A Framework for Generic State Estimation in Computer Vision Applications

Cristian Sminchisescu, Alexandru Telea

A Framework for Generic State Estimation in Computer Vision Applications

Cristian Sminchisescu[1], Alexandru Telea[2]

[1] INRIA Rhône-Alpes,
Institut National Polytechnique de Grenoble, France,
Cristian.Sminchisescu@inrialpes.fr,
[2] Eindhoven University of Technology,
Dept. of Mathematics and Computer Science, The Netherlands,
alext@win.tue.nl

Abstract. Experimenting and building integrated, operational systems in computational vision poses both theoretical and practical challenges, involving methodologies from control theory, statistics, optimization, computer graphics, and interaction. Consequently, a control and communication structure is needed to model typical computer vision applications and a flexible architecture is necessary to combine the above mentioned methodologies in an effective implementation. In this paper, we propose a three-layer computer vision framework that offers: a) an application model able to cover a large class of vision applications; b) an architecture that maps this model to modular, flexible and extensible components by means of object-oriented and dataflow mechanisms; and c) a concrete software implementation of the above that allows construction of interactive vision applications. We illustrate how a variety of vision techniques and approaches can be modeled by the proposed framework and we present several complex, application oriented, experimental results.

1 Introduction

Experimenting and building systems in computational vision poses several major challenges. First, such systems involve methodologies from various areas, such as object modeling, optimization, control theory, statistics and computer graphics. Secondly, a control and communication structure is necessary to build complete classes of vision application in terms of the above methodologies. Finally, in order to satisfy the different demands of particular vision applications, one needs a software architecture where they can be built in a complete, extensible, flexible and interactive manner.

These are some reasons for which relatively few generic-purpose vision software systems exist. Most such systems are monolithic software architectures built around specialized linear algebra or numerical optimization modules such as Netlib [13] and LAPACK [1], image processing tools such as SUSAN [18] and Intel's Image Processing Library [8], or basic visualization tools such as OpenGL [24] and Geomview [12]. Few such systems, if any, have a high-level architecture able to integrate the many aspects

To appear ICCV-ICVS 2001

of a computer vision problem (computations, control, visualization, user interaction). Moreover, their design often lacks the simplicity, extensibility and completeness required to build various vision applications out of generic, reusable components. For instance, Intel's Open Source Computer Vision Library [8] offers various functionalities, such as camera calibration, image thresholding, image-based gesture recognition, but does not structure them in a modular, extensible, and customizable way. Target Jr [19], intended object-oriented, employs a one-way communication to its Netlib-based Fortran optimization routines. Since this control is hard-wired in its classes, Target Jr is not usable for applications that require user input or data monitoring during the optimization process. Finally, almost no computer vision software we know allows adding interactive manipulation and visualization of its data structures to the computational code in a simple, yet generic manner. Visual interactivity is important in several respects, such as the setting of correspondences between 3D vision models and 2D images, and for monitoring and control of the time evolution of vision applications.

Summarizing, many vision systems lack a generic control model. Secondly, they provide heterogenous functionalities that cannot be easily adapted to new application contexts.

In order to address these problems, we propose a generic high-level application model that covers a large class of vision approaches, a software architecture of this generic model that combines the flexibility, extensibility, and interactivity requirements, and an efficient and effective implementation of this architecture. The proposed application model is based on the idea that a large range of vision applications share the concept of generic optimal state estimation. Such applications involve:

- a *model* characterized by its state
- a generative *transformation* which predicts discretized model features in the observation space, based on a current state configuration
- an *association* of predicted and extracted features in the observation space to evaluate a configuration cost
- a *control strategy* that updates the model state such that the evaluation cost meets an optimality criterium

This application model is powerful enough to cover many techniques such as deformable models/dynamical systems, optimization based methods and sampling based methods, as well as combinations of them (mixed methods). The application model is flexible and extensible as it does not make any assumptions about the ways the model is represented or discretized, the type and number of extracted features (cues), how the association is done, and the underlying strategy used to estimate and evolve the model state. From the proposed model, we derive an architecture that provides desired extensibility, flexibility and modularity requirements in terms of object-oriented and dataflow mechanisms, and finally, we propose an implementation of the above architecture which allows building complex vision applications with visual parameter monitoring and interactive control.

The rest of the paper describes the proposed vision framework, as follows. Section 2 describes the generic vision application model we propose. Section 3 describes an architecture that combines the model's genericity with the flexibility, extensibility, and modularity requirements, and presents a concrete implementation of the architecture. Section 4 presents several vision applications constructed in the proposed framework.

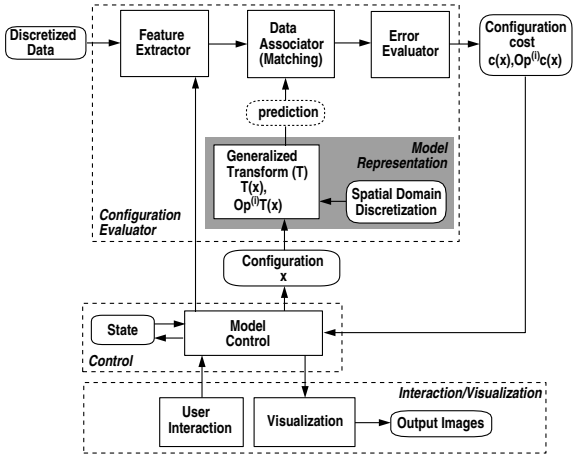


Fig. 1. Vision application model

2 Application Model

The proposed application model exploits the observation that many vision applications are based on the generic state estimation concept. This concept involves optimal estimates of a model's parameters based on a (possibly temporal) sequence of observations. In detail, our application model involves the following elements (see also Fig. 1 which depicts these elements and the data streams between them):

1. a "representational" *discretization* in a spatial domain.
2. a composite (generally non-linear) *generalized transformation* (T), parameterized in terms of the current configuration (typically an instance of the state) which generates predictions in the observation space for points in the discretized domain. This item and the previous one form the *model representation*.
3. a sequence of (possibly temporal) observations or extracted *features*.
4. a way to associate predictions to features to evaluate a *configuration cost* or higher order operators associated with it (their computation typically needs equivalent quantities of T).
5. a *strategy* to evolve the model state based on the evaluation of configuration costs or higher-order operators associated to it, such as its gradient or Hessian, in order to match an optimality criterium (see Fig. 1).
6. several *user interaction* and *visualization* components responsible for the application building and scene exploration, including interactive model-data couplings, 2D and 3D renderings, as well as classical graphics user interfaces (GUI).

The application model elements outlined above are detailed in the following sections.

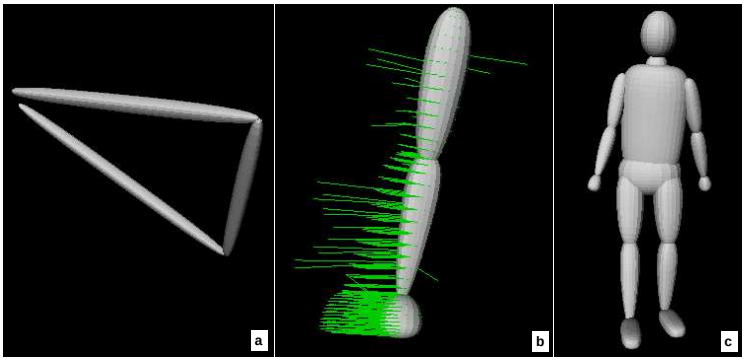


Fig. 2. Various hierarchical models used during tracking

2.1 Configuration Evaluator

The basic task of the configuration evaluator is to compute the cost of a configuration or higher order operators (gradient or Hessian) associated with it. Their evaluation is typically performed in terms of equivalent quantities computed for the model/data generalized transform (T).

Model Representation The model is spatially discretized in a domain Ω. For any point $u \in \Omega$, we can compute a prediction in the observation space $x = T(q, u)$. The Jacobian matrix of the generalized transformation makes the connection between differential quantities in parameter and observation spaces:

$$\dot{x} = \frac{\partial T}{\partial q}\dot{q} = L\dot{q} \tag{1}$$

The process of model estimation involves a data association problem between individual model feature predictions r_i and one or more observations that we shall generically denote $\bar{r}_i$ (with additional subscripts if these are several). We refer to $\Delta r_i(x) = \bar{r}_i - r_i(x)$ as the feature prediction error.

Feature Extraction, Data Association and Error Evaluation These components extract the various types of information used in vision applications. Typical datasets include 2D image data, namely edge/contour/silhouette information, optical fbw information, or 3D range data obtained, for instance, from a multi-camera system.

A subsequent data associator or matching stage establishes correspondences between model predictions and data features. The matching is either explicit such as in the case of an optical fbw module or a 3D model to range features, or implicit, when every predicted model feature has already a computed cost on a potential surface (see below and Fig. 3). In the case of explicit matched features a separate error evaluation stage computes the feature prediction error, based on a certain error distribution. Common error distributions include both well-known non-robust Gaussian ones and robustified ones that model the total (inlier plus outlier) distribution for the observation, e.g.:

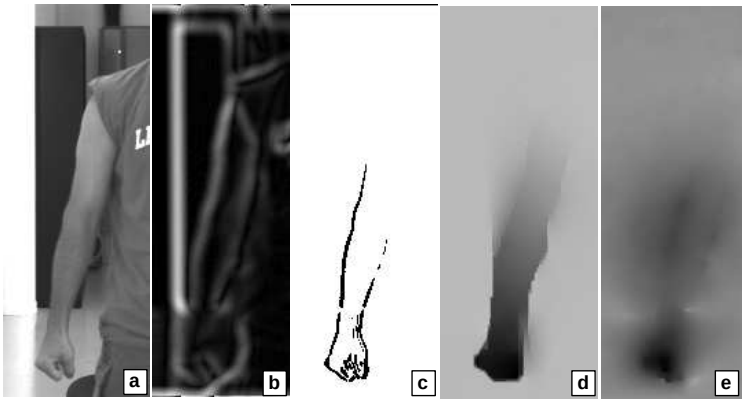


Fig. 3. Image processing operators: original image (a), edge detection (b), motion boundaries (c), robust flow in horizontal (d) and vertical (e) directions

$$\rho_i(s,\sigma) = \nu(1 - e^{-\frac{s}{\sigma^2}}) \tag{2}$$

where ν, σ control the outlier threshold and robust influence, respectively, s is the squared feature prediction error, $s = \Delta r_i W_i \Delta r_i^{\top}$, and W_i is a symmetric positive definite weighting matrix associated with feature i.

For implicit feature matching, let us consider the case of edges, where a potential surface can be build such that it has gaps in the places in the images corresponding to edge features or steep changes in intensity. The image potential is computed for every image frame from the image intensity $I(x,y)$ as follows:

$$\prod(x,y) = -\beta \left\| \nabla (G_\sigma * I)(x,y) \right\| \tag{3}$$

where σ determines the width of the Gaussian function G_σ, $*$ denotes the convolution operator, and β determines the potential surface steepness. The potential is generating a 2D force field, given by:

$$f_{image}(x,y) = -\nabla \prod(x,y) \tag{4}$$

2.2 State Representation and Control

The state representation and control components constitute the core of the application model. Various state representations correspond to various control strategies. Typical ones include unimodal, Gaussian state representation and multiple-hypothesis (sample-based) representations. Generic control strategies include:

– *continuous* ones (deformable models, continuous optimization) which generally assume unimodal state representation and evaluation of the cost function and its higher order operators (gradient, Hessian, etc.);

– *discrete* ones which only involve cost function evaluations and sampling methods for focusing the search effort;
– *mixed* strategies that involve a combination of the first two.

Regardless of the state representation and control employed, there is a unique interface between these components and the configuration evaluator (Fig. 1).

Dynamical Systems and Deformable Models A deformable model estimates the state of the system by numerical integrating a synthetic dynamical system that encodes the fitting error. The Lagrangian equations governing its evolution can be written as:

$$M\ddot{q} + D\dot{q} + Kq = f_q, f_q = \int L^\top e \tag{5}$$

where $M = \int \delta L^\top L$, $D = \int \gamma L^\top L$, and $K = diag(k_{si})$, with δ and γ being tuning parameters, k_{si} being the stiffness associated with the parameter i, and f_q are generalized "forces", acting on the state parameters and e is a distance error in the observation space (typically an $L2$ norm).

Continuous Optimization Based Methods Optimization based methods perform a non-linear estimation in terms of the generalized transformation gradient and Hessian. Popular optimizers include second order damped Newton trust region methods that choose a descent direction by solving the regularized system [6]:

$$(H + \lambda W)\delta q = -g \tag{6}$$

where W is a symmetric positive-definite matrix and λ is a dynamically chosen weighting factor. For robust error distribution specific gradient and Hessian approximations have to be derived in terms of the robustifiers [16]. For least squares problems, $g = \int L^\top e$ and $H \approx \int L^\top L$.

Sampling Methods Sampling methods, usually known in vision under the generic name of CONDENSATION [3], are discrete methods that propagate the entire parameter distribution in time as a set of hypotheses, or samples, with their associated probability. In each frame, the entire distribution is resampled (i.e. recomputed and reweighted) based on new image observations. As these methods do not traditionally have a continuous component, evaluating the distribution in our scheme requires only the evaluation of a configuration, but no other associated higher-order operators. The computational burden lies in the strategies for sampling the parameter distribution in order to locate typical sets, i.e. areas where most of the probability mass is concentrated. Known strategies include importance, partitioned (Gibbs) or annealing based sampling methods (see [9]). We developed an uniform interface such that various discrete sampling methods can be parameterized by the search strategy.

Mixed Continuous/Discrete Methods Mixed continuous/discrete methods have been proposed recently [16] in an effort to combine generic robustness properties of sample-based techniques with the local informed, accuracy and speed properties of continuous ones. The state is represented effectively as a set of hypotheses. In each frame, each hypothesis is subject to a continuous optimization followed by a hypothesis generation based on the uncertain directions of the continuous estimate. Finally, the current set of hypotheses is pruned and propagated to the next time step.

2.3 Interaction/Visualization

The interaction and visualization modules are responsible for scene exploration and manipulation, parameter monitoring and control, and interactive application building. While further details are provided in Section 3, we point out that the dual, simultaneous type of control, user driven vs. application driven, imposes particular demands on the system design, namely *consistency* and *control* problems: user input has to be consistently integrated and propagated into the application data structures (even during application driven execution), while the structuring of individual application components has to allow external interaction during the execution of their different operations. We address these problems by means of automatically enforcing dataflow consistency and by using object-orientation in order to design components with open, non-encapsulated, control.

3 Architecture

The above sections present a generic model that can accommodate a large class of vision applications. These applications share the conceptual structure described in Fig. 1 in terms of component functionalities and intercommunication. To make this application model viable, one must produce a software implementation of it that complies with the requirements of modularity, simple application construction and extension, and flexible control specification, discussed in the previous sections.

We have achieved the above combination by choosing a specific software architecture for implementing the discussed conceptual model. This architecture is based on the combination of two design principles: object-orientation and dataflow, as follows.

3.1 Object Orientation

Object orientation (OO) provides a sound, complete framework for modeling a system as a set of related software modules, by what is called a *class hierarchy* [7]. Specific mechanisms such as subclassing allow the incremental construction of application functionality by specializing a few basic concepts, as well as an effective reuse of the written code. In our case, we have implemented our vision application model as a C++ class library that specializes a few base classes that correspond to the generic concepts shown in Fig. 1. The fixed characteristics of the model, such as the data interfaces between the generic components in Fig. 1, reside the library's base classes. Subclasses add specific functionality to the basic components, in an orthogonal way. For example, different

control strategies or model parameterizations, error evaluations, or feature detection techniques can be added easily, independently on each other, and without modifying the basic software architecture.

3.2 Dataflow

Object orientation effectively models the static, structural relations between the framework's components. To model the dynamic, control relations, we added the *dataflow* concept to the C++ classes. We structure a vision application as a network of classes that have data inputs, outputs, and an update operation. Application execution is driven by the network structure: once a class's input change, the class reads the new input, updates its output, and triggers the execution of the other classes connected to its output, thus enforcing the *dataflow consistency*. A datafbw architecture allows constructing complex control strategies in a simple, yet flexible way, by connecting together the desired components in the desired network. Keeping the vision computational code inside the components and the control strategy outside them, in the network structure, has two advantages: the vision components are simple to write, extend, and understand, and they are directly reusable in applications having different control scenarios.

3.3 Implementation

Writing a vision application in the above setup implies constructing and updating the desired datafbw network. We have made this process flexible, by integrating our vision C++ library in the VISSION datafbw application environment [20]. In VISSION, the datafbw networks are constructed interactively by assembling iconic representations of C++ classes in a GUI network editor (see Fig. 5 b). Figure 4 a,b show two such networks for the vision applications discussed in the next section. The graphical icons are actual subclasses of the vision components shown in Fig. 1.

Besides providing a simple, intuitive way to construct the application, this solution offers several other advantages as an implementation for our architecture. First, VISSION provides graphics user interfaces (GUIs) automatically for all the classes of a network, thus allowing for parameter changes and monitoring. Figure 5 a shows such an interface for the 3D range data tracking application discussed in the next section. Secondly, once a parameter is changed, the traversal and update of the datafbw network is performed automatically, thus enforcing the *dataflow consistency*. Thirdly, VISSION can dynamically load different C++ class libraries. Consequently, we integrated our vision library with the Open Inventor library which provides several direct manipulation tools and 2D and 3D viewers by which monitoring the time evolution of a vision experiment and setting up model-data correspondences can be done in a simple, interactive way, as described in the next section.

There exist several datafbw application environments similar to VISSION, such as AVS [22] or Khoros [25]. From an end user point of view, these environments offer the same visual programming, datafbw, and user interface facilities as VISSION. However, integrating our vission C++ library in such environments would pose several architectural problems. First, these environments are primarily designed to integrate C or FORTRAN code. Integrating C++ libraries, such as our vision library or the Open Inventor

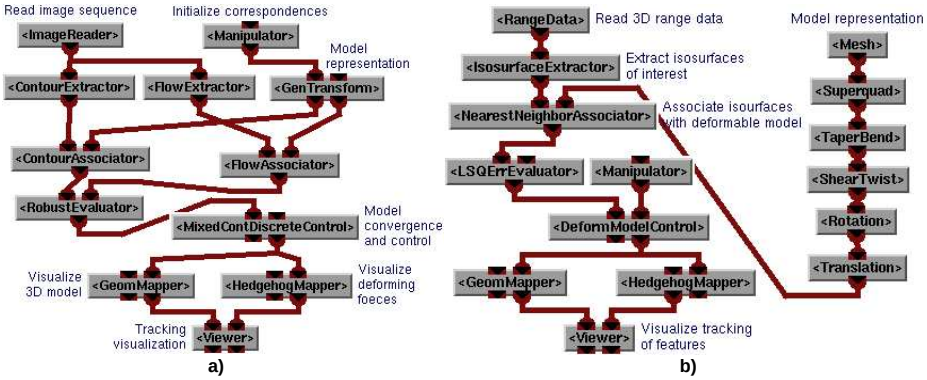


Fig. 4. Tracking networks for human motion (a) and 3D range data (b)

library, is a complex task. Secondly, these environments assume that the code to be integrated is developed based on an application programmer interface (API) provided by the environment. Such an API usually contains data types and functions via which the user code and the environment communicate. In our case, however, we wish to keep our vision library independent on a specific environment API. In this way, our vision library can be deployed in different environments, such as custom turnkey applications, with little or no modification.

4 Applications

In this section we discuss applications involving shape estimation and tracking both on 2D monocular image data and 3D range data. These applications that we have implemented in our framework are typical examples for the state estimation approach based on temporal observations.

In order to model a large class of objects or structures, we have implemented a hierarchical multi-object parametric representation (see Fig. 2), by subclassing the model representational components (Fig. 1). More precisely, any node $u_i \in \Omega$ corresponding to one of the objects discretization domain can be transformed into a 3D point $p_i = p_i(x)$, and subsequently into an image prediction $r_i = r_i(x)$, by means of a composite non-linear transformation:

$$r_i = T_i(x) = P(p_i = A(x_a, x_i, D(x_d, u_i))) \tag{7}$$

where D represents a sequence of parametric deformations which construct the corresponding part in a self-centered reference frame, A represents a chain of transformations that position the corresponding part in the hierarchy, and P represents the perspective projection of the camera, in case we work with 2D data. The entire transformation T (see Subsection 2.1) is assembled in terms of the individual transforms T_i.

4.1 2D Image-Based Temporal Estimation

In this subsection, we illustrate the flexibility of the presented framework by a temporal estimation application based on 2D image sequences. The examples we show involve various types of model representations, various types of feature extractors and data associators as well as different control strategies and interaction modules.

Human Body Tracking We use human body motion sequences that consist of an arm and an entire human body motion (see Figs. 9 and 8). They both involve complex motions with significant self-occlusion, in cluttered backgrounds, including subjects wearing loose clothing. In order to accommodate the complexity of such an application, we have derived a novel mixed-type control method, combining robust (Eqn.2) continuous optimization and sampling along the uncertain directions of the continuous estimate (see Section 2.2).

For model representation, we use the hierarchical scheme mentioned above and superquadric ellipsoids with tapering and bending deformations as basic representational primitives. The estimation involves the articulations degrees of freedom, namely 6 for the arm and 30 for the human model.

We employ both edge and intensity features as cues during estimation. The optical flow has been extracted based on a robust multi-scale computation and the edges have been weighted based on the motion boundary map extracted from that computation (see Fig. 3). Furthermore, we used a multiple association scheme for the assignment of model contour predictions to image observations. Consequently, the Feature Extractor and Data Associator modules have been implemented by chaining together the corresponding Extractors/Associators for contours and optical flow, weighted accordingly (see Fig. 4 a).

A 3D model to 2D image Manipulator interaction module has been employed for specifying correspondences between the model joints and the subject joints in the image, in order to initialize the model in the first frame of the sequence. Subsequent estimation, both for the model initial pose and proportions as well as articulation parameters during the next frames, is fully automated based on image cues. Finally, the model geometry and deforming forces are visualized by the GeomMapper and the HedgehogMapper modules respectively, within an interactive Viewer module.

Incremental Model Acquisition and Tracking The bicycle sequence in Figure 10 uses a mixed model representational structure and a control strategy based on deformable models for the state estimates (Section 2.2). Only contours are used as image cues initially, based on a single association scheme using implicit matching in a computed image potential (Section 2.1). User interaction was needed during model initialization as for the human sequence.

The tracking started with a minimal hierarchical model, a frame consisting of 3 parametric shapes. New model components, with different discretization and parameterizations, are discovered and recovered during tracking based on geometric consistency checks (see [17] for details). Particularly, lines moving consistently with the bicycle frame, but not part of the initial parameterization, are identified, reconstructed,

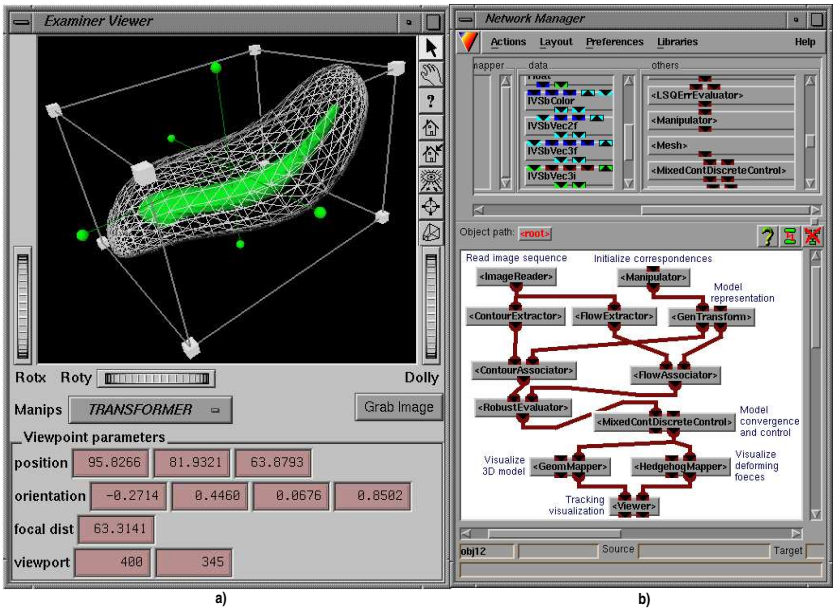


Fig. 5. Data viewer and model manipulator interface (a). A vision application in the visual dataflow network editor (b)

and integrated in the model representation, and further used to improve the tracking robustness as additional cues. Reconstructed model (predicted) lines are displayed in red while image extracted lines (features) are displayed in green. New corresponding cues for line alignment are added to the existing ones (based on model contours). The extended model representation finally consists of heterogenous components: different discretizations (points and lines) and parameterizations (deformable shapes and rigid lines).

This application needs flexibility at all levels of the vision system: we need various representations and discretizations for model components, various feature extractors and data associators corresponding to each component prediction/observation couple, and a way to integrate each component contribution as a cue in the model control. For instance, in a deformbale model control, Jacobians corresponding to each component representational mapping need to be evaluated.

4.2 3D Flow Feature Tracking

In this section, we present a 3D feature tracking and reduced modeling application based on information extracted from fluid dynamics simulations. We employ a model representation based on a superquadric ellipsoid with tapering, bending, shearing and twisting parametric transformations. Next, we employ a control strategy based on deformable models (see Section 2.2). The application's network is shown in Fig. 4 a.

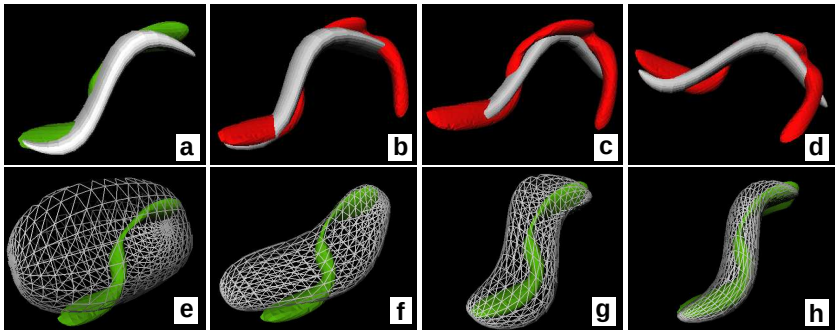


Fig. 6. 3D range data tracking examples

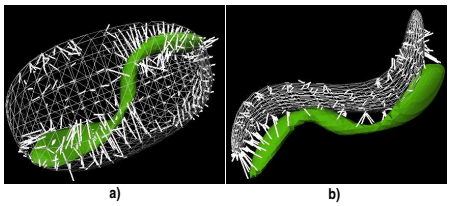


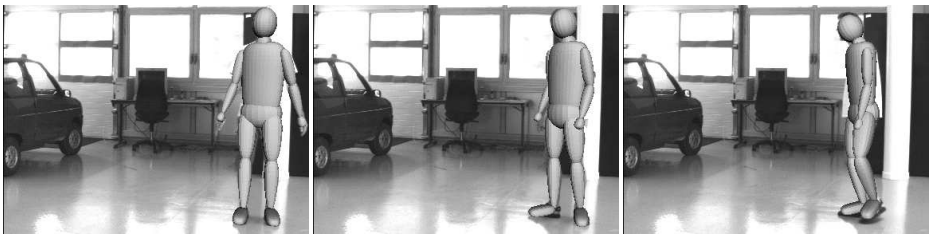
Fig. 7. 3D flow features, model, and deforming forces

In this application, isosurfaces of vorticity magnitude are extracted from a 3D 128^3 time-dependent CFD dataset[1] by a specialized IsosurfaceExtractor module. In the first frame, a desired feature is selected for tracking. The model shape and position are adjusted to approximate the selected feature's geometry by using the GUIs and Manipulator tools mentioned in Sec. 3.3 (Fig. 5 a). Figures 6 e-h show the model convergence (rendered in wireframe) towards a given isosurface geometry, during the initialization phase. The model is first rigidly attracted towards the feature, then starts bending to approximate feature's snake-like form. We used a Data Associator based on nearest neighbor correspondences to match model discretized points to points on the isosurface of interest. Figures 6 a-d show several time instants after initialization when the tracking is automatic. The deforming forces for the initial, respectively converged model state, are shown in Fig. 6.

5 Conclusions

We have presented a layered framework for the modeling, design and implementation of integrated computer vision applications. Our proposal includes: (1) an application model in the form of generic state estimation, involving a control and communication structure able to cover a large class of vision paradigms; (2) an architecture able to support flexibility, extensibility and interactivity demands based on object-oriented and

[1] Dataset courtesy of N. Zabusky, D.Silver and X. Wang [15]



(a) (b) (c)

Fig. 8. Frames during human tracking sequence

(a) (b) (c)

Fig. 9. Frames during hand tracking sequence

datafbw principles; and (3) an effective and efficient implementation of this architecture.

We have subsequently presented how various vision technique classes, like the ones based on dynamical systems/deformable models, continuous optimization, random sampling, as well as combination of them, can be naturally integrated in the proposed framework. Finally, we have presented several complex integrated vision applications which have been easily built within our framework.

References

1. E. ANDERSON, Z. BAI, C. BISCHOF ET AL, *LAPACK User's Guide*, SIAM Philadelphia, 1995.
2. A.BARR *Global and local deformations of solid primitives.* Comp. Graphics, 18:21-30, 1984.
3. M.ISARD AND A.BLAKE *CONDENSATION – conditional density propagation for visual tracking.* IJCV, 29, 1, 5–28, (1998).
4. D.DECARLO AND D.METAXAS *Combining Information using Hard Constraints.* IEEE CVPR 1999.

(a) (b) (c)

Fig. 10. Model and incrementally recovered line feature tracking

5. E.DICKMANS AND V.GRAEFE *Applications of dynamic monocular machine vision.* Machine Vision and Applications, 1:241-261, 1988.
6. R.FLETCHER *Practical Methods of Optimization.* John Wiley, 1987.
7. E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
8. INTEL OPEN SOURCE COMPUTER VISION LIBRARY, available at http://developer.intel.com/software/opensource/cvfl/
9. D.J.C.MACKAY *An Introduction to Monte Carlo Methods.* Learning in Graphical Models, M.Jordan editions, 1999.
10. D.METAXAS AND D.TERZOPOULOS *Shape and nonrigid motion estimation through physics-based synthesis.* IEEE TPAMI, 15(6):580-591, 1993.
11. A.PENTLAND *Automatic extraction of deformable part models.* IJCV, 4:107-126, 1990.
12. M. PHILLIPS, S. LEVY, AND T. MUNZNER *Geomview - An Interactive Geometry Viewer*, Notices A.M.S., October 1993.
13. NETLIB REPOSITORY *available at www.netlib.org*
14. F.SOLINA AND R.BAJCSY *Recovery of Parametric models from range images: the case of superquadrics with local and global deformations.* IEEE TPAMI, 12(2):131-146, 1990.
15. D. SILVER AND X. WANG *Tracking and Visualizing Turbulent 3D Features.* IEEE TVCG, 3(2), June 1997.
16. C.SMINCHISESCU AND B.TRIGGS *A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking*, INRIA Research Report No. 4208, June 2001.
17. C.SMINCHISESCU, D.METAXAS, AND S.DICKINSON *Incremental Model-Based Estimation Using Geometric Consistency Constraints*, INRIA-Research Report No.4209, June 2001.
18. S.M. SMITH AND J.M. BRADY *SUSAN - a new approach to low level image processing*, IJCV, 23(1):45-78, May 1997.
19. TARGETJR SOFTWARE AND DOCUMENTATION, available at http://www.targetjr.org/
20. A.C. TELEA, J.J. VAN WIJK, VISSION*: An Object Oriented Dataflow System for Simulation and Visualization*, Proc. IEEE VisSym'99, Springer, 1999.
21. D.TERZOPLOULOS, A.WITKIN, AND M.KASS *Constraints on deformable models:Recovering 3D shape and non-rigid motion.* Artifi cial Intelligence, 36(1):91-123,1988.
22. C. UPSON, T. FAULHABER, D. KAMINS, D. LAIDLAW, D. SCHLEGEL, J. VROOM, R. GURWITZ, AND A. VAN DAM, *The Application Visualization System: A Computational Environment for Scientific Visualization.*, IEEE Computer Graphics and Applications, July 1989, 30–42.

23. J. WERNECKE, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Addison-Wesley, 1993.
24. M. WOO, J. NEIDER, T. DAVIS, D. SHREINER, *The OpenGL Programming Guide*, 3rd edition, Adison-Wesley, 1999.
25. KHORAL RESEARCH INC., *Khoros Professional Student Edition*, Khoral Research Inc, 1997.