rigid tracking problems in computer vision.

Notes

1. Proofs of all the lemmas in this paper can be found at Cheung (2003).
2. All of the movie clips can be found at http://www.cs.cmu.edu/~german/research/Journal/IJCV/Theory/. Lower resolution versions of some of the movies are also included in the supplementary movie SFSAT_11_Theory.mpg.

References

Aggarwal, J., Cai, Q., Liao, W., and Sabata, B. 1994. Articulated and elastic non-rigid motion: A review. In *Proceedings of IEEE Workshop on Motion of Non-rigid and Articulated Objects'94*, pp. 16–22.

Ahuja, N. and Veenstra, J. 1989. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 11(2):137–149.

Baumgart, B.G. 1974. Geometric modeling for computer vision. Ph.D. thesis, Stanford University.

Besl, P. and McKay, N. 1992. A method of registration of 3D shapes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Bottino, A. and Laurentini, A. 2000. Non-intrusive silhouette based motion capture. In *Proceedings of the Fourth World Multiconference on Systemics, Cybernetics and Informatics SCI 2001*, pp. 23–26.

Buehler, C., Matusik, W., McMillan, L., and Gortler, S. 1999. Creating and rendering image-based visual hulls. Technical Report MIT-LCS-TR-780, MIT.

Buehler, C., Matusik, W., and McMillan, L. 2001. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering*.

Cheung, G., Baker, S., and Kanade, T. 2003. Visual hull alignment and refinement across time:a 3D reconstruction algorithm combining shape-frame-silhouette with stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, MI.

Cheung, G. 2003. Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Tracking and Rendering. Ph.D. thesis, Carnegie Mellon University.

Delamarre, Q. and Faugeras, O. 1999. 3D articulated models and multi-view tracking with silhouettes. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece.

Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Statistical Society*, B 39:1–38.

Dennis, J. and Schnabel, R. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, NJ.

Irani, M., Hassner, T., and Anandan, P. 2002. What does the scene look like from a scene point? In *Proceedings of European Conference on Computer Vision (ECCV'02)*, Copenhagen, Denmark. pp. 883–897.

Jain, A. 1989. *Fundamentals of Digital Image Processing*. Prentice Hall.

Joshi, T., Ahuja, N., and Ponce, J. 1994. Towards structure and motion estimation from dynamic silhouettes. In *Proceedings of IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pp. 166–171.

Joshi, T., Ahuja, N., and Ponce, J. 1995. Structure and motion estimation from dynamic silhouettes under perspective projection. Technical Report UIUC-BI-AI-RCV-95-02, University of Illinois Urbana Champaign.

Kakadiaris, I. and Metaxas, D. 1998. 3D human body model acquisition from multiple views. *International Journal on Computer Vision*, 30(3):191–218.

Ke, Q. and Kanade, T. 2001. A subspace approach to layer extraction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai, HI.

Kim, Y. and Aggarwal, J. 1986. Rectangular parallelepiped coding: A volumetric representation of three dimensional objects. *IEEE Journal of Robotics and Automation*, RA-2:127–134.

Krahnstoever, N., Yeasin, M., and Sharma, R. 2001. Automatic acquisition and initialization of kinematic models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01), Technical Sketches*, Kauai, HI.

Krahnstoever, N., Yeasin, M., and Sharma, R. 2003. Automatic acquisition and initialization of articulated models. In *To appear in Machine Vision and Applications* (to accepted).

Kurazume, R., Nishino, K., Zhang, Z., and Ikeuchi, K. 2002. Simultaneous 2D images and 3D geometric model registration for texture mapping utilizing reflectance attribute. In *Proceedings of Asian Conference on Computer Vision (ACCV'02)*, vol. 1, pp. 99–106.

Kutulakos, K. and Seitz, S. 2000. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218.

Laurentini, A. 1991. The visual hull: A new tool for contour-based image understanding. In *Proceedings of the Seventh Scandinavian Conference on Image Analysis*, pp. 993–1002.

Laurentini, A. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 16(2):150–162.

Laurentini, A. 1995. How far 3D shapes can be understood from 2D silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195.

Laurentini, A. 1999. The visual hull of curved objects. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece.

Lazebnik, S., Boyer, E., and Ponce, J. 2001. On computing exact visual hulls of solids bounded by smooth surfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, Kauai HI.

Martin, W. and Aggarwal, J. 1983. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–174.

Matusik, W. 2001. Image-based visual hulls. Master's thesis, Massachusetts Institute of Technology.

Matusik, W., Buehler, C., Raskar, R., Gortler, S., and McMillan, L. 2000. Image-based visual hulls. In *Computer Graphics Annual Conference Series (SIGGRAPH'00)*, New Orleans, LA.

Mendonca, P., Wong, K., and Cipolla, R. 2000. Camera pose estimation and reconstruction from image profiles under circular motion. In *Proceedings of European Conference on Computer Vision (ECCV'00)*, Dublin, Ireland, pp. 864–877.

Mendonca, P., Wong, K., and Cipolla, R. 2001. Epipolar geometry from profiles under circular motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):604–616.

Moezzi, S., Tai, L., and Gerard, P. 1997. Virtual view generation for 3D digital video. *IEEE Computer Society Multimedia*, 4(1).

Noborio, H., Fukuda, S., and Arimoto, S. 1988. Construction of the octree approximating three-dimensional objects by using multiple views. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 10(6):769–782.

Okutomi, M. and Kanade, T. 1993. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363.

Poelman, C. and Kanade, T. 1992. A paraperspective factorization method for shape and motion recovery. Technical Report CMU-CS-TR-92-208, Carnegie Mellon University, Pittsburgh, PA.

Potmesil, M. 1987. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40:1–20.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. 1993. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

Quan, L. and Kanade, T. 1996. A factorization method for affine structure from line correspondences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, CA, pp. 803–808.

Rusinkiewicz, S. and Levoy, M. 2001. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, pp. 145–152.

Sawhney, H. and Ayer, S. 1996. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(8):814–830.

Shanmukh, K. and Pujari, A. 1991. Volume intersection with optimal set of directions. *Pattern Recognition Letter*, 12:165–170.

Szeliski, R. 1993. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1):23–32.

Szeliski, R. 1994. Image mosaicing for tele-reality applications. Technical Report CRL 94/2, Compaq Cambridge Research Laboratory.

Szeliski, R. and Golland, P. 1998. Stereo matching with transparency and matting. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, pp. 517–524, Bombay, India.

Tomasi, C. and Kanade, T. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154.

Vijayakumar, B., Kriegman, D., and Ponce, J. 1996. Structure and motion of curved 3D objects from monocular silhouettes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, CA, pp. 327–334.

Wheeler, M. 1996. Automatic Modeling and Localization for Object Recognition. PhD thesis, Carnegie Mellon University.

Wong, K. and Cipolla, R. 2001. Head model acquisition and silhouettes. In *Proceedings of International Workshop on Visual Form (IWVF-4)*.

Wong, K. and Cipolla, R. 2001. Structure and motion from silhouettes. In *Proceedings of International Conference on Computer Vision (ICCV'01)*, Vancouver, Canada.

Zhang, Z. 1994. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152.