Constraint Fusion for Recognition and Localization of Articulated Objects

Yacov Hel-Or *
Dept. of Applied Math. and Comp. Science
The Weizmann Institute of Science
76100 Rehovot, Israel

Michael Werman
Institute of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

February 15, 1995

Abstract

This paper presents a method for localization and interpretation of modeled objects that is general enough to cover articulated and other types of constrained models. The flexibility between the components of the model is expressed as spatial constraints that are fused into the pose estimation during the interpretation process. The constraint fusion assists in obtaining a precise and stable pose of each of the object's components and in finding the correct interpretation. The proposed method can handle any constraint (including inequalities) between any number of different components of the model. The framework is based on Kalman filtering.

*Current address: NASA Ames Research Center, M/S 262-2, Moffet Field, CA 94035-1000. e-mail: toky@white.stanford.edu

1 Introduction

Estimating the pose of a 3D object from images or other sensed data is a classical problem in computer vision. Quite often, a model of the object is known and this information is used to estimate the pose of the object in the world. This problem is known as model-based pose determination and is used in many applications such as object recognition, object tracking, robot navigation, motion detection, etc. A complementary problem to the pose determination problem is the interpretation problem that deals with the correspondence between the given sensory data and the model features. This correspondence is necessary in localization procedures that are based on local features of the model. Both, the positioning and the interpretation problems are well documented in the literature (for example: [11, 6, 7, 19]), however, the majority of the papers deal with 3D rigid objects and little attention has been given to articulated or constrained objects (e.g. [4, 8, 20, 24]).

An articulated object is an object composed of a set of rigid components connected at joints that allow certain degrees of freedom. These joints can be, for example, *prismatic joints* that allow relative translation between components, or *revolute joints* that allow relative rotation of the components about a point (see Figure 1). An example of such an object is a robot arm made up of several rigid components connected by movable joints. In this case, each model joint enforces a constraint on the spatial location of the body's components, thus, the problem of articulated objects is a special case of the general study of constrained models. We extend the definition of the problem to models that include general constraints such as co-linearity or co-planarity of the model components, angle relationships, etc, and also include inequality constraints such as a limited range of distances between points or a limited range of angles. We call these kind of models *constrained models*.

Existing methods that deal with constrained objects are restricted to deal with articulated models (e.g. [4, 8, 20, 24]). They deal with constraints that are due to prismatic or revolute joints. In this paper we present a general framework that can deal with all types of spatial constraints and is not limited to any particular type. The method presented, solves the interpretation and the localization problems simultaneously where constraints and measurements are considered and fused incrementally. Fusion of constraints into the pose determination of the components enables the information obtained on the pose of any single component to be propagated to all other components through the mutual constraints. In this manner,

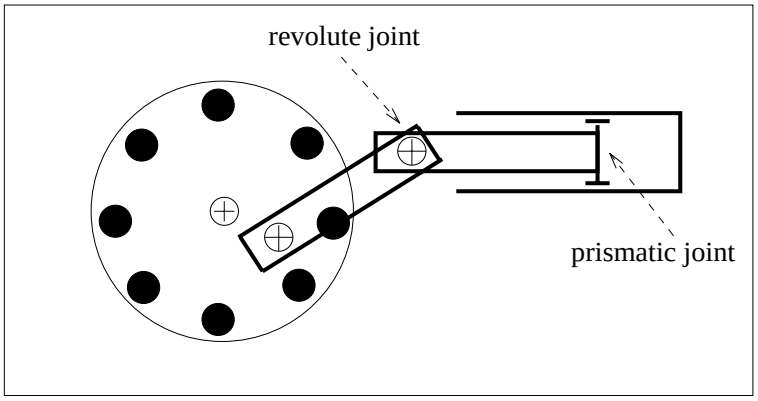

Figure 1: An articulated object composed of several rigid components connected by two revolute joints and a single prismatic joint.

the estimated solution takes into account all the existing measurements and all the defined constraints. In addition, this process enables a simple and efficient interpretation strategy. The fusion of the constraints and the measurements is performed using the Kalman filter.

We deal here with models consisting of a set of feature points, such as maximum curvature, segment endpoints or corners. The measurements taken on these points are noisy.

2 Formal Description of the Problem

A *constrained model* M of a $3D$ object consists of a set of rigid components

$$M = \{C_i\}_{i=1\cdots n} \quad .$$

Each component C_i has its own local coordinate system and consists of a set of feature points whose locations are:

$$C_i = \{\mathbf{u}_{i,j}\}_{j=1\cdots m_i} \quad .$$

$\mathbf{u}_{i,j}$ is a 3 dimensional vector, representing the location of the j^{th} point in the i^{th} component and is given in the local coordinate system of C_i. A set of points forming a component is rigid but the collection of components are not rigid. For each component C_i there is an associated parameter-vector $\mathbf{T}_i$ representing the position of C_i relative to the viewer-centered frame of reference. Hence, $\mathbf{T}_i$ is a six dimensional vector describing the location and the orientation of the local coordinates system of C_i relative to the viewer coordinates system. Since the components are restricted in their location due to flexible joints, the model includes, in

addition to the representation of each component, a set of constraints that describe the mutual relationships between the components. These constraints are of the form:

$$\psi_k(\mathbf{T}_p, \mathbf{T}_q, \cdots) = 0 \quad .$$

Each constraint may involve a single model component, such as a known location or a known orientation of the component, or several components as in the case of a revolute or prismatic joint between two components, a known distance between components, etc. Each constraint is expressed by an appropriate equation, for example, in an articulated constraint two components, C_p and C_q, are linked at a rotational point whose location is given by $\mathbf{u}_{p,i}$ in the local coordinates of C_p, and by $\mathbf{u}_{q,j}$ in the local coordinates of C_q. In such a case the constraint equation will be:

$$T_p(\mathbf{u}_{p,i}) - T_q(\mathbf{u}_{q,j}) = 0 \quad .$$

where T_s is the transformation function defined by the parameters in $\mathbf{T}_s$.

As previously mentioned, the model may also consist of inequality constraints of the form $\psi(\mathbf{T}_p, \mathbf{T}_q, \ldots) > 0$. Let us assume, for the moment, that the constraints are restricted to equality constraints, and we will later describe the direct extension of these constraints to inequality constraints.

A *measurement* M' of a constrained object is represented by a collection of noise contaminated measurements and their uncertainties:

$$M' = \{(\hat{\mathbf{u}}'_{i,j}, \Lambda_{i,j})\}_{i=1\cdots n \ ; \ j=1\cdots m_i} \quad .$$

$\hat{\mathbf{u}}'_{i,j}$ - is a noise-contaminated measurement of the real location-vector $\mathbf{u}'_{i,j}$, associated with the j^{th} measured point of the i^{th} component. Both, $\hat{\mathbf{u}}'_{i,j}$ and $\mathbf{u}'_{i,j}$ are represented in a viewer-centered frame of reference. It is possible to have more than one measurement for a model point.
$\Lambda_{i,j}$ - is the covariance matrix depicting the uncertainty in the sensed vector $\hat{\mathbf{u}}'_{i,j}$. We do not constrain the dimensionality of the measured data but allow it to be $3D$ (stereo, range finder, etc.) or $2D$ (orthographic or perspective projection).

A *matching* (correspondence) between the model M and the measurement M' is a collection of pairs of the form

$$matching = \{\mathbf{u}_{i,j}, \hat{\mathbf{u}}'_{i,j}\} \quad ,$$

which represents the correspondence between the model points and the measured points. For simplicity we mark every model point and its matched measurement with the same indices.

The problem :
Given a model M and a measurement M', for each component C_i, find the measured points that correspond to its feature points and estimate its position $\mathbf{T}_i$. It is important to note that the solution $\{\mathbf{T}_i\}_{i=1\cdots n}$ *must* satisfy the model constraints:

$$\{\psi_k(\mathbf{T}_p, \mathbf{T}_q, \cdots) = 0\}_{k=1\cdots r} \quad .$$

3 Background and Related Works

Extensive studies can be found in the literature dealing with interpretation and pose estimation of rigid objects from measurements, however, little attention has been given to pose estimation of articulated or constrained objects. Several studies can be found that deal with special cases of constrained objects, namely, articulated objects having prismatic or revolute joints, most of them in the context of recognition ([4, 8, 20, 24]). In general, the existing methods dealing with this problem can be divided into two main paradigms:

1. **Divide and conquer methods**
   The basic and naive method is to decompose the object into its parts and to estimate the pose of each part separately. Grimson [9, 8] follows this paradigm in order to identify a family of $2D$ objects that differ in scale-factor, stretch factor or the angles between parts. In his approach, the pose that was estimated for a part, is followed by a verification step that tests whether the part satisfies the defined constraints (up to a predefined threshold). Grimson uses the pose estimate of a part to constrain the possible matchings of the neighboring part, however this estimate is not used in the pose estimation of the neighboring part. Shakunaga [26] follows a similar method for estimating the orientation of a flexible body composed of parts joined by revolute joints.

   Although the simplicity of this method is attractive, it is obvious that it is unsatisfying since it does not exploit the fact that different components do belong to the same object; Evaluating the pose of each part separately may result in constraints not being satisfied between the parts. Furthermore, no mutual information passes between parts and each object component is located using only its associated measurements. Additional

information that can be obtained from measurements of neighboring components is not considered, thus, not all available information is exploited. Consequently, the interpretation procedure which, in some procedures, is aided by the pose estimation, is impaired as well.

2. **Parametric methods**

   It is possible to eliminate the need to explicitly handle the constraints by decreasing the number of parameters that describe the pose of the object (so that the number of free parameters equals the degrees of freedom of the object). The remaining parameters are estimated during the estimation process. In this way, the constraints are expressed implicitly by the free parameters. For example, the pose of an articulated object in 2D having two components connected by a revolute joint, can be described by the translation and the orientation of each component (6 parameters) with an additional constraint due to the joint between the components (Figure 2a). Alternatively, the pose of the object can be described by the translation and orientation of one of the components and the relative angle between the two components (4 parameters) (Figure 2b). The latter description eliminates the need to consider a constraint in the estimation process. Note, that the classical approach to pose estimation of a rigid object consisting of several feature points, falls in this category; The rigidity constraint between the feature points are expressed implicitly when the pose of the entire object is described by its six free parameters.

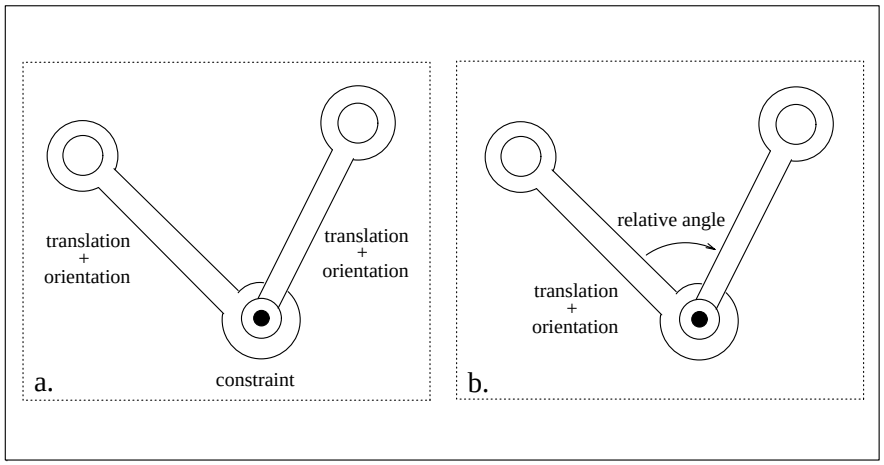

Figure 2: Two possible parameter sets for defining the pose of an articulated object having a revolute joint.

Lowe [20, 21] follows this method and estimates the free parameters of the viewpoint and of the model using Newton iterations into which additional techniques are incorporated in order to ensure convergence. A similar method was used by Brooks [5] in the well known system *ACRONYM*. Mulligan et. al. [24] use the same approach for estimating the positions of an excavator's arm, however, in their work, the flow of pose information from one arm to the next is in one direction, thus the pose of the boom influences the pose of the bucket but not vice versa.

The main problem in the method of parameter reduction is the need for defining the dependence of each measurement on all the free parameters during the estimation process. The definition of the dependence is problematic for two reasons:
First, the complexity of this definition increases with the number of free parameters that each point is dependent on. Second, in most cases, as the number of components of the object is greater, the order of the nonlinearity of the dependence equations is higher. This results in a more complex and less stable solution especially when using iterative methods based on linear approximation of the nonlinear equations (such as in [21]). An additional drawback of this method is the difficulty in finding the correct free parameters. The difficulty of selecting the parameters increases with the number of constraints and with the number of parts participating in each constraint. Moreover, when dealing with an interactive system (such as in [13]) where the constraints are incorporated dynamicly at run time, it is difficult to find the free parameters, due to the necessity of applying symbolic mathematical methods during run time.

4 Constraint Fusion Method

In the two kind of methods described in the last section there are no explicit consideration of constraints in the estimation process. Either, the constraints are not considered in the divide and conquer methods or they are implicitly expressed, by reducing the number of estimated parameters, in the parametric methods. The constraint fusion method, suggested in this paper, considers explicitly both, measurements and constraints, in the estimation process. The pose of the object parts is estimated to conform optimally with the measurements while satisfying the model constraints. The method we suggest is a general scheme which overcomes the drawbacks of the other methods.

The idea is to treat both measurements and constraints similarly while varying only their

associated uncertainty. The constraints are considered as perfect "measurements" with zero uncertainty whereas the measurements themselves (the actual measurements) have uncertainty greater than zero. In other words the actual measurements are considered *soft* constraints whereas the constraints are considered *strong*. The fusion of the actual measurements and the constraints during the pose estimation process is performed using the Kalman filter and it is in accord with [2].

The rest of the paper is organized as follows: We first describe the solution for estimating the pose of one rigid component using the Kalman filter tool. This solution follows the method we suggested in [14]. For simplicity of explanation we assume in this part that the matching is given. Following, we elaborate in detail two methods for estimating the pose of a constrained object where each component of the object has a single model point. The first solution solves the problem using a batch process and the second solution uses an iterative process which is preferable when the matching is not given. We then elaborate the interpretation process and explain how it is interlaced with the pose estimation process. Finally, we expand the paradigm to include multiple-point components. Results on simulated and real cases follow.

5 Pose Estimation of One Rigid Component

In order to facilitate the explanation of our method we first briefly describe the estimation of the pose of a single rigid component given noisy measurements. The solution follows the method we suggested in [14]. Since there is only one component in this object we omit in this section the index denoting the component number. Therefore, the model M is represented by a set of points: $M = \{\mathbf{u}_i\}$, and the measurements of M are represented by $M' = \{(\hat{\mathbf{u}}'_j, \Lambda_j)\}$. The aim is to estimate a transformation $\mathbf{T}$ that optimally maps the model points onto the corresponding measured points, given the matching between them. $\mathbf{T}$ is a vector representing a rigid $3D$ transformation and describes the position of the measured object M' in the $3D$ scene.

The method described in [14] estimates the transformation in two phases:
In the first phase, all the measurements are changed to be $3D$ measurements by appropriately updating the uncertainty matrices associated with the measured points. A $2D$ measurement, which is a projection (perspective or orthographic) onto a $2D$ plane, is regarded as a $3D$

measurement with infinite uncertainty in the direction of the projection. Therefore, the dimensionality of the measurements is encoded in the covariance matrix where the uncertainty depends both on the measurement noise and on the type of measurement. Details explaining the construction of the covariance matrix and the measurement unification are given in [14]. In the second phase, $\mathbf{T}$ is estimated from all the measured points using the *Kalman-filter* fuser [17, 23]. The estimation process is composed of an incremental refinement, for which at each step $k-1$, there exists an estimate $\hat{\mathbf{T}}^{k-1}$ of the transformation $\mathbf{T}$ and a covariance matrix Σ^{k-1} which represents the "quality" of the estimate $\hat{\mathbf{T}}^{k-1}$:

$$\Sigma^{k-1} = E\{(\hat{\mathbf{T}}^{k-1} - \mathbf{T})(\hat{\mathbf{T}}^{k-1} - \mathbf{T})^T\} \quad .$$

Given a new measurement $(\hat{\mathbf{u}}'_k, \Lambda_k)$ the current estimate is updated to be $\hat{\mathbf{T}}^k$ with the associated uncertainty Σ^k. The accuracy of the estimate increases, as additional matches are fused, i.e. $\Sigma^k \leq \Sigma^{k-1}$ ($\Sigma^{k-1} - \Sigma^k$ is nonnegative definite). The process terminates as soon as the uncertainty satisfies our criterion for accuracy or no additional matches can be supplied.

5.1 The Kalman Filter Fuser for Static Parameter Estimation

The Kalman filter (K.F) fuser is a tool for parameter estimation from given measurements. In our case the parameter vector to be estimated is the transformation vector $\mathbf{T}$ that represents the rigid transformation of the object from its local coordinates to the viewer coordinates and is composed of two components:

- The translation component, expressed by the vector t;

$$\mathbf{t} = (t_x, t_y, t_z)^T \quad . \tag{1}$$

- The rotation component, described by the quaternion $\tilde{\mathbf{q}}$ [16]:

$$\tilde{\mathbf{q}} = (q_0, \mathbf{q}) = (q_0, q_1 i + q_2 j + q_3 k) \quad .$$

Considering these two components, the parameter vector to be estimated during the filtering process is a seven dimensional vector:

$$\mathbf{T} = \begin{pmatrix} \tilde{\mathbf{q}} \\ \mathbf{t} \end{pmatrix} \quad .$$

The Kalman filter fuser produces an estimate $\hat{\mathbf{T}}$ of the transformation vector, given the measurements of the point locations. At each step, the fuser receives three inputs and supplies a single output. The inputs are:

1. An *a priori* estimate of the evaluated parameter vector and the uncertainty associated with it. In our case, in the k^{th} step, the *a priori* estimation will be the estimate evaluated at the previous step $\hat{\mathbf{T}}^{k-1}$ and its associated covariance Σ^{k-1}. The covariance matrix Σ^0 in the initial step will be set to infinity since no *a priori* knowledge about $\mathbf{T}$ is assumed.

2. The current measurement and its uncertainty, in our case $(\hat{\mathbf{u}}'_k, \Lambda_k)$ which, following the first phase, can be assumed to be three dimensional.

3. A mathematical relationship between the evaluated parameters and the measurements. This mathematical relationship should be linear in the evaluated parameters. In our case the relationship is:

$$h_k(\mathbf{u}_k, \mathbf{u}'_k, \mathbf{T}) = R\mathbf{u}_k + \mathbf{t} - \mathbf{u}'_k = 0 \quad , \tag{2}$$

   where R is a rotation matrix constructed from the quaternion $\tilde{\mathbf{q}}$ [25]. The equation $h_k(\mathbf{u}_k, \mathbf{u}'_k, \mathbf{T}) = 0$ is not linear as required in the K.F., therefore, we use the *extended Kalman filter* (E.K.F.) [17, 23] which is a generalization of the Kalman filter to non-linear systems where transition from step $k-1$ to step k is performed using a linear approximation of h_k by taking the first order Taylor expansion around $(\hat{\mathbf{T}}^{k-1}, \hat{\mathbf{u}}'_k)$.

The output of the K.F fuser is an updated estimation of the evaluated parameters and its associated uncertainty; in our case $\hat{\mathbf{T}}^k$ and Σ^k respectively. The K.F. fuser is of the form:

$$\hat{\mathbf{T}}^k = f(\hat{\mathbf{T}}^{k-1}, \Sigma^{k-1}, \mathbf{u}_k, \hat{\mathbf{u}}'_k, \Lambda_k, h_k) \quad .$$

Thus, at each stage k, there is no need of retaining any of the previously considered measurements. Only the current estimate $\hat{\mathbf{T}}^{k-1}$ and its associated uncertainty Σ^{k-1} need be retained. The K.F. equations in our case are given in Appendix A.

The K.F. updating equations yield an unbiased estimate of $\mathbf{T}$ which is optimal in the sense of the linear minimal variance criterion [1]. In the case where the measurement noise is a Gaussian process (which is a reasonable assumption, considering the numerous sources of noise),

the K.F. fuser gives an estimate that is also optimal in the sense of the maximum-likelihood criterion. In this case, the estimate coincides with the conditional expectation:

$$\hat{\mathbf{T}}^k = E\{\mathbf{T}|\{\hat{\mathbf{u}}'_i\}_{i=1\ldots k}\} \quad .$$

Note, that the general K.F. deals with a parameter vector that is changing with time, whereas in our case the estimated transformation, $\mathbf{T}$, is static and does not change during the estimation process. Therefore, our case uses a degenerated form of the K.F. and we call its fusion process a *K.F. fuser*.

Further details of the rigid object process such as how to solve the correspondence problem simultaneously with the pose problem and computational aspects such as, time complexity, stabilization methods, parallelization and convergence can be found in [12].

6 Constrained Objects Having a Single Point Per Component

The simplest case of a constrained object is where each of its components consists of a single model point. In order to simplify the explanation of the pose determination of general constrained objects, we first elaborate the solution in this case, and then expand the solution to include multiple-point components. When an object consists of single point components, its model is represented by:

$$M = \{C_k\}_{k=1\cdots n}$$

where each component C_k has a single model point whose location is $\mathbf{u}_k$. Without loss of generality, we choose this point to be located at the origin of the local coordinates associated with C_k, i.e: $\mathbf{u}_k = (0,0,0)^T$. Measurements of the locations of the model points are obtained. For simplicity assume n measurements are obtained, $\{(\hat{\mathbf{u}}'_i, \Lambda_i)\}_{i=1\cdots n}$, a single measurement for each model point, represented in the viewer-centered coordinates. Additionally, assume in this case that the measurements are $3D$ data. The latter assumption is due to the inability to deduce the $3D$ position of an isolated point from a single $2D$ measurement. The transformation of the k^{th} component, $\mathbf{T}_k$, is composed only of the translation vector $\mathbf{t}_k$ since the rotation part $\tilde{\mathbf{q}}_k$ is irrelevant for an isolated point. Therefore, the general position vector, $\mathbf{T}$, to be estimated in such a case consists of the translation vectors of all the model

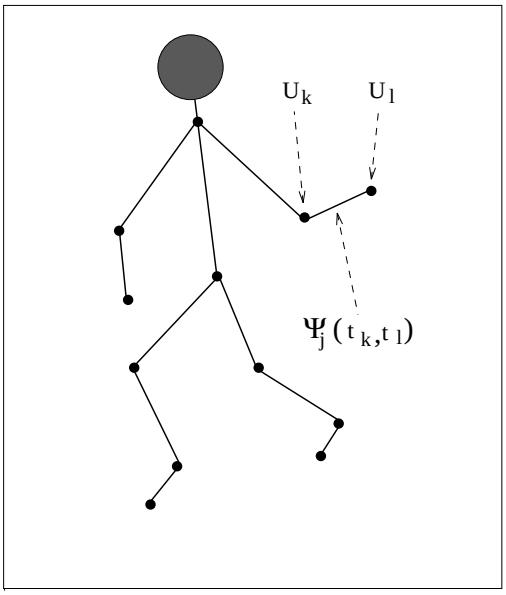

Figure 3: An articulated object. The black points represent model features and the segments represent constraints between adjacent points.

components:

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_n \end{pmatrix} \quad .$$

Since the model points are located at the origin of the local coordinates the translation vector $\mathbf{t}_k = (x'_k, y'_k, z'_k)^T$ also describes the position of the k^{th} point in the viewer centered frame of reference. However, the evaluated estimation must satisfy a set of constraints:

$$\{\psi_j(\mathbf{T}) = 0\}_{j=1\cdots r} \quad .$$

For the specific case of an articulated object the constraints are:

$$\psi_j(\mathbf{t}_k, \mathbf{t}_l) = \|\mathbf{t}_l - \mathbf{t}_k\|^2 - d^2_{(k,l)} = 0$$

where $d_{(k,l)}$ represents the constant Euclidean distance between two adjacent points, $\mathbf{u}_k$ and $\mathbf{u}_l$, in the object (see Figure 3).

6.1 Solving the System Using A Batch Mode of K.F.

As stated, enforcing the model constraints onto the pose solution is performed by considering the constraints as additional artificial "measurements" having zero uncertainty. The

zero uncertainty of these "measurements" assures that the constraints are satisfied in the final solution. The constraint "measurements" are fused as a single measurement which is composed of *all* the constraints together (batch).

Fusion of the constraints is performed, similar to the fusion of actual measurements, using the E.K.F. tool where the evaluated parameters are denoted by the vector $\mathbf{T}$. The fusion is performed in one step of the K.F. fuser, followed by several *local iterations* [17] in order to reduce the influence of the linearization effect on the final solution. The inputs supplied to the K.F. fuser are the following:

- *A priori* estimate input:
  From the actual measurements we construct an *a priori* estimate of the evaluated transformation:

$$(\hat{\mathbf{T}}^0, \Sigma^0) = \left[\begin{pmatrix} \hat{\mathbf{u}}'_1 \\ \hat{\mathbf{u}}'_2 \\ \vdots \\ \hat{\mathbf{u}}'_n \end{pmatrix}, \begin{pmatrix} \Lambda_1 & & 0 \\ & \ddots & \\ 0 & & \Lambda_n \end{pmatrix} \right] = [\hat{\mathbf{u}}', \Lambda] \tag{3}$$

  which takes into consideration *all* the actual measurements.

- Measurement input:
  From the constraint equations we construct a set of artificial perfect "measurements" having zero uncertainty .

- Measurement model input:
  The mathematical relationship between the measurements and the evaluated vector is a concatenation of all the linear approximations of the constraint equations.

Formally, assume we are given the constraint $\psi_j(\mathbf{T}) = 0$. We regard $\hat{\mathbf{T}}^0$ (Equation 3) as an initial approximation of $\mathbf{T}$ and linearize $\psi_j(\mathbf{T})$ around $\hat{\mathbf{T}}^0$ obtaining:

$$\psi_j(\mathbf{T}) = 0 \approx \psi_j(\hat{\mathbf{T}}^0) + \frac{\partial \psi_j}{\partial \mathbf{T}}(\mathbf{T} - \hat{\mathbf{T}}^0) \quad .$$

This equation can be rewritten as a linear equation:

$$\mathbf{z}_j = H_j \mathbf{T} \tag{4}$$

where

$$\begin{aligned} \mathbf{z}_j &= -\psi_j(\hat{\mathbf{T}}^0) + (\frac{\partial \psi_j}{\partial \mathbf{T}})\hat{\mathbf{T}}^0 \\ \text{and} \quad H_j &= \frac{\partial \psi_j}{\partial \mathbf{T}} \end{aligned}$$

The matrix H_j is of dimensions $\dim(\psi_j) \times 3n$ representing the linear relationship between $\mathbf{z}_j$ and $\mathbf{T}$. Note that no random noise is added to this equation, thus $\mathbf{z}_j$ is a perfect "measurement". The rest of the constraints are similarly linearized and appended to Equation 4, so that a vector equation is obtained:

$$\mathbf{z} = H\mathbf{T} \tag{5}$$

where

$$\mathbf{z} = \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_m \end{pmatrix} \quad \text{and} \quad H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_m \end{bmatrix} \quad . \tag{6}$$

$\mathbf{z}$ is the "measurement" vector of size $d = \sum \dim(\psi_j)$ with an associated uncertainty matrix of zeroes. H is a $d \times 3n$ matrix that describes the mathematical relationship between $\mathbf{z}$ and the evaluated vector $\mathbf{T}$. In the case where there is more than a single measurement per feature point, these measurements and their uncertainties that were not already considered in the *a priori* estimation are appended to Equation 5 as well.

Given the inputs described above, the estimate for $\mathbf{T}$ obtained from the K.F. updating equations is (Appendix A):

$$\hat{\mathbf{T}} = \hat{\mathbf{T}}^0 + \Lambda H^T (H\Lambda H^T)^{-1}(\mathbf{z} - H\hat{\mathbf{T}}^0) \quad . \tag{7}$$

Multiplying both sides of the equation by H we obtain:

$$H\hat{\mathbf{T}} = H\hat{\mathbf{T}}^0 + (H\Lambda H^T)(H\Lambda H^T)^{-1}(\mathbf{z} - H\hat{\mathbf{T}}^0) = \mathbf{z}$$

i.e. the obtained solution indeed satisfies the constraints as defined in Equation 5.

An example of producing the K.F. input for a simple articulated object similar to that in Figure 4 is given in Appendix B. Some examples of this articulated object in different positions are shown in Figure 5. In these examples the measurements are represented by rectangles having width and length proportional to the s.t.d. of the measurements. The dotted lines connect each measurement with its associated model point.

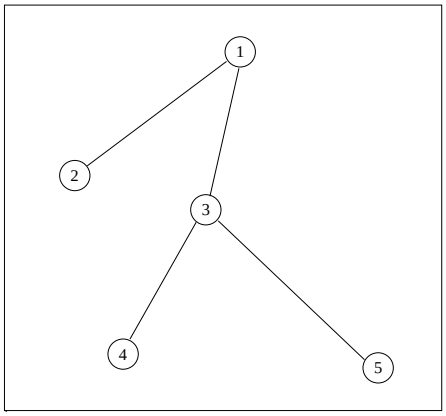

Figure 4: An articulated object composed of five single point components and four articulated constraints.

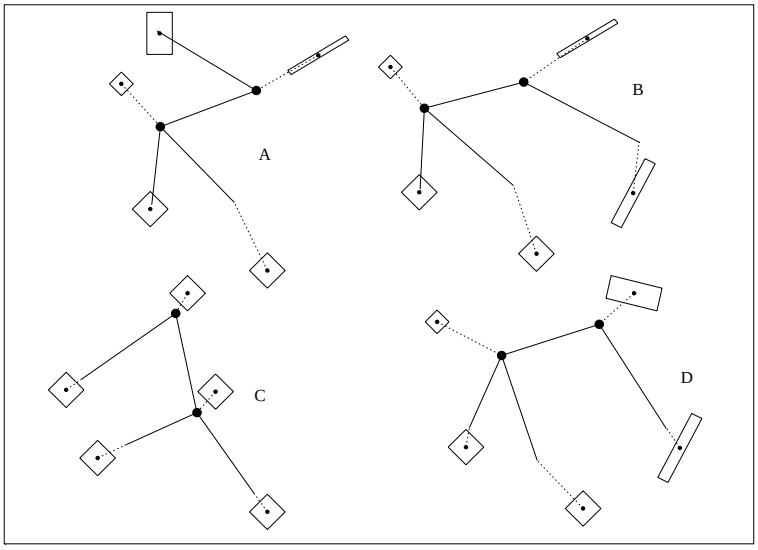

Figure 5: Four examples of solutions for the articulated object of Fig. 4. Joints are represented by black circles and measurements are represented by rectangles. The width and the length of each rectangle is proportional to the s.t.d. of the measurement. The dotted lines connect every measurement with its associated model point.

6.1.1 Adding An Initial Guess

Using the described inputs, the *a priori* estimate is based on the actual measurements. Although these measurements supply a good *a priori* estimate, in some cases, it seems beneficial to allow the user a possibility of providing an initial guess based on external knowledge, such as, an "expected" viewpoint and "expected" positional relationships between components. This can be helpful in obtaining a better linearization during the first stages of the

process resulting in a more reliable convergence. Adding an initial guess is easily obtained by slightly varying the input to the K.F. fuser:

- *A priori* estimate input: is taken as the *a priori* estimate supplied by the user and is associated with infinite uncertainty:

$$(\hat{\mathbf{T}}^0, \Sigma^0) = \left[\begin{pmatrix} \hat{\mathbf{t}}_1^0 \\ \hat{\mathbf{t}}_2^0 \\ \vdots \\ \hat{\mathbf{t}}_n^0 \end{pmatrix}, \begin{pmatrix} \infty & & 0 \\ & \ddots & \\ 0 & & \infty \end{pmatrix} \right]$$

- Measurement input: is a concatenation of all the actual measurements and all the constraint "measurements":

$$[\tilde{\mathbf{z}}, \mathrm{cov}\{\tilde{\mathbf{z}}\}] = \left[\begin{pmatrix} \mathbf{z} \\ \hat{\mathbf{u}}' \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & \Lambda \end{pmatrix} \right]$$

  where $\hat{\mathbf{u}}'$, Λ and $\mathbf{z}$ are as defined in Equations 3, 6 above.

- Measurement model input: is defined in the following equation:

$$\tilde{\mathbf{z}} = \begin{pmatrix} H \\ I \end{pmatrix} \mathbf{T}$$

  where H is defined in Equation 5.

This calculation is general enough to include the case where $\mathbf{T}^0 = \hat{\mathbf{u}}'$ which is the former case where the actual measurements are taken as the *a priori* estimate.

6.2 Solving the System using an Incremental Process

Using the batch approach for solving systems that include both measurements and constraints has the following disadvantage: The method must assume that the matching between model points and measurements is given in advance. This assumption is not acceptable in many applications especially in recognition systems. In order to overcome this problem we use an incremental approach.

In the incremental method, as in the batch method, the state-vector $\mathbf{T}$ is a composition of all the pose parameters of the model points, however in this method the actual measurements and the constraint "measurements" are fused sequentially. At each step k, the current

estimate $(\hat{\mathbf{T}}^k, \Sigma^k)$ is updated to be $(\hat{\mathbf{T}}^{k+1}, \Sigma^{k+1})$ by fusing a single measurement $(\hat{\mathbf{u}}'_k, \Lambda_k)$ or a single constraint $(\mathbf{z}_k, 0)$. The fusion is performed using the K.F. in a similar manner as described above for the batch process, where the *a priori* estimate input is a user defined initial guess with an associated infinite uncertainty (Section 6.1.1).

If the current measurement is an actual measurement, the measurement input is $(\hat{\mathbf{u}}'_k, \Lambda_k)$ and the mathematical relationship between $\mathbf{T}$ and $\mathbf{u}'_k$ is simply

$$\mathbf{u}'_k = [0, \cdots, 1, \cdots, 0]\mathbf{T}$$

where the 1 is in the k^{th} position. If we fuse a "perfect" measurement, the measurement input is $\mathbf{z}_k$ and the measurement model is $\mathbf{z}_k = H_k\mathbf{T}$ as described above in Equation 4.

The sequential fusion of the measurements is possible due to the assumption that there is no correlation between the noise of different measurements (i.e. Λ in Eq. 3 is block diagonal). Although, the batch method has better convergence properties and seems to better approach the Cramer Rao bound [29], the advantage of the incremental method is its ability to easily incorporate a matching (interpretation) process into the estimation process as will be described in the following section.

7 The Measurement Interpretation using Constraints

Using the incremental approach described above we can incorporate techniques that solve the interpretation problem using a pruning search in the correspondence space. These techniques regard the correspondence problem as a search problem in a graph. This graph defines a pairing between the model features and the measured features. The basic scheme behind these methods is to prune parts of the graph which represent impossible pairings. Faugeras et al. [7, 6, 3] and Grimson et al. [11] follow similar methods for rigid objects. They represent all possible matches in an *interpretation tree* (I.T.). in which every level corresponds to a model feature, and every node in that level denotes a match between the model feature and some measured feature. Every path in the I.T., from root to leaf, defines a full interpretation, i.e, defines for each measurement the corresponding model feature. Both, Grimson and Faugeras search for a consistent path in the I.T., using depth first search. Grimson restricts the search by defining local constraints in the model, such as limiting the range of distances and angles between model features. Any branch of the I.T. that con-

tradicts the local constraints is pruned. Grimson extended this method to parameterized $2D$ objects [9] which include articulated objects. However this extension suffers from two deficiencies: First, this method is developed to deal with particular types of constraints (articulated constraints) and it is not general enough to automatically handle any type of constraint. Second, in this method, a model constraint assists only in pruning irrelevant matches that are directly related to this particular constraint but is not used in pruning further matches that are indirectly related to the constraint (related through other constraints).

Faugeras et al. present a similar method for a rigid object. For every partial path in the I.T., they evaluate the (rigid) transformation of the object that is consistent with the matches in the path. This transformation is applied to the model features and these are used to further restrict the remaining matches. In this framework, the pose of the object is solved simultaneously with the interpretation problem. We extend this technique to deal with constrained objects in which the model constraints assist in the interpretation process:

Assume we want to match the measurement $(\hat{\mathbf{u}}'_i, \Lambda_i)$ with the j^{th} model point. From the current estimate $(\mathbf{T}^{cur}, \Sigma^{cur})$, we extract an estimate of the location $\mathbf{u}'_j$: $(\hat{\mathbf{t}}_j^{cur}, \Sigma_j^{cur})$ and evaluate the Mahalanobis distance between $\hat{\mathbf{t}}_j^{cur}$ and $\hat{\mathbf{u}}'_i$:

$$\delta = (\hat{\mathbf{t}}_j^{cur} - \hat{\mathbf{u}}'_i)^T (\Sigma_j^{cur} + \Lambda_i)^{-1} (\hat{\mathbf{t}}_j^{cur} - \hat{\mathbf{u}}'_i) \quad .$$

If δ is greater than a predefined threshold, the match is rejected. The greater the number of measurements and constraints fused prior to the match, the more precise is the estimate $\hat{\mathbf{t}}_j^{cur}$ (and so the uncertainty Σ_j^{cur} is smaller) and the elimination of irrelevant measurements is more effective. Therefore, there is great importance to the order of the points being fused (matched) since before matching the j^{th} model point, we would like the system to obtain as much information as possible on the location estimate $\hat{\mathbf{t}}_j$ so that the match verification is significant. Thus, at each step of the process the next point to be matched should be one associated with previously matched points through constraints, so that previous information (measurements and constraints) can be exploited. Before trying to match a measurement to a certain point we fuse as many constraints as possible associated with this point (and with previously matched points). The following algorithm follows this idea.

Denote by ψ_k $(k = 1 \ldots m)$ the constraints of the model and by $point(\psi_k)$ the set of points on which ψ_k depends. The order of fusion of the measurements and the constraints is obtained

from the following algorithm:

1. $FusedPoints = \emptyset \quad ; \quad FusedConst = \emptyset$
2. $PointList = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\} \quad ; \quad ConstList = \{\psi_1, \ldots, \psi_r\}$
3. **while** ($ConstList \neq \emptyset$ **and** $PointList \neq \emptyset$) **do**
   (a) **for each** ($\psi_k \in ConstList$ s.t. $point(\psi_k) \in point(FusedConst)$) **do**
      i. fuse ψ_k
      ii. delete ψ_k from $ConstList$ and add it to $FusedConst$
   (b) **if** there exist ($\mathbf{u}_k \in PointList$ s.t. $\mathbf{u}_k \in point(FusedConst)$) **do**
      i. find and fuse a matched measurement for $\mathbf{u}_k$
      ii. delete $\mathbf{u}_k$ from $PointList$ and add it to $FusedPoints$
   (c) **else if** there exists $\psi_k \in ConstList$ s.t. $(point(\psi_k) \cap FusedPoints) \neq \emptyset$ **do**
      i. fuse ψ_k
      ii. delete ψ_k from $ConstList$ and add it to $FusedConst$
   (d) **else** select an arbitrary $\psi_k \in ConstList$ and **do** steps i-ii in (c).

To summarize this algorithm, it chooses the next point or constraint to be fused according to a series of decreasing priorities:

1. An unfused constraint whose associated points are all associated with previously fused constraints.
2. An unfused point associated with already fused constraints.
3. An unfused constraint part of whose associated points have already been matched.
4. An unfused constraint none of whose associated points have been matched.

To illustrate the methodology, assume the hyper-graph depicted in Figure 6. Each dashed closed curve (edge in the hyper-graph) represents a constraint. The points associated with the constraint are encompassed by the curve. Assume, in this example, that we start the process by fusing constraint ψ_1. The order of fusion given by the algorithm will be as follows: $\psi_1 \rightarrow \mathbf{u}_1 \rightarrow \mathbf{u}_2 \rightarrow \psi_2 \rightarrow \mathbf{u}_3 \rightarrow \psi_3 \rightarrow \psi_4 \rightarrow \mathbf{u}_4 \rightarrow \mathbf{u}_5 \rightarrow \psi_5 \rightarrow \mathbf{u}_6$.

In the case where a good match for a model point can not be found due to occlusion or inability to obtain measurement about its position, we synthesize an artificial measurement

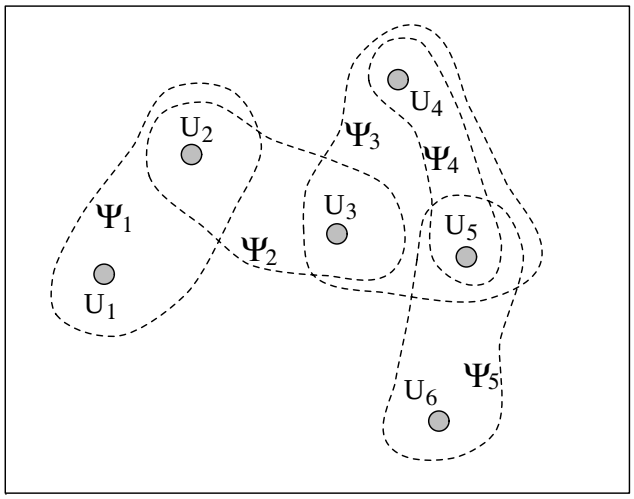

Figure 6: A hyper-graph representing model points and their associated constraints. A model point is represented by a full dot and a constraint is represented by a dashed closed curve, surrounding its associated points.

for the model point and associate it with an infinite uncertainty so that its influence on the rest of the process will be minimal. This scheme can also be helpful in fusing a constraint ψ_k when some of its associated points $point(\psi_k)$ are unavailable.

In the case where a constraint ψ_k is non-linear, we need a reasonable guess for the locations of its associated points to serve as linearization points during the fusion of ψ_k. However, it is possible that at time of fusion of ψ_k, part of its associated points have not yet been matched. In such a case, the matching step is performed simultaneously with the fusion of ψ_k. Thus, if the match is rejected, ψ_k must be relinearized about new matched measurements and ψ_k must be fused again.

The proposed algorithm is general and additional strategies can be applied to improve it. It is possible to improve the matching process assisting with intrinsic features (such as, color or local shape). It is possible to use more sophisticated strategy to choose the first costraint to be fused. This could be the constraint associated with the easiest point to match or the one that is associated with the most reliable measurements.

8 Constrained Objects Having Multiple-Point Components

We easily extend the solution for objects having one point per component to objects that have multiple-point components. For every component C_k, one must estimate the transformation $\hat{\mathbf{T}}_k$ which describes the pose of the component. The transformation $\mathbf{T}_k$ is composed of a quaternion $\tilde{q}_k$ and a translation vector $\mathbf{t}_k$ as described in Section 5 for rigid objects.

The process of evaluating all the transformations $\{\mathbf{T}_i\}$ is similar to the methods previously described for models having a single point per component:

1. Using the batch approach - The solution consists of two computational phases. In the first phase the transformation $\hat{\mathbf{T}}_k$ is estimated for each component C_k using the measurements associated with the points belonging to this component. In this phase the constraints existing between the components are not taken into consideration and the position of each component is estimated as if it was a single rigid object. The pose estimation of each component is computed as elaborated in Section 5. At the end of the first phase we have a set of evaluated transformations and their uncertainties:

$$\hat{\mathbf{T}} = \begin{pmatrix} \hat{\mathbf{T}}_1 \\ \hat{\mathbf{T}}_2 \\ \vdots \\ \hat{\mathbf{T}}_n \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} \Sigma_1 \\ \Sigma_2 \\ \vdots \\ \Sigma_n \end{pmatrix} \tag{8}$$

   In the second phase we consider $(\hat{\mathbf{T}}, \Sigma)$ as an *a priori* estimation for the transformations and we consider the set of constraints as artificial perfect "measurements". Fusing the constraint "measurements" with the *a priori* estimations is performed as in the single-point component case.

2. Using the incremental approach - The evaluated parameter vector in this method is $\mathbf{T}$ as given in Equation 8. Since each component is considered a rigid object the information obtained from a measurement is fused as described in Section 5. The information obtained from a constraint is fused as described in Section 6.2. The order in which the measurements and constraints are fused follows the algorithm given in Section 7 with the following two changes:
   The constraints are now associated with components rather than with single points,

therefore, in the algorithm, the components $\{C_k\}$ replace the model points. Additionally, following the fusion of the constraints associated with component C_k, we find and fuse matched measurements for all the features points $\{\mathbf{u}_{k,i}\}_{i=1\cdots m_k}$ (the feature points belonging to component C_k). The interpretation method for each component is similar to that for a rigid body, as presented in [14], where in this case there is additional *a priori* information about $\mathbf{T}_k$ from the previously fused constraints. This order of interpretation ensures that prior to fusion of a point in any component, all available information from neighboring components and mutual constraints have been exploited in order to assist in rejecting irrelevant matches.

In the case where every component contains several model points, there is no need to restrict the measurements to be 3D since the pose of the component can be estimated from projections (2D measurements) as mentioned in Section 5 and elaborated in [14].

9 Inequality Constraints

Inequality constraints appear in many man made objects. These constraints can assist in rejection of inconsistent interpretations that contradict the inequalities. Examples of such inequality constraints can be found in articulated models such as scissors and robot arms that are limited in the range of feasible angles between parts. Another example can be found in the work by Grimson [9, 10] where points are restricted to match segments of the model by limiting the range of distances between points.

The inequality constraints can be reduced to equality constraints by rewriting a given constraint:

$$g(x) \geq 0$$

as an equality constraint

$$g(x) - \lambda^2 = 0$$

where λ is a new variable that is added to the state vector and is estimated during the filtering process. Thus, every inequality constraint increases by one the dimensionality of the vector to be estimated. The initial *a priori* uncertainty associated with the parameter λ is infinite.

10 Computational Aspects

10.1 The Initial Guess

When dealing with the K.F. solution to the problem of constrained systems, one should note that similar to all other methods of constrained optimization [15] the K.F. method may erroneously converge to a local minima. Thus, here too the solution depends on the initial guess, i.e. on the *a priori* estimate supplied to the filter. However, we emphasize that from our experience of simulations of constrained systems, the system does converge to the global minima when a reasonable *a priori* estimate is given, and we find that its basin of attraction is quite large. Figure 7 displays results of simulations of a constrained system, for two *a priori* estimates. It can be shown that the *a priori* estimate input influences the final solution of the system.

10.2 Stability

Several problems may prevent the local iterations from converging or cause the solution to oscillate about the true solution. These problems arise from two main sources:

1. The fusion of perfect measurements with noisy measurements may create an ill-posed matrix which must be inverted during the filtering process. Inversion of such matrices creates computational imprecisions during the process.

2. The linear approximation of the non linear measurement model (which includes the constraints) may create imprecisions that can prevent convergence. This is specifically true when the *a priori* estimate is distant from the true solution (since linearization is performed around this *a priori* estimate).

In order to improve the linear approximation of non-linear measurements (and constraints) we employ local-iterations [17] in the batch K.F. process. In the incremental method, local iterations are applied at each fusion step. Additionally, the incremental method applies *global iterations* (see [17]) which repeats the full estimation process while performing linearization of the non-linear equations about the solution obtained at the end of the previous global iteration. The global iterations in the incremental method have the same effect as the local iterations in the batch method, however the former will require fewer iterations since the additional local iterations assist in the convergence.

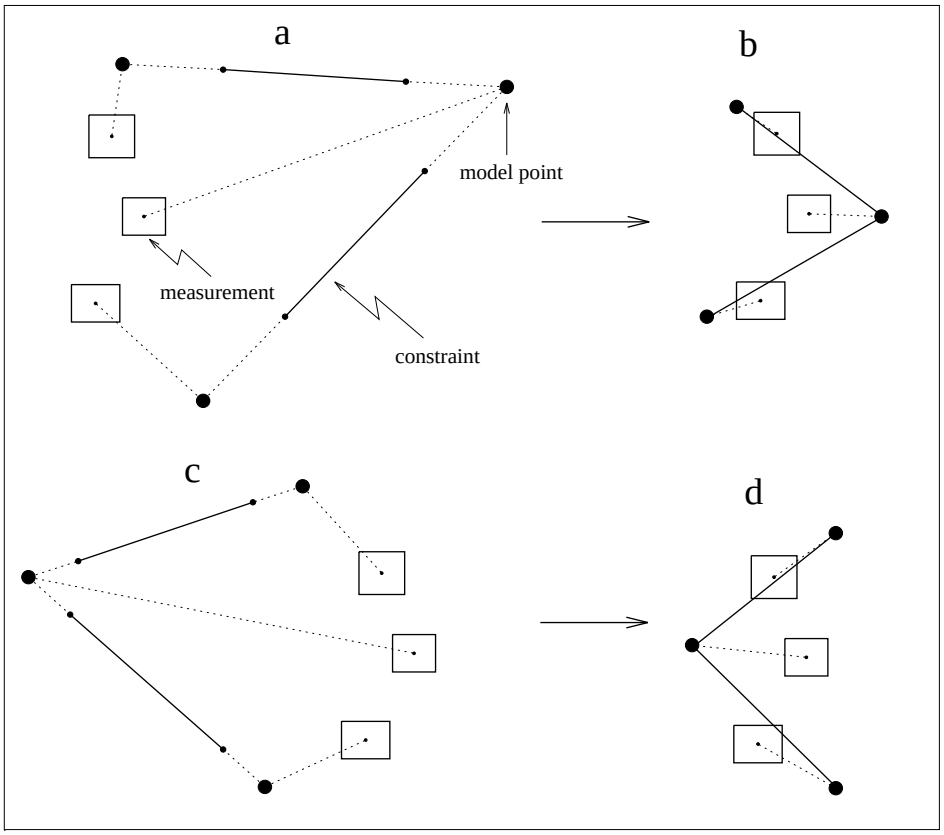

Figure 7: Two examples of *a priori* estimates (a and c) and their corresponding solutions (b and d). Model points are represented by black circles, measurements and their uncertainty by rectangles and distance constraints by bold lines. The dotted lines connect between a measurement and its associated model point.

From our experience of simulated constrained systems, we find that quite often the iterations oscillate about the true solution, specifically when the actual (noisy) measurements are distant from the true locations. This phenomenon is similar to the oscillations of a physical system of springs about a stable state in a frictionless environment (see [12] for a physical analogy to the K.F.). We extend the K.F. process to deal efficiently with this problem by adding a damping force to the process.

Incorporating a damping force is a common technique in optimization methods. There is a known trade-off between the rate of convergence and the reliability of the convergence, as a function of the damping strength added to the system. Adding a damping force to the K.F. process is simple in the case where the *a priori* estimate input is a user defined

initial guess with an associated infinite uncertainty (Section 6.1.1). In this case, the *a priori* estimate $\hat{\mathbf{T}}^0$ is given a finite uncertainty rather than an infinite one. Additionally, during the iterative process, we continuously update the *a priori* estimate input by taking the resulting estimate obtained at the previous step. This method adds a damping force, in the direction of the *a priori* estimate and can be shown to improve the condition of the inverted matrix in the K.F. updating equations. This is similar to convergence of a physical system of springs containing friction. As the uncertainty of the *a priori* estimate is smaller, the damping force is larger. We should note that Lowe [21], in dealing with pose estimation of articulated objects using minimization of free parameters, also includes a damping factor to stabilize the solution. Additionally he discusses the relation between the strength of the damping force, and the rate and assurance of the convergence. This relation was defined by Levenberg and Marquardt [18, 22] and holds in our case as well: When the damping factor is small (i.e. the *a priori* uncertainty is large), the process is similar to the Newton iteration, which ensures fast convergence but a small basin of attraction. As the damping factor increases (smaller *a priori* uncertainty), the process is more similar to regular gradient-descent methods resulting in decreasing incremental steps but increasing the basin of attraction. Marquardt [22] suggest a simple algorithm for adjusting the damping strength at each step. This algorithm is easily implemented in our framework. Implementing this method gave very good results and allowed a smooth and stable convergence of the process. However, it must be to note that as the number of non-linear constraints increases, the stability of the process decreases, and a more accurate initial guess is required in order to guarantee convergence. Convergence results on simulated constrained systems with and without damping factor can be seen in [12].

Another advantage of using the damping force, is that the initial guess inserted into the system based on some external knowledge (such as "expected" position), can serve not only as a linearization point but also as an additional, already matched, measurement that can assist in rejecting false matches. This can be done only if the initial guess has some finite uncertainty (as in this case). This idea is used in an elegant way; while in the matching stage the initial guess contributes its information, in the final stage, where the real measurement has been fused, the initial guess does not bias the solution as having an additional measurement.

10.3 Existence of a Solution

Till now we assumed that a solution exists for the constrained system. However, this assumption is not always true; there are cases where the constraints conflict with one another and no solution exists. Furthermore, even when no conflicts arise in the system, some cases can not be solved in the usual method. In order to analyze those cases in which the system does not have a solution, let us consider the batch implementation (described in Section 6.1) where the actual measurements are supplied to the filter as an *a priori* estimate input and the constraints are given as "measurements" with zero uncertainty. In this implementation, the K.F. equations involve inversion of the matrix: $H\Lambda H^T$ (see Equation 7). The process will fail when this matrix is singular. Thus, it is sufficient to study the singularity cases of this matrix. H is a $d \times 3n$ matrix, Λ is an $3n \times 3n$ matrix and $H\Lambda H^T$ is a $d \times d$ matrix ($d = \sum dim(\psi_j)$ and n is the number of model points). Let us first suppose that $k = 3n < d$ in which case $Rank(H) \leq k$, therefore also $Rank(H\Lambda H^T) \leq k < d$. But $H\Lambda H^T$ is a $d \times d$ matrix, therefore it is singular. In other words, there is no solution when the system is over-constrained (more constraint equations than unknowns). On the other hand, suppose that $d < k$ but some of the constraints are dependent on other constraints. This means that if constraint linearization is performed at the true solution, some rows of H will be linearly dependent on other rows. In this case $H\Lambda H^T$ is singular since $Rank(H) < d$ hence $Rank(H\Lambda H^T) < d$. The cases when $H\Lambda H^T$ can not be inverted can be easily identified in the course of the solution process. When this happens, we can try an additional linearization point, since singularity may be the result of a special configuration of the linearization point. When this fails and there is a reason to suspect that the constraints are not contradicting we can attempt to find a maximal set of rows in H which are linearly independent and try to solve the resulting system (in the incremental implementation the last constraint is eliminated, in the batch implementation $H\Lambda H^T$ is inverted using SVD decomposition).

10.4 Complexity

The complexity of each iteration of the K.F. is $\max[O(d^3), O(k^2 d)]$ where d is the dimensionality of the measurement vector and $k = 3n$ is the dimensionality of the state vector. This complexity is due to matrix inversion and matrix multiplications during the computation of the K.F. equations.

In the batch process k is proportional to the number of model points and d depends on the type of implementation: If the measurement input consists only of constraint "mea-

surements" we have $d = \sum_i \dim(\psi_i)$. In the implementations which add a user defined *a priori* estimate to the system the measurement input also includes the actual measurements, and then $d = \sum_i \dim(\psi_i) + \dim(T)$. However, the system must not be over-constrained, so $\sum_i \dim(\psi_i) \leq \dim(T)$. Thus, $d \leq 2\dim(T)$ and the upper bound of the complexity of each iteration is $O(n^3)$.
In the incremental process every fusion of a single measurement requires time complexity of $O(n^2)$ (the dimensionality d of each measurement is constant in this case). Thus, the complexity of a single global iteration is, similar to the batch method, $O(n^3)$ (assuming the number of local iterations is restricted).

The rate of convergence (number of iterations) also depends on the implementations: If no damping factor is added to the system, the convergence rate of the system is equivalent to that of the Newton iterations which is quadratic near the solution [28]. When a damping factor is use to stabilize the process, the rate of convergence decreases as the strength of the damping increases.

Run time can be reduced if the matrix $M = H\Lambda H^T$ is not inverted at each iteration. Instead, one can use an approximation of M obtained from previous stages. This approach is appropriate in those implementations where the rate of convergence is low (due to high damping factor) so the state-vector $\mathbf{T}$, and accordingly the matrix M, do not change greatly between iterations ($\mathbf{T}$ is the linearization point producing M). In terms of run time, the simulated examples shown in Figure 9 run on a Sparc2 workstation for about 1-2 seconds.

11 Results

11.1 Simulated Data

We applied the constraint fusion method to estimate the pose of a $2D$ constrained model consisting of single point components. We used the parametric modeler described in [13] which we developed based on our techniques. The modeler enables the definition of constraint graphs using the following types of constraints: co-linearity of three points, a particular distance between two points, a particular distance between a point and a fixed location, constraining a point to lie on a fixed line and constraining a point to be on one side of a fixed line (inequality constraint). It was demonstrated that the algorithm is capable of computing solutions to complex models. Not surprisingly, the batch and the incremental techniques

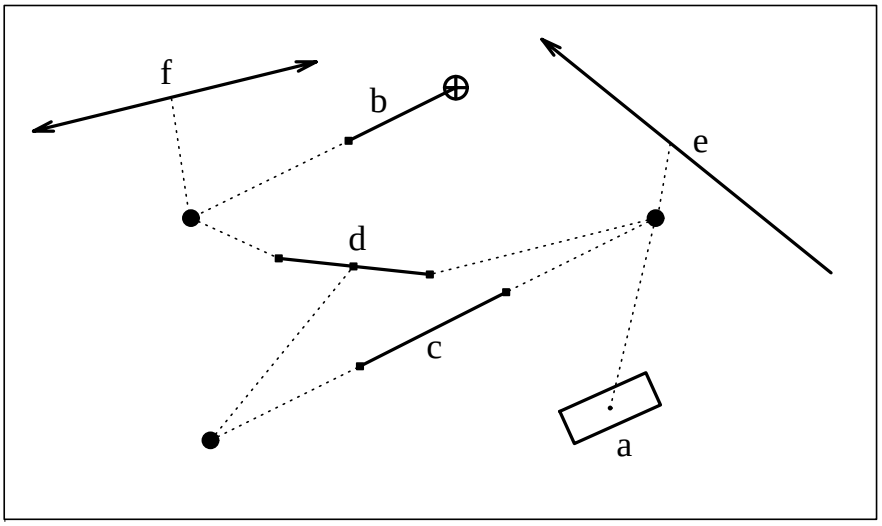

Figure 8: An example of measurements and constraints with three model points. The model points to be positioned are shown as full circles. A measurement (a) is represented as a rectangle positioned at the measurement location and having width and length proportional to the s.t.d. of the measurement. A *fixed location distance* constraint (b) is represented by a line segment with one endpoint at the fixed location and the other connected to the constrained point. An *inter-point distance* constraint (c) is represented by a line segment connecting the constrained points. A *three points co-linearity* constraint (d) is represented by a fixed length line segment connected on both its ends and its middle to the constrained points. A *point on right of line* constraint (e) is represented by an arrow-headed line segment connected to a model point. A *point on line* constraint (f) is represented by a double arrow-headed line connected to a model point.
The connections between constraints and associated model points, are marked by dashed lines.

gave the same solution for different model configurations.

The following figures were created using this software. In the figures, model points and constraints are represented as shown in Figure 8 and described in the caption. Figure 9 shows four complex examples which include inequality constraints. As can be seen, the solutions conserve the model constraints. Further examples with various initial guesses and different measurements can be found in [12].

In order to confirm the validity of the results, we applied the suggested methods to several simple examples containing only two degrees of freedom. The obtained results were compared to "energy" maps describing the Mahalanobis distance $(\hat{\mathbf{T}} - \hat{\mathbf{u}}')^T \Lambda^{-1} (\hat{\mathbf{T}} - \hat{\mathbf{u}}')$ at

each permitted position. Figure 10 shows two examples of constrained objects having two degrees of freedom denoted θ and ϕ. The energy maps corresponding to these examples describe the energy of the system for every θ, ϕ pair (Figure 10-bottom). The displays show that the solution obtained by the constraint fusion method (marked as X in the energy map) indeed corresponds to the minimum energy solution.

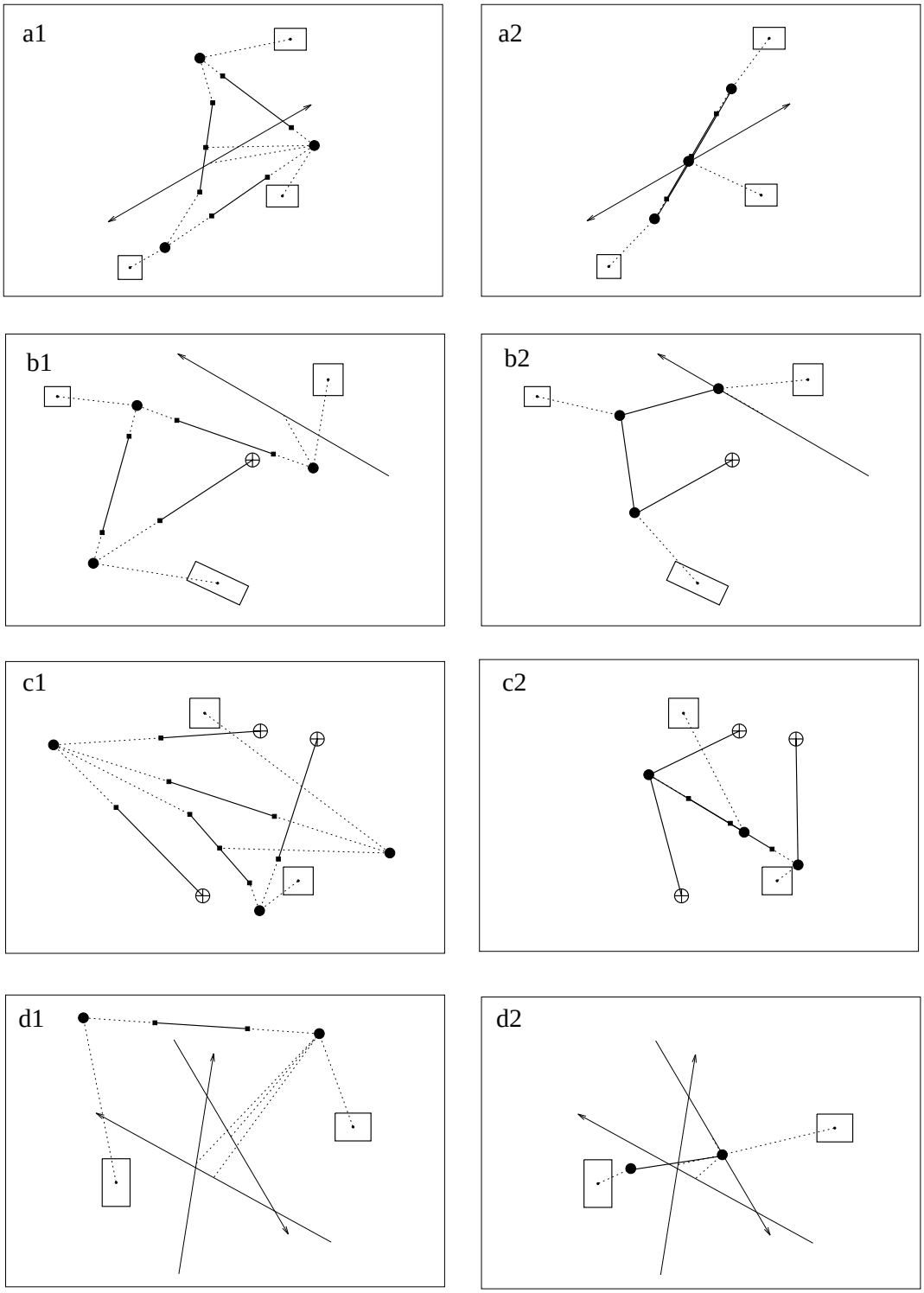

Figure 9: Some examples of constrained models including inequality constraints. Four examples are shown, the original input (measurements, constraints and model points) are shown on the left and the solution is shown on the right.

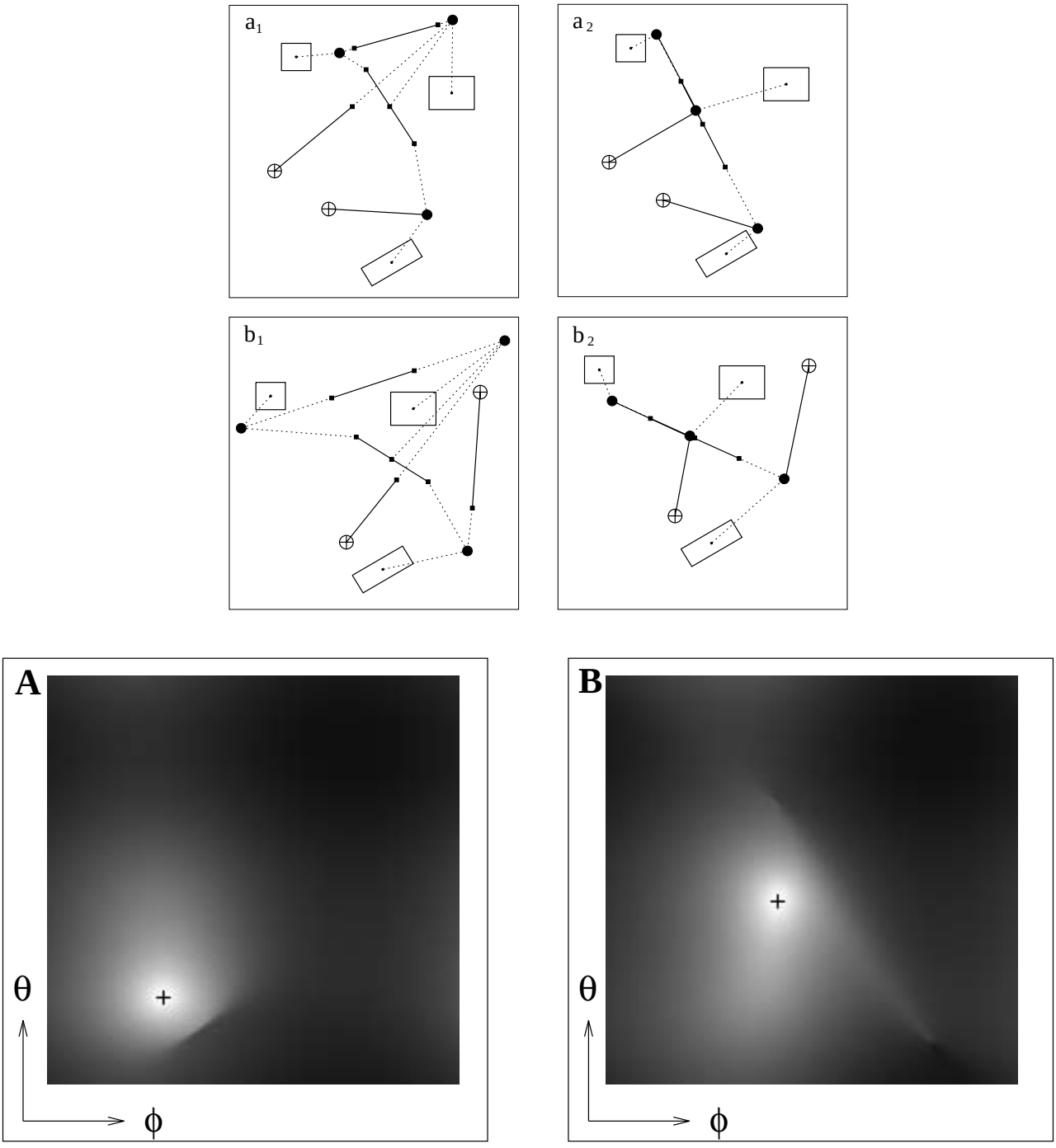

Figure 10: Two cases (a and b) of a constrained model with two degrees of freedom (denoted θ and ϕ). a_1, b_1 - the initial guess of the solution. a_2, b_2 - the final solution. The energy maps corresponding to each case is shown at the bottom. White values denote low energy of the system. The black crosses mark the θ, ϕ corresponding to the final solution of a_2, b_2 as obtained using our method. It is seen that the final solution indeed corresponds to the minimum energy.

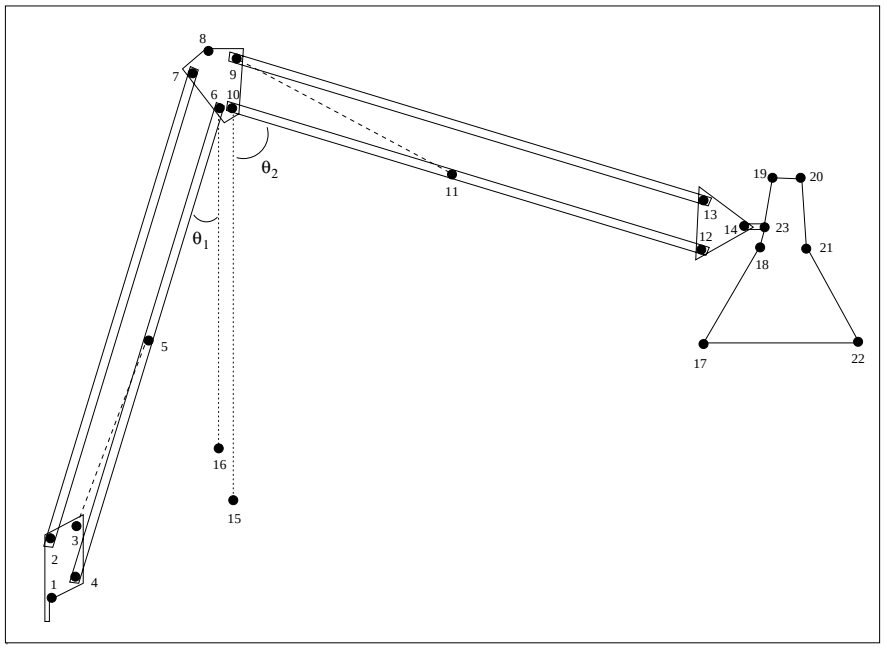

Figure 11: A schematic diagram of a lamp model having 23 points.

11.2 Real Image Data

We used the constraint fusion method to estimate the position of a real articulated 3D object from 2D images. The articulated model used, is a desk lamp shown in Figure 12, having 5 degrees of freedom. We consider the lamp model as a 23-point model (as shown in Figure 11) and we included the following constraints into the model:

- 31 constant distance constraints between pairs of points in the model (for example points 10 and 12).
- 3 parallel constraints between 2 pairs of points (between points 6 and 16 and points 10 and 15).
- 15 co-planar constraints between 4 or more points (points 10,12,13 and 9 are constrained to be co-planar).
- 2 co-linear constraints between 3 points (points 4, 5 and 6 are co-linear).

Measurements of the 3D location of the points and the measurement uncertainty were obtained from stereo image pairs. This data is noisy due to digitization, inconsistent lighting and imprecise feature matching. The uncertainties due to noise were modeled according to

the auto-correlation of the image features [27]. We estimated the pose of the lamp components from the noisy 3D measurements and from the constraints, using the constraint fusion technique. The evaluated vector is a $23 \cdot 3$ dimensional location vector composed of the 23 locations of the model points. Figures 12a and 12b show 2 examples of lamp images having 2 different positions. Figures 12c and 12d show the corresponding results as synthetic images created from the estimated location vector. As can be seen, there is high correlation between the real model location and the synthesized reconstruction.

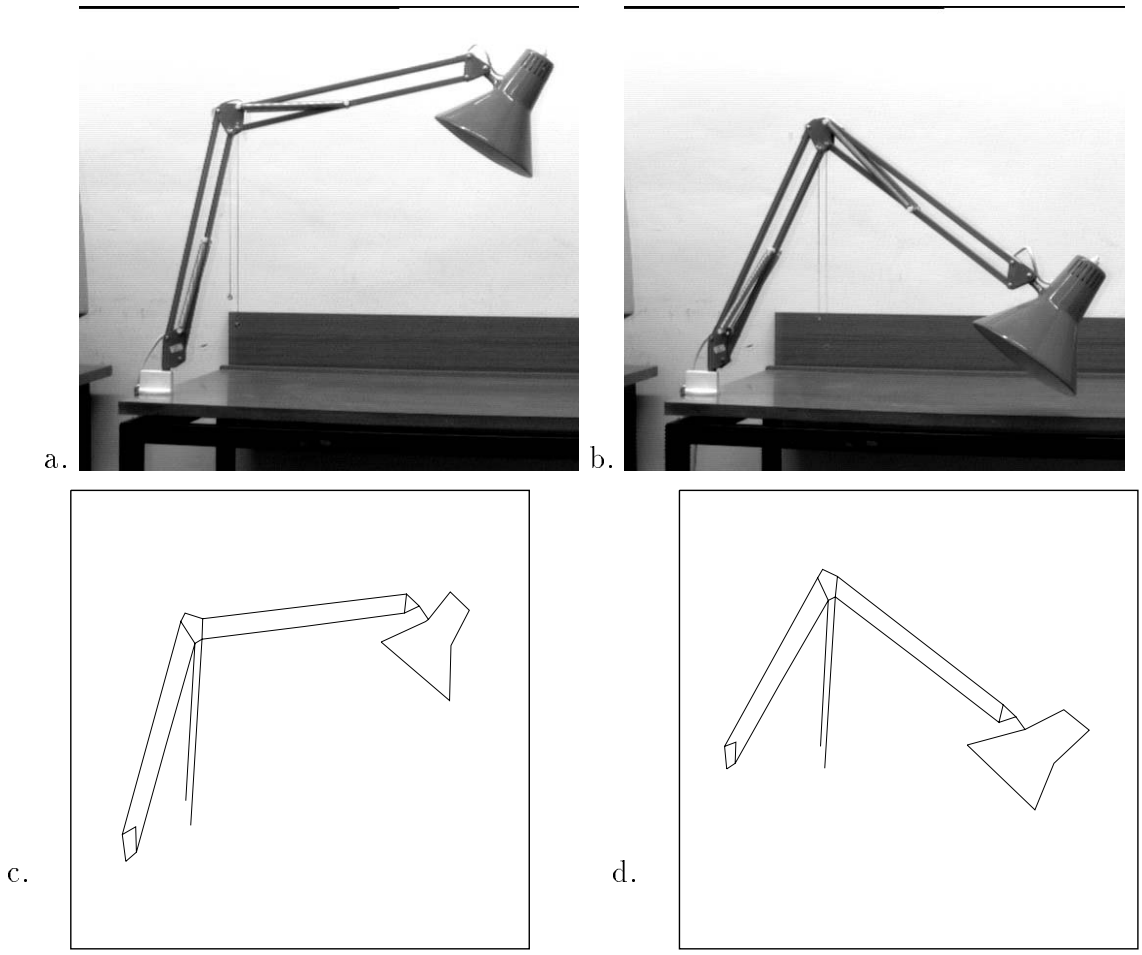

Figure 12: a-b) Images of a desk lamp at different positions. c-d) The corresponding result shown as a synthetic image created from the estimated location vector.

Additionally, the angles θ_1 and θ_2 (shown in Figure 11) were physically measured in several positions of the lamp. These values were also extracted from the pose estimate obtained with our method. The real values and the constructed values for four typical examples are

compared in the following table:

		pose A	pose B	pose C	pose D
Real Values	θ_1	17.0	12.0	14.0	26.0
	θ_2	61.0	103.0	84.0	56.0
Reconstructed Values Without Constraints	θ_1	20.2	13.6	13.7	29.3
	θ_2	58.4	104.6	80.2	53.4
Reconstructed Values With Constraints	θ_1	16.5	12.3	14.1	25.8
	θ_2	60.8	104.2	84.7	57.4

The table shows the improvement in the reconstructed angle values when the fusion includes the constraints. The difference between the real angle value and the reconstructed value decreases when the constraints are fused. The s.t.d of these differences is 2.60 for the reconstruction without fusing the constraints and 0.73 for the reconstruction with constraint fusion.

The importance of propagating the pose information of each component to its neighboring components, is shown in Figures 13 and 14. Figure 13 shows several views of a synthesized lamp reconstructed only from the 3D measurements taken on the lamp shown in Figure 12b. Figure 14 shows the same views after mutual information was propagated between the components through the constraints. The improvement is significant, as demonstrated.

11.3 The Measurement Interpretation

Figure 15 shows a limited part of the interpretation tree (I.T.) which is constructed for the desk lamp interpretation. This I.T. is used for the matching process as described in Section 7. Each node on the kth level of this I.T. represents a possible matching between the kth model point, as numbered in Figure 11, and some particular measured point. The measurements are numbered according to their real correspondence (i.e. the true match of the kth measured point is the kth model point). The score of each match is shown at the appropriate node where the value is the Mahalanobis distance δ calculated using the formula given in Section 7. For each level in the I.T. we show the three best scored nodes. The model constraints are fused during the parsing of the I.T. as described by the algorithm in Section 7. The distance constraints between model points k and j (denoted by $dist(k,j)$)

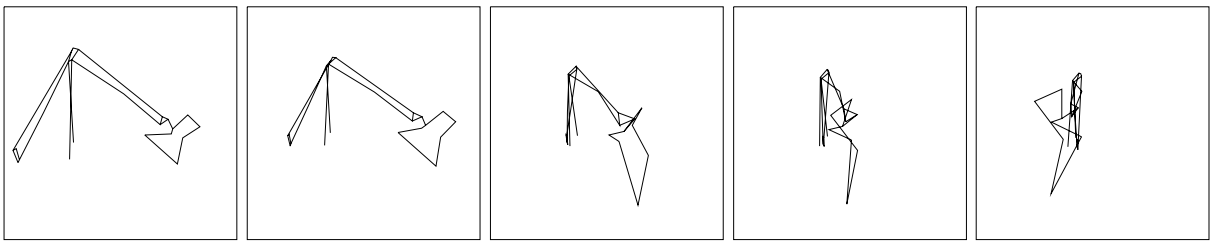

Figure 13: Several views of a synthesized lamp reconstructed from the 3D measurements only.

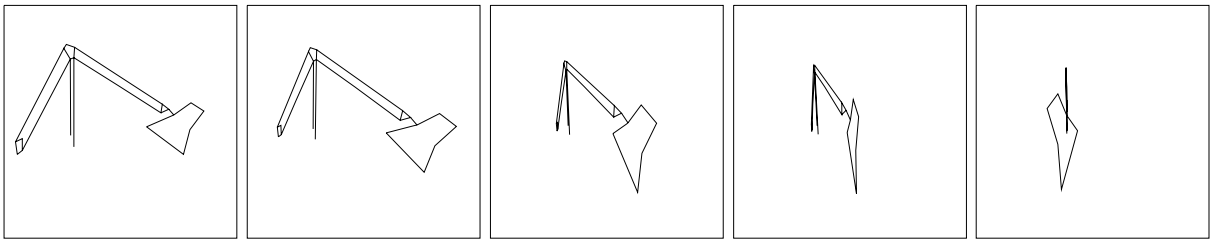

Figure 14: Several views of a synthesized lamp reconstructed from the 3D measurements and the model constraints.

are shown in the figure at the level at which they are fused. As can be seen the score of the correct matches are, in most cases, significantly lower than the erroneous matches. An example where the score of the correct match is not significantly lower can be seen in the 6^{th} level where measurement no. 6 scores 0.95 and measurement no. 10 scores 3.8. This relatively small difference is caused by the fact that the distance between point 5 and 6 is indeed similar to the distance between point 5 and 10. Therefore, the constraint $dist(5,6)$ which has been fused prior to this stage can not give a significant indication as to which measurement is the correct one. In such a case, the IT should be explored in both directions.

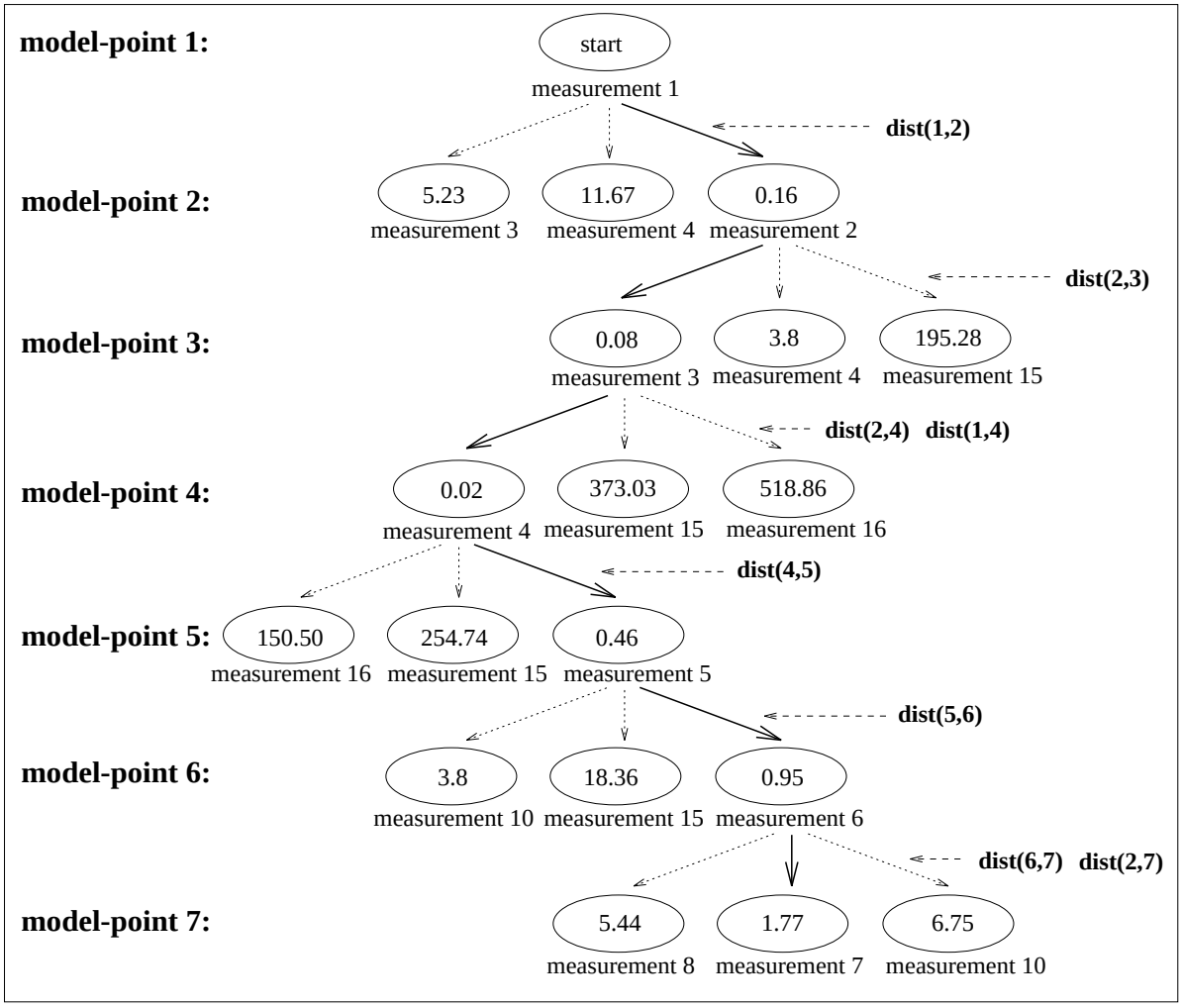

Figure 15: Results of the matching algorithm for the lamp model. A section of the pruned interpretation tree is displayed. Every level of the tree corresponds to one model point, and each node at a particular level corresponds to a possible match between the model point and a measured point. The score of each match is shown at the node where the value is the Mahalanobis distance of this match. For each level in the interpretation tree the three best scored nodes are shown. The distance constraints (denoted by $dist(k, j)$) are shown at the level at which they are fused.

12 Conclusion

This paper presented a framework based on Kalman filtering and constraint fusion for model based pose estimation and interpretation that is general enough to cover articulated and other types of non-rigidly constrained models. The constraints are general and can be associated with any number of different parts of the model. The validity of the framework was shown on real and simulated images.

The constraint fusion method has several advantages over existing methods:

1. In any pose estimation method, exploiting the information supplied by the measurement requires a definition of the functional dependence between the measurement and the estimated parameters. In the parametric methods (Section 3), this dependence is not simple since it must include all the parameters on which the measurement depends. Additionally, the order of the nonlinearity of the dependence equations increases with the number of parameters. In the constraint fusion method, the functional dependence includes only the local parameters (i.e. only the parameters that define the transformation of the measured component) since the dependence of the measurement on other parameters is expressed through the constraints of the model. This local dependence is simply defined and is not as highly nonlinear as that of the parametric methods. Additionally, there is no need to reduce the constrained parameter space into a set of free parameters (as is performed in the parametric methods), thus the definition of the set of parameters to be estimated is simple.

2. Using the constraint fusion method, constraints can be added at run time and therefore this method can be used in interactive systems. In the parametric methods, the constraints must be given and analysed prior to the estimation process.

3. The information obtained on the position of a given component is propagated to all other components of the model through the constraints between them. Thus the estimated pose of a certain component takes into consideration all the existing measurements and all the defined constraints (this is not true in the divide and conquer methods).

4. The proposed scheme enables an efficient simultaneous matching procedure that allows incremental fusion of additional matches that improve the pose estimation and the search complexity in the I.T.

5. The existing methods of pose estimation of constrained models, deal with articulated objects and with constraints that are due to prismatic or revolute joints between the model components. In the constraint fusion method we are not limited to any type of constraints and can deal with all types of constraints including co-linearity, co-planarity, constant distance, constant angle, etc. Additionally, we deal with inequality constraints such as limited range of distances between points or limited range of angles.

6. The constraint fusion method uses K.F. tools and so includes the advantages associated with the K.F. such as explicitly dealing with the measurement uncertainty, simple updating of the solution given additional measurements, easy parallelization, and the possibility of using an efficient matching strategy.

There are computational aspects that were not covered in this paper such as methods to speed up the computation and to reduce the time complexity using Optimal Smoothing. This technique is described in [12].

This framework has also been carried over to graphics in which it is used in a parametric modeler system [13]. Additionally, the method can be easily extended to handle flexible models made of elastic materials. This extension can be performed by associating non-zero uncertainty with the artificial "measurements" produced from the constraints. Current work extends the framework to deal with dynamic articulated systems. This extension is relevant to keyframe animation and inverse kinematics.

Appendix:

A The Extended Kalman Filter Equations for Pose Estimation of a Single Component

Assume an estimation process for a seven dimensional parameter vector $\mathbf{T}$:

$$\mathbf{T} = \begin{pmatrix} \tilde{\mathbf{q}} \\ \mathbf{t} \end{pmatrix}$$

At each step k we have, from the previous step, an *a priori* estimate of the evaluated vector $\hat{\mathbf{T}}^{k-1}$ with an associated uncertainty matrix Σ^{k-1}, and a new measurement $\hat{\mathbf{u}}'_k$ with its uncertainty Λ_k. The mathematical relationship between the measurement and the evaluated

vector is represented by a vector equation:

$$h_k(\mathbf{u}'_k, \mathbf{T}) = 0$$

In our case:

$$h_k = R\mathbf{u}_k + \mathbf{t} - \mathbf{u}'_k = \mathbf{0} \quad ,$$

where R and $\mathbf{t}$ are the rotational matrix and the translation vector and can be constructed from $\mathbf{T}$. In our case h_k is non-linear, therefore, transition from step $k-1$ to step k is performed using a linear approximation of h_k by taking the first order Taylor expansion around $(\hat{\mathbf{T}}^{k-1}, \hat{\mathbf{u}}'_k)$:

$$h_k(\mathbf{u}_k, \mathbf{u}'_k, \mathbf{T}) = \mathbf{0} \approx \hat{h}_k(\mathbf{u}_k, \hat{\mathbf{u}}'_k, \hat{\mathbf{T}}^{k-1}) + \frac{\partial h_k}{\partial \mathbf{u}'_k}(\mathbf{u}'_k - \hat{\mathbf{u}}'_k) + \frac{\partial h_k}{\partial \mathbf{T}}(\mathbf{T} - \hat{\mathbf{T}}^{k-1}) \tag{9}$$

Equation 9 can be rewritten as a linear equation:

$$\mathbf{z}_k = H_k\mathbf{T} + \eta_k \quad ,$$

where

$$\begin{aligned}
\mathbf{z}_k &= -\hat{h}_k(\mathbf{u}_k, \hat{\mathbf{u}}'_k, \hat{\mathbf{T}}^{k-1}) + (\frac{\partial h_k}{\partial \mathbf{T}})\hat{\mathbf{T}}^{k-1} \\
H_k &= \frac{\partial h_k}{\partial \mathbf{T}} = \left[\frac{\partial h_k}{\partial \tilde{\mathbf{q}}}, \frac{\partial h_k}{\partial \mathbf{t}}\right] = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 & 1 & 0 & 0 \\ -d_3 & -d_2 & d_1 & d_0 & 0 & 1 & 0 \\ d_2 & -d_3 & -d_0 & d_1 & 0 & 0 & 1 \end{bmatrix} \\
\eta_k &= \frac{\partial h_k}{\partial \mathbf{u}'_k}(\mathbf{u}'_k - \hat{\mathbf{u}}'_k)
\end{aligned}$$

and where

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = 2 \begin{bmatrix} q_0^{k-1} & -q_3^{k-1} & q_2^{k-1} \\ q_1^{k-1} & q_2^{k-1} & q_3^{k-1} \\ -q_2^{k-1} & q_1^{k-1} & q_0^{k-1} \\ -q_3^{k-1} & -q_0^{k-1} & q_1^{k-1} \end{bmatrix} \mathbf{u}_k \quad .$$

(see [25] for explanations about the representation of a rotation matrix by a unit quaternion). $\mathbf{z}_k$ represents the new three dimensional "measurement" vector, H_k is a 3×7 matrix denoting a linear connection between the "measurement" and the actual transformation $\mathbf{T}$. Both $\mathbf{z}_k$ and H_k can be derived from $\hat{\mathbf{u}}'_k$, $\mathbf{u}_k$ and $\hat{\mathbf{T}}^{k-1}$. The term η_k is the noise in the "measurement"

$\mathbf{z}_k$ and satisfies:

$$\begin{aligned} E\{\eta_k\} &= 0 \\ E\{\eta_k \eta_k^T\} &= \left(\frac{\partial h_k}{\partial \mathbf{u}'_k}\right) \Lambda_k \left(\frac{\partial h_k}{\partial \mathbf{u}'_k}\right)^T = W_k \\ E\{\eta_k \eta_\ell^T\} &= 0 \quad k \neq \ell \quad . \end{aligned}$$

The K.F. fuses the new measurement with the *a priori* estimate of the evaluated vector $\mathbf{T}$ and produces an updated estimate $\hat{\mathbf{T}}^k$ with the associated uncertainty Σ^k. The K.F. fuser equations are:

$$\begin{aligned} &\textit{state estimate update}: && \hat{\mathbf{T}}^k = \hat{\mathbf{T}}^{k-1} + K_k(\mathbf{z}_k - H_k \hat{\mathbf{T}}^{k-1}) \\ &\textit{state covariance update}: && \Sigma^k = \Sigma^{k-1} - K_k H_k \Sigma^{k-1} \\ &\textit{Kalman gain matrix}: && K_t = \Sigma^{k-1} H_k^T (H_k \Sigma^{k-1} H_k^T + \Lambda_k)^{-1} \quad . \end{aligned}$$

The constraint of the quaternion $\|\tilde{\mathbf{q}}\|^2 = 1$ is not necessarily conserved in the estimated transformation $\hat{\mathbf{T}}^k$. In order to enforce this constraint an additional artificial measurement is fused. The additional measurement is derived by linearizing the constraint equation around $\hat{\mathbf{T}}^{k-1}$:

$$1 + (\hat{\mathbf{T}}^{k-1})^T \begin{bmatrix} I_4 & 0 \\ 0 & 0 \end{bmatrix} \hat{\mathbf{T}}^{k-1} = 2(\hat{\mathbf{T}}^{k-1})^T \begin{bmatrix} I_4 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{T}$$

where the measurement (the left side of the equation) is associated with zero uncertainty.

B Batch Solution for an Articulated Object

Assume the articulated object as illustrated in Figure 4. This object has five point components $\{\mathbf{u}_i\}_{i=1}^5$ and four articulated constraints $\{\psi_j(\mathbf{T})\}_{j=1}^4$. Each constraint is of the form

$$\psi_j(\mathbf{t}_k, \mathbf{t}_l) = \|\mathbf{t}_l - \mathbf{t}_k\|^2 - d_{k,l}^2 = 0 \tag{10}$$

representing the constant Euclidean distance existing between the points $\mathbf{u}_k$ and $\mathbf{u}_l$. Additionally, assume a single measurement $(\hat{\mathbf{u}}'_k, \Lambda_k)$ for each model point $\mathbf{u}_k$. Linearization of Equation 10 around $(\hat{\mathbf{u}}'_k, \hat{\mathbf{u}}'_l)$ yields:

$$\frac{1}{2}(d_{k,l}^2 + \mathbf{v}_{k,l}^2) = \mathbf{v}_{k,l}\mathbf{t}_l - \mathbf{v}_{k,l}\mathbf{t}_k$$

where

$$\mathbf{v}_{k,l} = \hat{\mathbf{u}}'_l - \hat{\mathbf{u}}'_k \quad .$$

Linearizing all the constraint equations and concatenating them into a single vector equation gives:

$$\mathbf{z} = H\mathbf{T} \tag{11}$$

where

$$\mathbf{z} = \frac{1}{2}\begin{pmatrix} d_{1,2}^2 + \mathbf{v}_{1,2}^2 \\ d_{1,3}^2 + \mathbf{v}_{1,3}^2 \\ d_{3,4}^2 + \mathbf{v}_{3,4}^2 \\ d_{3,5}^2 + \mathbf{v}_{3,5}^2 \end{pmatrix} \quad ; \quad H = \begin{bmatrix} \mathbf{v}_{2,1} & -\mathbf{v}_{2,1} & 0 & 0 & 0 \\ \mathbf{v}_{3,1} & 0 & -\mathbf{v}_{3,1} & 0 & 0 \\ 0 & 0 & \mathbf{v}_{4,3} & -\mathbf{v}_{4,3} & 0 \\ 0 & 0 & \mathbf{v}_{5,3} & 0 & -\mathbf{v}_{5,3} \end{bmatrix} \quad ; \quad \mathbf{T} = \begin{pmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_5 \end{pmatrix} \quad .$$

Solving the illustrated example is performed by supplying the K.F. fuser with the following inputs:

1. The *a priori* estimate for the evaluated transformation will be:

$$\left[\begin{pmatrix} \hat{\mathbf{u}}'_1 \\ \vdots \\ \hat{\mathbf{u}}'_5 \end{pmatrix}, \begin{pmatrix} \Lambda_1 & & 0 \\ & \ddots & \\ 0 & & \Lambda_5 \end{pmatrix} \right]$$

2. The measurement is the vector $\mathbf{z}$ (of size 15) and its associated uncertainty - a 15×15 zero matrix.

3. The mathematical relationship between $\mathbf{z}$ and the evaluated parameters $\mathbf{T}$, is given by Equation 11 as formulated above.

References

[1] B.D.O. Anderson and J.B. Moore. *Optimal Filtering.* Prentice-Hall, Inc., N.J., 1979.

[2] N. Ayache. *Artificial Vision for Mobile Robots Stereo Vision and Multisensory Perception.* MIT Press, 1991.

[3] N. Ayache and O.D. Faugeras. Building, registrating, and fusing noisy visual maps. In *International Conf. on Computer Vision*, pages 73–82, 1987.

[4] A. Beinglass and H.J. Wolfson. Articulated object recognition, or: How to generalize the generalized hough transform. In *Conference on Computer Vision and Pattern Recognition*, pages 461–466, 1991.

[5] R.A. Brooks. Model-based 3-D interpretation of 2-D images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5:140–150, 1983.

[6] O.D. Faugeras, N. Ayache, and B. Faverjon. A geometric matcher for recognizing and positioning 3D rigid objects. In *Conference on Artificial Intelligence Applications*, pages 218–224, 1984.

[7] O.D. Faugeras and M. Hebert. The representation, recognition, and positioning of 3D shapes from range data. In A. Rosenfeld, editor, *Techniques for 3D Machine Perception*, pages 13–51. Elsevier Science, 1986.

[8] W.E.L. Grimson. Recognition of object families using parameterized models. In *International Conf. on Computer Vision*, pages 93–101, 1987.

[9] W.E.L. Grimson. On the recognition of parametrized 2-D objects. *International Journal of Computer Vision*, 2:353–372, 1989.

[10] W.E.L. Grimson. On the verification of hypothesized matches in model-based recognition. In *European Conference on Computer Vision*, pages 489–498, 1990.

[11] W.E.L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[12] Y. Hel-Or. *Pose Estimation from Uncertain Sensory Data.* PhD thesis, Inst. of Computer Sciense, The Hebrew University of Jerusalem, 1993.

[13] Y. Hel-Or, A. Rappoport, and M. Werman. Relaxed parametric design with probabilistic constraints. In *Solid Modelling-93*, 1993.

[14] Y. Hel-Or and M. Werman. Pose estimation by fusing noisy data of different dimensions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):195–200, February 1995.

[15] D.M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill, 1972.

[16] B.K.P. Horn. *Robot Vision.* MIT Press, 1986.

[17] A.H. Jazwinski. *Stochastic Process and Filtering Theory.* Academic Press, 1970.

[18] K. Levenberg. A method for the solution of certain non-linear problems in least-squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[19] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[20] D.G. Lowe. Stabilized solution for 3-D model parameters. In *European Conference on Computer Vision*, pages 408–412, 1990.

[21] D.G. Lowe. Fitting parametrized 3-D models to images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:441–450, May 1991.

[22] D.W. Marquardt. An algorithm for least-squares estimation of non-linear problems. *Journal Soc. Indust. Appl. Math.*, 11:431–441, 1963.

[23] P.S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, 1979.

[24] I.J. Mulligan, A.K. Mackworth, and P.D. Lawrence. A model-based vision system for manipulator position sensing. In *Workshop on Interpretation of 3D Scenes, Austin, Texas*, pages 186–193, 1989.

[25] B. Sabata and J.K. Aggarwal. Estimation of motion from a pair of range images: A review. *Computer Vision, Graphics and Image Processing*, 54(3):309–324, Nov 1991.

[26] T. Shakunaga. Pose estimation of jointed structures. In *Conference on Computer Vision and Pattern Recognition*, pages 566–572, 1991.

[27] A. Shmuel and M. Werman. Active vision: 3D depth from an image sequence. In *International Conference on Pattern Recognition*, pages 48–54, 1990.

[28] G. Strang. *Introduction to applied mathematics*. Wellesley-Cambridge Press, 1986.

[29] J. Weng, N. Ahuja, and T.S. Huang. Optimal motion and structure estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 144–152, 1989.