Bootstrap Initialization of Nonparametric Texture Models for Tracking

Kentaro Toyama[1] and Ying Wu[2]

[1] Microsoft Research, Redmond, WA 98052, USA
kentoy@microsoft.com
[2] University of Illinois (UIUC), Urbana, IL 61801, USA
yingwu@ifp.uiuc.edu

Abstract. In *bootstrap initialization* for tracking, we exploit a weak prior model used to track a target to learn a stronger model, without manual intervention. We define a general formulation of this problem and present a simple taxonomy of such tasks.

The formulation is instantiated with algorithms for bootstrap initialization in two domains: In one, the goal is tracking the position of a face at a desktop; we learn color models of faces, using weak knowledge about the shape and movement of faces in video. In the other task, we seek coarse estimates of head orientation; we learn a person-specific ellipsoidal texture model for heads, given a generic model. For both tasks, we use nonparametric models of surface texture.

Experimental results verify that bootstrap initialization is feasible in both domains. We find that (1) independence assumptions in the learning process can be violated to a significant degree, if enough data is taken; (2) there are both domain-independent and domain-specific means to mitigate learning bias; and (3) repeated bootstrapping does not necessarily result in increasingly better models.

1 Introduction

Often, we know something about the target of a tracking task in advance, but specific details about the target will be unknown. For example, in desktop interfaces, we are likely to be interested in the moving ellipsoid that appears in the image, but we may not know the user's skin color, 3D shape, or the particular geometry of the facial features. If we could learn this additional information during tracking, we could use it to track the same objects more accurately, more efficiently, or more robustly.

This problem, which we call *bootstrap initialization for tracking* arises whenever the target object is not completely known *a priori*. In Section 2, we propose an abstract formulation of bootstrap initialization. Section 3 offers a taxonomy of bootstrap initialization problems and reviews previous work. Sections 4 and 5 discuss experiences with two domains in which the learned models are nonparametric models of target texture. We take the Bayesian perspective that models represent a tracking system's "belief" about the target. Experiments show how different data sampling techniques affect learning rate and quality (Section 5).

2 Bootstrap Initialization Formulation

Given a prior belief about the target (inherited from the system designer), the goal of bootstrap initialization is to learn a more useful model of the target, using only information gained during actual tracking. We now introduce an abstract framework for this concept. Reference to Figure 1 will clarify the notation.

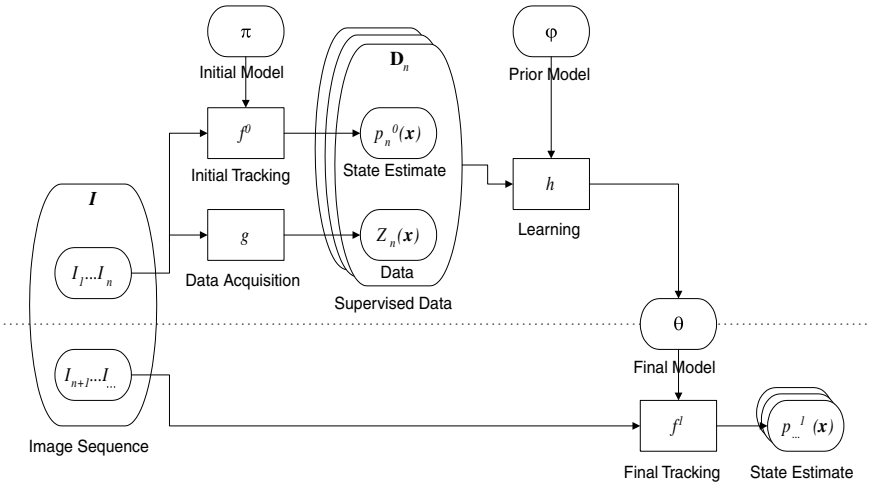


Fig. 1. Abstract formulation of bootstrap initialization.

The goal of tracking is the recovery of the configuration of the target, $\mathbf{x}_m \in \mathcal{X}$, at time t_m, given a model of the target, a sequence of images, $\mathcal{I}_m = \{I_1 \ldots I_m\}$ (where I_i is the image taken at time t_i).

Automatic initialization is impossible without some starting belief about the target. So, we assume that there is an *initial model*, $\boldsymbol{\pi}$, that can be used to track targets with some reliability. This means that there is some prespecified, *initial tracking function*, $f^0_{\boldsymbol{\pi}}(\mathcal{I}_m)$, which, given a sequence of images and an initial model, returns a distribution, $p^0_m(\mathbf{x})$, for the probable configurations that the target assumes at time t_m. (We will assume that tracking functions return distribution functions and that should it be necessary, some way to choose a single estimate $\hat{\mathbf{x}}$, given $p^0_m(\mathbf{x})$, is also specified.) The initial model represents a belief about the target that the system designer passes on to the algorithm, *e.g.*, the target is blue, it is round, it matches template A, etc. We leave the form of the initial model intentionally vague; what matters is the existence of an $f^0_{\boldsymbol{\pi}}$ that incorporates the information contained in $\boldsymbol{\pi}$. The initial model need not be particularly reliable, as long as it correlates to some degree with characteristics of the target that distinguish it from non-target objects.

For bootstrap initialization, we also posit a data *acquisition function*, $g(\mathcal{I}_m)$, which takes an image sequence and returns observations, $\mathbf{Z}_m(\mathbf{x})$, defined over the

state space. Note that $\mathbf{Z}$ maps target states to observations. The observations are of the type that will be relevant to the *final model*, the model to be acquired.

The final model is represented by $\boldsymbol{\theta}$. $\boldsymbol{\theta}$ will contain all of the information relevant to the *final tracking function*. Knowing $\boldsymbol{\theta}$ allows the final tracking function, $f^1_{\boldsymbol{\theta}}(\mathcal{I}_m)$, to output $p^1_m(\mathbf{x})$. We assume that the final model itself has some prior, denoted $\boldsymbol{\phi}$, which contains the same *type* of information contained in $\boldsymbol{\theta}$. That is, $f^1_{\boldsymbol{\phi}}(\mathcal{I}_m) - \boldsymbol{\phi}$ not $\boldsymbol{\theta}$ – makes sense, although it may not provide good tracking output. In general, we will be concerned with determining a final model for time $t > t_n$, for $n \geq 1$.

Next, let a pair of corresponding observations and tracking output be denoted $\mathbf{D}_i = (p^0_i(\mathbf{x}), \mathbf{Z}_i(\mathbf{x}))$. By expressing its degree of belief in intervals of $\mathcal{X}$, $p^0_i(\mathbf{x})$ effectively provides supervision that points out which values in the range of $\mathbf{Z}_i(\mathbf{x})$ are more or less reliable as training data. Thus, $\mathbf{D}_i$ represents a single instance of supervised data annotated by the initial tracking function.

Let $\mathcal{D}_n = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$. These are fed to a learning function, $h(\mathcal{D}_n, \boldsymbol{\phi})$, which takes the available data and learns the final model.

This framework has been structured broadly enough to encompass existing forms of bootstrap initialization. As an example, consider a recursive estimation scheme such as the Kalman filter. In our framework, the Kalman filter corresponds to the following: $n = 1$, $f^1 = f^0$, the models $\boldsymbol{\pi}, \boldsymbol{\phi}$, and $\boldsymbol{\theta}$ contain state and covariance parameters, $\boldsymbol{\phi} = \boldsymbol{\pi}$, and h updates $\boldsymbol{\phi}$ to $\boldsymbol{\theta}$. Because $f^1 = f^0$, and $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ share similar structure, the Kalman filter can (and does) iterate bootstrap initialization by setting $\boldsymbol{\pi}_i = \boldsymbol{\theta}_{i-1}$.

The preference for the function $f^1_{\boldsymbol{\theta}}$ over $f^0_{\boldsymbol{\pi}}$ supplies the entire *raison d'être* for bootstrap initialization; we expect at least one of the following to be true:

- $f^1_{\boldsymbol{\theta}}$ is more accurate than $f^0_{\boldsymbol{\pi}}$, *e.g.*, for ground truth target configuration $\mathbf{x}^*$,

$$\| \arg\max_{\mathbf{x}} p^1(\mathbf{x}) - \mathbf{x}^* \| < \| \arg\max_{\mathbf{x}} p^0(\mathbf{x}) - \mathbf{x}^* \|,$$

- $f^1_{\boldsymbol{\theta}}$ can be computed more efficiently than $f^0_{\boldsymbol{\pi}}$,
- $f^1_{\boldsymbol{\theta}}$ is more robust than $f^1_{\boldsymbol{\theta}}$, or
- $f^1_{\boldsymbol{\theta}}$ is otherwise preferable to $f^0_{\boldsymbol{\pi}}$.

We anticipate that in most applications, the forms of $\mathbf{x}$, $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, f^0, and f^1 will be well understood. Thus, the interesting problems in bootstrap initialization are in the design of the acquisition function, g, and the learning function, h.

3 Taxonomy and Related Work

To help restrict our attention to a small subset of bootstrap problems we will consider a taxonomy for the common types of initialization problems in tracking. We propose a classification of initialization problems based on the following axes:

- Does the final model learn information about **the object geometry or the surface texture** of the target?

- Does the final model involve information about **static or dynamic** properties of the target?
- Is the final model **parametric or nonparametric**?
- Does the learning take place **adaptively or in batch mode**?
- Is the information contained in the initial and final **models same or different**?

Very little prior work addresses automatic initialization for the sake of tracking, but there is a body of related work that fits the bootstrap initialization paradigm.

Classic structure from motion algorithms can be thought to learn the static (rigid) geometry of an object, often outside of any explicit parametrization. Most such work is cast as a batch problem, and rarely as an initialization step for tracking, but there are exceptions. In some facial pose tracking work, for example, 3D points on the face are adaptively estimated (learned) using Extended Kalman Filters [1, 9]. Care must be used to structure the EKF correctly [4], but doing so ensures that as the geometry is better learned, tracking improves, as well.

Other research focuses on learning textural qualities. Again, in the domain of facial imagery, there is work in which skin color is modeled as a parametrized mixture of n Gaussians in some color space [11, 12]. Work here has covered both batch [11] and adaptive [12] learning with much success. The preferred learning algorithm for parameter learning in these instances is expectation-maximization.

Although color distributions are a gross quality of object texture, learning of localized texture is also of interest. Work here focuses on intricate facial geometry and texture, using an array of algorithms to recover fine detail [7].

Finally, there is research in learning of dynamic geometry – the changing configuration (pose or articulation) of a target. The most elementary type occurs with the many variations of the Kalman filter, which "learns" a target's geometric state [3]. In these cases, the value of the learned model is fleeting, since few targets ever maintain fixed dynamics. More interesting learning focuses on models of motion. Existing research includes learning of multi-state motion models of targets which exhibit a few discrete patterns of motion [8, 13].

Our work focuses on bootstrap initialization of nonparametric models for the static texture of faces. In contrast with previous work, we explicitly consider issues of automatic learning during tracking without manual intervention.

4 Nonparametric Texture Models

In our first example, we consider learning a skin-color distribution model of a subject's face, using a contour tracking algorithm to offer samples of target skin color. We use this model for color-based tracking of facial position.

In the second, we learn a person-specific 3D texture model of a subject's head, using a generic model to provide supervisory input. The 3D model is used to estimate approximate head orientation, given head location in an image.

The models we use will be nonparametric in the sense adopted by common statistical parlance. For example, the skin-color distributions are modeled by histograms. Strictly speaking, histograms have a finite number of parameters equal to the number of bins, but they can also be considered discrete approximations to elements of a nonparametric function space. Likewise, the 3D texture models we use are discrete approximations to a continuous surface model of texture.

4.1 Color PDF

Tracking of faces using color information is popular both for its speed and simplicity. Previous techniques require manual initialization or parameter tuning in order to achieve optimal performance [11, 17]. At best, a manually initialized model adapts over time [12]. Below, we consider automatic initialization of a color model that bootstraps from prior knowledge about object shape and movement.

Framework Instantiation We will assume that the goal is estimation of $\mathbf{x} = (x, y, s)$, the position and scale of an approximately upright face at a desktop computer.

The initial model, $\boldsymbol{\pi}$, encapsulates *a priori* knowledge of user's heads at a PC. In particular, they are likely to project edges shaped like an ellipse at a limited range of scales, and they are likely to exhibit motion from time to time. Given this knowledge and an incoming image sequence, $\mathcal{I}_m$, the initial tracking function, f^0, performs adjacent frame differencing (with decay [5]) on frames I_m and I_{m-1} to detect moving edges and follows this with simple contour tracking [3] to track the most salient ellipse.

The acquisition function, $g(\mathcal{I}_m)$, returns the following observation function: $\mathbf{Z}_m(\mathbf{x}) = I_m(\mathbf{x})$ – the mapping from state to observation simply returns the RGB pixel value of the pixel at the center of the tracked ellipse in the current image (other schemes such as averaging among a set of pixels are also possible and may reduce noise).

Finally, we consider the form of the prior, $\boldsymbol{\phi}$, and posterior, $\boldsymbol{\theta}$, of the final model. Both are represented by normalized histograms, which serve as approximations to the pdf of skin-color. The histogram itself will be represented by a Dirichlet distribution. The reasons for this choice will be explained in the next section. Observed pixel values will be represented by the random variable $\mathbf{U} \in \mathcal{U}$.

Given a likelihood function for skin color, it is a simple matter to define a final tracking function that tracks a single face in an image by computing spatial moments [11, 12].

Bootstrap Initialization Algorithm We now describe our bootstrap initialization algorithm for learning color pdfs of skin, assuming we have a body of data, $\mathcal{D}$, from some frames of tracking.

First, we can cast the goal of bootstrap initialization in this case to be $p(\mathbf{U}|\mathcal{D}_n, \boldsymbol{\phi})$ (recall that $\mathcal{D}_n$ the body of supervised data acquired between time

t_1 and t_n). We can determine $p(\mathbf{U}|\mathcal{D}_n, \boldsymbol{\phi})$ if we know the final model, $p(\boldsymbol{\theta}|\mathcal{D}_n, \boldsymbol{\phi})$. The latter can be computed by Bayes' Rule:

$$p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\phi}) = \frac{p(\boldsymbol{\theta}|\boldsymbol{\phi})p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\phi})}{p(\mathcal{D}|\boldsymbol{\phi})}, \tag{1}$$

where the marginal likelihood, $p(\mathcal{D}|\boldsymbol{\phi})$, is given by

$$p(\mathcal{D}|\boldsymbol{\phi}) = \int p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\phi})p(\boldsymbol{\theta}|\boldsymbol{\phi})d\boldsymbol{\theta}. \tag{2}$$

We can then compute $p(\mathbf{U}|\mathcal{D}, \boldsymbol{\phi})$ by marginalizing over $\boldsymbol{\theta}$,

$$p(\mathbf{U}|\mathcal{D}, \boldsymbol{\phi}) = \int p(\mathbf{U}|\boldsymbol{\theta}, \boldsymbol{\phi})p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\phi})d\boldsymbol{\theta}. \tag{3}$$

In general, neither the posterior probability in Equation 1 nor the integral in Equation 3 are easy to compute, since expressions for $p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\phi})$ and $p(\boldsymbol{\theta}|\boldsymbol{\phi})$ can be arbitrarily complex. Fortunately, there are approximations that simplify the analysis. We discretize $\mathcal{U}$ and assume that our distributions can be captured by *conjugate distributions* [2], which provide tractable, analytical solutions under certain assumptions about the models.

First, we discretize the observed variable, $\mathbf{U}$, such that it can assume any of r possible values, $\mathbf{u}^1, \ldots, \mathbf{u}^r$. Assume that the final model parameters are given by $\boldsymbol{\theta} = \{\theta_1, \ldots \theta_r\}$, with $\theta_k \geq 0$, and $\sum_{k=1}^{r} \theta_k = 1$, and that the likelihood function for $\mathbf{U}$ is given by

$$p(\mathbf{U} = \mathbf{u}^k|\boldsymbol{\theta}, \boldsymbol{\phi}) = \theta_k, \tag{4}$$

for $k = 1, \ldots, r$. Clearly, we can represent any pdf to arbitrary precision by varying r. In our case, we use 32^3 bins, where each of the RGB color channels is quantized into 32 discrete values.

If the data, $\mathcal{D}_n$ can be reduced to n independent observations of $\mathbf{U}$, the process of observation is a multinomial sampling, where a sufficient statistic [2] is the number of occurrences of each θ_k in $\mathcal{D}_n$. As mentioned earlier, we force the algorithm to choose one observation per frame as follows: For each $\mathbf{D}_i$, we choose the pixel at $\mathbf{Z}_{\mathbf{X}'}$, where $\mathbf{x}' = \arg\max_{\mathbf{X}} p^0(\mathbf{x})$. Then, if we let N_k be equal to the total number of occurrences of θ_k in the data ($N = \sum_{k=1}^{r} N_k$), then

$$p(\mathcal{D}_n|\theta, \boldsymbol{\phi}) = \prod_{k=1}^{r} \theta_k^{N_k}. \tag{5}$$

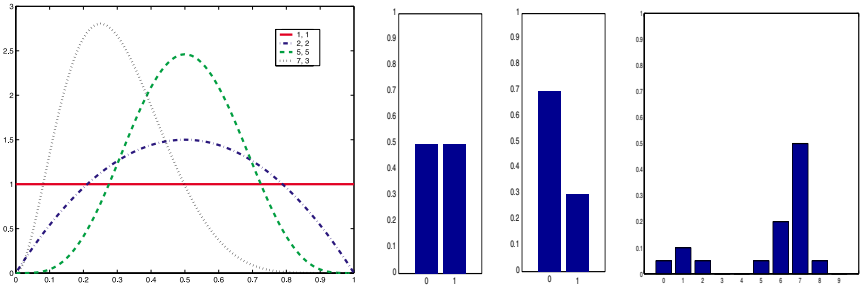
What remains now is to determine the form of the prior, $p(\theta|\boldsymbol{\phi})$. We choose *Dirichlet distributions*, which when used as a prior for this example, have several nice properties. Among them are the fact that (1) a Dirichlet prior ensures a Dirichlet posterior distribution, and (2) there is a simple form for estimating $p(\mathbf{U}|\mathcal{D}, \boldsymbol{\phi})$, which is our eventual goal. The Dirichlet distribution is as follows:

$$p(\boldsymbol{\theta}|\boldsymbol{\phi}) = \mathrm{Dir}(\boldsymbol{\theta}|\alpha_1, \ldots, \alpha_r) \tag{6}$$

$$\equiv \frac{\Gamma(\alpha)}{\prod_{k=1}^{r} \Gamma(\alpha_k)} \prod_{k=1}^{r} \theta_k^{\alpha_k - 1}, \tag{7}$$

where α_k is a "hyperparameter" for the prior, with $\alpha_k > 0$, $\alpha = \sum_{k=1}^{r} \alpha_k$, and $\Gamma(\cdot)$ is the Gamma function [2].

Properly, a Dirichlet distribution is a unimodal distribution on a $(r-1)$-dimensional simplex. When used to represent a distribution of a single variable with r bins, it can be interpreted as a distribution of distributions. In our case, we use it to model the distribution of possible distributions of $\mathbf{U}$, where $p(\mathbf{U} = \mathbf{u}^k | \mathcal{D}, \boldsymbol{\phi})$ is the expected probability of $\mathbf{u}^k$ integrated over $\boldsymbol{\theta}$ (Equation 9). Examples of Dirichlet distributions for $r = 2$ (also known as Beta distributions) are given in Figure 2.



(a) (b) (c) (d)

Fig. 2. Examples of (a) 2-parameter Dirichlet functions (Beta functions) and (b,c) their corresponding 2-bin histograms. A 10-parameter Dirichlet function could represent the histogram in (d).

As distributions of distributions, Dirichlet distributions contain more information than a single pdf alone. For example, while the pdf shown in Figure 2(b) is the expected pdf for any Beta distribution with $\alpha_1 = \alpha_2$, the Beta distribution also gives us information about our confidence in that pdf. Specifically, as $\alpha = \alpha_1 + \alpha_2$ increases, our confidence in the expected pdf increases as well. This is illustrated by the increased peakedness corresponding to increasing α in Figure 2(a).

With this prior, the posterior becomes

$$p(\boldsymbol{\theta} | \mathcal{D}, \boldsymbol{\phi}) = \mathrm{Dir}(\boldsymbol{\theta} | \alpha_1 + N_1, \ldots, \alpha_r + N_r), \tag{8}$$

and the probability distribution for $\mathbf{U}_{n+1}$ is

$$p(\mathbf{U}_{n+1} = \mathbf{u}^k | \mathcal{D}, \boldsymbol{\phi}) = \int \theta_k p(\boldsymbol{\theta} | \mathcal{D}, \boldsymbol{\phi}) d\boldsymbol{\theta} = \frac{\alpha_k + N_k}{\alpha + N}. \tag{9}$$

The surprising consequence of the discretization of $\boldsymbol{\theta}$ and the assumption of the Dirichlet prior is the simple form of Equation 9. Effectively, we need only count the number of samples in the data for each bin of the histogram. Also, note how the expression appeals to our intuition: First, if $\alpha_k = 1$ for all k (a flat, low-information prior, which we use in our implementation), then the probability of

observing $\mathbf{u}^k$ is $(N_k + 1)/(N + r)$, which asymptotically approaches the fraction that $\mathbf{u}^k$ is observed in the data. Second, as the number of observations increases, the effect of the prior diminishes; in the limit, the influence of the prior vanishes. Lastly, we find a particularly intuitive form for expressing our prior beliefs. Our relative sense for how often each of the $\mathbf{u}^k$ occurs is decided by the relative values of α_k, and the confidence with which we believe in our prior is determined by their sum, α.

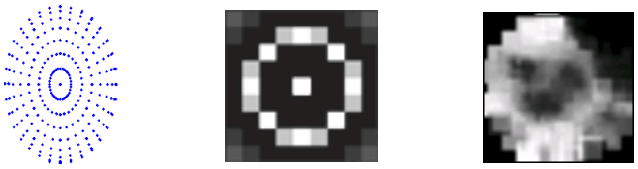
4.2 3D Feature-Mapped Surface

In our second example, we consider the task of estimating a person's approximate head pose, given head location in an image. We distinguish "head pose" from "facial pose" by the range of applicability: facial pose is restricted to images where most of the face is visible.

In contrast to pose-tracking techniques that give precise pose estimates for close-up, well-lit facial images of known subjects [6, 9, 14–17], we consider coarse, but robust, estimation of pose for unknown subjects under a variety of circumstances. By using a generic model to provide initial pose estimates, we can learn a new model that is tailored to that person.

Framework Instantiation The output is $\mathbf{x} = (r_x, r_y, r_z)$, the rotational pose of a person's head. We will assume that other parameters (position and scale, for example) have been recovered by other means [3, 10].

In this case, the initial model, $\boldsymbol{\pi}$, the final model, $\boldsymbol{\theta}$, and the prior for the final model, $\boldsymbol{\phi}$ all take the same form: We model heads as ellipsoids with a set of points on the surface. Each point, indexed by j $(1 \leq j \leq m)$, is represented by its coordinates, $\mathbf{q}_j$ (lying on the ellipsoid surface), and a pdf representing the belief probability, $p_j(\mathbf{z}|\mathbf{x})$ – the belief that given a particular pose, the point j will project observation $\mathbf{z}$. Model points are placed at the intersections of regularly-spaced latitudinal and longitudinal lines, where "north pole" coincides with the front of the head (see Figure 3(a)).



(a) (b) (c)

Fig. 3. (a) Model point distribution; (b) rotation-invariant sum of Gabor wavelets for determining local edge density; (c) coefficients for a learned model viewed exactly frontally, for one kernel.

The domain of the pdfs stored at model points form a feature vector space. An element of this space, $\mathbf{z}$, is a 5-element feature vector consisting of the transform

coefficients when five convolution kernels are applied to a pixel in the image. For a model point j, $\mathbf{z}^j$ is the feature vector for the pixel on which point j would project via scaled orthographic projection (assuming fixed orientation $\mathbf{x}$). The kernels extract information about local "edge density," which tends to be consistent for corresponding points of people's heads across different illumination conditions [19].

The acquisition function, g, returns the observation function $\mathbf{Z}$, where $\mathbf{Z}(\tilde{\mathbf{x}})$ is the concatenation of the feature vectors, $\{\mathbf{z}^j\}$, observed at points in the image which correspond to the orthographically projected points, $\{j\}$, of the model when oriented with pose $\tilde{\mathbf{x}}$ (for $1 \leq j \leq m$). For model points j that occur in the hemisphere not facing the image plane, $\mathbf{z}^j$ is undefined.

Because the underlying models are the same, the tracking functions, f^0 and f^1 are identical. In particular, they simply compute the maximum likelihood pose. Given a cropped image of a head, the image is first rescaled to a canonical size and histogram-equalized. The resulting image is convolved with the five templates described above. Finally, we compute

$$\mathbf{x}^* = \arg\max_{\mathbf{X}} p(\mathbf{x}|\mathbf{Z}) = \arg\max_{\mathbf{X}} p(\mathbf{Z}|\mathbf{x}), \tag{10}$$

using Bayes' Rule, where we ignore the normalization constant and assume a constant, low-information prior over possible head poses. More detail on the pose estimation algorithm is presented elsewhere [19].

Bootstrap Initialization Algorithm Given a set of pose-observation pairs, $\mathcal{D}$, where the pose pdfs are generated using a generic head model, bootstrapping a person-specific model proceeds as follows.

Let $\mathbf{s}_j = \{\mathbf{z}_i^j : \text{ the } j\text{-th element of } \mathbf{Z}_i(\arg\max_{\mathbf{X}} p_i^0(\mathbf{x})), \forall i\}$. That is, $\mathbf{s}_j$ represents the set of all observations that would project to model point j, if, for each pose-observation pair, the pose estimated to have the maximum likelihood is used.

Once all of the data is collected for each model point, j, we estimate the pdf for that point. In our implementation, we approximate the pdf with a single Gaussian whose mean and covariance coincide with that for $\mathbf{s}_j$. This is consistent with a Bayesian approximation of the model pdfs with a low-information prior, ϕ, which contains Gaussian pdfs with zero mean and very large, constant covariances at each model point. The data at each model point is thus assumed to be indepedent of data at other points – this is not the case, but experiments suggest independence serves as a reasonable approximation.

5 Results and Analysis

Both learning algorithms were implemented as described above. Initial results indicate that the bootstrapping algorithms work as expected – in both cases, the final model is learned without manual intervention, when only an initial model was available.

Annotation within:	$0^\circ - 45^\circ$		$45^\circ - 90^\circ$		$90^\circ - 135^\circ$		$135^\circ - 180^\circ$	
Model Type	Rot Y	Rot X	Rot Y	Rot X	Rot Y	Rot X	Rot Y	Rot X
person-specific	10.4	5.7	14.8	6.8	16.9	5.9	28.5	8.7
generic model	19.2	12.0	33.6	16.3	38.0	15.7	47.5	13.2
bootstrap 1	14.2	8.7	23.2	12.2	26.5	9.7	49.7	13.9
bootstrap 2	14.7	8.3	22.1	13.2	25.8	10.4	46.5	13.9
bootstrap 3+	14.5	9.0	25.1	15.2	26.6	11.7	50.5	14.4
preproc+bs1	13.9	8.8	22.9	12.4	26.3	9.7	49.4	14.0

Fig. 4. Average estimation errors.

For the skin-color initialization task, Figure 7 shows an example input image (a) and the corresponding skin-color map (b) using the final model learned over 60 frames during 2 seconds of tracking.

For the head-pose task, Figure 4 displays average error rates over 4 different angular ranges. The values indicate errors averaged over runs on 10 separate sequences of people recorded by different cameras, under different illumination conditions, and at varying distances from the camera. "Ground truth" pose was determined by hand because many of the data sequences were from prerecorded video. For testing purposes, all errors of the algorithm are measured with respect to the annotation.

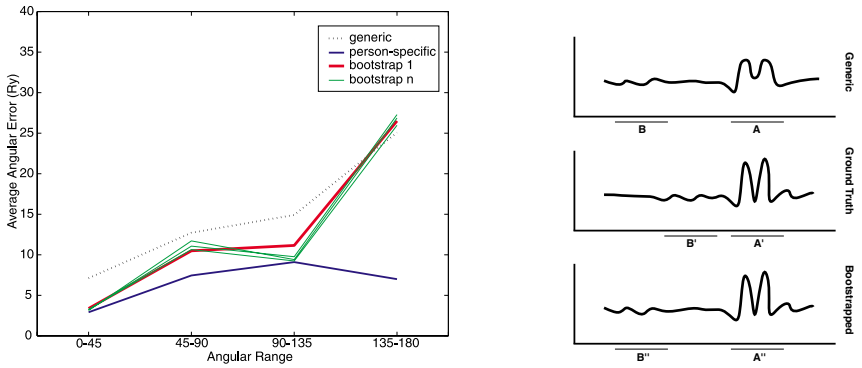
Because texture is more stable on the face than in hair, results were far more accurate when all or part of the face was actually visible. Thus, we report errors averaged over four regions of the pose space. The columns in Figure 4 show the range for which errors were averaged. These numbers indicate the difference in rotation about the y-axis between the annotated face normal and the camera's optical axis. A typical result for a single individual is shown in Figure 5(a).

The results suggest that no unreasonable approximations have been made – bootstrapping works as expected. Nevertheless, because our algorithms are based on independence assumptions which do necessarily hold, we examine the effect that algorithmic choices have on the final outcome.

5.1 Data Dependencies

Both of the learning algorithms presented are based on the assumption that data is acquired independently and without bias from the distribution the models try to capture. How likely and how important is it that these assumptions hold?

In the case of generating learning examples from tracking, the acquired data is unlikely to represent independent samples for several reasons. First, the image sequences involved in tracking generally exhibit little change from frame to frame. We thus anticipate that data from adjacent frames will exhibit considerable temporal dependencies. Second, initial tracking functions are unlikely to track the target with high accuracy or precision (hence the need for bootstrap initialization at all). Thus, a certain amount of noise is expected to corrupt the training data. What is worse is if the initial tracking function (or the initial



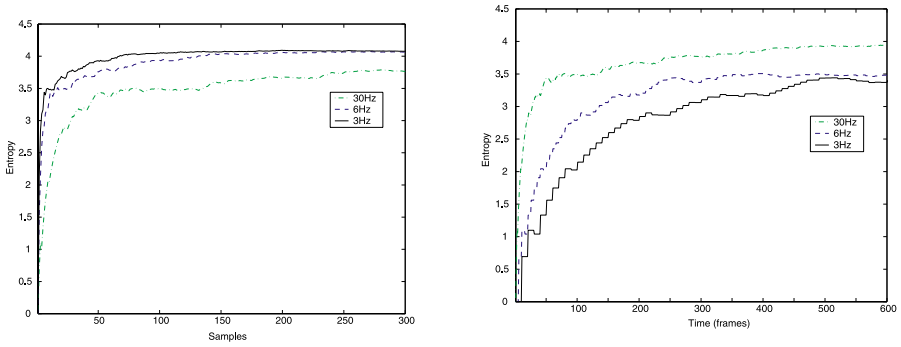
(a) (b)

Fig. 5. (a) Differences in estimation errors for generic model (top line) and bootstrap-iterated models (middle lines) for a typical subject. For comparison, the results for a person-specific model trained using manually annotated data are given as well (bottom line). (b) 1-D schematic representation of models. See Section 5.

model on which it is based) presents a consistent bias in its estimates, which propagates to the data. The acquisition function may also introduce biases in a similar manner.

Reducing Effects of Temporal Coherence: Dependencies due to temporal coherence can be reduced in one of two ways. An intuitive approach is to sample data at random instances in time that are a sufficient interval apart. A "sufficient interval" would be on the order of time such that tracked states appear conditionally independent. For example, in learning the skin-color model, instead of taking samples at 30Hz, we could sample at intervals determined by a Poisson process with intensity adjusted to sample every 0.3 seconds or so. In Figure 6(a), we plot the entropy, $H(X) = -\sum_{\mathbf{X}} p^1(\mathbf{x}) \log p^1(\mathbf{x})$, of the final model for skin color against the total number of data samples, where the lines represent variation in sampling frequency. We expect the entropy to converge as more samples are incorporated. We note that taking samples at lesser frequency increases the learning rate *per datum*, suggesting that temporal coherence can be broken through subsampling.

Alternatively, data can be taken over a long enough period of time that a representative sequence of tracking contexts (*i.e.*, spanning a wide range of target configurations and environmental contexts) is observed. Although the data may not be locally independent, the sufficient statistics of the data set should approximate those of a large, randomly sampled set. This behavior is evident in all of the plots in Figure 6, where the final models appear to converge to similar models, regardless of sampling frequency. The inversion in learning rates between Figure 6(a) and (b) suggests that one can trade off amount of data to process with time required to collect data.



(a) (b)

Fig. 6. Entropy of final model plotted against number/time of data samples. In (a), the x-axis represents number of data samples; in (b), the time required to collect the samples.

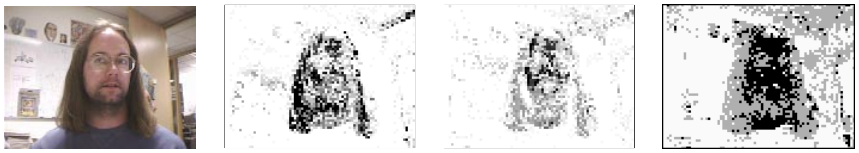
Weighting Data: Some of the problems with data acquisition may be alleviated if the initial tracking function returns a confidence value for its output. If we read these confidence values as indicators of the *effective sample size* that a particular datum represents, we can weight its contribution to the final model.

For both examples, we can simply multiply each piece of data by a value proportional to its confidence. It seems strange to say that a single datum can represent more than one sample, so for both skin-color and head texture models, we normalize all weights such that they fall in the interval $[0.0, 1.0]$. For the skin-color model, we use the residual from ellipse tracking to weight each set of observations (better fits correspond to greater confidence). In the case of the head-texture model, we weight by the normalized likelihood of the maximum likelihood pose, which is interpretable as an actual probability.

Performance improves for both cases. Results for the skin-color model are shown in Figure 7. Note how the pixel distribution is most concentrated in skin-colored regions in (c) because samples which were taken when tracking was unreliable were suppressed. This is in contrast to (b), where each sample was weighted evenly.

Reducing Bias from Tracking and Acquisition Functions: The problem we are least able to overcome is bias in the initial tracking function and the acquisition function, since they provide the supervisory data, $(p^0(\mathbf{x}), \mathbf{z}(\mathbf{x}))$. In the abstract, there is very little we can do to eliminate such a bias. But, there may be domain-specific solutions which help alleviate the problem.

For example, the skin-color model learns a pdf consisting of mostly skin color, together with a small contribution from pixels taken inadvertently from the background. If we can learn the distribution of background pixels, we can eliminate these with a Bayesian decision criterion to determine whether a given



(a) (b) (c) (d)

Fig. 7. (a) A raw image, as input to the final tracking function. (b-c) likelihood of pixel skin-color, based on learned final model (likelihoods scaled so that highest is darkest), with total weight of training data size kept constant: (b) 1 sample from each frame; (c) data weighted by ellipse residual (1 sample from each frame, for 70% of frames). (d) Bayesian decision to separate foreground from background: Black pixels mark foreground pixels; gray pixels mark potential foreground pixels which are more likely to be background.

pixel value, $\mathbf{u}^k$ is more likely to be skin or background. That is, $\mathbf{u}^k$ is skin if

$$p(\text{skin}|\mathbf{u}^k) > p(\text{bg}|\mathbf{u}^k) \tag{11}$$

$$p(\mathbf{u}^k|\text{skin})p(\text{skin}) > p(\mathbf{u}^k|\text{bg})p(\text{bg}), \tag{12}$$

where $p(\mathbf{u}^k|\text{skin})$ is acquired from Equation 9, $p(\mathbf{u}^k|\text{bg})$ can be acquired similarly by simply sampling pixels outside of the tracked ellipse (in practice, we collect entire frames of pixels from just a few frames), and $p(\text{skin})$ and $p(\text{bg})$ are set based on the relative area that they are expected to occupy in an image containing a face. See Figure 7(d) for an example in which only those pixels which occur frequently on the face, but very infrequently in the background are considered skin color. Modeling both skin and background as mixtures of a handful of Gaussians achieves a similar result [12], but without the granularity possible with a nonparametric model.

In the head orientation example, our original generic model exhibits a slight orientational bias – in Figure 3(c), a slight turn of the head to the left is visible. We can eliminate this bias by finding the angle at which the model appears most symmetrical and averaging the model with its reflection. Doing so does in fact reduce some of the error generated by the final model (see Figure 4, Row 6 vs. any of Rows 3-5).

Repeated Bootstrapping Finally, we mention the possibility of repeated bootstrapping. Clearly, if one model can be used to learn a second model, any combination of the first two models could be used to learn a third model. In Figure 1, f^1 and $\boldsymbol{\theta}(\mathbf{x})$ replace f^0 and $p^0(\mathbf{x})$, and bootstrap initialization iterates.

Strangely, in both of our examples, repeated bootstrapping does not appear to improve the final models. For learning skin-color, repeated bootstrapping is good for adapting to a changing color model [12], but for a fixed distribution, there is nothing more to be gained by going beyond the asymptotically learned color models. This is not surprising, since we have chosen to gather enough data in the first iteration to learn a good model.

Figures 4 and 5(a), show that even for the head-texture case, bootstrapping beyond the first iteration does not appear to improve pose estimates.

Figure 5(b) shows a one-dimensional schematic of what we believe is taking place. The x-axis shows the angular position on the model and the y-axis gives the extracted feature value. The top figure shows a generic model, the middle figure shows the ground truth model, and the bottom graph shows the bootstrapped model.

Pose estimation is equivalent to being given a short, noisy segment of the bottom function (the observation) and trying to find the best match displacement in the top function. In the case when we are trying to find a match to a uniquely-varying part of the model, such as Segment A', the corresponding segment is accurately localized (Segment A). This corresponds to cases when the front or side views (angular ranges 0-135) are being presented. Bootstrapping helps in this instance because the characteristics of the real model are transferred to the bootstrapped model.

When the observation segment is more homogeneous as in Segment B', the match is likely to be affected by noise and other small differences in the generic model, making matching inaccurate (Segment B). The bootstrapped model then merely recaptures the initial estimation error. This behavior was observed in many of the test images, where the bootstrapped model inhereted a tendency to misestimate back-of-head poses by a significant and *consistent* amount.

It is not clear whether the ineffectiveness of repeated bootstrapping should be expected in other similar cases of bootstrap initialization. One distant counterexample is the remarkable success of iterated bootstrapping in learning linear subspace models of faces [18].

6 Conclusion

We have presented bootstrap initialization, an abstract framework for using one model to guide the initialization of another model during tracking. We presented two examples of bootstrap initialization for tracking, using nonparametric models of target surface texture. In the first, we acquired a strong skin-color model of a user's face, given a weak edge-based model of user shape. In the second, we refined a generic model of head texture to suit a particular individual. Preliminary experiments show the potential for bootstrap initialization in tracking applications; in both cases, initial implementations were able to learn a bootstrapped model without undue concern for data dependencies in the acquired training data.

Additional experiments provided evidence toward the following tentative conclusions:

- Independence assumptions in the acquired training data can be violated to a great degree. Movement of target objects creates enough variation that the sufficient statistics of training data taken over an extended period of time closely match those of an ideally sampled data set.

– Dependencies in data can be removed through both generic and domain-specific strategies. Final models are better learned by taking advantage of such tactics.
– Repeated bootstrapping does not necessarily result in improved models.

In future work, we expect to delve deeper into theoretical limits of bootstrap initialization and more broadly into other tracking domains.

References

1. A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Patt. Anal. and Mach. Intel.*, 17(6), June 1995.
2. J.M. Bernardo and A.F.M. Smith. *Bayesian Theory*. John Wiley, Chichester, 1994.
3. S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. Computer Vision and Patt. Recog.*, pages 232–237, 1998.
4. A. Chiuso and S. Soatto. 3-D motion and structure causally integrated over time: Theory (stability) and practice (occlusions). Technical Report 99-003, ESSRL, 1999.
5. J.W. Davis and A.F. Bobick. The representation and recognition of action using temporal templates. In *CVPR97*, pages 928–934, 1997.
6. D. DeCarlo and D. Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *Proc. Computer Vision and Patt. Recog.*, pages 231–238, 1996.
7. P. Fua and C. Miccio. From regular images to animated heads: a least squares approach. In *Proc. European Conf. on Computer Vision*, pages 188–202, 1998.
8. M. Isard and A. Blake. ICondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. European Conf. on Computer Vision*, pages I:893–908, 1998.
9. T. S. Jebara and A. Pentland. Parametrized structure from motion for 3D adaptive feedback tracking of faces. In *Proc. Computer Vision and Patt. Recog.*, 1997.
10. J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. Int'l Conf. on Computer Vision*, pages I:572–578, 1999.
11. N. Oliver, A. Pentland, and F. Berard. LAFTER: Lips and face real time tracker. In *Proc. Computer Vision and Patt. Recog.*, 1997.
12. Y. Raja, S. J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, pages 228–233, 1998.
13. D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *Proc. European Conf. on Computer Vision*, pages 357–368, 1996.
14. A. Schoedl, A. Haro, and I. A. Essa. Head tracking using a textured polygonal model. In *Proc. Wkshp on Perceptual UI*, pages 43–48, 1998.
15. R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. In *Proc. Wkshp on Perceptual UI*, Banff, Canada, 1997.
16. H. Tao and T. S. Huang. Bezier volume deformation model for facial animation and video tracking,. In *Proc. IFIP Workshop on Modeling and Motion Capture Techniques for Virtual Environments (CAPTECH'98)*, November 1998.
17. K. Toyama. 'Look Ma, no hands!' Hands-free cursor control with real-time 3D face tracking. In *Workshop on Perceptual User Interfaces*, 1998.
18. T. Vetter, M. J. Jones, and T. Poggio. A bootstrapping algorithm for learning linear models of objects classes. In *Proc. Computer Vision and Patt. Recog.*, pages 40–46, 1997.
19. Y. Wu, K. Toyama, and T. S. Huang. Wide-range, person- and illumination-insensitive head orientation estimation. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, 2000.