Combined Structure and Motion Extraction from Visual Data Using Evolutionary Active Learning

Krishnanand N. Kaipa
Department of Computer Science
University of Vermont
Burlington VT, USA
kkrishna@uvm.edu

Josh C. Bongard
Department of Computer Science
University of Vermont
Burlington VT, USA
jbongard@uvm.edu

Andrew N. Meltzoff
Institute for Learning and Brain Sciences
University of Washington
Seattle WA, USA
meltzoff@u.washington.edu

ABSTRACT

We present a novel stereo vision modeling framework that generates approximate, yet physically-plausible representations of objects rather than creating accurate models that are computationally expensive to generate. Our approach to the modeling of target scenes is based on carefully selecting a small subset of the total pixels available for visual processing. To achieve this, we use the estimation-exploration algorithm (EEA) to create the visual models: a population of three-dimensional models is optimized against a growing set of training pixels, and periodically a new pixel that causes disagreement among the models is selected from the observed stereo images of the scene and added to the training set. We show here that using only 5 % of the available pixels, the algorithm can generate approximate models of compound objects in a scene. Our algorithm serves the dual goals of extracting the 3D structure and relative motion of objects of interest by modeling the target objects in terms of their physical parameters (e.g., position, orientation, shape, etc.), and tracking how these parameters vary with time. We support our claims with results from simulation as well from a real robot lifting a compound object.

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*modeling and recovery of physical attributes*

General Terms

Algorithms, design, experimentation

1. INTRODUCTION

Modeling of 3D objects from 2D images is still an unsolved computer vision problem. We present here a novel modeling framework that enables a stereo vision system to rapidly create simulations of what it observes. This process is based on carefully selecting a small subset of pixels from the images of the left and the right cameras, and using them to train a population of three-dimensional, physically-realistic models of the target scene. New pixels are requested from the camera images based on disagreements among the models. The extracted pixel is added to the training set and modeling continues. We show here that using only 5 % of the pixels, the algorithm generates physically-plausible models of the target scene. The vision literature abounds with techniques that are capable of localizing and tracking moving objects in a scene. However, our algorithm is driven by the dual goals of extracting the 3D structure and relative motion of objects. This is achieved by modeling the target objects in terms of their physical parameters (e.g., position, orientation, shape, etc.), and tracking how these parameters vary with time. In particular, we infer that the object has undergone a change in one of its parameters if there is a statistically significant change in its value between the initial frame and the final frame of the corresponding video footage; and that two objects are components of a same compound object if there is no statistically significant change in the distance between them over successive frames. We support our claims with the results of structure and motion extraction experiments conducted against simulated scenes and video footage of a real robot lifting a compound object. Moreover, we show that this method results in selection of pixels around the edges of observed objects, therefore leading to automated edge detection. The proposed method caters to specific visual perception requirements in social robotics. A primary goal in this context is to obtain models that can be used to simulate the physical repercussions of a teacher's actions, which is not possible using geometric representations. In order to equip social robots with rapid responses to their perceived environments, approximate yet physically-plausible representations of the observed entities take precedence over creating accurate models that are computationally expensive to generate.

The paper is organized as follows. Related prior vision approaches are outlined in Section 2. Scene modeling is described in Section 3.1. The estimation-exploration algorithm and its application to the vision modeling problem addressed here are presented in Section 3.2. Experimental results illustrating the basic working of the algorithm are reported in Section 4. The structure and motion extraction experiments comprising the main results of the paper are presented in Section 5. A discussion outlining some issues faced by the algorithm and remedies that can be explored in future work is provided in Section 6. Conclusions are discussed in Section 7.

2. PRIOR MODELING APPROACHES

Computer vision is a vast research area with innumerable techniques that have been developed in order to address problems such as object detection, object recognition, mo-

GECCO'09, July 8–12, 2009, Montréal, Québec, Canada.
Copyright 2009 ACM 978-1-60558-325-9/09/07 ...$5.00.

tion tracking, and 3D shape extraction. However, we have limited our literature survey to only those techniques that address the problems similar to the ones that we are interested in this paper. In standard threshold and fit model approaches, models from a large library of known templates are matched with the target image using a distance or a correlation metric over all the pixels. These methods fail when an unknown object (not found in the template library) is encountered in the target scene. However, our algorithm does not require such a model database and uses combinations of basic 3D shapes to model the compound objects in the target scene. Whereas the former approaches use a geometric representation of models, our approach uses a physically-plausible representation of models. Image-based modeling approaches have received considerable attention recently [10], [8], [7], [13], [16], [6], [11], [1]. Unlike 3D scanning techniques where range measurements are used to recover 3D structure, image-based rendering techniques try to generate synthetic views from a given set of original images. Appearance matching is an image-based method and uses raw or filtered pixel brightness values to perform recognition [11]. Our algorithm is an image-based technique. In particular, our method belongs to the class of techniques that generate synthetic models from multiple views of a target scene. The underlying representations of the models may differ from one technique to another. *Shape-from-Silhouettes* [13] is one of the earliest methods that used multiple images for 3D shape extraction. A 2D silhouette is the set of close contours that outline the projection of the object onto the image plane. The silhouette of the image along with the corresponding camera center defines a volume formed by the set of all rays that start from the camera center and pass through a point on the silhouette. The intersection of such volumes defined by silhouettes of several images taken around the object is called a visual hull [12] and yields a reasonable approximation of the real object. Matusik et al. [13] present an image-based method for real-time rendering of a dynamic scene. A visual hull is constructed by using the visible silhouette information from a series of reference images to determine a conservative shell that progressively encloses the actual object. Stereo correspondence is another popular technique that is used for structure extraction problems [14]. Basic to all stereo matching techniques is resolving the correspondence problem that deals with selecting a pixel in one of the images and finding its correspondence (or a corresponding pixel) in the other image. Once the correspondence of a selected pixel is found, the disparity at that pixel's location can be computed, which can then be used to compute the depth information at the pixel's location. Dense disparity maps (disparities at all the pixel locations) are typically used for 3D reconstruction applications.

The algorithm that we discuss in this paper mainly differs from these two standard approaches with respect to the model representation and the amount of pixel information used in the modeling process. Shape-from-Silhouettes uses a volumetric representation of models. In stereo correspondence, the reconstruction comprises a mesh of points in 3D space. However, models in our approach are composed of a combination of basic objects that are represented by physical parameters such as position, orientation, and size. Whereas all the pixels are used for visual processing in the silhouette and stereo based approaches, only a few pixels are carefully selected from the target images and used in the modeling process in our approach. For motion tracking applications, stereo correspondence methods use sparse disparity maps where visually significant pixels like those on corners instead of those on edges of an object have to be computed and monitored. However, our algorithm can simultaneously extract the motion of the object without using the visually significant pixels as the models are directly obtained in terms of 3D shape, size, position, and orientation.

The methods presented in this paper bear some similarity to the work of Bebis *et al.* [1], where the authors investigate the problem of recognizing real 2D or 3D objects from 2D intensity images. They use a model-based approach where the variety of 2D views depicting an object can be expressed as a combination of a small number of 2D views of the object. Objects in a scene are recognized by predicting their appearance through the combination of known views of the objects. This problem can be solved either in the image space or the space of parameters. The authors use genetic algorithms (GAs) to search these spaces. However, the authors assume a fixed set of models. In the work described here, models are created on the fly, and disagreement among them is used to select new pixels for further modeling.

3. METHODS

In this paper, active learning and evolutionary algorithms are combined to jointly infer the structure and motion of objects in a target scene. We assume that there is a left and right camera trained on this scene, and that we can extract arbitrary pixels from images taken by them. Two genetically-encoded models are optimized against these training pixels. We can translate a genome into a model of the scene by translating the genetically-encoded parameters into 3D objects that we introduce into a virtual space. We then place two virtual left and right cameras in this space and extract arbitrary pixels from images taken by them. By comparing the luminosities of pixels between the left physical and virtual cameras, and between the right physical and virtual cameras, we can determine the accuracy of a genetically-encoded scene. We use a hill climber to optimize both models against a growing number of carefully-selected pixels extracted from images taken by the physical cameras.

3.1 Model Synthesis

Our approach to the modeling of a target scene consists of synthesizing models using a small subset of pixels extracted from images (henceforth referred to as target images) taken of the target scene. A pair of left and right images of the target scene are captured from two cameras that are horizontally separated and whose axes are parallel. The models are composed of three basic geometric objects: cuboids, spheres and capsules. A capsule is a cylinder capped with hemispheres. For the sake of simplicity, in this paper, we restrict target scenes to greyscale images with the objects of interest cast against a light background. The objects of each model are parameterized by variables such as the object's shape O_t, position (X, Y, Z), orientation (R_x, R_y, and R_z), size (e.g. length d_1, width d_2, and height d_3 for cuboid), and luminosity ℓ_{obj}. The object phenotype is encoded by combining these parameters into a genotype representation: $\{O_t, X, Y, Z, R_x, R_y, R_z, d_1, d_2, d_3, \ell_{obj}\}$.

These parameters are used to construct a 3D scene in a virtual environment, and two virtual cameras with the same placement as that of the camera pair used to capture the target images are placed in the simulation, and capture images of the virtual scene. With the above framework in place, the modeling problem can now be solved by carrying out stochastic search over the space described by the genome. One possible fitness function L_{rmse} is computed as the root mean squared error of the individual pixel luminosities extracted from the model and those extracted from the target images evaluated over all the pixels:

$$L_{rmse} \quad = \quad \sqrt{\sum_{i=1}^{N_p}(\ell_i^{l_t} - \ell_i^{l_m})^2} + \sqrt{\sum_{i=1}^{N_p}(\ell_i^{r_t} - \ell_i^{r_m})^2} \quad (1)$$

where $\ell_i^{l_t}$ and $\ell_i^{l_m}$ are the luminosities of the i^{th} pixel for the left images taken of the target and model scene, respectively, $\ell_i^{r_t}$ and $\ell_i^{r_m}$ are the luminosities of the i^{th} pixel for the right images taken of the target and model scene, respectively, and N_p is the total number of image pixels. Luminosity ranges from 0 for black pixels to 255 for white pixels.

The process of requesting luminosities from all of the pixels and using this large set of data in order to compute L_{rmse} in (1) would demand a high computational cost, leading to slow reaction times, and hence would not be very useful for real-time vision applications. In order to overcome this problem, we propose to use the estimation-exploration algorithm (EEA) in which several independent models are optimized against a growing set of training pixels, instead of the full pixel array. In short, the training pixels are extracted from the target image based on model disagreement (the exploration phase), and the models are optimized against this growing set of training data using stochastic optimization (the estimation phase). A brief description and implementation of the EEA is given in the next section.

3.2 The Estimation-Exploration Algorithm

The EEA was introduced by Bongard and Lipson [3] as a coevolutionary algorithm that automates both model inference and the generation of useful training data. The EEA builds on query by committee [15], which demonstrated that the optimal method for choosing new training data is through model disagreement. Each cycle of the EEA consists of two optimization phases of *exploration* and *estimation*. During the exploration phase, candidate training data are sought using stochastic optimization such that, when supplied to the current model set, the outputs of the models are maximally divergent. This ensures that if this disagreement-inducing training data is sent to the target system, its response will indicate which models have hidden inaccuracies. These inaccurate models will then be automatically replaced during subsequent model optimization in the estimation phase. In contrast to this, in the field of identification for control [9], training data is passively collected from the target system and the model is not used to determine which experiment to perform next. However, the EEA intelligently chooses new training data so that the amount of data extracted from the target system is minimized, yet sufficiently accurate models of it are produced. For more details please refer to [3]. The EEA has been applied to several other learning problems [5], [4], [2].

3.3 Modeling of target scenes

The EEA was adapted for the computer vision task described above as follows. Initially, a pair of left and right images of the target scene are captured from two cameras that are horizontally separated and whose axes are parallel. The EEA starts by creating two random models A and B. Each model describes a set of virtual objects that reflect objects in the target scene. Each object is described by the parameters introduced in section 3.1. Now, a pair of virtual cameras with the same placement as that of the camera pair used to capture the target images generates the left and right perspectives of model A and model B. During the exploration phase, two sets of training pixels are generated corresponding to the left and right images of both the models. These two training sets are used in conjunction in order to optimize the models in the estimation phase. Now, we describe the exploration phase used to generate the training set for the left images of the models (the same description holds for the right images).

Exploration: In order to add a new pixel to the training set, a candidate pixel set C_p is obtained by selecting $n(= 200 < N_p)$ pixels, where N_p is the total number of pixels contained in the left image of the target scene, as well as the total pixels comprising the left images taken of the virtual scenes. The left image of the target scene, and the left images taken from any given model have the same size and the same dimensions. The set C_p are randomly drawn from a uniform distribution over the integer interval $[1, w] \times [1, h]$, where w and h are image width and image height, respectively. Each pixel from this set is then evaluated to determine how much disagreement it induces in the models in terms of their corresponding luminosities at that pixel's position. For example assume that both models describe a small dark object that appears near the center of the image, but the object's position is slightly different in each image. Then, a pixel chosen near the image center will cause model agreement (both the models will predict that that pixel will be dark in the left image taken from the target scene), as will a pixel taken near a corner of the image (both the models will predict that that pixel will be light in the left image taken of the target scene). However, a pixel near the object edges in the models may cause them to disagree (one may predict that pixel will appear dark, while the other model will predict it will appear light). It is these latter pixels that are useful for uncovering hidden inaccuracies in the models. The pixel among C_p that induces maximum disagreement is defined herein as the most distinguishing pixel. Subjective fitness is the fitness that is accessible to the coevolving individuals and is weakly correlated with the absolute fitness which is only used for benchmarking purposes [3]. We use the following subjective fitness function δ_i to evaluate the disagreement for each pixel $i \in C_p$:

$$\delta_i \quad = \quad |\lfloor(\ell_i^{m_1}/255)\rfloor - \lfloor(\ell_i^{m_2}/255)\rfloor| \qquad (2)$$

From (2), note that the pixel luminosities are collapsed into binary values ('1' for white and '0' for all other grey shades) before evaluating the disagreement. The image is binarized to distinguish foreground pixels from those of the background. Therefore, δ_i is a binary-valued function; if both the models predict that a pixel i belongs to either foreground or background, then the two models either totally agree ($\delta_i = 0$) at pixel i. However, if one model predicts that pixel i belongs to foreground and the other model predicts that it belongs to the background, then the models totally disagree ($\delta_i = 1$) at pixel i. Since pixel luminosities are binarized, the algorithm can also handle image blur so long as there is some contrast between foreground and background. However, there might be a very small offset in the estimation of the parameters from their nominal values. The most distinguishing pixel is located for the left images, and the most distinguishing pixel is located for the right images, the luminosities of those two pixels are extracted from the left and right image of the target scene, and are added to the training set. The estimation phase then commences.

Estimation: We use a hill climber for model optimization. Let $D_p^l(t)$ and $D_p^r(t)$ be the sets of distinguishing pixels at cycle t for left and right images of the models, respectively. The models are optimized against the growing set of training pixels, instead of the full pixel array. Therefore, the fitness

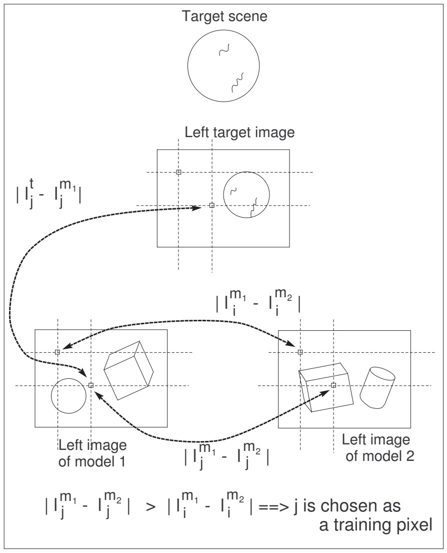

Target scene
Left target image
$| l_j^t - l_j^{m_1} |$
$| l_i^{m_1} - l_i^{m_2} |$
Left image of model 1
$| l_j^{m_1} - l_j^{m_2} |$
Left image of model 2
$| l_j^{m_1} - l_j^{m_2} | > | l_i^{m_1} - l_i^{m_2} |$ ==> j is chosen as a training pixel

Figure 1: An illustration of the EEA implementation only showing the left images.

function in (1) is modified as below:

$$L_{rmse}(t) = \frac{\sqrt{\sum_{i=1}^{N_{dp}^{l}(t)} (\ell_i^{lt} - \ell_i^{lm})^2}}{N_{dp}^{l}(t)} + \frac{\sqrt{\sum_{j=1}^{N_{dp}^{r}(t)} (\ell_j^{rt} - \ell_j^{rm})^2}}{N_{dp}^{r}(t)} \quad (3)$$

where $i \in D_p^l(t)$, $j \in D_p^r(t)$, and $N_{dp}^l(t)$, $N_{dp}^r(t)$ are the number of distinguishing pixels collected up to and including cycle t corresponding to the left and right images, respectively. Figure 1 illustrates the EEA implementation only for the left images. During the exploration phase, since pixel j causes more disagreement in the models than that caused by i, it is chosen as a training pixel. Training pixels that are found in a similar way for the right images are used in conjunction with the former for comparing the models with the target during the estimation phase. At each cycle, the genome of each model is subjected to random mutation and then evaluated. If the child model produced by the mutation has a lower error than the parent model, the parent is replaced with the child; otherwise, the child is discarded. The mutation is achieved by randomly selecting from one to five variables in the genome and perturbing their values using a Gaussian distribution. Since the variables are randomly selected, a variable may undergo mutation more than once during any cycle. For the integer object shape variable, if it is selected for mutation, its value is replaced with a new random value chosen from $[0, 1, 2]$. If an object changes shape due to mutation, the size parameters are altered so that the visual impact on the new image is slight, and there is therefore little discontinuity between successive model images. The above optimization scheme steps through N_{est} $(= 8)$ iterations after which the next exploration phase starts. The algorithm alternates between estimation and exploration until some termination criteria is reached.

4. PRELIMINARY EXPERIMENTS

In initial experiments, a single camera rather than stereo vision was used. The open-source physical simulator Open Dynamics Engine (ODE) was used to generate 3D simulations from the genomes. The virtual camera supplied with ODE was then used to capture a 2D pixel array from the 3D simulation. The error of a model was then calculated as the similarity between the luminosities of its pixels and the luminosities of the pixels at the same positions in the image taken from the scene. Since we use a physics based simulation engine, although the models run slightly slower than they would if we used other simple graphics based simulations, the speed is still sufficient for our purpose. However, the number of training pixels used to optimize the models directly affects the computation cost. Whereas the algorithm takes 40 seconds to model the synthetic black box (Figure 2(a)) when the luminosities of all pixels are requested, it takes only 1 to 2 seconds to model the same object when luminosities from only 200 pixels are requested.

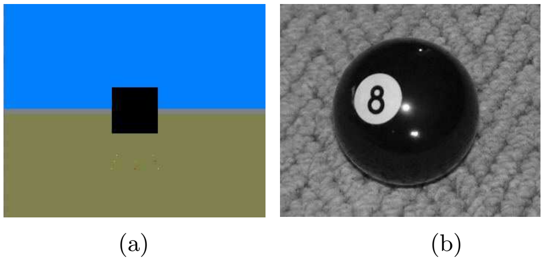


Figure 2: Target images used for the modeling experiments: (a) Emulation of a scene captured by a camera. (b) A real camera image.

The EEA algorithm described above is compared with a control algorithm that differs from the EEA in that rather than selecting new pixels for training based on model disagreement, pixels are chosen at random. Another possible control method could be selecting adjacent pixels to a randomly selected pixel that finds an object. However, in this paper, we restricted our control algorithm to the random pixel selection method. Whereas the subjective error computed using (3) is applied for optimization during the estimation phase, the objective error computed using (1) is monitored to determine the performance of the algorithm. The objective error is not used for model optimization, but rather to compare the performances of the EEA and the control algorithm variant that relies on random pixel selection. Thirty independent trials of both algorithm variants are conducted and the corresponding objective/subjective errors are logged for each trial. Since each trial optimizes two models, there are a total of 60 models that can be analyzed at each optimization step across the trials.

4.1 Modeling of a simulated scene

In the first set of experiments, a simulated scene was used as the target scene (Figure 2(a)). The EEA causes training pixels to be selected from areas of the image that are covered by one model, but not by another. When the luminosity of such pixels is extracted from the target image, this results in a rise in the subjective error of one or both of the models. Therefore, the growing subjective error acts as a driving force that forces the models to rapidly converge toward an accurate description of the target image. The mean objective error plots (averaged over 60 models) for both the control and the EEA variants are shown in Figure 3. The clear separation between the means and SEMs for the control and EEA variants demonstrates that the selection of training pixels based on model disagreement accelerates modeling, compared to random selection of training pixels.

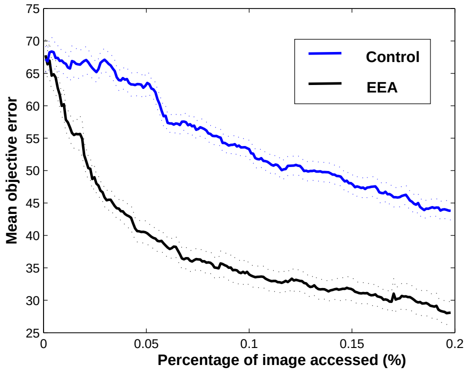


Figure 3: Mean objective error plot. Dots indicate ± standard error of the mean.

4.2 Modeling of a real target scene

In the next set of experiments, we choose a real camera image (Figure 2(b)) to test the modeling performance of the EEA. The target consists of a greyscale image of an 8-ball against a light colored background. The image has a resolution of 352 × 287 pixels, so the images captured from the models were also set to 352 × 287 pixels. A thresholding operation is carried out in order to enhance the contrast in the image. As before, 30 independent trials of both the EEA and random variants were run against this image. The mean objective error plot is shown in Figure 4. The divergence between the control and the EEA in this case is less significant when compared to that of the simulated black box. This is because the 8-ball occupies a larger proportion of space than that of the black box while the number of training pixels used is the same for both the cases.

4.3 Automatic edge detection

The emulated target image overlaid with the training pixels from all the thirty trials for the control and EEA cases are shown in Figures 5(a) and 5(b), respectively. Note that the training pixels are uniformly distributed over the image in the control case; however, in the EEA case, most of the pixels cluster around the edges of the observed object, thus performing edge detection automatically. Similar results for the case of the real image are shown in Figure 6. The clustering around the 8-ball image looks relatively thin when compared to that of the square because the total number of pixels displayed is the same for both the cases (30×200), but the circumference of the 8-ball is larger than that of the perimeter of the square, so the clustering is more diffuse.

The EEA causes the training pixels to be selected from areas within either, but not both of the two models where there is maximum disagreement in their luminosities. As the models converge toward the target, the areas of disagreement become constrained to the edges of the modeled objects. But because the modeled objects converge on an accurate description of the target objects, and thus converge toward each other, distinguishing pixels can only be found near the model objects' perimeters. Therefore, pixels begin to be chosen from around the edge of the target object.

5. STRUCTURE AND MOTION EXTRACTION EXPERIMENTS

Next, we apply our algorithm to jointly model the structure and motion of objects. The algorithm models observed objects in terms of position, orientation, shape, size, and luminance. The structure and motion of a compound object is extracted by tracking how the value of these parameters change during optimization, as pixels from successive frames of a video stream are extracted and added to the training set. In particular, we make the following inferences:

- the object has undergone a change in one of its parameters if there is a statistically significant change in its value between the initial frame and the final frame of the corresponding video footage.
- two objects are components of the same compound object, in spite of a change in their individual positions, if there is no statistically significant change in the distance between them over successive frames.
- as models are optimized, the parameter values of replaced models are recorded in a buffer. Systematic change in a model parameter therefore indicates change in the inferred scene.

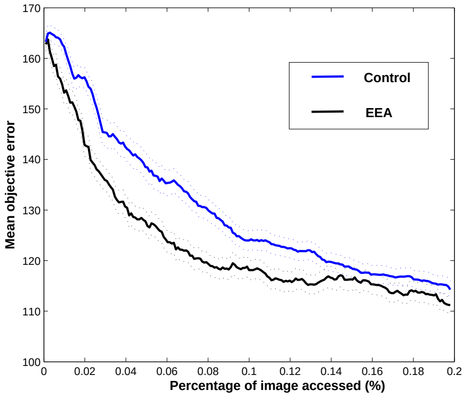


Figure 4: Mean objective error plot (averaged over 60 trials) for the real image

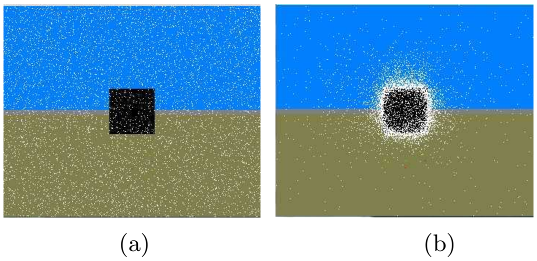


Figure 5: Synthetic target image overlaid with training pixels from all 30 trials: (a) Control. (b) EEA.

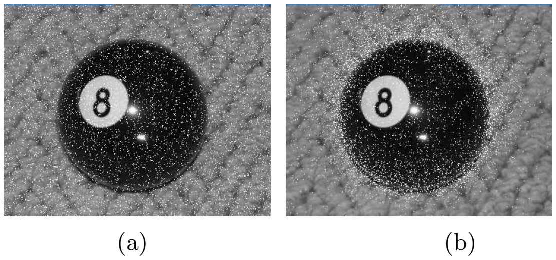


Figure 6: Real target image overlaid with training pixels from all 30 trials: (a) Control. (b) EEA.

5.1 Modeling a changing virtual scene

We consider a synthesized dumbbell that is initially at rest and makes a small vertical displacement. A sequence of 11 images with a resolution of 100 × 80 is extracted from video footage of this event (the initial and final frames are shown in Figure 7) and used as input to the algorithm. The dumbbell starts from rest at the position (0, 0, 2.5) in the first image, vertically displaces by 0.1 units from one image to the next in the following sequence of 10 images, and comes to a halt at position (0, 0, 3.5) in the final image. The left and the right virtual cameras are located at (8, -1.8, 2.5) and (8, 1.8, 2.5), respectively, with a horizontal camera separation of 3.6 units. Each cycle of the algorithm consists of modeling a single image in the image sequence. The first cycle of the algorithm consists of modeling the initial image over a growing training set of 400 pixels (5 % of the total number of pixels). The model parameters that are evolved at the end of the first cycle are used to initialize the parameter values in the second cycle and so on. The training pixels collected during the first cycle constitute the training buffer in the next cycle where each newly found training pixel replaces the oldest one in the previous training set. The first five cycles use the same image where the object is at rest in order to obtain a good initial model. The next 10 cycles model the images in which the object moves upward. A set of thirty independent trials are conducted and the evolved parameter values at the end of each cycle are logged for each trial. As earlier, we use two models in the EEA algorithm that compete with each other to approximate the target scene. Therefore, we have a total of 60 models that can be averaged to analyze the statistical performance of the algorithm. We observe that the left, the middle, and the right components of the target dumbbell are modeled by three objects O^s_L, O^s_M, and O^s_R, respectively. Figures 8(a - f) show the mean trajectories of the parameters of O^s_L, O^s_M, and O^s_R as a function of the number of modeling cycles. Note that there is a statistically significant change in the parameter Y (height) for all three objects. However, the mean trajectories of all the remaining parameters of the three objects (except W in the case of O^s_M) do not change significantly. From the trajectory of the width parameter W in the case of O^s_M (Figure 8(d)), it appears that the target object changes its width, which is a false interpretation. This may be attributed to the insufficient stereo information that is available given the long width of the rod, causing a lag in its approximation (W approaches its correct value at cycle 11. Refer to Figure 8(d)). However, this problem could be rectified by modifying the configuration of the camera pair placement appropriately. Therefore, it is clear that there is a consistent change only in the case of the parameter Y of all the three components that form the dumbbell. Hence, the algorithm can infer from the video footage that the dumbbell was displaced vertically.

The genome works with three objects, which fits the dumbbell well. When only two objects are used the algorithm partially models the dumbbell. When four objects are used it can still model the dumbbell. However, as the object complexity is not known a priori, the number of objects cannot be fixed and should be left as a free parameter. This implies that genes have to be added or deleted from the genome whenever the number of objects is increased or decreased, respectively.

5.2 Modeling a changing physical scene

Next, we consider a real-world experiment in which a single real camera is used to capture video footage of a robotic arm performing a simple task of lifting a red colored dumbbell upward. The robot used for this purpose is shown in Figure 10. The 11 frames extracted at 0.2 sec. intervals from a 2 sec. video clip are shown in Figure 11. The same procedure used to model the virtual scene is applied to the sequence of real images. However, before each image is input to the algorithm, it is processed by an appropriate filter that collapses the r, g, b values of red colored pixels to 0 and those of any pixel of a different color to 255. This is performed in order to allow the algorithm to identify the pixels of only a particular color as belonging to the foreground (or objects of interest) and the rest of the pixels as belonging to the background. Since a single camera is used in these experiments, the virtual cameras that are used to generate the stereo perspectives of the models are placed at the same location. Thirty independent trials are conducted and the evolved parameter values at the end of each cycle are logged for each trial. Similar to the case of the virtual scene, we observe that the left, the middle, and the right components of the real dumbbell are modeled by three objects O^r_L, O^r_M, and O^r_R, respectively. Figures 9(a - f) show the mean trajectories of the parameters for O^r_L, O^r_M, and O^r_R as a function of the number of modeling cycles. Since the stereo information is not available in this experiment, the depth parameter Z is incorrectly estimated. Therefore, it is not considered for analysis here. However, this limitation could be overcome by using a stereo rig to capture the video footage and using a non-zero separation between the virtual cameras. Again, note that there is a statistically significant change in the parameter Y of all the three objects. However, the mean trajectories of all the remaining parameters of the three objects (except X) do not show consistent change. Hence, it can be inferred that the real dumbbell has moved upward. The images of models evolved by the algorithm at the end of each cycle are shown along side the target images in Figure 11 from which it is clear that the models are able to track the vertical motion of the dumbbell. Since a single camera was used in this experiment, it is not possible to match the parameter values estimated by the algorithm with those of the real object. Experiments using a stereo camera pair are currently in progress which would allow such comparisons.

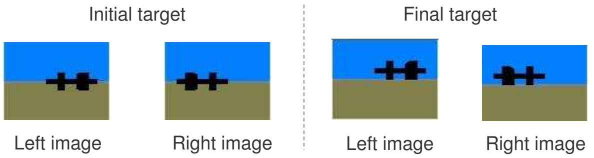
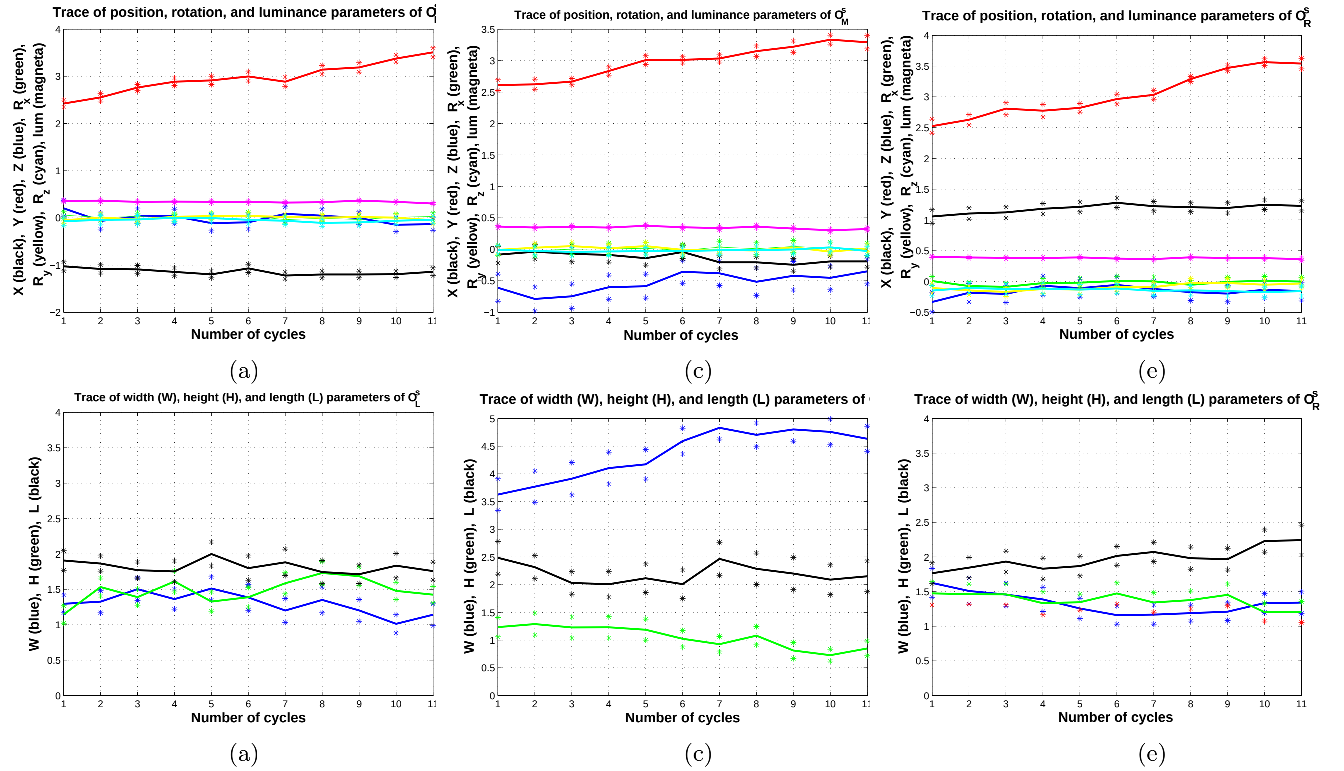


Figure 7: The initial and final frames of the synthetic moving target.

6. DISCUSSION

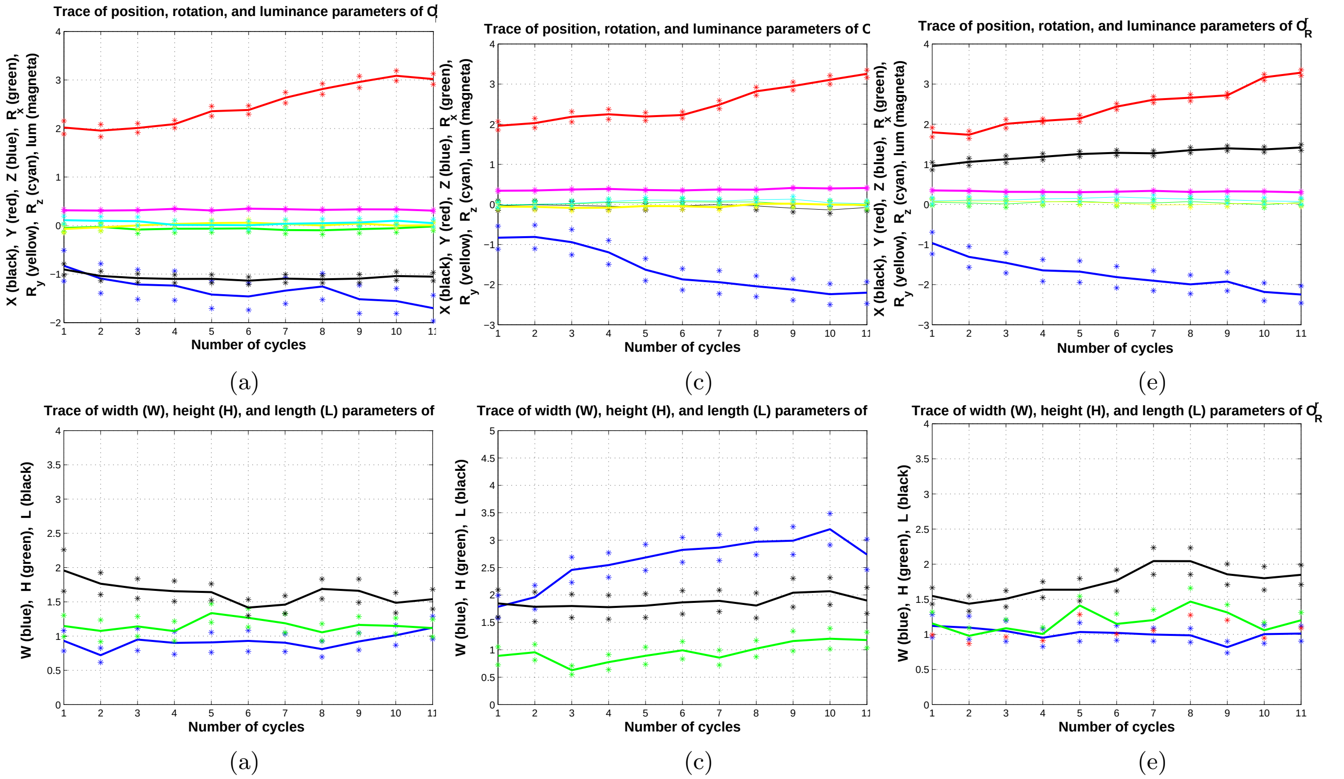
Our algorithm models simple objects in near real time (1 to 2 seconds on a 2.66 GHz, Intel Core 2 CPU based computer). However, it currently lacks the same capability in the case of compound objects. This is because the algorithm uses a combination of simple objects to model a compound object. Now, as the budget of objects increases, the dimensionality of the search space increases. As a simple hill climber is used for the optimization of the models, a few seconds is spent in the search process. However, more efficient search schemes like evolution strategies may be used to quickly optimize the models, thereby allowing the algorithm to run faster. The filtering process of the real images also re-



(a) (c) (e)

(a) (c) (e)

Figure 8: Mean trace of the various parameters of the left, the middle, and the right components that model the synthetic dumbbell (averaged over 60 trials; the '$*$' marks represent the standard error of the mean values): (a), (b) Left object O^s_L. (c), (d) Middle object O^s_M. (e), (f) Right object O^s_R.



(a) (c) (e)

(a) (c) (e)

Figure 9: Mean trace of the various parameters of the left, the middle, and the right components that model the real dumbbell: (a), (b) Left object O^r_L. (c), (d) Middle object O^r_M. (e), (f) Right object O^r_R.

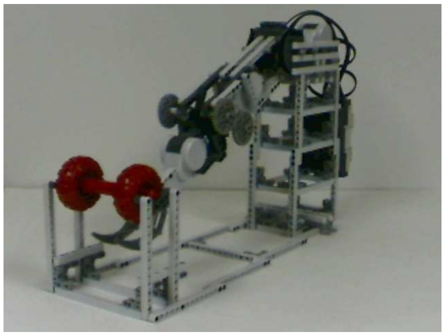


Figure 10: The real robot used in the experiments.

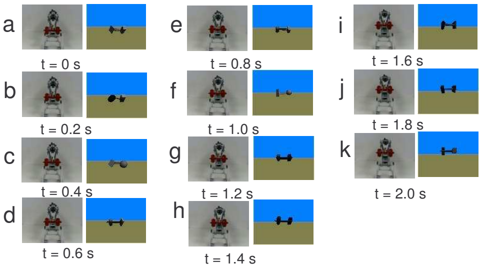


Figure 11: A sequence of 11 frames extracted from the video of a real robot lifting a dumbbell and the corresponding models generated by the algorithm.

duces the modeling speed. However, the algorithm requires the comparison between model images and the target image only at a few pixels. Therefore, it is sufficient to acquire and filter only the r, g, b values of the few pixels that are intelligently selected for training the models. This can further enhance modeling speed. Currently, the mean behavior of the parameter trajectories over several trials are analyzed to deduce the structure and motion of objects. However, the algorithm should have the capability to make similar inferences from a single trial. This can be achieved, for instance, by using thirty models (instead of just two as was used in this paper) that compete with each other and observing mean behavior over these models. The budget of objects is limited to three in the experiments used to model the compound objects. However, this restriction may be relaxed when a more efficient search technique is used, thereby allowing modeling of more complex objects. The fitness metric was chosen as a function of binarized luminosity to compare the models with the target images at the training pixel positions. However, with little modifications, we can extend this basic framework to algorithms that look for other parameters like absolute luminosity and texture templates. For example, currently only shapes like cuboids, spheres, and capsules are used as the basic elements. However, compound objects (e.g., dumbbell) that are successfully modeled can be added to this basis set of objects so that the algorithm could look for such templates in more complex scenes.

7. CONCLUSIONS

We have here presented a novel stereo vision modeling framework that will be useful for applications in which rapid generation of approximate yet physically-plausible models of target scenes is required. We have shown here that even though a very small subset of pixels from the target scene are queried, it is sufficient to rapidly generate approximate models of observed objects. We have shown that the technique automatically performs edge detection of the observed objects. Finally, we have shown how our method serves the dual purpose of extracting structure and motion of objects of interest in simple scenes using synthetic simulations as well as real world experiments.

8. REFERENCES

[1] G. Bebis, S. Louis, Y. Varol, and A. Yfantis. Genetic object recognition using combinations of views. *IEEE Trans. on Evol. Comp.*, 6(2):132–146, 2002.

[2] J. C. Bongard and H. Lipson. Active coevolutionary learning of deterministic finite automata. *Journal of Machine Learning Research*, 6, 2005.

[3] J. C. Bongard and H. Lipson. Nonlinear system identification using coevolution of models and tests. *IEEE Trans. on Evol. Comp.*, 9(4):361–384, 2005.

[4] J. C. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *PNAS*, 104(24):9943–9948, 2007.

[5] J. C. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314, 2006.

[6] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[7] C. H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96, 2004.

[8] W. T. Freeman, D. B. Anderson, P. Beardsley, C. N. Dodge, M. R. C. D. Weissman, W. S. Yerazunis, H. Kage, I. Kyuma, Y. Miyake, and K. Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, 18(3):42–53, 1998.

[9] M. Gevers. A decade of progress in iterative control design: From theory to practice. *Journal of Process Control*, 12(4):519–531, 2002.

[10] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. In *Proc. of Ninth IEEE International Conference on Computer Vision*, pages 641–647, 2003.

[11] E. Hadjidemetriou and S. K. Nayar. Appearance matching with partial data. In *DARPA Image Understanding Workshop (IUW)*, pages 1071–1078, 1998.

[12] A. Laurentini. The visual hull concept for silhouettes-based image understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[13] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Proc. of SIGGRAPH*, pages 369–374, 2000.

[14] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Intl. Journal of Comp. Vis.*, 47(1), 2002.

[15] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. of the Fifth Workshop on Computational Learning Theory*, pages 287–294, 1992.

[16] R. Yang, G. Welch, and G. Bishop. Real-time consensus-based scene reconstruction using commodity graphics hardware. *Computer Graphics Forum*, 22(2):207–216, 2003.