Lucas-Kanade 20 Years On: A Unifying Framework: Part 4

Simon Baker, Ralph Gross and Iain Matthews

CMU-RI-TR-04-14

Lucas-Kanade 20 Years On: A Unifying Framework: Part 4

Simon Baker, Ralph Gross, and Iain Matthews

CMU-RI-TR-04-14

Abstract

Since the Lucas-Kanade algorithm was proposed in 1981 image alignment has become one of the most widely used techniques in computer vision. Applications range from optical flow, tracking, and layered motion, to mosaic construction, medical image registration, and face coding. Numerous algorithms have been proposed and a variety of extensions have been made to the original formulation. We present an overview of image alignment, describing most of the algorithms in a consistent framework. We concentrate on the *inverse compositional* algorithm, an efficient algorithm that we recently proposed. We examine which of the extensions to the Lucas-Kanade algorithm can be used with the inverse compositional algorithm without any significant loss of efficiency, and which cannot. In this paper, the fourth and final part in the series, we cover the addition of priors on the parameters. We first consider the addition of priors on the warp parameters. We show that priors can be added with minimal extra cost to all of the algorithms in Parts 1–3. Next we consider the addition of priors on both the warp and appearance parameters. Image alignment with appearance variation was covered in Part 3. For each algorithm in Part 3, we describe whether priors can be placed on the appearance parameters or not, and if so what the cost is.

Keywords: Image alignment, unifying framework, the Lucas-Kanade algorithm, the inverse compositional algorithm, priors on the parameters, linear appearance variation, robust error functions.

1 Introduction

Image alignment consists of moving, and possibly deforming, a template to minimize the difference between the template and an image. Since its first use in the Lucas-Kanade algorithm [16], image alignment has become one of the most widely used techniques in computer vision. Besides optical flow, some of its other applications include tracking [8,14], parametric and layered motion estimation [7], mosaic construction [19], medical image registration [9], and face coding [10,17].

The usual approach to image alignment is gradient descent. A variety of other numerical algorithms have also been proposed [13], but gradient descent is the defacto standard. We propose a unifying framework for image alignment, describing the various algorithms and their extensions in a consistent manner. Throughout the framework we concentrate on the *inverse compositional* algorithm, an efficient algorithm that we recently proposed [3,5]. We examine which of the extensions to the Lucas-Kanade algorithm can be applied to the inverse compositional algorithm without any significant loss of efficiency, and which extensions require additional computation.

In this paper, the fourth and final part in the series, we cover the addition of priors on the parameters. We begin in Section 3 by considering priors on the warp parameters. We first show how to add priors on the inefficient Lucas-Kanade algorithm. We then proceed to show how the efficient inverse compositional algorithm [5] can be extended to allow priors on the parameters with minimal extra cost. We then briefly outline how the same approach can be used for all of the other algorithms in Parts 1-3 [1,2,4]. We end by describing a variety of applications.

In Section 4 we cover the addition of priors on both the warp and appearance parameters. Image alignment with linear appearance variation was covered in Part 3 [1] of this series. We begin Section 4 by describing how the simultaneous inverse compositional algorithm [1] can be extended to allow priors on both the warp and appearance parameters. For each algorithm in Part 3, we then describe whether priors can be placed on the appearance parameters or not, and if so what

the cost is. We end by describing a few applications of priors on the appearance parameters.

2 Review of Parts 1-3

2.1 Background: The Lucas-Kanade Algorithm

The original image alignment algorithm was the Lucas-Kanade algorithm [16]. The goal of Lucas-Kanade is to align a template image T(x) to an input image /(x), where x = (ar, *y)*^{T} is a column vector containing the pixel coordinates. If the Lucas-Kanade algorithm is being used to track an image patch from time *t* = 1 to time *t* — 2, the template T(x) is an extracted sub-region (a 64 x 64 window, maybe) of the image at *t* = 1 and /(x) is the image at *t* = 2.

Let W(x;p) denote the parameterized set of allowed warps, where p = ($p_{1?}$.. .p_n)^T is a vector of parameters. The warp W(x; p) takes the pixel x in the coordinate frame of the template *T* and maps it to the sub-pixel location W(x; p) in the coordinate frame of the image /. If we are tracking a large image patch moving in 3D we may consider the set of affine warps:

$$W(x;P) = \begin{pmatrix} (1+p_1)\cdot x + p_3\cdot y + p_5 \\ p_2\cdot x + (1+p_4)\cdot y + p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where there are 6 parameters p = *(pi,P2,Ps,P4,P$,PG)*^{T} as, for example, was done in [7]. In general, the number of parameters *n* may be arbitrarily large and W(x; p) can be arbitrarily complex. One example of a complex warp is the set of piecewise affine warps used in [3,10,18].

2.1.1 Goal of the Lucas-Kanade Algorithm

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template *T* and the image *I* warped back onto the coordinate frame of the template:

$$\sum_x [r(x) - /(W(x;p))]^2. \quad (2)$$

Warping 7 back to compute 7(W(x; p)) requires interpolating the image 7 at the sub-pixel locations W(x;p). The minimization in Equation (2) is performed with respect to p and the sum is performed over all of the pixels **x** in the template image T(x). Minimizing the expression in Equation (2) is a non-linear optimization even if W(x;p) is linear in p because the pixel values 7(x) are, in general, non-linear in x. In fact, the pixel values 7(x) are essentially un-related to the pixel coordinates x. To optimize the expression in Equation (2), the Lucas-Kanade algorithm assumes that a current estimate of p is known and then iteratively solves for increments to the parameters Ap; i.e. the following expression is (approximately) minimized:

$$\sum_{\mathbf{x}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathrm{p} + \mathrm{Ap}))]^2 \tag{3}$$

with respect to Ap, and then the parameters are updated:

$$\mathrm{p} \leftarrow \mathrm{p} + \mathrm{Ap}. \tag{4}$$

These two steps are iterated until the estimates of the parameters p converge. Typically the test for convergence is whether some norm of the vector Ap is below a threshold e; i.e. $||\mathbf{Ap}|| \leq$ e.

2.1.2 Derivation of the Lucas-Kanade Algorithm

The Lucas-Kanade algorithm (which is a Gauss-Newton gradient descent non-linear optimization algorithm) is then derived as follows. The non-linear expression in Equation (3) is linearized by performing a first order Taylor expansion of **7(W(x;** p + Ap)) to give:

$$\sum_{x} \left[\mathbf{r}(\mathbf{x}) - \mathbf{J}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \nabla \mathbf{J} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \mathbf{Ap} \right]^2 . \tag{5}$$

In this expression, V7 = (|^, $\frac{\partial J}{\partial}$ *s the *gradient* of image 7 evaluated at W(x;p); i.e. V7 is computed in the coordinate frame of 7 and then warped back onto the coordinate frame of *T* using

the current estimate of the warp W(x;p). (We follow the notational convention that the partial derivatives with respect to a column vector are laid out as a row vector. This convention has the advantage that the chain rule results in a matrix multiplication, as in Equation (5).) The term $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$^ is the *Jacobian* of the warp. IfW(x;p) = $(W_x(x;\ \mathrm{p})_7\ W_y(x;p))^T$ then:

$$\frac{dW}{dp} = \begin{pmatrix} \frac{dW_x}{dpi} & \frac{dW_x}{\partial p_2} & \ldots & \frac{dw_x}{dp_n} \\ \frac{dWy}{dpi} & \frac{OWy}{dp_2} & \ldots & \frac{dw_y}{dp_n} \end{pmatrix}. \tag{6}$$

For example, the affine warp in Equation (1) has the Jacobian:

$$\frac{dW}{dp} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}. \tag{7}$$

Equation (5) is a least squares problem and has a closed from solution which can be derived as follows. The partial derivative of the expression in Equation (5) with respect to Ap is:

$$-2\sum_{\mathbf{x}} \text{V/-^-j}^{T} \ \text{|T(x)-/(W(x;p))-V/^Apj.} \tag{8}$$

Then denote:

$$SD_{lk}(x) = \text{V / ^}\ dp, \tag{9}$$

the *steepest descent* images. Setting the expression in Equation (8) to equal zero and solving gives the closed form solution of Equation (5) as:

$$Ap = \sum_{\mathbf{x}} \text{ffi^SDftxJEfx)} \tag{10}$$

where iJ_{lk} is the n x n (Gauss-Newton approximation to the) *Hessian* matrix:

$$H_{lk} = \text{^}\sum_{x} \text{T SD^SD^x)} \tag{11}$$

The Lucas-Kanade Algorithm

Iterate:

(1) Warp I with $W(x; p)$ to compute $7(W(x; p))$
(2) Compute the error image $i£(x)$ using Equation (12)
(3) Warp the gradient VI with $W(x; p)$
(4) Evaluate the Jacobian $\hat{}$ at $(x; p)$
(5) Compute the steepest descent images $SDi_k(x)$ using Equation (9)
(6) Compute the Hessian matrix $\#i_k$ using Equation (11)
(7) Compute $£_x$ $SDj(x)E(x)$
(8) Invert the Hessian and compute $Ap = H\hat{}\}$ $£_x$ $SDj£(x)i?(x)$
(9) Update the parameters $p \ll- p$ -f Ap

until $||Ap|| \leq e$

Figure 1: The Lucas-Kanade algorithm [16] consists of iteratively applying Equations (10) & (4) until the estimates of the parameters p converge. Typically the test for convergence is whether some norm of the vector Ap is below a user specif] ed threshold e. Because the gradient V7 must be evaluated at $W(x; p)$ and the Jacobian $\hat{}$ must be evaluated at p, all 9 steps must be repeated in every iteration of the algorithm.

and:

$$E(x) = T(x)-/(W(x;p)) \tag{12}$$

is the *error image.* The Lucas-Kanade algorithm, summarized in Figure 1, consists of iteratively applying Equations (10) and (4). Because the gradient VJ must be evaluated at $W(x; p)$ and the Jacobian $\hat{}\frac{W}{p}$ at p, they both depend on p. In general, therefore, both the steepest-descent images and the Hessian must be recomputed in every iteration of the algorithm. See Figure 1.

2.1.3 Computational Cost of the Lucas-Kanade Algorithm

Assume that the number of warp parameters is *n* and the number of pixels in *T* is *N.* Step 1 of the Lucas-Kanade algorithm usually takes time O(n iV). For each pixel **x** in T we compute $W(x; p)$ and then sample / at that location. The computational cost of computing **W(x;p)** depends on W but for most warps the cost is O(n) per pixel. Step 2 takes time O(7V). Step 3 takes the same time as Step 1, usually *0(nN).* Computing the Jacobian in Step 4 also depends on W but for most warps the cost is O(n) per pixel. The total cost of Step 4 is therefore *0(nN).* Step 5

Table 1: The computational cost of one iteration of the Lucas-Kanade algorithm. If *n* is the number of warp parameters and TV is the number of pixels in the template T, the cost of each iteration is $O(n^2 TV - f n^3)$. The most expensive step by far is Step 6, the computation of the Hessian, which alone takes time $O(n^2 TV)$.

Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Total
O(nTV)	O(TV)	O(nTV)	O(nTV)	O(nTV)	$O(n^2N)$	O(nTV)	$O(n^3)$	O(n)	$O(n^2TV + n^3)$

takes time O(nTV), Step 6 takes time $O(n^2 TV)$, and Step 7 takes time O(nTV). Step 8 takes time $O(n^3)$ to invert the Hessian matrix and time $O(n^2)$ to multiply the result by the steepest descent parameter updated computed in Step 7. Step 9 just takes time O(n) to increment the parameters by the updates. The total computational cost of each iteration is therefore $O(n^2 TV + n^3)$, the most expensive step being Step 6. See Table 1 for a summary of these computational costs.

2.2 Part 1: The Inverse Compositional Algorithm

2.2.1 Goal of the Inverse Compositional Algorithm

As a number of authors have pointed out, there is a huge computational cost in re-evaluating the Hessian in every iteration of the Lucas-Kanade algorithm [11, 14, 19]. If the Hessian were constant it could be precomputed and then re-used. In [5] we proposed the inverse compositional algorithm as a way of reformulating image alignment so that the Hessian is constant and can be precomputed. Although the goal of the inverse compositional algorithm is the same as the Lucas-Kanade algorithm (see Equation (2)) the inverse compositional algorithm iteratively minimizes:

$$\sum_{\mathbf{x}} [\mathbf{T}(\mathbf{W}(\mathbf{x};\mathbf{A}\mathbf{p})) - \mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2 \quad (13)$$

with respect to $\mathbf{A}\mathbf{p}$ and then updates the warp:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\mathbf{A}\mathbf{p})^{-1} . \quad (14)$$

The expression:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\mathbf{A}\mathbf{p}) \equiv \mathbf{W}(\mathbf{W}(\mathbf{x};\mathbf{A}\mathbf{p});\mathbf{p}) \quad (15)$$

is the composition of 2 warps. For example, if W(x; p) is the affine warp of Equation (1) then: W(x;p)oW(x;Ap) =

$$\begin{pmatrix} (1 + pi) \cdot ((1 + Api) \cdot x + Ap_z \cdot y + \mathrm{Ap}_5) + \mathrm{p}_3 \cdot (\mathrm{Ap}_2 \cdot x + (1 + Ap_4) \cdot y + \mathrm{Ap}_6) + p_5 \\ p_2 \cdot ((1 + \mathrm{A}_{\mathrm{P1}}) \cdot x + Ap_3 \cdot y + \mathrm{Ap}_5) + (1 + \mathrm{p}_4) \cdot (Ap_2 \cdot x + (1 + \mathrm{Ap}_4) \cdot \mathrm{y} + \mathrm{Ap}_6) + \mathrm{p}_6 \end{pmatrix}' \tag{16}$$

i.e. the parameters of W(x; p) o W(x; Ap) are:

$$\begin{pmatrix} \mathrm{Pi} + \hat{\mathrm{A}}\mathrm{pi} + \mathrm{pi} \cdot \Delta p_1 + p_3 \cdot \Delta p_2 \\ \mathrm{p}_2 + \mathrm{Ap2} + \mathrm{p}_2 \cdot \Delta p_1 + p_4 \cdot \Delta p_2 \\ p_3 + \Delta p_3 + p_1 \cdot \Delta p_3 + p_3 \cdot \Delta p_4 \\ p_4 + \mathbf{Ap}_4 + p_2 \cdot \Delta p_3 + p_4 \cdot \mathrm{Ap}_4 \\ p_5 + \mathbf{Ap}_5 + \mathbf{px} \cdot \mathbf{Ap}_5 + p_3 \cdot \mathrm{Ap}_6 \\ p_6 + \Delta p_6 + p_2 \cdot \Delta p_5 + p_4 \cdot \mathbf{Ap}_6 \end{pmatrix}', \tag{17}$$

a simple bilinear combination of the parameters of W(x;p) and W(x; Ap). The expression W(x; Ap)""[1] is the inverse of W(x; Ap). The parameters of W(x; Ap)"[1] are:

$$\frac{1}{(1 + \Delta p_1) \cdot (1 + \mathrm{Ap}_4) - Ap_2 \cdot \Delta p_3} \begin{pmatrix} -\Delta p_1 - \Delta p_1 \cdot \Delta p_4 + Ap_2 \cdot \Delta p_3 \\ -Ap_2 \\ -\Delta p_3 \\ -\Delta p_4 - \Delta p_1 \cdot \Delta p_4 + \mathbf{Ap}_2 \cdot \Delta p_3 \\ -\Delta p_5 - \Delta p_4 \cdot \Delta p_5 + \mathbf{Ap}_3 \cdot \Delta p_6 \\ -\mathrm{Ap}_6 - \Delta p_1 \cdot \mathbf{Ap}_6 + \mathbf{Ap}_2 \cdot \mathbf{Ap}_5 \end{pmatrix}. \tag{18}$$

If (1 + Api) - (1 + Ap$_4$) - Ap$_2$ • Ap$_3$ = 0, the affine warp is degenerate and not invertible. All pixels are mapped onto a straight line in J. We exclude all such affine warps from consideration.

The Lucas-Kanade algorithm iteratively applies Equations (3) and (4). The inverse compositional algorithm iteratively applies Equations (13) and (14). Perhaps somewhat surprisingly, these two algorithms can be shown to be equivalent to first order in Ap. They both take the same steps as they minimize the expression in Equation (2). See [4] for the proof of equivalence.

2.2.2 Derivation of the Inverse Compositional Algorithm

Performing a first order Taylor expansion on Equation (13) gives:

$$\sum_{x} \left[T(\mathbf{W}(\mathbf{x};0)) + \nabla T \frac{\partial}{\partial p} \Delta p - I(W(x;p)) \right]^2 . \quad (19)$$

Assuming that $W(x;0)$ is the identity warp, the solution to this least-squares problem is:

$$\mathbf{\Delta p} = -H_{IC}^{-1} \sum_{x} SD_{IC}(x) E(n) \quad \mathbf{(20)}$$

where $SD_{IC}(x)$ are the steepest-descent images with I replaced by T:

$$SD_{IC}(x) = \nabla T \frac{\partial \mathbf{W}}{\partial p}, \quad (21)$$

H_{IC} is the Hessian matrix computed using the new steepest-descent images:

$$H_{IC} = \sum_{x} SD_{IC}(x)^T SD_{IC}(x), \quad (22)$$

and the Jacobian $\frac{\partial \mathbf{W}}{\partial p}$ is evaluated at $(x;0)$. Since there is nothing in either the steepest-descent images or the Hessian that depends on p, they can both be pre-computed. The inverse composition algorithm is summarized in Figures 2 and 3.

2.2.3 Computational Cost of the Inverse Compositional Algorithm

The inverse compositional algorithm is far more computationally efficient than the Lucas-Kanade algorithm. See Table 2 for a summary. The most time consuming steps, Steps 3-6, can be performed once as a pre-computation. The pre-computation takes time $O(n^2 N + n^3)$. The only additional cost is inverting $W(x;\mathbf{\Delta p})$ and composing it with $W(x;p)$. These two steps typically require $O(n^2)$ operations, as for the affine warp in Equations (16) and (18). Potentially these 2

The Inverse Compositional Algorithm

Pre-compute:

(3) Evaluate the gradient ∇T of the template $T(\mathbf{x})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$
(5) Compute the steepest descent images $\mathbf{SD}_{\text{ic}}(\mathbf{x})$ using Equation (21)
(6) Compute the Hessian matrix H_{ic} using Equation (22) and invert it

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
(2) Compute the error image $E(\mathbf{x})$ using Equation (12)
(7) Compute $\sum_{\mathbf{x}} \mathbf{SD}_{\text{ic}}^{\text{T}}(\mathbf{x}) E(\mathbf{x})$
(8) Compute $\Delta \mathbf{p} = -H_{\text{ic}}^{-1} \sum_{\mathbf{x}} \mathbf{SD}_{\text{ic}}^{\text{T}}(\mathbf{x}) E(\mathbf{x})$
(9) Update the warp $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$

until $\|\Delta \mathbf{p}\| \leq \epsilon$

Figure 2: The inverse compositional algorithm [3, 5]. All of the computationally demanding steps are performed once in a pre-computation step. The main algorithm simply consists of image warping (Step 1), image differencing (Step 2), image dot products (Step 7), multiplication with the inverse of the Hessian (Step 8), and the update to the warp (Step 9). All of these steps are effi cient and take time $O(n\,N + n^2)$.

Table 2: The computation cost of the inverse compositional algorithm. The one time pre-computation cost of computing the steepest descent images and the Hessian in Steps 3-6 is $O(n^2\,N + n^3)$. After that, the cost of each iteration is $O(n\,N + n^2)$ a substantial saving over the Lucas-Kanade iteration cost of $O(n^2\,N + n^3)$.

Pre-Computation	Step 3	Step 4	Step 5	Step 6	Total
	$O(N)$	$O(n\,N)$	$O(n\,N)$	$O(n^2\,N + n^3)$	$O(n^2\,N + n^3)$

Per Iteration	Step 1	Step 2	Step 7	Step 8	Step 9	Total
	$O(n\,N)$	$O(N)$	$O(n\,N)$	$O(n^2)$	$O(n^2)$	$O(n\,N + n^2)$

steps could be fairly involved, as in [3], but the computational overhead is almost always completely negligible. Overall the cost of the inverse compositional algorithm is $O(n\,N + n^2)$ per iteration rather than $O(n^2\,N + n^3)$ for the Lucas-Kanade algorithm, a substantial saving.

2.3 Part 2: The Choice of the Error Function

In Part 2 [2] we extended the inverse compositional algorithm to fit with a weighted L2 norm:

$$\sum_{\mathbf{x}} \sum_{\mathbf{y}} Q(\mathbf{x}, \mathbf{y}) \cdot [\,T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))\,] \cdot [\,T(\mathbf{y}) - I(\mathbf{W}(\mathbf{y}; \mathbf{p}))\,] \tag{23}$$

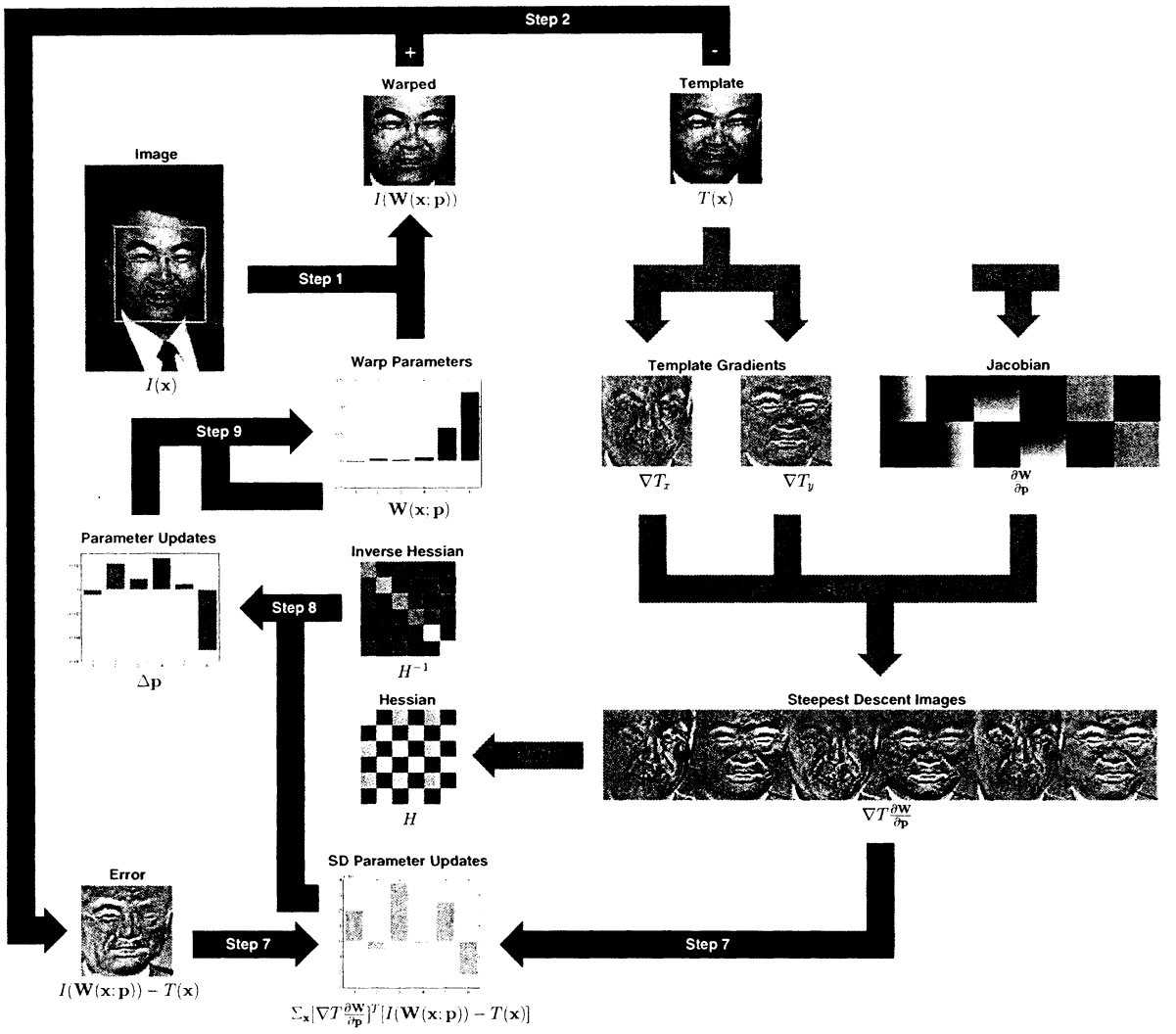

Figure 3: A schematic overview of the inverse compositional algorithm. Steps 3-6 (light-color arrows) are performed once as a pre-computation. The main algorithm simply consists of iterating: image warping (Step 1), image differencing (Step 2), image dot products (Step 7), multiplication with the inverse of the Hessian (Step 8), and the update to the warp (Step 9). All of these steps can be performed efficiently.

where $Q(\mathbf{x}, \mathbf{y})$ is an arbitrary symmetric, positive definite quadratic form. The Euclidean L2 norm is the special case where $Q(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x}, \mathbf{y})$; i.e. $Q(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{x} = \mathbf{y}$ and $Q(\mathbf{x}, \mathbf{y}) = 0$ otherwise. We therefore refer to the original inverse compositional algorithm as the Euclidean inverse compositional algorithm. The inverse compositional algorithm with a weighted L2 norm runs just as fast as the Euclidean inverse compositional algorithm. In Part 2 [2] we also extended the inverse

compositional algorithm to fit with a robust error function:

$$\sum_{x} p([T(x)-/(W(x;p))]^2) \tag{24}$$

where *p(t)* is a symmetric *robust error function* [15]. The naive robust inverse compositional algorithm (iteratively reweighted least squares) is far less efficient than the Euclidean algorithm. However, we described and evaluated two efficient approximations to the robust inverse compositional algorithm, one using the //-algorithm [12], the other using spatial coherence of p.

2.4 Part 3: Linear Appearance Variation

In Part 3 [1] we considered linear appearance variation described by a set A_{iy} $i = 1,\ldots,$ ra, of known appearance variation images. We first considered the Euclidean L2 norm: i.e. how to fit:

$$\sum_{x} \left[T(x) + \sum_{i=i}^{m} U(x) - J(W(x;p)) \right]_{J}^{2} \tag{25}$$

simultaneously with respect to the warp and appearance parameters, p and $A = (Ai,\ldots, A_m)^T$. We described the inefficient simultaneous inverse compositional algorithm, an efficient approximation to the simultaneous algorithm, the efficient project out algorithm, and the efficient normalization algorithm. We also considered fitting with a robust norm and linear appearance variation:

$$\sum_{\mathbf{x}} \varrho \left(\left[T(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 \right). \tag{26}$$

We described the inefficient robust simultaneous algorithm, the inefficient robust normalization algorithm, and efficient approximation to both of these algorithms using spatial coherence.

3 Priors on the Warp Parameters

3.1 The Lucas-Kanade Algorithm with a Prior

3.1.1 Goal of the Algorithm

We assume that the prior on the warp parameters p can be combined with the image alignment goal in Equation (2) to give the expression:

$$\sum_{x} [T(x) - I(W(x;p))]^2 + \sum_{i=1}^{K} F_i^2(p) \tag{27}$$

to be minimized with respect to the warp parameters p. The vector of functions F_I contains the prior on the parameters. It encourages the warp parameters p to take values such that $F_i^2(p)$ is small. (Note that the prior term $\sum_{i=1}^{K} F_i^2(p)$ can be generalized from the Euclidean L2 norm of the functions $F_2(p)$ to the weighted L2 norm $\sum_{ij} Q_{ij} F_i(p) F_j(p)$ where Q is an arbitrary symmetric, positive definite quadratic form. This generalization is analogous to the approach in Section 3 of Part 2 [2].) Following the approach in Section 2.1.1 the Lucas-Kanade algorithm assumes that p is known and solves for additive updates to the warp parameters Ap by approximately minimizing:

$$\sum_{x} [T(x) - I(W(x;p + \Delta p))]^2 + \sum_{i=1}^{K} F_i^2(p + \Delta p) \tag{28}$$

with respect to Ap. The algorithm then updates the parameters p <— **p** + **Ap.**

3.1.2 Derivation of the Algorithm

The derivation of the algorithm then follows Section 2.1.2. The expression in Equation (28) is linearized by performing first order Taylor expansions on the two terms to give:

$$\sum_{x} \left[T(x) - I(W(x;p)) - \nabla I \frac{\partial W}{\partial p} \Delta p \right]^2 + \sum_{i=1}^{K} \left[F_i(p) + \frac{\partial F_i}{\partial p} \Delta p \right]^2 . \tag{29}$$

The partial derivative of the expression in Equation (29) with respect to Ap is:

$$-2\sum_{\mathbf{x}}\left[\nabla I\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}}\left[T(\mathbf{x})-I(\mathbf{W}(\mathbf{x};\mathbf{p}))-\nabla I\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\Delta\mathbf{p}\right]+2\sum_{i=1}^{K}\left[\frac{\partial F_i}{\partial \mathbf{p}}\right]^{\mathrm{T}}\left[F_i(\mathbf{p})+\frac{\partial F_i}{\partial \mathbf{p}}\Delta\mathbf{p}\right]. \tag{30}$$

Setting the expression in Equation (30) to equal zero and solving gives the closed form solution:

$$\Delta\mathbf{p} = H_{\mathrm{lk},F_i}^{-1}\left[\sum_{\mathbf{x}}\left[\nabla I\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}}[T(\mathbf{x})-I(\mathbf{W}(\mathbf{x};\mathbf{p}))]-\sum_{i=1}^{K}\left[\frac{\partial F_i}{\partial \mathbf{p}}\right]^{\mathrm{T}}F_i(\mathbf{p})\right] \tag{31}$$

where the Hessian is:

$$H_{\mathrm{lk},F_i} = \sum_{\mathbf{x}}\left[\nabla I\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}}\left[\nabla I\frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]+\sum_{i=1}^{K}\left[\frac{\partial F_i}{\partial \mathbf{p}}\right]^{\mathrm{T}}\left[\frac{\partial F_i}{\partial \mathbf{p}}\right]. \tag{32}$$

These two expressions simplify to:

$$\Delta\mathbf{p} = H_{\mathrm{lk},F_i}^{-1}\left[\sum_{\mathbf{x}}\mathbf{SD}_{\mathrm{lk}}^{\mathrm{T}}(\mathbf{x})\,E(\mathbf{x})-\sum_{i=1}^{K}\left[\frac{\partial F_i}{\partial \mathbf{p}}\right]^{\mathrm{T}}F_i(\mathbf{p})\right] \tag{33}$$

and:

$$H_{\mathrm{lk},F_i} = H_{\mathrm{lk}} + \boldsymbol{r}\left|\frac{dFi}{T}\right|\frac{dFi}{\partial \mathbf{p}}\Big|. \tag{34}$$

The Lucas-Kanade algorithm with a prior then consists of iteratively applying Equation (33) and updating the warp parameters additively **p** «— **p** + Ap. The algorithm is summarized in Figure 4.

3.1.3 Computational Cost

Assume that the time taken to evaluate both $F_2(p)$ and ^p is O(n) for each *i*. Steps 2 and 5 therefore take an additional time *O(nK).* Step 6 takes extra time $O(n^2K)$ and Step 7 *O(nK).* The overall algorithm therefore takes $O(n^2N + n^2K + n^3)$ per iteration, compared to $O(n^2N + n^3)$

The Lucas-Kanade Algorithm with a Prior

Iterate:

(1) Warp/ with W(x; p) to compute 7(W(x;p))
(2) Compute the error image ^(x) and i^(p)
(3) Warp the gradient *VI* with W(x; p)
(4) Evaluate the Jacobian ^- at (x; p)
(5) Compute the steepest descent images SDik(x) and ^p
(6) Compute the Hessian matrix i?ik,F$_t$- using Equation (32)
(7) Compute £$_x$ SD£(x)£(x) - E$_{i=}$* i [fgf^(p)
(8) Invert the Hessian *H*^^ and compute Ap using Equation (33)
(9) Update the parameters p <— p -f Ap

until ||Ap|| ≤ e

Figure 4: The Lucas-Kanade algorithm with a prior consists of iteratively applying Equations (33) and (4). As well as computing the steepest descent images, the algorithm needs to compute the steepest descent direction of the prior *Y^iLi F?-* The Hessian *H*^^ is the sum of the Lucas-Kanade Hessian *H*^ and the Hessian of *Ylf=i Ff-* Overall the algorithm is little slower than the Lucas-Kanade algorithm itself.

Table 3: The computational cost of one iteration of the Lucas-Kanade algorithm with a prior. Steps 2, 5, and 7 take extra time *O(nK)* and Step 6 *O(n^2K).* Since *K* is usually small all *K* <C *N,* the extra computational cost is negligible compared to the cost of the conventional Lucas-Kanade algorithm.

Step 1	Step *2*	Step 3	Step 4	Step *5*	Step 6
O(nN)	***0(N + nK)***	**rO(n.^0**	O(niV)	0(niV4•*nK)*	O($n^2$7V +· ***n^2K)***

Step 7	Step8	Step 9	Total
*O(nN + **nK**)*	0(n^3)	O(n)	*O(n^2N + **n^2R' + n^3**)*

for the conventional Lucas-Kanade algorithm. Since *K* is usually small and *K* <C iV, the extra computational cost is negligible. The computational cost is summarized in Table 3.

3.2 The Inverse Compositional Algorithm with a Prior

3.2.1 Goal of the Algorithm

To derive the inverse compositional algorithm with a prior, we need the equivalent of Equation (28) in the inverse compositional framework. The first term is dealt with as in Section 2.2.1. The second term is more problematic. When we use the inverse compositional algorithm, we effectively

reparameterize the problem. This new parameterization needs to be reflected in the second term. Remember that with the inverse compositional algorithm, the update to the warp is:

$$\mathbf{W(x;p)} \leftarrow W(x;p) \circ W(x;\Delta p)^{-1}. \tag{35}$$

Let p' denote the parameters of $\mathbf{W(x;p)} \circ W(x;\Delta p)^{-1}$. For example, for the affine warp in Section 2, p' is derived by combining Equations (17) and (18). For simplicity, approximate the parameters of $W(x;\Delta p)^{-1}$ with the first order approximation $-\Delta p$. We therefore have:

$$p' = \begin{pmatrix} p'_1 \\ p'_2 \\ p'_3 \\ p'_4 \\ p'_5 \\ p'_6 \end{pmatrix} = \begin{pmatrix} p_1 + \Delta p_1 + p_3 \cdot \Delta p_2 \\ p_2 + \Delta p_2 + p_4 \cdot \Delta p_2 \\ p_3 + \Delta p_3 + p_1 \cdot \Delta p_3 + p_3 \cdot \Delta p_4 \\ p_4 + \Delta p_4 + p_2 \cdot \Delta p_3 + p_4 \cdot \Delta p_4 \\ p_5 + \Delta p_5 + p_1 \cdot \Delta p_5 + p_3 \cdot \Delta p_6 \\ p_6 + \Delta p_6 + p_2 + \Delta p_5 + p_4 \cdot \Delta p_6 \end{pmatrix}. \tag{36}$$

When we compute Δp in the inverse compositional algorithm, the first order equivalent additive update to the warp parameters is:

$$\frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \Delta p. \tag{37}$$

For the affine warp above:

$$\frac{dp'}{\partial \Delta \mathbf{p}} = \begin{pmatrix} 1+p_1 & p_3 & 0 & 0 & 0 & 0 \\ p_2 & 1+p_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1+p_1 & p_3 & 0 & 0 \\ 0 & 0 & p_2 & 1+p_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+p_1 & p_3 \\ 0 & 0 & 0 & 0 & p_2 & 1+p_4 \end{pmatrix}. \tag{38}$$

The equivalent of Equation (28) in the inverse compositional framework is then:

$$\sum_{\mathbf{x}} [T(W(x;\Delta p)) - I(W(x;p))]^2 + \sum_{i=1}^{K} F_i^2(\mathbf{p} + \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \Delta \mathbf{p}). \tag{39}$$

3.2.2 Derivation of the Algorithm

The derivation of the algorithm then follows Section 2.2.2. The expression in Equation (39) is linearized by performing first order Taylor expansions on the two terms to give:

$$\sum_{\mathbf{x}} \left[T(\mathbf{x}) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 + \sum_{i=1}^{K} \left[F_i(\mathbf{p}) + \frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \Delta \mathbf{p} \right]^2 . \quad (40)$$

The minimum of this quadratic has the closed form solution:

$$\mathbf{Ap} = -H_{ic}^{-1} \left[\sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))] + \sum_{i=1}^{K} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \right]^{\mathrm{T}} F_i(\mathbf{p}) \right] \quad (41)$$

where the Hessian is:

$$H_{\mathrm{ic},F_i} = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right] + \sum_{i=1}^{K} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \right]^{\mathrm{T}} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{dp'}{\partial \Delta \mathbf{p}} \right] . \quad (42)$$

These expressions simplify to:

$$\mathbf{Ap} = -H_{\mathrm{ic},F_i}^{-1} \left[\sum_{\mathbf{x}} \mathbf{SD}_{\mathrm{ic}}^{\mathrm{T}}(\mathbf{x}) E(\mathbf{x}) + \sum_{i=1}^{K} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \right]^{\mathrm{T}} F_i(\mathbf{p}) \right] \quad (43)$$

and:

$$H_{\mathrm{ic},F_i} = H_{\mathrm{ic}} + \sum_{i=1}^{K} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \right]^{\mathrm{T}} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \right] . \quad (44)$$

The inverse compositional algorithm with a prior (see Figure 5) then consists of iteratively applying Equation (43) and updating the warp $\mathbf{W(x;p)} \leftarrow \mathbf{W(x;p)} \circ \mathbf{W(x;} \Delta \mathbf{p})^{-1}$.

3.2.3 Computational Cost

Assuming that it takes time $O(n^2)$ to evaluate $J^{\wedge}$, the inverse compositional algorithm with a prior requires extra computation *0(nK)* in Step 2, and *0(n^2K)* in Steps 5a, 6a, and 7. An extra $O(n^3)$ is

The Inverse Compositional Algorithm with a Prior

Pre-compute:

(3) Evaluate the gradient VT of the template T(x)
(4) Evaluate the Jacobian ^ at (x; 0)
(5) Compute the steepest descent images $SD_{ic}(x)$ using Equation (21)
(6) Compute the Hessian matrix H_{IC} using Equation (22)

Iterate:

(1) Warp / with W(x; p) to compute **J(W(x;** p))
(2) Compute the error image *E(x)* and *F{(p)*
(5a) Compute f g j g
(6a) Compute the Hessian matrix $H\backslash_C \wedge F_l$ using Equation (44)
(7) Compute Ex SD£(x)£(x) + E* i [f ^ ^ C p)
(8) Invert the Hessian $H\backslash_Q \wedge i$ and compute Ap using Equation (43)
(9) Update the warp W(x; p) <- W(x; p) o W(x; Ap)"[1]

until ||Ap|| ≤ e

Figure 5: The inverse compositional algorithm with a prior consists of iteratively applying Equations (43) and (14). In addition to the steps taken by the Euclidean inverse compositional algorithm, the algorithm must compute the error in the prior (Step 2), the steepest descent direction for the prior (Step 5a), the combined Hessian (Step 6a), the modifi ed steepest descent update (Step 7) and the Gauss-Newton update (Step 8).

Table 4: The inverse compositional algorithm with a prior requires extra computation *O(nK)* in Step 2, $0(n^2K)$ in Steps 5a, 6a, and 7. Overall, the algorithm takes time $0(nN + n^2\ K\sim\backslash\text{-}n^z)$ per iteration. Assuming K <C N, the extra computational cost over the original inverse compositional algorithm is negligible.

Pre-Computation	Step 3	Step 4	Step 5	Step 6	Total
	$O(N)$	$0(nN)$	$0\{nN)$	$0\{n^2N)$	$0(n^2N)$

Per Iteration	Step 1	Step 2	Step 5a	Step 6a	Step 7
	O(niV)	$0(N + nK)$	$0(n^2K)$	$0(n^2K)$	$0(nN + n^2\ K)$

Step 8	Step 9	Total
$O(n^3)$	$\mathbf{O(n^2)}$	$0(nN + n^2K + n^3)$

also required in Step 8 to invert the Hessian. Overall, the algorithm takes time $O(n\ N + n^2\ K + n^s)$ per iteration, compared with $O(nJV + n^2)$ for the inverse compositional algorithm. See Table 2. Assuming K <C N and n <C AT, the extra computational cost is negligible. The computational cost of the inverse compositional algorithm with a prior is summarized in Table 4.

3.3 Extensions to the Inverse Compositional Algorithm

A natural question at this point is whether a prior on the warp parameters can be added to all the extensions to the inverse compositional algorithm in Parts 1-3. The answer is yes. The changes that need to be made are always the same, and so the details are omitted: (1) the extra term $\sum_{i=1}^{K} \left[\frac{\partial F_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}}\right]^T \wedge K P))$ must be added to the steepest descent parameter updates in Step 7, and (2) the extra term $Y\%=i\ [f \wedge J \wedge]^T\ [fjf|J\_\wedge]$ must be added to the Hessian. In the simultaneous inverse compositional algorithm in Part 3 (and all its variants), the steepest descent parameter update is an n 4- m dimensional vector (where there are *n* warp and *m* appearance parameters) and the Hessian is a *(n + m)* x *(n + m)* dimensional matrix. In these cases, the extra terms are added to the appropriate sub-vector and sub-matrix corresponding to the warp parameters. In all cases, the extra computational cost is $O(n^2 K)$, which if K <C N, is always negligibly small.

3.4 Applications

There are a wide variety of applications of putting priors on the warp parameters. We just mention three, although plenty more exist. The first example is to impose smoothness priors on the warp, as for example was done in [6]. The second example is to constrain a 2D AAM so that it is a valid instance of a 3D shape model, as in [20]. The final example is to constrain the shape parameters of an Active Appearance Model to remain within the bounds learnt from the training data [17]; e.g. to remain withing three standard deviations of the mean shape.

4 Priors on the Warp and Appearance Parameters

4.1 Review: The Simultaneous Inverse Compositional Algorithm

4.1.1 Goal of the Algorithm

In [1] we considered the optimization of:

$$\sum_{\mathbf{x}} \left[T(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 \tag{45}$$

simultaneously with respect to the warp and appearance parameters, $\mathbf{p}$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^{\mathrm{T}}$. The first algorithm we considered was the simultaneous inverse compositional algorithm. The simultaneous inverse compositional algorithm uses the inverse compositional parameter update on the warp parameters. The appearance parameters are updated additively. Composition does not have any meaning for them. The algorithm operates by iteratively minimizing:

$$\sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) + \sum_{i=1}^{m} (\lambda_i + \Delta\lambda_i) A_i(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 \tag{46}$$

simultaneously with respect to $\Delta\mathbf{p}$ and $\Delta\boldsymbol{\lambda} = (\Delta\lambda_1, \ldots, \Delta\lambda_m)^{\mathrm{T}}$, and then updating the warp $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$ and the appearance parameters $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$.

4.1.2 Derivation of the Algorithm

Performing a first order Taylor expansion on $T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p}))$ and $A_i(\mathbf{W}(\mathbf{x};\Delta\mathbf{p}))$ in Equation (46), and assuming as in Section 2.2.2 that $\mathbf{W}(\mathbf{x};\mathbf{0})$ is the identity warp, gives:

$$\sum_{\mathbf{x}} \left[T(\mathbf{x}) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} + \sum_{i=1}^{m} (\lambda_i + \Delta\lambda_i) \left(A_i(\mathbf{x}) + \nabla A_i \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 . \tag{47}$$

Neglecting second order terms, the above expression simplifies to:

$$\sum_{\mathbf{x}} \left[T(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \left(\nabla T + \sum_{i=1}^{m} \lambda_i \nabla A_i \right) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} + \sum_{i=1}^{m} A_i(\mathbf{x}) \Delta \lambda_i \right]^2 . \tag{48}$$

To simplify the notation, denote:

$$\mathbf{q} = \begin{pmatrix} \mathbf{p} \\ \boldsymbol{\lambda} \end{pmatrix} \quad \text{and similarly} \quad \Delta \mathbf{q} = \begin{pmatrix} \Delta \mathbf{p} \\ \Delta \boldsymbol{\lambda} \end{pmatrix}; \tag{49}$$

i.e. $\mathbf{q}$ is an $n+m$ dimensional column vector containing the warp parameters $\mathbf{p}$ concatenated with the appearance parameters $\boldsymbol{\lambda}$. Similarly, denote the $n+m$ dimensional steepest-descent images:

$$\mathbf{SD}_{\text{sim}}(\mathbf{x}) = \left(\left(\nabla T + \sum_{i=1}^{m} \lambda_i \nabla A_i \right) \frac{\partial \mathbf{W}}{\partial p_1}, \ldots, \left(\nabla T + \sum_{i=1}^{m} \lambda_i \nabla A_i \right) \frac{\partial \mathbf{W}}{\partial p_n}, A_1(\mathbf{x}), \ldots, A_m(\mathbf{x}) \right) . \tag{50}$$

Finally, denote the modified error image:

$$E_{\text{sim}}(\mathbf{x}) = T(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})). \tag{51}$$

Equation (48) then simplifies to:

$$\sum_{\mathbf{x}} [\, E_{\text{sim}}(\mathbf{x}) + \mathbf{SD}_{\text{sim}}(\mathbf{x})\, \Delta \mathbf{q} \,]^2 \tag{52}$$

the minimum of which is attained at:

$$\Delta \mathbf{q} = -H_{\text{sim}}^{-1} \sum_{\mathbf{x}} \mathbf{SD}_{\text{sim}}^{\text{T}}(\mathbf{x}) E_{\text{sim}}(\mathbf{x}) \tag{53}$$

where H_{sim}^{-1} is the Hessian with appearance variation:

$$H_{\text{sim}} = \sum_{\mathbf{x}} \mathbf{SD}_{\text{sim}}^{\text{T}}(\mathbf{x})\, \mathbf{SD}_{\text{sim}}(\mathbf{x}). \tag{54}$$

The Simultaneous Inverse Compositional Algorithm

Pre-compute:

(3) Evaluate the gradients VT and *VAi* for $i = 1, \ldots, m$

(4) Evaluate the Jacobian ^ at (x; 0)

Iterate:

(1) Warp I with W(x; p) to compute J(W(x; p))

(2) Compute the error image -$B_S$im(x) using Equation (51)

(5) Compute the steepest descent images $SD_Si_m(x)$ using Equation (50)

(6) Compute the Hessian matrix $H_s\backslash_m$ using Equation (54) and invert it

(7) Compute £$_x$SD?$_{sim}$(x)£$_{sim}$(x)

(8) Compute Aq - -*H*^ £$_x$ SD$_{s}$T$_m$(x)£$_{sim}$(x)

(9) Update W(x; p) <- W(x; p) o W(x; Ap)"^{1} and A <- A + AA

until ||Ap|| ≤ e

Figure 6: The simultaneous inverse compositional algorithm for appearance variation operates by iteratively applying Equations (50), (51), (53), and (54) to compute Aq. The incremental updates to the warp Ap and appearance AA parameters are then extracted from Aq and used to update the parameters in Step 9. Because the steepest descent images depend on the appearance parameters (see Equation (50)), Steps (5) and (6) must be performed in every iteration. See Table 5 for a summary of the computational cost.

In summary, the simultaneous inverse compositional algorithm for appearance variation proceeds by iteratively applying Equations (50), (51), (53), and (54) to compute Aq. The incremental updates to the warp **Ap** and appearance **AA** parameters are then extracted from Aq and used to update the warp W(x; p) <— W(x; p)oW(x; Ap)"^{1} and the appearance parameters A «— A+**AA.** Unfortunately the steepest descent images depend on the (appearance) parameters **AA** and so must be re-computed in every iteration. The result is the algorithm summarized in Figure 6.

4.13 Computational Cost

Overall the simultaneous inverse compositional algorithm is even slower than the Lucas-Kanade algorithm because the computational cost of most of the steps depends on the total number of parameters $n + m$ rather than just the number of warp parameters n. See Table 5 for a summary. Also see [1] for an efficient approximation to the simultaneous inverse compositional algorithm, as well as two other efficient algorithms, the project out algorithm and the normalization algorithm.

Table 5: The computation cost of the simultaneous inverse compositional algorithm. Overall the algorithm is even slower than the Lucas-Kanade algorithm because the computational cost of most of the steps depends on the total number of parameters n -f *m* rather than just the number of warp parameters *n.*

Pre-Computation	Step 3	Step 4	Total
	O(mN)	*O(nN)*	0((n + m)7V)

Per Iteration	Step 1	Step 2	Step 5	Step 6
	O(nN)	*O(mN)*	O((ra + m)JV)	O((n + m)^{2}7V + (n-f m)^3)

Step 7	Step 8	Step 9	Total
O((w + m)JV)	O((n + m)^2)	O(n^2 + *m)*	O((n + m)^{2}iV+(n + m)^3)

4.2 The Simultaneous Inverse Compositional Algorithm with a Prior

4.2.1 Goal of the Algorithm

We assume that the prior on the warp p and appearance A parameters can be combined with the image alignment goal in Equation (45) to give the expression:

$$\sum_{\mathbf{x}} \left[\mathbf{T(x)} + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathrm{p})) \right]^2 + \sum_{i=1}^{K} \mathbf{;G?(P;A)} \tag{55}$$

to be minimized simultaneously with respect to the warp and appearance parameters, p and A = (Ai,..., A_m)^T. The vector of functions *G{* contains the prior on the parameters. It encourages the warp p and appearance A parameters to take values such that G?$_i$(p; A) is small. As in Section 4.1.1, we use the inverse compositional parameter update on the warp parameters and the additive update on the appearance parameters. The algorithm operates by iteratively minimizing:

$$\mathbf{£}_{\mathrm{x}\ \mathrm{L}} \left[\mathbf{r(W(x;\ Ap))} + \sum_{i=1}^{m} (\lambda_i + \Delta\lambda_i) A_i(\mathbf{W}(\mathbf{x}; \mathrm{Ap})) \text{ - } \mathbf{7(W(x;\ p))} \right]^2$$

$$+ \sum_{i=1}^{K} G_i^2(\mathbf{p} + \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \Delta \mathbf{p}; \boldsymbol{\lambda} + \Delta \boldsymbol{\lambda}). \tag{56}$$

simultaneously with respect to Ap and $AA = (AAi, \ldots, AA_m)^T$, and then updating the warp $W(x; p) \leftarrow W(x; p) \circ W(x; Ap)^{-1}$ and the appearance parameters **A** «- **A** + **AA.**

4.2.2 Derivation of the Algorithm

Performing first order Taylor expansions on the two terms in Equation (56) gives:

$$\sum_{\mathbf{x}} [\, E_{\mathbf{sim}}(\mathbf{x}) + \mathbf{SD}_{\mathbf{sim}}(\mathbf{x})\, \Delta\mathbf{q}]^2 + \sum_{i=1}^{K} \left[G_i(\mathbf{p};\boldsymbol{\lambda}) + \frac{\partial G_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta \mathbf{p}} \Delta\mathbf{p} + \frac{\partial G_i}{\partial \boldsymbol{\lambda}} \Delta\boldsymbol{\lambda} \right]^2 . \qquad \textbf{(57)}$$

Denote the steepest descent direction of the prior G; by:

$$\mathbf{SD}_{G_i} = \left(\frac{\partial G_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta p_1}, \ldots, \frac{\partial G_i}{\partial \mathbf{p}} \frac{\partial \mathbf{p}'}{\partial \Delta p_n}, \frac{\partial G_i}{\partial \lambda_1}, \ldots, \frac{dG_i}{\partial \lambda_m} \right) . \qquad (58)$$

Equation (57) then simplifies to:

$$\sum_{\mathbf{x}} [\, E_{\mathbf{sim}}(\mathbf{x}) + \mathrm{SD}_{\mathrm{sim}}(\mathrm{x})\, \mathrm{Aq}]^2 + \overset{K}{Y,} [\, Gi(p;\, \mathrm{A}) + \mathrm{SD}_{\mathrm{Gj}}\, \mathrm{Aq}]^2 . \qquad (59)$$

The minimum of this expression is attained at:

$$\Delta\mathbf{q} = -H_{\mathbf{sim},G_i}^{-1} \left[\sum_{\mathbf{x}} \mathbf{SD}_{\mathbf{sim}}^{\mathbf{T}}(\mathbf{x}) E_{\mathbf{sim}}(\mathbf{x}) + \sum_{i=1}^{K} \mathbf{SD}_{G_i}^{\mathbf{T}} G_i(\mathbf{p};\boldsymbol{\lambda}) \right] \qquad (60)$$

where $H_{si}^{-}{}^{\wedge}{}_{G\cdot,1}$ is the Hessian:

$$H_s \backslash m, Gi \text{ —} \overset{\nabla}{\underset{\mathrm{X}}{\mathrm{z}}}{}^{\wedge} \mathrm{SD}_{\mathrm{sim}}^{\mathrm{T}}(\mathrm{x})\ \mathrm{SD}_{\mathrm{S}}\mathrm{i}_{\mathrm{m}}(\mathrm{x}) + \overset{K}{\overline{\gamma}}{}^{\wedge} \mathrm{SD}_{\mathrm{G}_{\cdot,1}}^{\mathrm{T}}\ \mathrm{SD}^{\wedge}{}_{\cdot}. \qquad (61)$$

The simultaneous inverse compositional algorithm with a prior proceeds by iteratively applying Equation (60) and updating the warp $\mathbf{W(x;p)} \leftarrow \mathbf{W(x;p)} \circ W(x; Ap)^{-1}$ and the appearance parameters A <— A + AA. The algorithm is summarized in Figure 7.

The Simultaneous Inverse Compositional Algorithm

Pre-compute:

(3) Evaluate the gradients VT and *VA{* for $i = 1, \ldots, n$

(4) Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at (x; 0)

Iterate:

(1) Waip *I* with W(x; p) to compute **7(W(x;** p))

(2) Compute the error image $E_s \backslash_m(x)$ and G_2-(p; A)

(5) Compute the steepest descent images $SD_S j_m(x)$ and $\mathbf{SD}_{G_i}$

(6) Compute the Hessian matrix $i?_S im,Gi^{usm}$ g Equation (61) and invert it

(7) Compute Ex $SD^\wedge_m(x)^\wedge_{sim}(x)$ + *EfLi* $SD^\wedge.G.$-fo A)

(8) Compute Aq using Equation (60)

(9) Update W(x; p) <- W(x; p) o W(x; Ap)$^{-1}$ and A <- A + AA

until ||Ap|| $\leq$ e

Figure 7: The simultaneous inverse compositional algorithm with a prior. In Steps 2, 5, 6, 7, and 8, additional processing is required for the prior G_z, but otherwise the fbw of the algorithm is the same.

Table 6: The computation cost of the simultaneous inverse compositional algorithm with a prior. Assuming *K* <C *N*, the extra computational cost over the simultaneous inverse compositional algorithm is negligible.

Pre-Computation	Step 3	Step 4	Total
	0(mN)	*0(nN)*	***0((n + m)N)***

Per Iteration	Step 1	Step 2	Step 5
	O(n7V)	***0(mN + (n + m)K)***	O((n + ra)iV+(n^2 + m)if)

Step 6	Step 7
O((n + *m)*^{2} *N* + *(n* + m)^3 + *(n* + ra)^2 *K)*	*0{(n* + ***m)N*** + ***{n*** + ***m)K)***

Step 8	Step 9	Total
O((n + *m)*^{2}*)*	O(n^2 + *m)*	**O((n -f** ***m)***^{2} ***N*** + ***(n*** + **m)**^{2} ***K*** + ***(n*** + ***mf)***

4.2.3 Computational Cost

Assume that the time taken to evaluate both G,-(p; A) and $\frac{\partial W}{\partial p}$ is O(n + *m)* for each i Step 2 therefore requires extra time O((n + *m) K),* Step 5 requires O((n^2 + m) *K),* Step 6 requires O((n + *m)*^{2} *K),* and Step 7 requires O((n + *m)K).* Overall, the extra computational cost is O((n + m)^2 AT) which is negligible \f *K* <^ *N.* The computational cost is summarized in Table 6.

4.3 Extension to Other Algorithms

Although a prior on the warp parameters can be added to all of the algorithms in Parts 1-3, the same is not true of a prior on the warp and appearance parameters. A prior on the warp and appearance parameters cannot even be added to all the algorithms in Part 3. Such a prior can be added to the efficient and robust extensions of the simultaneous inverse compositional algorithm. See Part 3. A prior on the appearance parameters, however, cannot be applied to the other algorithms in Part 3. In particular a prior on the appearance parameters cannot be added to either the project out or normalization algorithms (on any of their variants.) Neither of those algorithms explicitly solves for the appearance parameters using gradient descent. Instead, the appearance parameters as solved for implicitly using linear algebra. In general, the prior of the appearance parameters will not be a linear constraint and so the closed form solutions for the appearance parameters used by both of those algorithms are inapplicable. Hence, when placing a prior on the appearance parameters, one of the variants of the simultaneous inverse compositional algorithm must be used.

4.4 Applications

Perhaps the most natural application of priors on the appearance parameters is to constrain the (shape and) appearance of an Active Appearance Model to remain within the bounds learnt from the training data [17]; e.g. to remain withing three standard deviations of the mean.

5 Conclusion

5.1 Summary

We have shown how to add priors on the parameters to the algorithms considered in Parts 1-3 of this series [1,2,4]. In Section 3 we considered priors on the warp parameters. We first showed how a prior on the warp parameters can be added to the Euclidean inverse compositional algorithm.

We then described how the same approach can be used with all of the other algorithms in Parts 1-3 [1,2,4], In all cases, the extra computational cost is generally very small.

In Section 4 we considered priors on the warp and appearance parameters. We showed how such a prior can be added to the simultaneous inverse compositional algorithm, and its variants [1]. The same approach is not applicable to the other algorithms in Part 3 [1], in particular the project out and normalization algorithms. These algorithms do not explicitly solve for the appearance parameters using gradient descent. Instead they solve for the appearance implicitly using closed form solutions that are no longer applicable when a prior is placed on the appearance parameters.

5.2 Discussion

In the previous parts of this series we discussed which algorithm is the best one to use. If a prior on the warp parameters is all that is needed, the answer is the same as without the prior (and depends on the exact situation, as discussed in Parts 1-3.) The additional computational cost is roughly the same for all algorithms, and is generally negligible (unless the prior is computationally time consuming to evaluate.) If a prior on both the warp and appearance parameters is needed (or just a prior on the appearance parameters), the only options are the simultaneous inverse compositional algorithm. The only choice then is whether to use the efficient approximation or not.

5.3 Future Work

Although the Lucas-Kanade algorithm is now well over 20 years old [16], the subject still remains of substantial research interest. Recent developments in linear appearance modeling [8,10,14,17] have made the subject even more relevant. Ultimately, the subject retains its interest because the Lucas-Kanade algorithms minimizes one of the two most fundamental distance measures in computer vision, the image intensity difference (the other error function being the distance between features). There are a wide variety of possible areas for future work, including, but not limited to,

avoiding local minimal, aligning filtered images, and 3D to 2D image alignment.

Acknowledgments

The research described in this paper was conducted under U.S. Department of Defense contract N41756-03-C4024.

References

[1] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 3. Technical Report CMU-RI-TR-03-35, Robotics Institute, Carnegie Mellon University, 2003.

[2] S. Baker, R. Gross, I. Matthews, and T. Ishikawa. Lucas-Kanade 20 years on: A unifying framework: Part 2. Technical Report CMU-RI-TR-03-01, Robotics Institute, Carnegie Mellon University, 2003.

[3] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090-1097,2001.

[4] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, 2003.

[5] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221-255, 2004.

[6] S. Baker, I. Matthews, and J. Schneider. Image coding with active appearance models. Technical Report CMU-RI-TR-03-13, Robotics Institute, Carnegie Mellon University, April 2003.

[7] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237-252, 1992.

[8] M. Black and A. Jepson. Eigen-tracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 36(2): 101-130, 1998.

[9] G.E. Christensen and H.J. Johnson. Image consistent registration. *IEEE Transactions on Medical Imaging*, 20(7):568-582, 2001.

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

[11] F. Dellaert and R. Collins. Fast image-based tracking by selective pixel integration. In *Proceedings of the ICCV Workshop on Frame-Rate Vision*, pages 1-22, 1999.

[12] R. Dutter and P.J. Huber. Numerical methods for the nonlinear robust regression problem. *Journal of Statistical and Computational Simulation*, 13:79–113, 1981.

[13] M. Gleicher. Projective registration with difference decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 331–337, 1997.

[14] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[15] P.J. Huber. *Robust Statistics*. John Wiley & Sons, 1981.

[16] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[17] I. Matthews and S. Baker. Active Appearance Models revisited. *International Journal of Computer Vision*, 2004. In Press.

[18] S. Sclaroff and J. Isidoro. Active blobs. In *Proceedings of the 6th IEEE International Conference on Computer Vision*, pages 1146–1153, 1998.

[19] H.-Y. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 16(1):63–84, 2000.

[20] J. Xiao, S. Baker, I. Matthews, and R. Gross. Real-time combined 2D+3D active appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004. In Press.