Building Roadmaps of Minima and Transitions in Visual Models

Cristian Sminchisescu, Bill Triggs

Building Roadmaps of Minima and Transitions in Visual Models

Cristian Sminchisescu
University of Toronto, Department of Computer Science
Toronto, Ontario, M5S 3G4, Canada
crismin@cs.toronto.edu, www.cs.toronto.edu/~crismin
Bill Triggs
INRIA Rhône-Alpes, GRAVIR-CNRS
655 avenue de l'Europe, 38330 Montbonnot, France
Bill.Triggs@inrialpes.fr, www.inrialpes.fr/movi/people/Triggs

Abstract

Becoming trapped in suboptimal local minima is a perennial problem when optimizing visual models, particularly in applications like monocular human body tracking where complicated parametric models are repeatedly fitted to ambiguous image measurements. We show that trapping can be significantly reduced by building 'roadmaps' of nearby minima linked by transition pathways *— paths leading over low 'mountain passes' in the cost surface — found by locating the* transition state *(codimension-1 saddle point) at the top of the pass and then sliding downhill to the next minimum. We present two families of transition-state-finding algorithms based on local optimization. In* eigenvector tracking*, unconstrained Newton minimization is modified to climb uphill towards a transition state, while in* hypersurface sweeping*, a moving hypersurface is swept through the space and moving local minima within it are tracked using a constrained Newton method. These widely applicable numerical methods, which appear not to be known in vision and optimization, generalize methods from computational chemistry where finding transition states is critical for predicting reaction parameters. Experiments on the challenging problem of estimating 3D human pose from monocular images show that our algorithms find nearby transition states and minima very efficiently, but also underline the disturbingly large numbers of minima that can exist in this and similar model based vision problems.*

Keywords: *Model based vision, global optimization, saddle points, 3D human tracking.*

1 Introduction

Many visual modeling problems can be reduced to cost minimization in a high dimensional parameter space. Local minimization is usually feasible, but practical cost functions often have large numbers of local minima and it can be very difficult to ensure that the desired one is found. Exhaustive search is seldom feasible in more than about 2–3 dimensions, so global optimizers are typically built around heuristics for finding 'good places to look next'. This includes both deterministic methods like branch-and-bound and pattern search, and stochastic importance samplers like simulated annealing [26], genetic algorithms and tabu search [18].

Unfortunately, global optimization remains expensive with any of these methods. In this paper we develop an alternative strategy based on building 1-D 'roadmaps' linking the salient nearby minima. Each local minimum has a basin of attraction within which local optimization converges to it. Roughly speaking, our strategy is to start from a given minimum and to search for low lying 'mountain passes' that lead away from its basin of attraction, finding the new minima they lead to by sliding downhill using local optimization. The procedure is repeated recursively to build up a roadmap linking the neighboring minima. More precisely, any minimum-peak-cost path connecting two minima passes through a special state at its highest point, at which the path crosses the brow of the pass and begins to descend. Such "transition states" are saddle points (the gradient of the cost function vanishes there) at which the cost has a local maximum in the direction of the path and a local minimum in all orthogonal directions. Technically, transition states are "codimension 1" saddle points — ones with one negative and (in n dimensions) $n-1$ positive principal curvatures (Hessian eigenvalues). At the heart of our approach are two families of efficient numerical methods for locating the transition states surrounding a given local minimum (see, for instance fig. 4 on page 15).

Although there are many algorithms for finding local minima, it seems that finding transition states has generally been considered to be intractable, and we know of little previous work in the vision or optimization communities on methods for this. However such methods do exist in the computational chemistry / solid state physics community, where transition states are central to the theory of chemical reactions[1]. Both of our classes of transition-state-locating algorithms have roots in chemistry, and both are based on modified forms of local Newton minimization. **Eigenvector tracking** redefines a standard damped Newton minimization to converge uphill to a saddle point of the desired signature, while **hypersurface sweeping** sweeps a moving hypersurface through the space, using a constrained Sequential Quadratic Programming like iteration to track moving local minima within it. These methods should be useful in many visual modeling problems where local minima cause difficulties. Examples include model based tracking, reconstruction under correspondence ambiguities, and various classes of camera pose and calibration problems. In this paper, we present experimental results on monocular model based human pose estimation.

Contents: §2 describes our motivation and overall strategy. §3 discusses related trajectory methods in computational chemistry and global optimization. §4 introduces damped Newton methods for locating transition states, and discusses their relation to local minimization algorithms. §5 and §6 detail our Eigenvector Tracking and Hypersurface Sweeping algorithms. §7 illustrates the methods on a 2D toy problem. §8 presents our main experiments on locating minima during monocular

[1] Atomic assemblies can be modeled in terms of the potential energy induced by interactions among their atoms, *i.e.* by an energy function defined over the high-dimensional configuration space of the atoms' relative positions. A typical assembly spends most of its time near an energy minimum (a stable or quasi-stable state), but thermal perturbations may sometimes cause it to cross a transition state to an adjacent minimum — a chemical reaction. The peak energy of the lowest transition path joining two minima is the main determinant of the likelihood of such a perturbation, and hence determines the dominant reaction rate and mechanism.

reconstruction and tracking of 3D human body pose. §9 concludes the paper and discusses possible directions for future research.

2 Motivation and Overall Strategy

Overall strategy: Below we will focus on numerical strategies for locating nearby transition states starting from a given minimum. Here we briefly describe how these are used to build "roadmaps" of local minima, thus allowing quasi-global (or at least, somewhat less local) minimization of smooth cost functions with many local minima. Our basic strategy is simply to locate transition states (saddle points that have exactly one negative principal curvature) with one of the methods described below, then to slide downhill using local optimization to find the neighboring minima they lead to. Transition states are interesting because the highest point of any locally-minimal-peak-cost path between two minima of a smooth function is necessarily a transition state, and lower-cost paths tend to lead to lower-cost minima. However, the methods studied here can be generalized to find saddle types with larger numbers of negative principal curvatures, if desired.

The transition-state based search method is initialized at one or more known minima, and a working queue of local minima is maintained. At each step, the lowest cost minimum is popped off the queue, transition state searches in several directions are initialized from it, and for each successful search the corresponding neighboring minimum is found by local descent, checked for rediscoveries (within numerical tolerance), and if novel, re-enqueued. In favorable cases the algorithm can be run to completion[2], but if there are too many minima to enumerate explicitly, it can be limited to a fixed number of searches or function evaluations, in which case it tends to find nearby and relatively low-lying minima. This is useful in tracking, as mistracking is more likely to lead to a nearby minimum than a distant one.

Advantages of the transition state based approach: The above method works efficiently in many dimensions, and in practice it is close to linear in the number of minima that need to be enumerated. This good performance is made possible by the fact that transition state location methods are able to use local properties of the cost function to steer the search towards globally meaningful regions (transition states), from which other minima are easily found. In comparison, exhaustive search is infeasible in more than a few dimensions, and methods that search for nearby minima by patterned or purely random sampling also tend to become inefficient in high dimensional problems. Volume increases rapidly with radius in high dimensions, so the low-cost 'cores' of minima tend to be extremely small compared to their higher-cost surroundings. Thus, if random samples based at one minimum are spread widely enough to reach an adjacent minimum — and neighboring minima are

[2]It is straightforward to modify this heuristic search method to run indefinitely and converge with probability one simply by adding a random sampling element that is applied, perhaps with an exponentially decreasing probability in order not to degrade the efficiency of the method (*e.g.* consider the following: any time the number of function evaluation or saddle searches reaches an upper bound, say U_b, sample a point at random, optimize to find a local minimum, include it in the working set and increase U_b).

often separated by significant distances in parameter space (see fig. 7 below) — they will necessarily also be spread rather thinly owing to the large volume covered, so they can easily miss the minimum altogether, and even if not, they are much more likely to hit the high-cost shoulders of its basin of attraction than in its low-cost 'core'. Hence, even with random or pattern sampling, it is necessary to add a local optimization stage to push samples towards the high-likelihood regions in the 'cores' of the minima.

Moreover, in many applications where the cost function is very ill-conditioned, the sampling pattern needs to be tailored to the local shape of the cost surface to avoid massive sample wastage, as local minima tend to be distributed preferentially along low-cost-change / ill-conditioned directions. In particular, in monocular human tracking, there is substantial ill-conditioning owing to depth recovery ambiguities, and for any given image of the person there are many — usually thousands — of possible 3D kinematic solutions that need to be well sampled, all distributed along the ill-conditioned depth degrees of freedom. The above observations led us to develop the Covariance Scaled Sampling method [43, 46], but transition state search methods turn out to be even more effective at discovering such minima, as they easily find the transition states at the heads of elongated low-cost valleys. See [47] for a complementary minimum enumeration method, that exploits the forward/backward symmetry in the monocular human pose cost function in order to locate kinematic minima efficiently. See also [42] for a smoothing algorithm that reconstructs multiple plausible state trajectories for models where the multiplicity of solutions persists during tracking. Prior knowledge on typical human poses can be further used to learn application specific low-dimensional continuous generative models that are less ambiguous and allow more efficient search [41].

Difficulties of transition state based approach: Many efficient methods exist for finding local minima of smooth high dimensional cost surfaces. Minimization allows strong theoretical guarantees as the reduction in the function value provides a clear criterion for monitoring progress. For example, for a bounded-below function in a bounded search region, any method that ensures 'sufficient decrease' in the function at each step is 'globally convergent' to some local minimum [15]. Finding saddle points is much harder as there is no universal progress criterion and no obvious analogue of a 'downhill' direction. Newton-style iterations provide rapid *local* convergence near the saddle, but it is not so obvious how to find sufficiently nearby starting points. We will consider several methods that extend the convergence zone. We are mainly interested in saddles as starting points for finding adjacent minima, so we will focus on methods that can be started from a minimum and tuned to find nearby transition states. Efficient 'rubber band relaxation' methods also exist for finding the transition state(s) linking two given minima [35], but we will not need to consider this.

3 Transition State Location Strategies

We start with a brief overview of the computational chemistry / solid state physics literature on locating transition states. This literature should be accessible to vision workers with high-school

chemistry and a working knowledge of optimization. However the underlying ideas can be difficult to disentangle from chemistry-specific heuristics, and some of the references are rather naive about numerical optimization issues. We therefore give a self-contained treatment of generalizations of two of the most promising approaches below, in the language of numerical analysis.

A transition state is a local minimum along its $n-1$ positive curvature directions in parameter space, but a local maximum along its remaining negative curvature one. So transition state search methods are often formulated as a series of $(n-1)$-D minimizations, while moving or maximizing along the remaining direction. The main differences lie in the methods of choosing which directions to use.

Eigenvector tracking methods [10, 23, 7, 53, 54, 31, 22, 38, 33, 21, 11, 5] choose the ascent direction to be an eigendirection of the local cost function's Hessian (second derivative) matrix. A Newton based local minimization method is modified to locally increase the cost along the chosen curvature eigendirection, while still reducing it along the remaining eigendirections. In particular, if the lowest (most negative) curvature direction is chosen, the method attempts to find the lowest gradient path to a transition state by walking along the 'floor' of the local cost 'valley'. However success can not be guaranteed, as such valleys may continue indefinitely without leading to a saddle point [25, 49, 24]. Also, the method is not as canonical as it may seem: the eigendecomposition depends non-trivially on the coordinate system used, so, *e.g.*, changing the relative scaling of variables changes the trajectory followed; and to ensure progress, "the same" eigenvector must be followed at each step, but there is no natural identification between eigenvectors at different points, so there is no canonical way to ensure this sameness.

An early eigenvector tracking method [23] used explicit Newton minimization in the $(n-1)$-D space obtained by eliminating the coordinate with the largest overlap with the desired up-hill direction. Later quasi-Newton methods use Lagrange multipliers [7, 53, 54] or shifted Hessian eigenvalues [38, 33, 21, 11, 5] to ensure that the cost function is increased along the 'uphill' eigendirection while still being minimized in the orthogonal subspace. Owing to the lack of a global identification between eigenspaces at different points, maintaining a consistent direction to follow can be delicate and several competing methods exist, including using a fixed eigenvector index [23, 7, 53, 54, 31, 22] and following the eigendirection that is best aligned (has greatest dot product) with the current one [38, 33, 21, 11, 5].

Eigenvector tracking can be viewed as a 'virtual cost minimization' obtained by inverting the sign of both the 'uphill' eigenvalue and its corresponding gradient component [30, 21] (see below). This gives an intuitive algebraic analogy with minimization, but none of minimization's convergence guarantees as the virtual cost function changes at each step.

Trajectory methods from global optimization [14, 20, 6] are based on the idea of tracing routes through stationary points of a high-dimensional cost function. Some early 2D methods are based on alternating descents and ascents along eigendirections [14], without any clear ascent heuristic. "Golf" methods [20], aim at exploring different minima and equilibrate at a certain energy level

(assumed known a-priori). There are also randomized methods that speed up classical molecular dynamics simulations by generating trajectories on a modified potential with reduced well depth [51, 52] and lower potential at transition states. This insures both increased inter-minimum transition rates (and thus faster simulations), and unbiased behavior, in that the correct relative rates of different transitions are preserved in the modified potential. An application of a generalization of these methods to computer vision appears in [45]. Branin type methods [6, 3] are based on solving differential equations derived from the gradient stationarity condition, rather than explicitly minimizing the cost function. However, managing the singularities and bifurcations that arise during the iteration is an open problem.

Space Sweeping and Filling Methods: The above methods can fail to make global progress, *e.g.* owing to looping. Space sweeping methods [10, 1, 2, 4, 30, 19, 28, 17] guarantee more systematic progress by using a moving potential or hypersurface to sweep candidate points though the space in such a way that they often pass through transition states. Crippen & Sheraga's early method [10] builds an uphill path by stepping along a prespecified direction and minimizing in the hyperplane orthogonal to the direction at each step. Mousseau & Barkema use a similar but less rigorous technique based on changing the gradient sign in one direction followed by conjugate root-finding in the other directions [30]. Barkema [4] pushes the solution away from a known minimum by adding a steadily expanding repulsive spherical bias potential centred on the minimum, and optimizing subject to this soft constraint. New minima are found but the method does not attempt to pass exactly through saddle points. Abashkin & Russo [1, 2] minimize on successively larger-radius hyperspheres centered at a minimum, and also provide a method for refining approximate saddle point locations. The use of hyperspheres forces the search to move initially along the valley floor of the cost surface [24], so usually at most two distinct saddles can be found. Below we will show how to steer the initial search along any desired direction by using ellipsoidal surfaces. Closely related "*penalty methods*" from global optimization (filled function, tunneling) attempt to modify the cost function to prevent reconvergence to the currently known local minima, thus forcing the discovery of new ones [19, 28, 17]. As the extents of the basins of attraction are not known in advance, this requires the addition of increasingly strong repulsive potentials centered at the known minima, until a transition state is crossed and a new solution is found. The optimization becomes more complex as more minima are discovered because a repeller potential must be included for each known minimum. Including a large number of repellers also leads to an increasingly flat and contorted cost function. Repeller ideas are similar to those used in 'tabu search' methods [18], which try to avoid trapping and cyclic behavior by forbidding or penalizing moves that return to recently-visited states. Tabu search is traditionally a combinatorial method, but it can also be applied in the continuous domain by choosing a suitable problem discretization.

All of the above methods depend on particular choices. The hypersurface and repulsion based methods depend on the local shape and alignment of the moving hypersurfaces or potentials, while the eigenvector and valley following methods all depend on the local coordinate system — even

linear changes of coordinates completely change the eigenvalues and vectors, the notion of slowest-ascent valleys, *etc*. These dependencies are much stronger than in minimization, where they enter only via Levenberg-Marquardt style step damping (pure Newton iteration being coordinate-independent). They can be a nuisance, but they also allow initial search directions to be varied, so that many transition states can potentially be found starting from any given minimum.

The following sections detail several basic saddle point location methods. Damped Newton iteration §4 is useful for refining estimated saddle points but its convergence domain is too limited for general use. Eigenvector tracking §5 extends the convergence domain by operating a virtual Newton optimization (thus remaining relatively lightweight), but may be sensitive to the 'eigenvector following' heuristics. Hypersurface sweeping §6 is better founded in that it provides more guarantees of global progress, but no single sweep finds all saddle points and it is more complex to implement.

4 Damped Newton Methods for Finding Transition States

We first consider local damped Newton-like methods for finding saddle points. These provide rapid local convergence near saddle points, but higher level strategies are needed to ensure more global convergence. Let $f(\mathbf{x})$ be the cost function being optimized over n-D parameter vector $\mathbf{x}$, and let $\mathbf{g} \equiv \frac{\partial f}{\partial \mathbf{x}}$ be the function's **gradient** and $\mathbf{H} \equiv \frac{\partial^2 f}{\partial \mathbf{x}^2}$ be its **Hessian** at the current point $\mathbf{x}$. We seek transition states, *i.e.*stationary points ($\mathbf{g}(\mathbf{x})=\mathbf{0}$) at which the Hessian has one negative and $n-1$ positive eigenvalues. If there is a stationary point at $\mathbf{x}+\boldsymbol{\delta}\mathbf{x}$, first order Taylor approximation at $\mathbf{x}$ gives:

$$\mathbf{0} \;=\; \mathbf{g}(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) \;\approx\; \mathbf{g}(\mathbf{x}) + \mathbf{H}\,\boldsymbol{\delta}\mathbf{x} \tag{1}$$

Solving this linear system for $\boldsymbol{\delta}\mathbf{x}$ and iterating to refine the approximation gives the **Newton iteration**:

$$\mathbf{x} \;\leftarrow\; \mathbf{x}+\boldsymbol{\delta}\mathbf{x} \qquad \text{with update} \qquad \boldsymbol{\delta}\mathbf{x} \;=\; -\mathbf{H}^{-1}\mathbf{g} \tag{2}$$

When started sufficiently close to any regular stationary point ('regular' means roughly that $\mathbf{H}$ is nonsingular and 2^{nd} order Taylor expansion converges), Newton's method converges to it, but how close you need to be is a delicate point in practice.

For Newton-based *minimization*, the convergence can be globalized by adding suitable damping to shorten the step and stabilize the iteration. Most formulations use the **damped Newton** update:

$$\boldsymbol{\delta}\mathbf{x} \;=\; -(\mathbf{H}+\lambda\mathbf{D})^{-1}\mathbf{g} \tag{3}$$

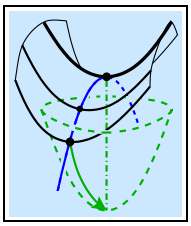
where $\mathbf{D}$ is a positive diagonal matrix (often the identity). The **damping factor** $\lambda > 0$ is manipulated by the algorithm to ensure stable and reliable progress downhill towards the minimum. Damping can be viewed as Newton's method applied to a modified local model for f, whose gradient at $\mathbf{x}$ is unchanged but whose curvature is steepened to $\mathbf{H}+\lambda\mathbf{D}$.

More generally, to stabilize Newton iteration near a saddle point, the negative Hessian curvatures must be made more negative, and the positive ones more positive. More globally, 'uphill motion' directions must be given sufficiently strong negative curvature, and 'downhill motion' ones sufficiently strong positive curvature. This can be conveniently expressed in a Hessian eigenbasis $\mathbf{H} = \mathbf{V}\,\mathbf{E}\,\mathbf{V}^\top$, where $\mathbf{E} = \mathrm{diag}(\lambda_1, ..., \lambda_n)$ are the eigenvalues and the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{H}$ with respect to the current coordinates. In this basis, the undamped Newton update becomes $\boldsymbol{\delta}\mathbf{x} = -\mathbf{V}\,(\bar{g}_1/\lambda_1, \ldots, \bar{g}_n/\lambda_n)^\top$, where $\bar{g}_i \equiv (\mathbf{V}^\top\mathbf{g})_i$ are the eigen-components of the gradient. Damping can be introduced by replacing this with:

$$\boldsymbol{\delta}\mathbf{x} = -\mathbf{V}\,\mathbf{u}(\lambda), \quad \text{where} \quad \mathbf{u}(\lambda) \equiv \left(\tfrac{\bar{g}_1}{\lambda_1+\sigma_1\lambda}, ..., \tfrac{\bar{g}_n}{\lambda_n+\sigma_n\lambda}\right)^\top = \left(\tfrac{\sigma_1\bar{g}_1}{\sigma_1\lambda_1+\lambda}, ..., \tfrac{\sigma_n\bar{g}_n}{\sigma_n\lambda_n+\lambda}\right)^\top \tag{4}$$

where $\sigma_i = \pm 1$ is a desired sign pattern for the λ_i. Damping $\lambda > \max_i(-\sigma_i\lambda_i, 0)$ ensures that the denominators are positive, so that the iteration moves uphill to a maximum along the eigendirections with $\sigma_i = -1$ and downhill to a minimum along the others. At each step this can be viewed as the minimization of a virtual local function with curvatures $\lambda + \sigma_i\lambda_i$ and sign-flipped gradients $\sigma_i\bar{g}_i$. However the virtual model changes at each step and f itself is *not* minimized, so none of the usual convergence guarantees of well-damped minimization apply.

As in minimization, λ must be varied to ensure smooth progress. There are two main strategies for this: **Levenberg-Marquardt** methods manipulate λ directly, while **trust region** ones maintain a local region of supposed-'trustworthy' points, and choose λ to ensure that the step stays within it, for instance $\|\boldsymbol{\delta}\mathbf{x}(\lambda)\| = \|\mathbf{u}(\lambda)\| \lesssim r$ where r is a desired 'trust radius'. (Such a λ can be found efficiently with a simple 1-D Newton iteration started at large λ [15]). In either case, one monitors model accuracy metrics such as the second-order-Taylor based relative f-prediction error:



$$\beta = \left|\tfrac{f(\mathbf{x}+\boldsymbol{\delta}\mathbf{x})-f(\mathbf{x})}{\mathbf{g}^\top\boldsymbol{\delta}\mathbf{x}+\boldsymbol{\delta}\mathbf{x}^\top\mathbf{H}\boldsymbol{\delta}\mathbf{x}/2} - 1\right| \tag{5}$$

as well as convergence criteria. Low accuracy indicates that the damping should be increased (larger λ or smaller r), and high accuracy that it can safely be decreased to allow longer steps and speed convergence (*e.g.*, by scaling λ or r up or down by fixed constants).

As in minimization, if the exact Hessian is unavailable, quasi-Newton approximations based on previously computed gradients can be used. Since positive definiteness is no longer required, non-positive update rules such as Powell's are generally preferred to the standard BFGS one [15, 5]:

$$\mathbf{H} \leftarrow \mathbf{H} - \frac{\boldsymbol{\delta}\mathbf{x}^\top\boldsymbol{\xi}}{\|\boldsymbol{\delta}\mathbf{x}\|^4}\boldsymbol{\delta}\mathbf{x}\,\boldsymbol{\delta}\mathbf{x}^\top + \frac{\boldsymbol{\xi}\,\boldsymbol{\delta}\mathbf{x}^\top + \boldsymbol{\delta}\mathbf{x}\,\boldsymbol{\xi}^\top}{\|\boldsymbol{\delta}\mathbf{x}\|^2} \quad \text{where} \quad \boldsymbol{\xi} = \mathbf{g}(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) - \mathbf{g}(\mathbf{x}) - \mathbf{H}(\mathbf{x})\boldsymbol{\delta}\mathbf{x} \tag{6}$$

5 Eigenvector Tracking

Eigenvector tracking methods take the above Hessian eigenbasis method and add a heuristic for selecting which eigendirection(s) to modify. Basically, once the coordinate system has been fixed,

the remaining freedom in (4) is the choice of the signs σ_i. The damped iteration tries to move uphill to a maximum along directions with $\sigma_i = -1$, and downhill to a minimum along directions with $\sigma_i = +1$, *i.e.* it tries to find a stationary point whose principal curvatures λ_i have the same signs as the σ_i. So for minimization we need $\sigma_i = +1$, while for a transition state search exactly one σ_i must be made negative. (This holds irrespective of the λ_i and $\bar{g}_i$ at the *current* state, which affect only the amount of damping required for stability).

The question of which eigenvalue to change is less obvious than it might seem. To ensure continued progress we need to flip "the same" eigenvalues at each step. Unfortunately, there is no globally well defined correspondence rule linking eigenvectors at different points, so there is no rigorous definition of "sameness" and heuristics must be used. For transition state searches we only need to track a single eigenvector (the one that is given $\sigma_i = -1$), so we will concentrate on this case. A simple approach would be to choose a fixed direction in space (perhaps the initial eigendirection) and take the eigenvector with maximal projection along this direction. But many such directions are possible and the most interesting transition states may happen not to have negative curvature along the particular direction(s) chosen. Alternatively, we can try to track a given eigendirection as we move. The problem is that globally, eigendirections are by no means stable. Eigenvalues change as we move about the space, but generically (in codimension 1) they never cross. When they approach one another, the eigenbasis of their 2D subspace becomes ill-conditioned and slews around through roughly 90°. Seen from a large enough scale, the eigenvalues appear to cross with more or less constant eigenvectors, but on a finer scale there is no crossing, only a smooth but rapid change of eigendirection that is difficult to track accurately. Whichever of the two behaviors is desired, it is difficult to choose a step length that reliably ensures it, so the numerical behavior of eigenvector-tracking methods can sometimes be sensitive to fine scale steps. In fact, the imprecise coarse scale view is probably the desired one: if we are tracking a large eigenvalue and hoping to reduce it to something negative, it will have to "pass through" each of the smaller eigenvalues. Tracking at too fine a scale is fatal as it (correctly) prevents such crossings, instead making the method veer off at right angles to the desired trajectory. Even without these problems, there is no guarantee that a saddle point of the desired signature is found — the trajectory might simply continue to climb indefinitely. Also, as with other damped Newton methods, the whole process is dependent on the affine coordinate system used. Nevertheless, eigenvector tracking is relatively lightweight, simple to implement, and it often works well in practice. Generalization to k-th order saddles is straightforward: the above procedure is applied to the lowest k (or some k) eigenvectors, and these are tracked simultaneously. At each step, we simply choose the damping factor λ to ensure that the corresponding directions have negative augmented curvatures.

5.1 Implementation Details

Our implementation of eigenvector tracking is summarized in fig. 1. We use the damped Newton saddle step (4), moving away from the minimum by reversing the sign of the gradient in the tracked

Eigenvector Tracking Transition State Search

Initialization

Set starting point $\mathbf{x}_0$, initial tracking direction $\mathbf{t}{=}\mathbf{v}_e$ for some e, and initial trust radius r.

Eigenvector Tracking Loop

A. At $\mathbf{x}_k$, find $f_k, \mathbf{g}_k, \mathbf{H}_k$, the Hessian eigen-decomposition $(\lambda_i, \mathbf{v}_i)$ and eigen-basis gradient $\bar{g}_k$. Set n_- to the number of negative eigenvalues. If the problem has active internal constraints, project $\mathbf{t}$ onto the constraint surface.

B. If $k > 0$ choose $e = \max_i |\mathbf{v}_i^\top \mathbf{t}|$. Set $\alpha = |\mathbf{v}_e^\top \mathbf{t}|$ and $\mathbf{t} = \mathbf{v}_e$.

C. If $\lambda_e > 0$ set $\bar{g}_e = -\bar{g}_e$. Take an undamped Newton step if $n_-{=}1$ and $\|\boldsymbol{\delta}\mathbf{x}\| \leq r$. Otherwise take a damped one (4) with λ chosen so that $\|\boldsymbol{\delta}\mathbf{x}\| \leq r$.

D. Find the new f and the modeling error β (5). If $\beta > 0.3$ (say) or $\alpha < 0.7$ (say), shrink the trust radius r by say 50%. Otherwise, if $\beta < 0.2$ (say) and we took a damped step, grow r by say 40%.

E. If $\beta \geq 1$ go to step C (try a shorter step). If $\|\mathbf{g}_k\| < \epsilon$ return success if $n_-{=}1$, failure otherwise (convergence / failure exit case). Otherwise, go to step A (do next iteration).

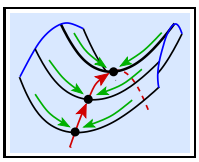
Figure 1: Summary of the eigenvector tracking algorithm for transition state search.

eigendirection if this has positive curvature. The damping $\lambda > 0$ is controlled to keep the step within a trust radius r and to dominate any undesired negative eigenvalues. The trust radius is set by monitoring the accuracy (5) of the local model for f.

In some of our target applications, the underlying problem also has bound constraints that need to be maintained. For hypersurface sweeping (below) this just adds additional constraints to the within-hypersurface minimizations, but for eigenvector tracking we introduced a trust region step calculation routine that uses a projection strategy to handle constraints on $\mathbf{x}$, and also projects the eigenvector-tracking direction $\mathbf{t}$ along the constraints to ensure stability.

6 Hypersurface Sweeping

Eigenvector trackers do not enforce any notion of global progress, so they can sometimes cycle or stall. To prevent this we can take a more global approach to the transition state searches '$(n{-}1)$-D minimization and 1D maximization'. Hypersurface sweeping approaches sweep an $(n{-}1)$-D hypersurface across the parameter space — typically a moving hyperplane or an expanding hyper-ellipsoid centered at the initial minimum — tracking local minima within the hypersurface and looking for temporal maxima in their function values. The intuition is that as the hypersurface sweeps towards a transition state, and assuming that it approaches along its cone of negative curvature directions, the $(n{-}1)$-D mini-

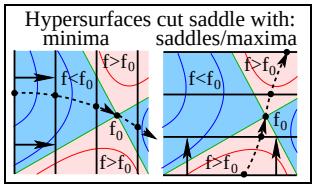


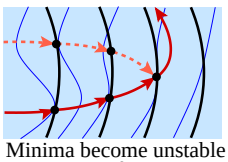
mization forces the hypersurface-minimum to move along the lowest path leading up to the saddle's 'col', and the 1-D maximization detects the moment at which the col is crossed. The method can not stall or cycle as the hypersurface sweeps through each point in the space exactly once.

The moving hypersurface can be defined either implicitly, in terms of the level sets $c(\mathbf{x}) = t$ of some parameter space function $c(\mathbf{x})$ (a linear form for hyperplanes, a quadratic one for hyperellipsoids...), or explicitly in terms of a local parameterization $\mathbf{x} = \mathbf{x}(\mathbf{y}, t)$ with respect to some hypersurface-t-parameterizing $(n-1)$-D vector $\mathbf{y}$. The minimum-tracking problem becomes:

$$\text{local\_max}_t \; f_c(t) \qquad \text{where} \qquad f_c(t) \equiv \begin{cases} \text{local\_min}_{c(\mathbf{x})=t} \, f(\mathbf{x}) \\ \text{local\_min}_{\mathbf{y}} \, f(\mathbf{x}(\mathbf{y}, t)) \end{cases} \tag{7}$$

There are some caveats. Firstly, we may need to find and track several minima, as different local minima on the hypersurface typically lead to different transition states. Secondly, even if we could track every local minimum within the hypersurface, this would not suffice to find every transition state: each family of hypersurfaces is 'blind' to some transition state orientations. Transitions that are approached in 'downhill' rather than 'uphill' directions — ones that are cut in negative curvature directions (the hypersurface's tangent space intersects the transition states' cone of negative curvature directions) — appear as saddle points or local maxima within the hypersurface, and so can not be found by tracking minima alone. Finally, some local maxima of $f_c(t)$ do not indicate transition states. The minimum being tracked can also simply disappear (topologically annihilate with another within-hypersurface saddle point), causing the tracked point to suddenly fall away to some other minimum on the hypersurface. This generates an abrupt 'sawtooth'-shaped maximum in $f_c(t)$. In more detail, at any stationary-within-hypersurface point, the projection of $\mathbf{g}=\frac{\partial f}{\partial \mathbf{x}}$ vanishes, so f's isosurface is necessarily tangent to the local hypersurface: $\mathbf{g} \propto \frac{\partial c}{\partial \mathbf{x}}$ or $\mathbf{g}^\top \frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \mathbf{0}$. The point is a within-hypersurface-minimum, -saddle, or -maximum respectively as the cost isosurface has higher/mixed/lower signed curvature than the local hypersurface (*i.e.* as the isosurface is locally inside/mixed/outside the hypersurface). At points where the moving hypersurface transitions from being outside to being mixed w.r.t. the local isosurface, the minimum being tracked disappears and the solution drops abruptly to some other hypersurface-minimum, producing a 'sawtooth' maximum in $f_c(t)$. The sweep can continue from the new minimum so this is not a problem as far as finding subsequent minima is concerned, but if transition states are needed it is important to eliminate the sawtooth maxima. (Similarly, minima of $f_c(t)$ correspond to true local minima of $f(\mathbf{x})$ — and there is no problem of 'blindness' in this case — *except* when they result from a sawtooth transition).




Minima become unstable when isosurfaces curve less than hypersurfaces

As each family of hypersurfaces is blind to some transition state orientations, it is useful to try several different families. For example, hyperplanes find forwards-looking transitions while hyperellipsoids find outward-looking ones. If we start the track at a minimum $\mathbf{x}_0$ of $f(\mathbf{x})$, the initial

tracking direction is determined by the hyperplane normal or the ellipsoid shape and the Hessian $\mathbf{H} = \mathbf{H}(\mathbf{x}_0)$. First consider the family of hyper-ellipsoids $c(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_0)^\top \mathbf{A}\,(\mathbf{x}-\mathbf{x}_0) = t$, where $\mathbf{A}$ is some positive definite matrix. To second order, $f(\mathbf{x})$ generically has exactly two local minima on an infinitesimal ellipsoid $c(\mathbf{x}) = t$: the $\pm$ directions of the smallest eigenvector of the matrix pencil $\mathbf{A}+\lambda\,\mathbf{H}$. (Numerically, these can be found by generalized eigendecomposition of $(\mathbf{A}, \mathbf{H})$, or standard eigendecomposition of $\mathbf{L}^{-\top}\mathbf{H}\mathbf{L}^{-1}$ where $\mathbf{L}\mathbf{L}^\top$ is the Cholesky decomposition of $\mathbf{A}$). For most $\mathbf{A}$ there are thus only two possible initial trajectories for the moving minimum, and so at most two nearest transition states will be found. To find other nearby transition states we need to modify $\mathbf{A}$. We can enforce any desired initial direction $\mathbf{u}$ by taking the 'neutral' search ellipsoids $\mathbf{A} = \mathbf{H}$ (on which f is constant to second order, so that all initial directions are equally good) and flattening them slightly relative to the cost isosurfaces in the direction $\pm\mathbf{u}$. To satisfy the Lagrange multiplier condition for a constrained minimum, $\frac{\partial c}{\partial \mathbf{x}} \propto \frac{\partial f}{\partial \mathbf{x}}$, we can take:

$$\mathbf{A} = \mathbf{H} + \mu \frac{\mathbf{g}\,\mathbf{g}^\top}{\mathbf{u}^\top \mathbf{g}} \tag{8}$$

where $\mathbf{g} = \mathbf{H}\,\mathbf{u}$ is the cost gradient (and hence isosurface normal) for a small displacement $\mathbf{u}$, and μ is a positive constant, say $\mu \sim 0.1$ for mild flattening. Similarly, for hyperplanes $c(\mathbf{x}) = \mathbf{n}^\top(\mathbf{x}-\mathbf{x}_0) = t$ with normal $\mathbf{n}$, the initial tracking direction is $\mathbf{u} = \pm\mathbf{H}^{-1}\mathbf{n}$, so to search in direction $\mathbf{u}$ we need to take $\mathbf{n} = \mathbf{H}\,\mathbf{u}$. (Note that, particularly if $\mathbf{H}$ is ill-conditioned, more neutrally scaled $\mathbf{A}, \mathbf{n}$ tend to produce to initial directions that are closely aligned with $\mathbf{H}$'s smallest eigenvector, which is likely to lead to poor search diversity).

By finding and tracking within-hypersurface saddle points rather than just within-hypersurface minima, the sweeping method can in principle be generalized to find saddle points with several negative eigen-curvatures, but similar caveats would apply.

6.1 Hypersurface Sweeping Equations

This section summarizes the equations needed to implement hypersurface sweeping, for both implicitly-specified and parametrically-specified hypersurfaces. For the implicit approach, let $\mathbf{g}_c \equiv \frac{\partial c}{\partial \mathbf{x}}$ and $\mathbf{H}_c \equiv \frac{\partial^2 c}{\partial \mathbf{x}^2}$ be the gradient and Hessian of the hypersurface constraint function $c(\mathbf{x})$. The hypersurface constraint is enforced with a Lagrange multiplier λ, solving:

$$\frac{\partial}{\partial \mathbf{x}}\,(f+\lambda\,c) \;=\; \mathbf{g} + \lambda\,\mathbf{g}_c \;=\; \mathbf{0} \qquad \text{subject to} \qquad c = t \tag{9}$$

If we are currently at $(\mathbf{x}, \lambda)$, second order Taylor expansion of these equations for a constrained minimum at $(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}, \lambda+\delta\lambda)$ gives the standard **sequential quadratic programming** update rule for $(\boldsymbol{\delta}\mathbf{x}, \delta\lambda)$:

$$\begin{pmatrix} \mathbf{H}_\lambda & \mathbf{g}_c \\ \mathbf{g}_c^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}\mathbf{x} \\ \delta\lambda \end{pmatrix} \;=\; -\begin{pmatrix} \mathbf{g} + \lambda\,\mathbf{g}_c \\ c - t \end{pmatrix} \qquad \text{where} \qquad \mathbf{H}_\lambda \;\equiv\; \mathbf{H}+\lambda\,\mathbf{H}_c \tag{10}$$

(The $\lambda \mathbf{H}_c$ term in the Hessian is often dropped for simplicity. This slows the convergence but still gives correct results).

Similarly, in the parametric approach let $\mathbf{J} \equiv \frac{\partial}{\partial \mathbf{y}}\mathbf{x}(\mathbf{y}, t)$. The chain rule gives the reduced gradient $\mathbf{g}_\mathbf{y}$ and Hessian $\mathbf{H}_\mathbf{y}$:

$$\mathbf{g}_\mathbf{y} = \mathbf{J}\,\mathbf{g} \tag{11}$$

$$\mathbf{H}_\mathbf{y} = \mathbf{J}\,\mathbf{H}\,\mathbf{J}^\top + (\frac{\partial}{\partial \mathbf{y}}\mathbf{J}) \cdot \mathbf{g} \tag{12}$$

These can be used directly in the Newton update rule $\boldsymbol{\delta}\mathbf{y} = -\mathbf{H}_\mathbf{y}^{-1}\,\mathbf{g}_\mathbf{y}$. In particular, if we eliminate one $\mathbf{x}$-variable — say x_n so that $\mathbf{y} = (x_1, ..., x_{n-1})$ and $x_n = x_n(\mathbf{y}, t)$ — we have:

$$\mathbf{J} = \left(\mathbf{I} \,|\, \tfrac{\partial x_n}{\partial \mathbf{y}}\right), \qquad \mathbf{g}_\mathbf{y} \;=\; \tfrac{\partial f}{\partial \mathbf{y}} + g_n\,\tfrac{\partial x_n}{\partial \mathbf{y}}, \qquad \mathbf{H}_\mathbf{y} \;=\; \mathbf{J}\,\mathbf{H}\,\mathbf{J}^\top + g_n\,\tfrac{\partial^2 x_n}{\partial \mathbf{y}^2} \tag{13}$$

To save optimization work and for convergence testing and step length control, it is useful to be able to extrapolate the position and value of the next minimum from existing values. This can be done, *e.g.*, by linear extrapolation from two previous positions, or analytically by solving the constrained minimum state update equations $(\mathbf{g}+(\lambda+\delta\lambda)\mathbf{g}_c)(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) = \mathbf{0}$ or $\mathbf{g}_\mathbf{y}(\mathbf{y}+\boldsymbol{\delta}\mathbf{y}, t+\delta t) = \mathbf{0}$ to first order, assuming that $\mathbf{x}, t$ is already a minimum and $t \to t+\delta t$:

$$(\boldsymbol{\delta}\mathbf{x},\ \delta\lambda) \;=\; \tfrac{1}{\mathbf{g}_c^\top \mathbf{H}_\lambda^{-1}\mathbf{g}_c}\left(\mathbf{H}_\lambda^{-1}\mathbf{g}_c,\ -1\right)\,\delta t \tag{14}$$

$$\boldsymbol{\delta}\mathbf{y} \;=\; -\mathbf{H}_\mathbf{y}^{-1}\left(\tfrac{\partial \mathbf{J}}{\partial t}\,\mathbf{g} + \mathbf{J}\,\mathbf{H}\,\tfrac{\partial \mathbf{x}}{\partial t}\right)\,\delta t \qquad \boldsymbol{\delta}\mathbf{x} \;=\; \mathbf{J}\,\boldsymbol{\delta}\mathbf{y} + \tfrac{\partial \mathbf{x}}{\partial t}\,\delta t, \tag{15}$$

Standard Taylor expansion of $f(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}(t))$ then gives:

$$f_c(t+\delta t) \;\approx\; f_c(t) + f_c'\,\delta t + \frac{1}{2} f_c''\,\delta t^2 \tag{16}$$

with $f_c' \;=\; \mathbf{g}\,\frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}$ and $f_c'' \;=\; \frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}^\top \mathbf{H}\,\frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}$. For step length control, we can either fix δt and solve for $\boldsymbol{\delta}\mathbf{x}$ or $\boldsymbol{\delta}\mathbf{y}$ (and hence $\mathbf{x}+\boldsymbol{\delta}\mathbf{x} \equiv \mathbf{x}(\mathbf{y}+\boldsymbol{\delta}\mathbf{y}, t+\delta t)$), or fix a desired trust region for $\boldsymbol{\delta}\mathbf{x}$ or $\boldsymbol{\delta}\mathbf{y}$ and work backwards to find a δt giving a step within it.

6.2 Implementation Details

The hyper-ellipsoid sweeping method that we used in the experiments below is summarized in fig. 2. We start at a local minimum and use centered, curvature-eigenbasis-aligned ellipsoidal hypersurfaces flattened along one eigendirection, say the e^{th}. This restricts the initial search to an eigendirection (the e^{th}). This limitation could easily be removed, but it gives a convenient, not-too-large set of directions to try. All calculations are performed in eigen-coordinates and the minimum is tracked using variable elimination (13) on x_e. In eigen-coordinates, the on-hypersurface constraint becomes:

$$\sum_i (x_i/\sigma_i')^2 = t^2 \tag{17}$$

Hyper-ellipsoid Sweeping Transition State Search

1. Initialization

Given initial minimum $\mathbf{x}_0$ with Hessian $\mathbf{H}$, eigen-decompose $\mathbf{H}$ to $(\lambda_i, \mathbf{v}_i)$ with principal radii $\sigma_i = 1/\sqrt{\lambda_i}$. Choose an initial search eigen-direction e. Shrink σ_e by say 20% and prepare to eliminate x_e. Set initial step $\mathbf{x}_1 = \mathbf{x}_0 + t_1 \sigma_e \mathbf{v}_e$ where t_1 is say 3. Go to step 2.B.

2. Loop, Updating Hypersurface and Minimizing

A. k=k+1. Estimate an initial $\mathbf{x}_k$ by linear extrapolation to the trust radius. Compute the resulting t_k.

B. Minimize f on the t_k ellipsoid to get $f_c(t_k)$: $\mathbf{y}_k = \arg\min_{\mathbf{y}} f(\mathbf{x}_k(\mathbf{y}, t_k))$.

C. Compute $f_c' = \frac{\partial}{\partial t} f_c(t_k)$. If $f_c' < \epsilon$ we are near or past saddle: go to step 3.A. Otherwise go to step 2.A.

3. Line Search for Transition State Refinement

A. If $|f_c'| < \epsilon$, exit.

B. k=k+1. Estimate t_{saddle} by linear interpolation of last two f_c' values.

B. Optimize $\mathbf{y}_k$ as in step 2.B and go to step 3.A.

Figure 2: Our hyper-ellipsoid sweeping algorithm for transition state search.

where the σ_i' are the principal standard deviations, except that the e^{th} (eliminated) one is shrunk by say 20%. Taking $\mathbf{y} = (x_1, ..., x_{e-1}, x_{e+1}, ..., x_n)$ and solving for x_e gives:

$$x_e(\mathbf{y}, t) \;=\; \pm\sigma_e' \sqrt{t^2 - \textstyle\sum_{i \neq e} (x_i/\sigma_i')^2} \tag{18}$$

The necessary derivatives are easily found. We use an $\mathbf{x}$-based trust region to choose the time step. First we estimate an initial $\mathbf{x}_{k+1}$ by linear extrapolation:

$$\mathbf{x}_{k+1} \;\approx\; \mathbf{x}_k + r\,\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|} \tag{19}$$

where r is a trust region radius for $\boldsymbol{\delta}\mathbf{x}$, then we solve for the corresponding t_{k+1} using the ellipsoid constraint, and finally we optimize $\mathbf{x}_{k+1}$ on this ellipsoid.

In our target application there are also underlying equality or bound constraints on the model. These are handled simply by including them as additional constraints in the hypersurface minimization.

7 2D Examples

This section illustrates our eigenvector tracking and hypersurface sweeping methods on the **Müller potential**, a very simple 2D example that is often used to demonstrate such methods in computa-

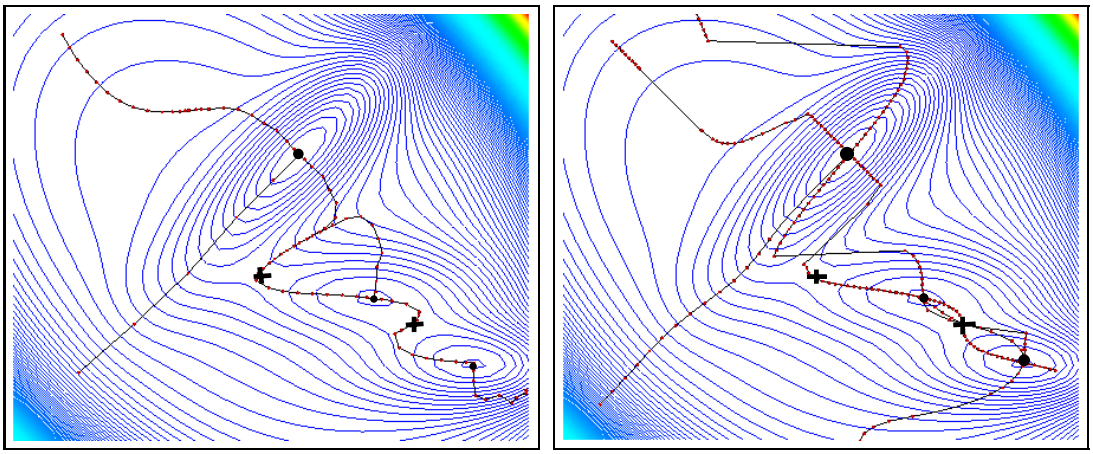


Figure 3: Trajectories for the eigenvector following (left) and hyper-ellipsoid sweeping (right) algorithms on the Müller cost surface, initialized along the $\pm$ eigendirections of the 3 minima (the position of minima is shown with dots, the saddles are shown with crosses).

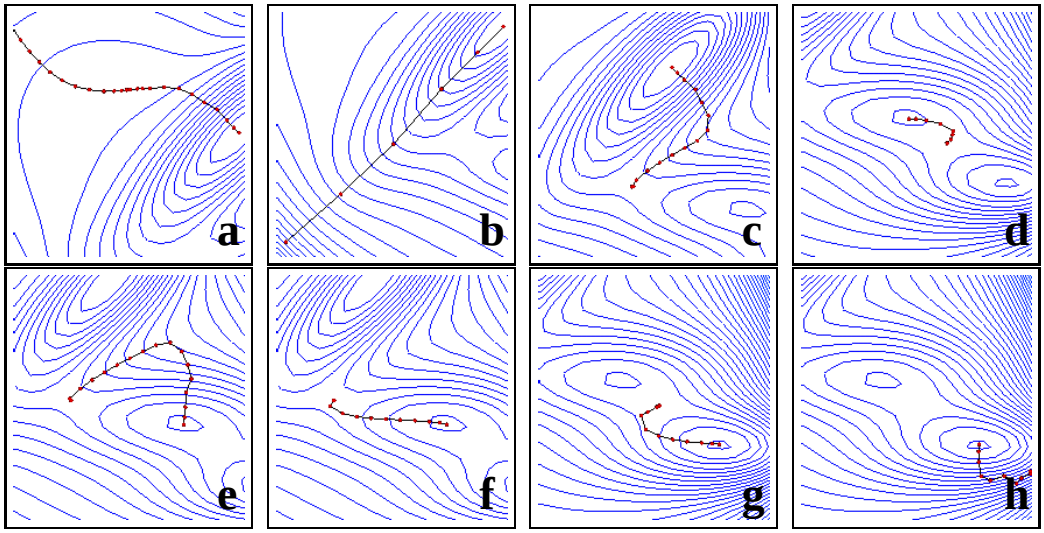


Figure 4: **(a)-(h)** Trajectories for eigenvector following on the Müller cost surface, started from different minima along different principal curvature directions. Note that the trajectories **(a), (b)** and **(h)** do not converge — in fact, no saddles exist in those regions of the space.

tional chemistry. The Müller potential has the form

$$V(x,y) = \sum_{i=1}^{4} A_i \, e^{a_i(x-x_i)^2 + b_i(x-x_i)(y-y_i) + c_i(y-y_i)^2} \tag{20}$$

where $\mathbf{A} = (-200, -100, -170, 15)$, $\mathbf{a} = (-1, -1, -6.5, 0.7)$, $\mathbf{b} = (0, 0, 11, 0.6)$, $\mathbf{c} = (-10, -10, -6.5, 0.7)$, $\mathbf{x} = (1, 0, -0.5, -1)$, $\mathbf{y} = (0, 0.5, 1.5, 1)$. It has three local minima M_1, M_2, M_3 separated by two saddle points S_1, S_2. The minima are separated by around one length unit, and

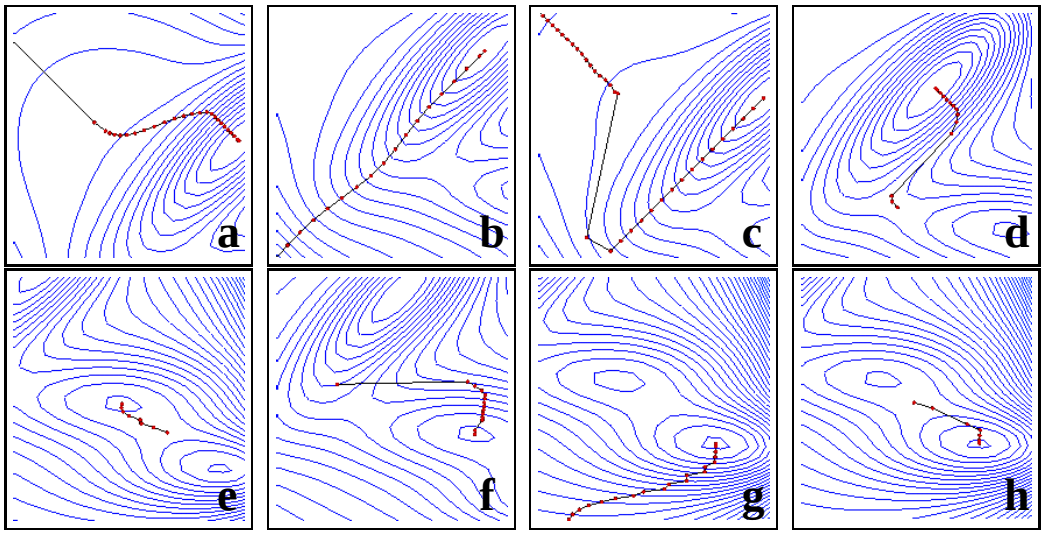


Figure 5: **(a)-(h)** Trajectories for hypersurface sweeping on the Müller cost surface, started from different minima along different principal curvature directions. The trajectories **(a), (b), (c)** and **(g)** do not converge to saddles. **(b)** and **(c)** use the same initial search direction but different ellipsoid flattening factors. **(c)** is flattened less than **(b)**, which allows a 'sawtooth' transition to occur, so that the track jumps abruptly northwards to a new within-hypersurface minimum. **(f)** shows another example — the saddle is crossed in an abrupt jump, without passing through the transition state (but the subsequent minimum will later be found, as required). These effects are intrinsic: they are *not* caused by discretization, step size effects, *etc*.

the transition states are around 100–150 energy units above the lowest minimum. See fig. 3 for a concise summary of results for the Eigenvector tracking (ET) and Hypersurface Sweeping (HS) methods. In the figure, the position of minima is shown with dots and that of the saddles with crosses.

Figs 4 and 5 respectively plot the trajectories of the eigenvector tracking and hypersurface sweeping methods, with initial search along all four principal curvature directions of each minimum. The hypersurface sweeping algorithm is also run for extended trajectories through several saddles and minima in fig. 6. In this simplistic example all of the minima and transition states can actually be found by a single sweep, but this is unusual in more complex problems.

8 Monocular Human Pose Reconstruction and Inter-Frame Tracking

In this section, we apply the transition state location and local minima mapping methods to the difficult problem of 3D articular human tracking from monocular image sequences.

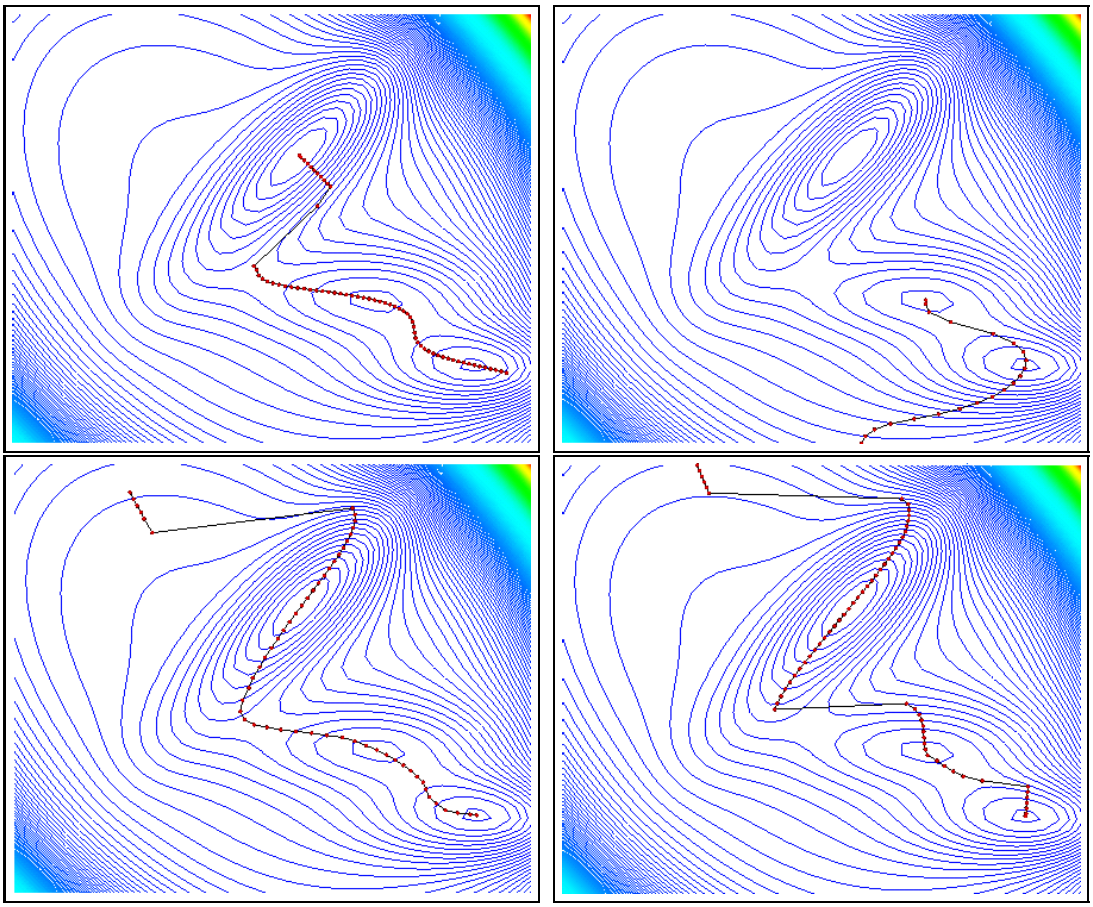


Figure 6: Trajectories for hypersurface sweeping on the Müller cost surface, for trajectories started in different minima, but not stopped after the first saddle detection. Several saddles and minima are found in each sweep.

8.1 Previous Work on Monocular Human Tracking

There is a large literature on human motion tracking, but relatively few works tackle the 3D-from-monocular case, where local minima are particularly troublesome owing to poor control of the depth degrees of freedom. We will mention only a few works that make contributions to the search for local minima problem in the context of generative 3D models. Deutscher *et al* use an annealed sampling method and multiple cameras to widen the search and limit the presence of spurious minima [12]. (Their sampling procedure resembles Neal's [32], except that Neal also includes an additional importance sampling correction designed to improve mixing). During annealing, the search for parameters is driven by noise proportional with their individual variances [13]. Considered as an improved (implicit) search space decomposition mechanism, an early method of this type was proposed by Gavrila & Davis [16] to efficiently sample partial kinematic chains. Adaptively identifying and sampling parameters with high variance is useful, but kinematic parameters usually

have quite strong interactions that make simple axis-aligned sampling questionable. For monocular reconstruction, it is important to realize that the principal axes of uncertainty change drastically depending on the viewing direction (as noted for instance in [43, 40, 46]). Sidenbladh *et al* use an intensity based cost function and focus search in the neighborhood of known trajectory pathways by particle filtering with importance sampling based on either a learned walking model or a database of motion snippets [36, 37]. Choo & Fleet [9] combine particle filtering and hybrid Monte Carlo sampling to estimate 3D human motion, using a cost function based on joint re-projection error given input from motion capture data. Sminchisescu & Triggs [43, 40, 46] argue that an effective random sampler must combine all three of cost-surface-aware covariance scaling, a sampling distribution with widened tails for deeper search, and local optimization (because deep samples usually have very high costs, and hence will not be resampled even if they ultimately lead to other minima). All of these works note the difficulty of the multiple-minimum problem and attempt to develop techniques or constraints (on the scene, motion, number of cameras or background) to tackle it. In a complementary approach to the one presented here, Sminchisescu & Triggs [45] proposed MCMC procedures that actively modify the effective cost function to stochastically drive the samples towards transition states, thus reducing the trapping effects that plague classical MCMC sampling. The cost modifications use gradient and curvature information to focus the search on local neighborhoods that have transition-state-like characteristics. Sminchisescu *et al* [48] give modified MCMC methods that speed up sampling from the equilibrium distribution by including long-range jumps based on prior knowledge of the function's dominant minima (which could be provided, *e.g.*, by the algorithms presented in this paper). Methods that search thoroughly over the static model-image matching cost are an effective basis for very general trackers. Sminchisescu & Jepson [42] propose a smoothing algorithm that computes multiple plausible trajectories for weakly identifiable nonlinear dynamical systems (like the ones resulting from 3d human modeling and tracking) where the multiplicity of solutions in the model-image matching cost persists over temporal states. Therefore, the trajectory distribution remains multimodal after both filtering and smoothing. Ambiguities can be resolved to a certain extent using prior knowledge. Sminchisescu & Jepson [41] use joint angle training sets typical of various application domains to learn constrained low-dimensional generative models using non-linear embedding. Because the learned representations are global and continuous, methods like the ones proposed here can be used for efficient search in a lower-dimensional space.

8.2 Human Modeling

Here, we very briefly review the human model, priors, and image features that we use for the below experiments. For more details, see [43, 40, 46].

Representation $\mathbf{x}$: The 3D body model used in the human pose and motion estimation experiments here consists of a kinematic 'skeleton' of articulated joints controlled by angular joint parameters, covered by a 'flesh' built from superquadric ellipsoids with additional global deformations. For the

experiments here we estimate typically 32 joint parameters.

Observation Likelihood $p(\mathbf{r}|\mathbf{x})$: Robust model-to-image matching cost metrics are evaluated for each predicted image feature, and the results are summed over all observations to produce the image contribution to the parameter space cost function. We use a robust combination of extracted-feature-based metrics and intensity-based ones such as optical flow and robustified normalized edge energy. We also give results for a simpler joint correspondence likelihood designed for model initialization, based on squared distances between reprojected model joints and their specified image positions.

Prior Constraints $p_s(\mathbf{x})$: The model incorporates both hard constraints (for joint angle limits) and soft priors for collision avoidance between body parts. These ensure the parameter estimates remain in the feasible region.

Estimation: We apply Bayes rule and minimize the total negative log-likelihood posterior probability, to give multiple locally MAP parameter estimates:

$$\log p(\mathbf{x}|\mathbf{r}) \;\propto\; \log p_s(\mathbf{x}) + \log p(\mathbf{r}|\mathbf{x}) \;=\; \log p_s(\mathbf{x}) - \sum e(\mathbf{r}_i|\mathbf{x}) \tag{21}$$

Here, $p_s(\mathbf{x})$ is the prior on the model parameters that ensures feasibility, $e(\mathbf{r}_i|\mathbf{x})$ is the cost density associated with observation i, and the sum is over all observations (assumed independent).

Equation (21) gives the model likelihood in a single image, under the model constraints but without initial state or temporal priors[3]. Adopting the Bayesian tracking framework, the temporal prior at time t is determined by the previous posterior $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ and the system dynamics $p_d(\mathbf{x}_t|\mathbf{x}_{t-1})$, where we have collected the observations at time t into vector $\mathbf{r}_t$ and defined $\mathbf{R}_t = \{\mathbf{r}_1, \ldots, \mathbf{r}_t\}$. The posterior at t becomes:

$$p(\mathbf{x}_t|\mathbf{R}_t) \;\propto\; p(\mathbf{r}_t|\mathbf{x}_t)\, p_s(\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p_d(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1}) \tag{22}$$

Together $p_d(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ form the time t prior $p(\mathbf{x}_t|\mathbf{R}_{t-1})$ for the image correspondence search (21). The integral on the r.h.s. of (22) is approximated as a mixture distribution of MAP estimates given by the local optima and their covariances from (21). See [46] for details on the approximation and [48] for methods to speed-up MCMC sampling from the equilibrium distribution using long-range jumps based on prior knowledge about minima structure.

8.3 Human Pose Estimation and Inter-Frame Tracking Experiments

Here we show examples from a set of experiments on locating local minima for pose estimation and inter-frame tracking in monocular images, using cost surfaces based on various different combinations of image cues and a 32 d.o.f. articulated human body model. The figures show examples of minima found in likelihood models based on image contours and optical flow (fig. 9), contours and silhouette-image data (fig. 11 – note that some of these minima can be removed using more complex

[3] Local optima of the current observation cost are obtained using the prior initialization followed by robust constraint-consistent local optimization, as described in [43, 40, 46].

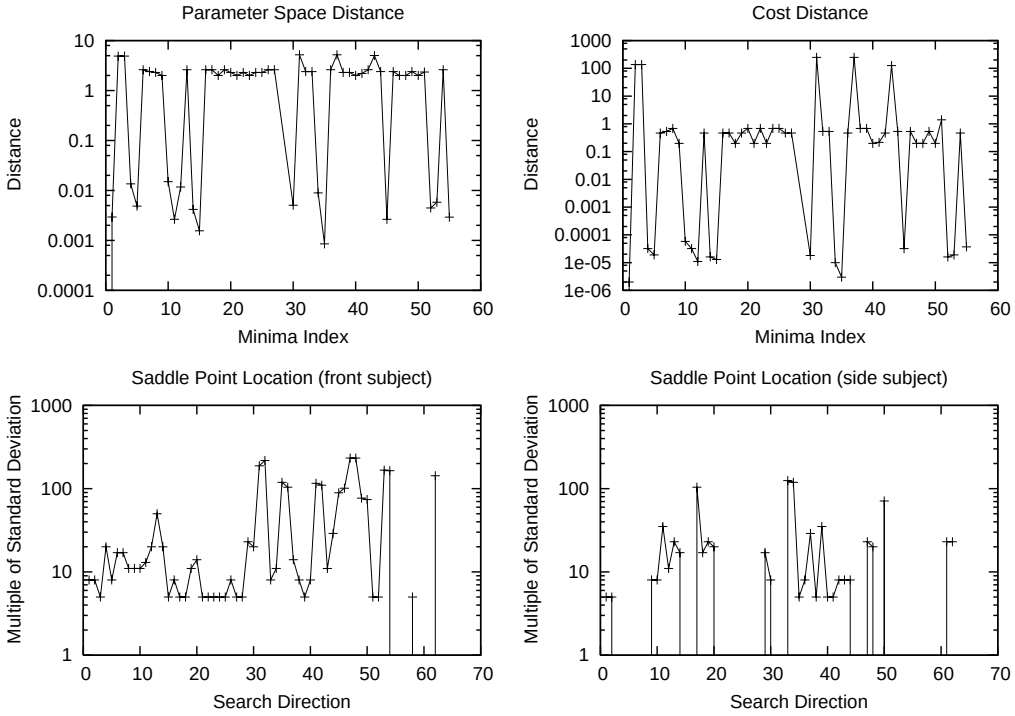


Figure 7: Summary of typical search results from a given minimum along ± 32-eigendirections. Top row: parameter space distance and cost difference between the initial minimum and its discovered neighbors. Second row: initial minimum to transition state distances in standard deviations for a frontal pose with hypersurface sweeping, and a partly side-on one with eigenvector tracking. (See figs 12 and 13 for the corresponding for visual results).

silhouette-based cost functions, as proposed in [39]), and model-to-image joint correspondences (figs 12 and 13). We also give more extensive quantitative results in table 1. In each case, the model was initialized in a minimum found using the local optimization algorithm from [43, 40, 46], and transition state searches were initiated along its principal curvature directions. For each transition state found, local descent [43, 40, 46] was used to find the corresponding neighboring minimum. The algorithm was globalized by repeating the process from the new minima found (checking for duplicates), until either no new minima are found or the permitted total run time has elapsed.

Fig. 7 captures some more quantitative information about the methods, here for the joint correspondence cost function (see §8.2). This problem is well adapted to illustrating the algorithm, as its cost surface is highly multimodal. Of the 32 kinematic d.o.f., about 10 are subject to 'reflective' ambiguities (forwards *vs*. backwards slant in depth). This potentially creates around $2^{10} = 1024$ local minima in the cost surface [27], although some of these are physically infeasible and hence forbidden in any constraint-consistent optimization. Indeed, given the large number of minima that exist, we find that it is very difficult to ensure initialization to the 'correct' pose with this kind of data. The first row in fig. 7 displays the parameter space and cost distances of the 56 minima found

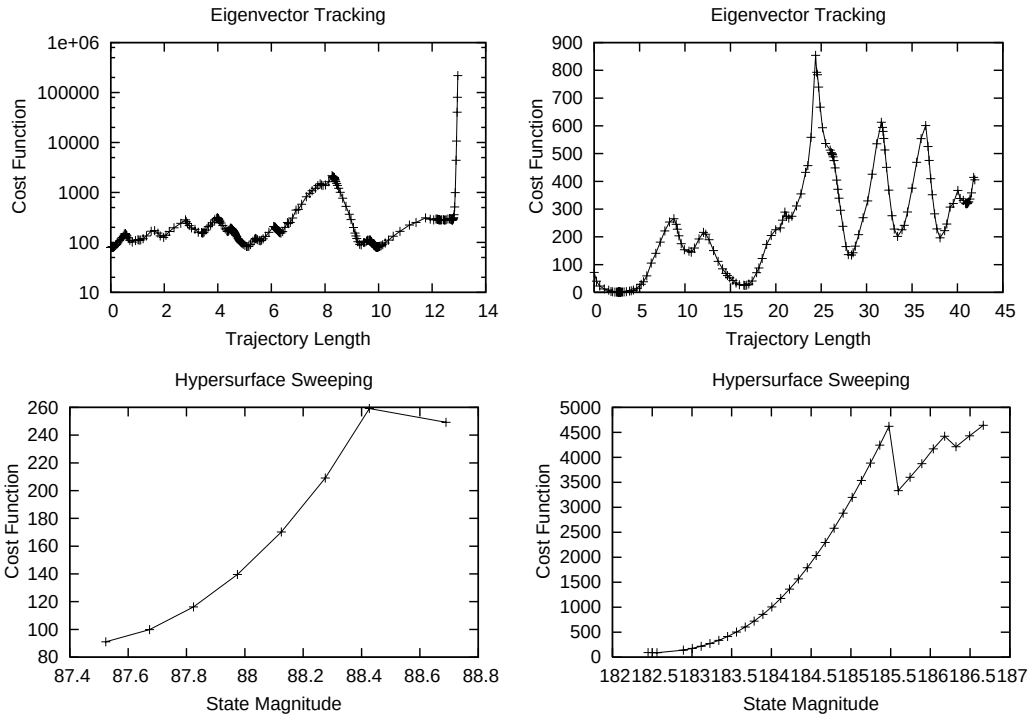


Figure 8: Top row: typical cost profiles for eigenvector tracking with soft (left) and hard (right) joint angle constraints. Soft constraints often lead to 'wall-climbing' divergences when the search hits a joint limit. The active-set projection strategy used in the hard-constraint tracker alleviates this, but introduces abrupt variations of the path direction whenever a constraint is hit, which cause the cost derivative to vary abruptly. Second row: The hyper-ellipsoid method does not require special joint limit processing — the joint constraints can just be included with the hypersurface constraint — but its cost profiles have characteristic 'sawtooth' transitions whenever the tracked minimum becomes locally unstable on the hypersurface.

during a set of 64 constrained searches (the $\pm$ directions of the 32 eigenvectors of an initial minimum, distances being measured w.r.t. this minimum, in radians and meters for the parameter space). The second row again shows parameter space distances, but now measured in standard deviations and for saddles rather than minima, for the same frontal view and for a slightly more side-on one (fig. 12 and fig. 13). The plots reveal the structure of the cost surface, with nearby saddles at 4–10 standard deviations and progressively more remote ones at 20–50, 80–100 and 150–200 standard deviations. It follows that no multiple-minimum exploration algorithm can afford to search only within the 'natural' covariance scale of its current minima: significantly deeper sampling is needed to capture even nearby minima (as previously noted, *e.g.* by [43, 46]).

In fig. 8 we show sample cost profiles for typical runs of joint-constrained eigenvector following (top) and hypersurface sweeping (bottom) search. In the eigenvector method, it is preferable to represent the joint limits using a 'hard' active set strategy by projecting the tracking direction

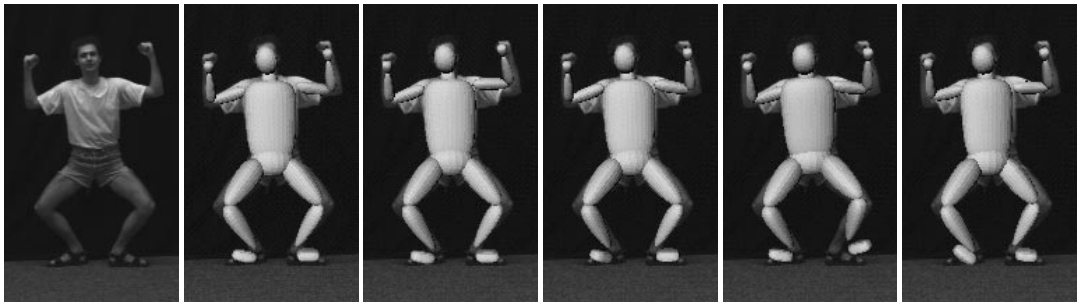


Figure 9: Minima of image-based cost functions: contour and optical flow likelihood. The model is initialized in one minimum (second figure), and search trajectories for other minima (along different principal curvature directions) are initiated from there. The other figures show some of the other minima found. These correspond to incorrect contour assignments or to configurations where the intensity robustifiers turn off (see text).

onto the active constraint set (row 1 right) rather than soft constraints (row 1 left): the stiff 'cost walls' induced by the soft constraints tend to force the eigenvector follower into head-on collision with the wall, with the cost climbing rapidly to infinity. The active set strategy avoids this problem at the price of more frequent direction changes as the joint limits switch on and off (row 1 right). The hyper-ellipsoid method (row 2) produces trajectories that do not require special joint limit processing, but its cost profiles have characteristic sawtooth edges (row 2 right) associated with sudden state readjustments on the hypersphere at points where the tracked minimum becomes locally unstable[4].

Figs 9 and 11 show minima for costs based on various combinations of image cues. In fig. 9 the minima correspond to a small interframe motion, using contour and robust optical flow information. This case has relatively few, but closely spaced local minima owing to the smoothing/quadratic effect of the flow. (Remoter minima do still exist at points where the robust contributions of sets of flow measurements turn off, particularly when these coincide with incorrect edge assignments). Fig. 11 shows minima arising from a silhouette and edge based cost function. To illustrate why such minima occur, we first run a toy experiment [39], that moves a model forearm over an image forearm — see fig. 10. As one can see in the bottom row, the cost has multiple minima, essentially owing to the attribution of both model edges to different sections of the same image edge. Fig. 11 shows how such incorrect limb assignments affect the full model. The minima shown include 'reflective' (depth-related) ambiguities, incorrect edge assignments and singular 'inside-silhouette' configurations (some of these can be alleviated to some extent by augmenting the contour and silhouette likelihood terms as in [39]).

[4]Note the different sources of trajectory instabilities in the two methods: in eigenvector tracking they are due to the projection of the current trajectory step onto the joint-constraint surface, while for hypersurface sweeping, the joint constraints can be enforced alongside the hypersurface one but there are instabilities due to topological transitions of within-constraint minima.

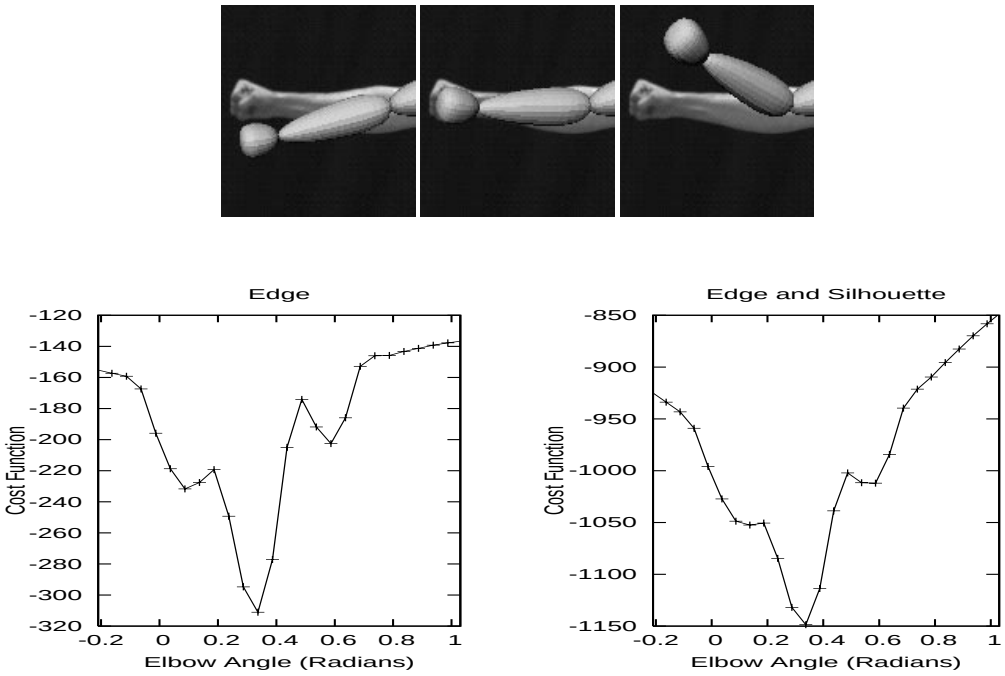


Figure 10: Cost Function Experiment. The elbow joint is varied as shown in the top row, and the corresponding edge cost is monitored. The corresponding cost function profile is multi-modal, with characteristic 'shoulders' in which one edge of the image limb contributes to the matching cost of the opposite edge of the model limb. These persist even when further visual cues (here a silhouette term) are fused into the cost.

Figs 12 and 13 show depth/pose ambiguities for frontal and half-profile views, under the model to image joint correspondence cost function (which is not subject to image matching ambiguities). The flexible and hard-to-observe arm-shoulder complex tends to induce more minima than the legs. We also find that profile views (fig. 13) tend to generate fewer minima than frontal ones (fig. 12), perhaps due to presence of body-part non-self-intersection and joint constraints that render many 'purely reflective' minima infeasible.

Finally, to provide a more quantitative evaluation of the minimum finding efficiency of the eigenvector-tracking and hypersurface-sweeping algorithms, we selected 6 initial minima for a frontal view and 6 more for a half-profile view, using the joint correspondence cost function and our 32 d.o.f. body model. Starting from these configurations, we initiated transition state searches along the ± 32-eigendirections at each minimum, giving a total of $12 \times 64 = 768$ search trials for each algorithm. In this experiment we also set an upper-bound of 50 iterations per search (the search is counted as a failure if a saddle has not been found at this point). Table 1 reports the number of minima found by each method, and the medians and standard deviations of their parameter space distances and function values. Both methods locate neighboring minima with good success rates.

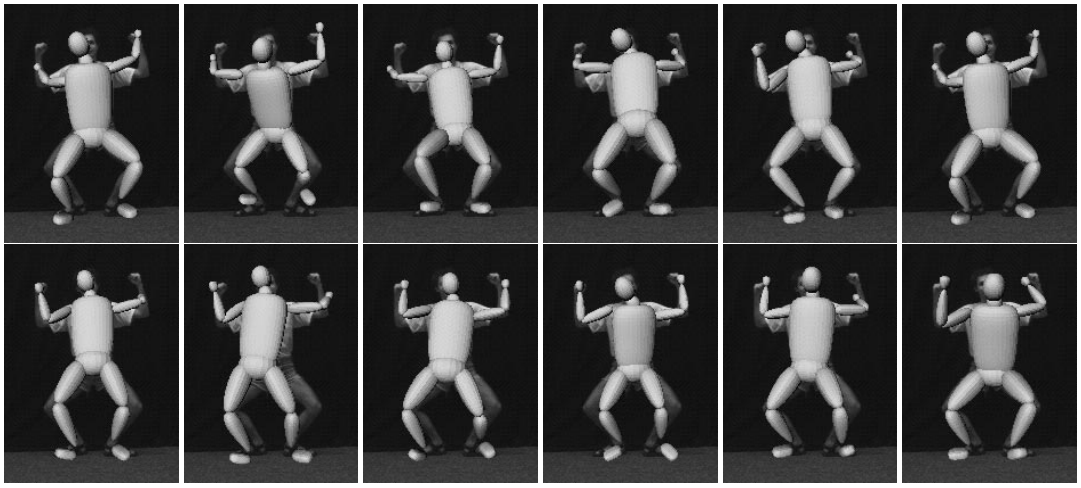


Figure 11: Minima of image-based cost functions: contour and silhouette likelihood. The model is initialized at one minimum (fig. 9, second figure) and search trajectories for other minima are initiated from there. The minima shown include 'reflective' (depth-related) ambiguities, incorrect edge assignments and singular 'inside-silhouette' configurations. More complex silhouette-based cost functions, as proposed in [39] can be further used eliminate some of the latter spurious minima (see text).

In this experiment, hypersurface sweeping locates a few more minima than eigenvector tracking (60% *vs*. 55%), but also finds slightly more remote and higher cost minima. When successful, the eigenvector tracking method typically needs around 17–19 iterations to locate a transition state, whereas hypersurface sweeping needs about 14–16. In contrast, once a transition state is located, local descent finds the neighboring minimum in around 7–10 iterations.

Computationally, the cost of an iteration is determined by function evaluation costs and by the cost of Hessian manipulation for each method. These in turn depend on the number of measurements and the dimension of the parameter space, respectively. ET needs eigendecomposition of the Hessian which is about 4-5 times more expensive than the Cholesky decomposition required for a local descent iteration. HS needs Hessian eigendecomposition only for the initial selection of the hypersurface shape, and any subsequent step involves a local optimization that in practice, we found, converged in about 3-5 iterations. For the articulated pose problem studied here, we also found that the above differences are dominated by the cost function evaluation, the manipulation of the Hessian being comparatively fast for 32 dimensions. For example, on a 2.2Ghz Pentium processor, the average iteration cost for the simple joint correspondence cost function was about 0.14s for local descent, and about 0.31s for ET and 0.62s for HS. For more expensive image-based cost functions, the timings were 1.21s for local descent, 1.67s for ET and 4.42s for HS.

Figure 12: ‘Reflective’ kinematic ambiguities under the model/image joint correspondence cost function in a frontal view. The model is initialized at one minimum (second row, leftmost figure), and search trajectories are initiated for other minima along different principal curvature directions. The images show some of the new minima found, from the original camera viewpoint (superposed on the source image) and from a synthetic overhead viewpoint. Note the pronounced forwards-backwards character of these reflective minima, and the large parameter space distances that often separate them.

9 Conclusions and Research Directions

This paper has described two classes of deterministic, local optimization based algorithms for finding ‘transition states’ (saddle points with 1 negative eigenvalue), and hence neighboring minima, of high-dimensional cost functions with multiple minima. These methods allow us to build topological ‘roadmaps’ of the nearby local minima and the transition states that lead to them. They are based on methods developed in computational chemistry, but here generalized, clarified and adapted for use in computational vision. Experiments on the difficult problem of articulated 3D human pose from monocular images show that our algorithms can stably and efficiently recover large numbers

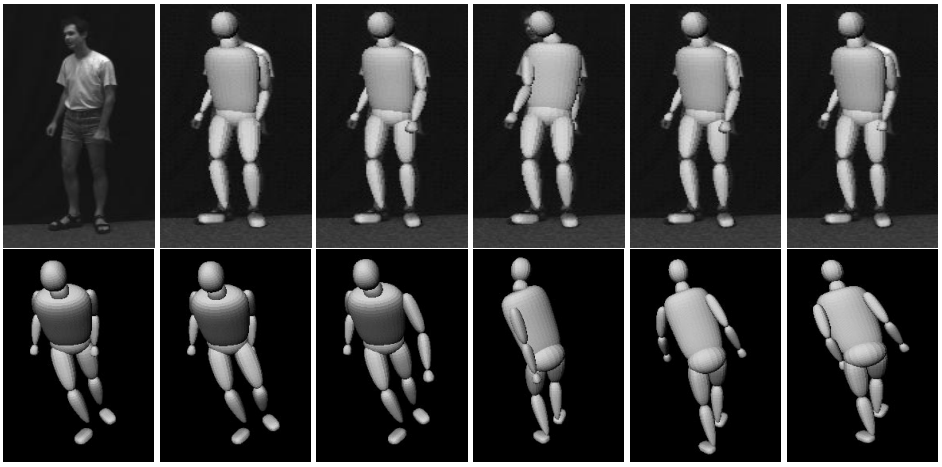


Figure 13: ‘Reflective’ kinematic ambiguities under the model/image joint correspondence cost function, for a half-profile view. For explanation, see fig. 12 caption.

METHOD	NUMBER OF DETECTED MINIMA	MEDIAN PARAMETER DISTANCE	MEDIAN STANDARD DEVIATIONS	MEDIAN COST
ET	421 (55%)	4.45	79.6	3.39
HS	457 (60%)	5.01	91.6	3.87

Table 1: Quantitative results for the distribution of minima found by eigenvector tracking and hypersurface sweeping. The results represent a total of 768 local searches (forwards and backwards along the 32-eigendirections of 6 frontal and 6 half-profile minima). Hypersurface sweeping finds 9% more minima than eigenvector tracking, but they are about 12–15% further away and have about 14% higher cost on average.

of nearby transition states and minima, but also serve to underline the very large numbers of minima that exist in this problem.

Our methods should be useful for many other minimum-rich problems in vision, including structure from motion. They could also potentially be used to quantify the degree of ambiguity of different cost functions, which in the long term may aid the design of less ambiguous cost functions based on higher-level features and groupings.

Although the current methods are a great improvement over previous ones, they are not foolproof and much remains to be done in this area. Damped Newton iteration is useful for refining estimated saddle points but its convergence domain is too limited for general use. Eigenvector tracking extends the convergence domain but can be sensitive to the ‘same eigenvector’ heuristic used. Hypersurface sweeping is better founded in that it provides some guarantee of global progress, but no single sweep finds all saddle points and it is more complex to implement. Future research directions include deriving heuristics to select promising up-hill directions for search initialization,

and an analysis of the benefits of different types of hypersurfaces and of ways to adaptively evolve them based on the local cost function structure. It should be also interesting to explore the use of local saddle point moves as alternatives to less informed long range Markov-chain steps in stochastic simulations.

Acknowledgements

This work was begun while the first author was a PhD student at INRIA Rhône-Alpes. It was supported by an Eiffel Scholarship, the EU R&D project VIBES. We thank Alexandru Telea at Eindhoven University of Technology, for generous help with the visualization tools used to display the results in this paper.

References

[1] Y. Abashkin and N. Russo. Transition State Structures and Reaction Profiles from Constrained Optimization Procedure. Implementation in the Framework of Density Functional Theory. *J. Chem. Phys.*, 1994.

[2] Y. Abashkin, N. Russo, and M. Toscano. Transition States and Energy Barriers from Density Functional Studies: Representative Isomerization Reactions. *International Journal of Quantum Chemistry*, 1994.

[3] N. Anderson and G. R. Walsh. A Graphical Method for a Class of Branin Trajectories. *Journal of Optimization Theory and Applications*, 1986.

[4] G. T. Barkema. Event-Based Relaxation of Continuous Disordered Systems. *Physical Review Letters*, 77(21), 1996.

[5] J. M. Bofill. Updated Hessian Matrix and the Restricted Step Method for Locating Transition Structures. *Journal of Computational Chemistry*, 15(1):1–11, 1994.

[6] F. Branin and S. Hoo. A Method for Finding Multiple Extrema of a Function of n Variables. *Numerical Methods of Nonlinear Optimization*, 1972.

[7] C. J. Cerjan and W. H. Miller. On Finding Transition States. *J. Chem. Phys.*, 75(6), 1981.

[8] A. Chiuso, R. Brockett, and S. Soatto. Optimal Structure from Motion: Local Ambiguities and Global Estimates. *International Journal of Computer Vision*, 39(3):195–228, 2000.

[9] K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *IEEE International Conference on Computer Vision*, 2001.

[10] G. M. Crippen and H. A. Scheraga. Minimization of Polypeptide Energy. XI. The Method of Gentlest Ascent. *Archives of Biochemistry and Biophysics*, 144:462–466, 1971.

[11] P. Culot, G. Dive, V. H. Nguyen, and J. M. Ghuysen. A Quasi-Newton Algorithm for First-Order Saddle Point Location. *Theoretica Chimica Acta*, 82:189–205, 1992.

[12] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

[13] J. Deutscher, A. Davidson, and I. Reid. Articulated Partitioning of High Dimensional Search Spacs associated with Articulated Body Motion Capture. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[14] I. Fiodorova. Search for the Global Optimum of Multiextremal Problems. *Optimal Decision Theory*, 1978.

[15] R. Fletcher. Practical Methods of Optimization. In *John Wiley*, 1987.

[16] D. Gavrila and L. Davis. 3-D Model Based Tracking of Humans in Action:A Multiview Approach. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 73–80, 1996.

[17] R. Ge. The Theory of the Filled Function Method for Finding a Global Minimizer of a Nonlinearly Constrained Minimization Problem. *J. of Comp. Math.*, 1987.

[18] F. Glover. Tabu search - part II. *ORSA Journal of Computing 2*, pages 4–32, 1990.

[19] A. Goldstein and J. Price. On Descent from Local Minima. *Mathematics of Computation*, 1971.

[20] A. Griewank. Generalized Descent for Global Optimization. *Journal of Optimization Theory and Applications*, 1981.

[21] T. Helgaker. Transition-State Optimizations by Trust-Region Image Minimization. *Chemical Physics Letters*, 182(5), 1991.

[22] G. Henkelman and H. Jonsson. A Dimer Method for Finding Saddle Points on High Dimensional Potential Surfaces Using Only First Derivatives. *J. Chem. Phys.*, 111(15):7011–7022, 1999.

[23] R. L. Hilderbrandt. Application of Newton-Raphson Optimization Techniques in Molecular Mechanics Calculations. *Computers & Chemistry*, 1:179–186, 1977.

[24] F. Jensen. Locating Transition Structures by Mode Following: A Comparison of Six Methods on the Ar_8 Lennard-Jones potential. *J. Chem. Phys.*, 102(17):6706–6718, 1995.

[25] P. Jorgensen, H. J. A. Jensen, and T. Helgaker. A Gradient Extremal Walking Algorithm. *Theoretica Chimica Acta*, 73:55–65, 1988.

[26] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 1983.

[27] H. J. Lee and Z. Chen. Determination of 3D Human Body Postures from a Single View. *Computer Vision, Graphics and Image Processing*, 30:148–168, 1985.

[28] A. Levy and A. Montalvo. The Tunneling Algorithm for the Global Minimization of Functions. *SIAM J. of Stat. Comp.*, 1985.

[29] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 289–296, 1998.

[30] N. Mousseau and G. T. Berkema. Traveling Through Potential Energy Lanscapes of Desordered Materials: The Activation-Relaxation Technique. *Physical Review E*, 57(2), 1998.

[31] L. J. Munro and D. J. Wales. Defect Migration in Crystalline Silicon. *Physical Review B*, 59(6):3969–3980, 1999.

[32] R. Neal. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 2001.

[33] J. Nichols, H. Taylor, P. Schmidt, and J. Simons. Walking on Potential Energy Surfaces. *J. Chem. Phys.*, 92(1), 1990.

[34] J. Oliensis. The Error Surface for Structure from Motion. Technical report, NECI, 2001.

[35] E. M. Sevick, A. T. Bell, and D. N. Theodorou. A Chain of States Method for Investigating Infrequent Event Processes Occuring in Multistate, Multidimensional Systems. *J. Chem. Phys.*, 98(4), 1993.

[36] H. Sidenbladh, M. Black, and D. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *European Conference on Computer Vision*, 2000.

[37] H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conference on Computer Vision*, 2002.

[38] J. Simons, P. Jorgensen, H. Taylor, and J. Ozmen. Walking on Potential Energy Surfaces. *J. Phys. Chem.*, 87:2745–2753, 1983.

[39] C. Sminchisescu. Consistency and Coupling in Human Model Likelihoods. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 27–32, Washington D.C., 2002.

[40] C. Sminchisescu. *Estimation Algorithms for Ambiguous Visual Models–Three-Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences*. PhD thesis, Institute National Politechnique de Grenoble (INRIA), July 2002.

[41] C. Sminchisescu and A. Jepson. Generative Modeling for Continuous Non-Linearly Embedded Visual Inference. In *International Conference on Machine Learning*, Banff, 2004.

[42] C. Sminchisescu and A. Jepson. Variational Mixture Smoothing for Non-Linear Dynamical Systems. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Washington D.C., 2004.

[43] C. Sminchisescu and B. Triggs. Covariance-Scaled Sampling for Monocular 3D Body Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 447–454, Hawaii, 2001.

[44] C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. In *European Conference on Computer Vision*, volume 1, pages 566–582, Copenhagen, 2002.

[45] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *European Conference on Computer Vision*, volume 1, pages 769–783, Copenhagen, 2002.

[46] C. Sminchisescu and B. Triggs. Estimating Articulated Human Motion with Covariance Scaled Sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[47] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 69–76, Madison, 2003.

[48] C. Sminchisescu, M. Welling, and G. Hinton. A Mode-Hopping MCMC Sampler. Technical Report CSRG-478, University of Toronto, submitted to *Machine Learning Journal*, September 2003.

[49] J. Q. Sun and K. Ruedenberg. Gradient Extremals and Stepest Descend Lines on Potential Energy Surfaces. *J. Chem. Phys.*, 98(12), 1993.

[50] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In Springer-Verlag, editor, *Vision Algorithms: Theory and Practice*, 2000.

[51] A. F. Voter. A Method for Accelerating the Molecular Dynamics Simulation of Infrequent Events. *J. Chem. Phys.*, 106(11):4665–4677, 1997.

[52] A. F. Voter. Hyperdynamics: Accelerated Molecular Dynamics of Infrequent Events. *Physical Review Letters*, 78(20):3908–3911, 1997.

[53] D. J. Wales. Finding Saddle Points for Clusters. *J. Chem. Phys.*, 91(11), 1989.

[54] D. J. Wales and T. R. Walsh. Theoretical Study of the Water Pentamer. *J. Chem. Phys.*, 105(16), 1996.