Enforcing Non-Positive Weights for Stable Support Vector Tracking

Simon Lucey
Robotics Institute, Carnegie Mellon University

slucey@ieee.org

Abstract

In this paper we demonstrate that the support vector tracking (SVT) framework first proposed by Avidan is equivalent to the canonical Lucas-Kanade (LK) algorithm with a weighted Euclidean norm. From this equivalence we empirically demonstrate that in many circumstances the canonical SVT approach is unstable, and characterize these circumstances theoretically. We then propose a novel "non-positive support kernel machine" (NSKM) to circumvent this limitation and allow the effective use of discriminative classification within the weighted LK framework. This approach ensures that the pseudo-Hessian realized within the weighted LK algorithm is positive semidefinite which allows for fast convergence and accurate alignment/tracking. A further benefit of our proposed method is that the NSKM solution results in a much sparser kernel machine than the canonical SVM leading to sizeable computational savings and much improved alignment performance.

1. Introduction

Rather than exhaustively searching through all possible warps it is often more efficient in terms of computation to perform a non-linear optimization by iterating two steps when performing object alignment/tracking. First, approximately minimize some cost typically by making some sort of linear or quadratic approximation around the current warp parameters. Second, update the current parameter estimate and then repeat step one until convergence. This approach is typically referred to as a *gradient descent* approach to object alignment/tracking [3]. The most notable application of this concept in computer vision is the classic Lucas-Kanade (LK) [14] algorithm although many variations upon this idea now exist in computer vision literature [3].

It has been recently found that their is an inherent advantage in performing gradient descent object alignment/tracking using a discriminative rather than a generative model [1, 16, 13, 15]. The appeal of discriminative approaches lies in their ability to generalize to situations

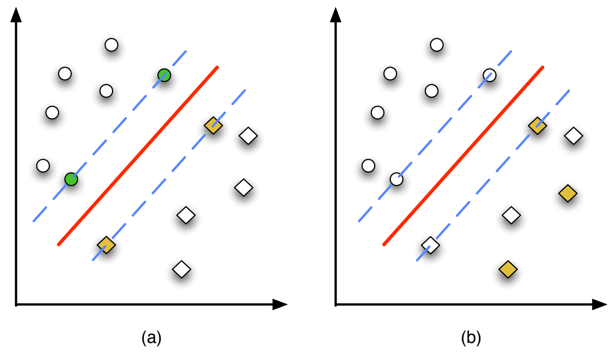

Figure 1. A major contribution in our paper is the ability to train a sparse kernel-machine with non-positive alpha weights which we refer to as a "non-positive support kernel machine" (NSKM). (a) A separating plane defined by a regular SVM requiring both positively and negatively weighted support vectors. (b) A separating plane defined by "only" negatively weighted support vectors. The filled circles and diamonds are termed positively and negatively weighted "support vectors" respectively. Ensuring negatively weighted alpha weights ensures the weight matrix employed within the LK alignment algorithm remains positive semidefinite which is critical for stable alignment/tracking behaviour.

where the appearance of the object being aligned/tracked has not been seen previously. One of the most notable and elegant approaches to discriminative gradient descent object alignment/tracking was the seminal work of Avidan [1] concerning his support vector tracking (SVT) framework. In this work Avidan employed a support vector machine (SVM) trained using a homogeneous quadratic kernel. Through some manipulation he found this SVM can be reformulated to take a similar form as the standard LK solution for optical flow. Avidan then demonstrated how this SVT outperformed the classic LK approaches for tracking. What makes the SVT framework and more generally the LK algorithm perform well in many alignment/tracking tasks is that it iteratively approximates the problem of finding the correct warp as minimizing an unconstrained *convex* quadratic optimization problem[1]. As noted in optimization

[1]Note that the quadratic function $f(\mathbf{x}) = (1/2)\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} + r$ is considered convex if $\mathbf{P} \succ 0$ (i.e., if $\mathbf{P}$ is positive definite) [6]. If $\mathbf{P}$ is positive semidefinite (i.e., $\mathbf{P} \succeq 0$) but not positive definite we refer to this

literature [6] the solution to any convex quadratic problem has a unique solution which can be found explicitly. Unfortunately, if the quadratic problem is *not* convex then there exists no method for explicitly finding the minima (there may instead exist no minima or multiple minimas). Within the canonical LK algorithm the quadratic problem being iteratively solved is always assured of being "close" to convex so this concern can be largely ignored. Within the SVT framework, however, we demonstrate that no such guarantees can be made. This dilemma forms the central thesis of our paper. The contributions of our paper are as follows:-

• Demonstrating that when the SVT framework is recast in the form of the weighted LK algorithm, the weight matrix learnt has no guarantees of being positive semidefinite. When the weighting matrix is not positive semidefinite the performance of the SVT framework can suffer catastrophically as it is no longer guaranteed of iteratively solving a quadratic optimization problem that is "close" to convex (Section 3).

• To remedy this problem we propose the employment of our novel "non-positive support kernel machine" (NSKM) which learns a sparse kernel machine with non-positive weights (see Figure 1). The NSKM results in a sparser solution than the canonical SVM approach leading to sizeable computational savings during alignment (Section 4).

• We demonstrate that our NSKM approach offers superior alignment to current generative approaches for LK type alignment when the appearance variation of the object has not been seen previously (Section 5).

Related Work: There has been substantial work conducted in the field of gradient descent alignment/tracking in the presence of appearance variation. Notably, Black and Jepson [5] proposed a gradient-descent approach to alignment which employed principal component analysis (PCA) to model object appearance variation. Within the LK framework they employed a distance from feature space (DFFS) error (i.e., reconstruction error) criterion to solve for warp displacement. In this work a robust error function was also employed to deal with occlusion and some degree of unseen appearance variation. Variations on the employment of a DFFS error criterion have been proposed by a number of authors, notably Hager and Belhumeur [10], with a good review of these variations being conducted in [2]. These approaches, however, can be considered generative, rather than discriminative, as they employ only positive (aligned) examples during learning and optimize an error function based on how well the generative model (i.e., PCA) can reconstruct the object. As a result it is well understood these approaches suffer from problems when trying to generalize to previously unseen object appearance variation.

quadratic function as being "close" to convex. In this situation convexity can be enforced by adding a weighted identity matrix (i.e., $\mathbf{P} \leftarrow \mathbf{P} + \sigma\mathbf{I}$) to $\mathbf{P}$ to enforce convexity.

An excellent review on the shortcomings of PCA models for gradient-descent alignment was performed by Gross et al. [8] concerning active appearance model (AAM) alignment.

Other than the work of Avidan [1], which is of central focus in this paper, there has recently been a plethora of work performed in the area of discriminative gradient descent object alignment/tracking. An approach that is closest in essence to the work of Avidan can be found in the recent work of Liu [13] for AAM face alignment. In this approach the authors trained a classifier, using a gentle boost procedure, to reliably classify "aligned" and "non-aligned" faces. Although reporting good alignment performance this approach employed a steepest descent optimization strategy to fitting. As a result the approach is susceptible to the same drawbacks seen in nearly all steepest descent methods, specifically how to choose a good step size and the problem of slow convergence.

Other work related to Avidan's method are approaches that attempt to learn a static displacement expert using discriminative regression methods. Williams et al. [16] learnt a displacement expert, which takes an image as input and returns the warp displacement, by using a relevance vector machine (RVM). Similarly, Saragih and Goecke [15] recently learned a displacement expert through a boosting procedure for AAM alignment. Again, although receiving good performance these approaches are hindered by the static nature of the learnt displacement expert. Additionally, the convergence properties of such approaches are poorly understood and largely based on heuristics.

2. Lucas-Kande Alignment

The standard Lucas-Kanade (LK) alignment algorithm [3, 14] attempts to find the parametric warp $\mathbf{p}$ that minimizes the weighted L2 norm between the template image T and source image Y[2],

$$\arg\min_{\mathbf{p}} ||Y(\mathcal{W}(\mathbf{z};\mathbf{p})) - T(\mathcal{W}(\mathbf{z};\mathbf{0}))||^2_{\mathbf{D}} \quad (1)$$

The vector $\mathbf{z} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]^T$ is a concatenation of individual pixel *2D* coordinates $\mathbf{x}$ within the template image T. One can map the image positions $\mathbf{z}$ to a set of new positions $\mathbf{z}'$,

$$\mathbf{z}' = \mathcal{W}(\mathbf{z};\mathbf{p}) = [\mathcal{W}(\mathbf{x}_1;\mathbf{p})^T, \ldots, \mathcal{W}(\mathbf{x}_N;\mathbf{p})^T]^T \quad (2)$$

based on the warp parameters $\mathbf{p}$. The warp function is canonically translation $\mathcal{W}(\mathbf{x};\mathbf{p}) = \mathbf{x} + \mathbf{p}$, but $\mathcal{W}(\mathbf{x};\mathbf{p})$ has been extended to numerous other types of warps such

[2]For convenience we employ the notation that $||\mathbf{a} - \mathbf{b}||^2_{\mathbf{C}} = (\mathbf{a} - \mathbf{b})^T\mathbf{C}(\mathbf{a} - \mathbf{b})$, where $\mathbf{a}$ and $\mathbf{b}$ are column vectors and $\mathbf{C}$ is a symmetric matrix.

as affine [5, 3] and piece-wise affine [8]. When we refer to $T(\mathbf{x})$ it refers to a single pixel intensity value at the image coordinate $\mathbf{x}$. Similarly, when we refer to $T(\mathbf{z}) = \left[T(\mathbf{x}_1)^T, \ldots, T(\mathbf{x}_N)^T\right]^T$ this refers to a N dimensional vector concatenation of pixel intensity values corresponding to each coordinate $\mathbf{x}$ within the vector $\mathbf{z}$. From here on we shall refer to $T(\mathcal{W}(\mathbf{z}; \mathbf{0}))$ as $T(\mathbf{z})$ because $W(\mathbf{z}; \mathbf{0})$ is an identity warp. The vector $Y(\mathbf{z}')$ is always the same size as $T(\mathbf{z})$ and refers to the pixel values taken from the source image we want to align. The weight matrix $\mathbf{D}$ is canonically chosen to be an identity matrix, however, there has been a lot of recent work [1, 5] demonstrating how different weight matrices can be defined/learned for improved robustness and performance. The details on how $\mathbf{D}$ is selected is a major part of our paper and shall be discussed in detail in Section 3.

Rather than exhaustively searching all possible values of $\mathbf{p}$, which becomes exponentially expensive as the dimensionality of $\mathbf{p}$ increases, the LK algorithm derives and solves a continuous optimization problem. The approach takes the first order Taylor series approximation around $Y(\mathbf{z}')$ to recast the non-linear optimization problem in Equation 1 as an unconstrained quadratic optimization,

$$\arg\min_{\Delta\mathbf{p}} ||Y(\mathbf{z}') + \mathbf{J}^T\Delta\mathbf{p} - T(\mathbf{z})||^2_{\mathbf{D}} \tag{3}$$

where $\Delta\mathbf{p}$ is the warp of the source image $Y(\mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p}))$ that minimizes the weighted sum of squared differences (SSD) between the template $T(\mathbf{z})$ given that $\mathbf{p}$ is our initial guess. One can solve for $\Delta\mathbf{p}$ through,

$$\Delta\mathbf{p} = \mathbf{H}^{-1}\mathbf{J}\mathbf{D}\left[T(\mathbf{z}) - Y(\mathbf{z}')\right] \tag{4}$$

where the pseudo-Hessian matrix is defined as,

$$\mathbf{H} = \mathbf{J}\mathbf{D}\mathbf{J}^T \tag{5}$$

and the Jacobian of $Y(\mathbf{z}')$ is defined as,

$$\mathbf{J} = \frac{\partial Y(\mathbf{z}')}{\partial \mathbf{p}} \tag{6}$$

Since we took the first order Taylor series approximation around $Y(\mathbf{z}')$ in Equation 3 we must iteratively update $\mathbf{z}' = \mathcal{W}(\mathbf{z}; \mathbf{p}) \leftarrow \mathcal{W}(\mathbf{z}; \mathbf{p} + \Delta\mathbf{p})$ and then solve for a new $\Delta\mathbf{p}$ in Equation 4 until convergence or a maximum number of iterations are performed. This type of non-linear optimization is commonly referred to as *Gauss-Newton* optimization [3].

Other least-squares variants have been evaluated in [3] such as the *Newton* and *Levenberg-Marquardt* optimization for the LK algorithm. The Levenberg-Marquardt optimization is especially useful when $\mathbf{H}$ is positive semidefinite but not positive definite as it adds a weighted identity matrix to $\mathbf{H}$ to ensure the quadratic in Equation 3 has a single solution. As pointed out in [3] assuming $\mathbf{H}$ is positive definite, even though we only have guarantees that $\mathbf{H}$ is positive

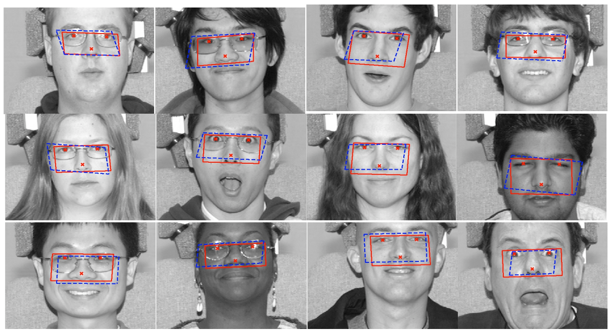
Figure 2. Example images from the MultiPIE database [9] used in our alignment experiments. Eye and nose ground-truth points are denoted on each face with red"x"s with the resulting red solid line bounding box taken around those coordinates. A blue dashed line bounding box denotes an instance where the ground-truth points are perturbed by random noise ranging between 5-10 *root mean squared point error* (RMS-PE). The images within the MultiPIE database exhibit a large amount of expression variation making alignment challenging.

semidefinite, still leads to robust performance and fast convergence in practice [3]. As we can see in Equation 5, $\mathbf{H}$ is guaranteed of being positive semidefinite if the weight matrix $\mathbf{D} \succeq 0$.

3. Support Vector Tracking

SVMs have been used to great effect in a plethora of different machine learning tasks [7, 4] as they are able to incorporate positive and negative examples within their framework and provide good generalization through the principle of "maximizing the margin". A SVM can be employed for alignment by estimating the parametric warp $\mathbf{p}$ that maximizes,

$$\arg\max_{\mathbf{p}} \sum_{j=1}^{l} b_j\gamma_j k(Y(\mathcal{W}(\mathbf{z}; \mathbf{p})), T_j(\mathbf{z})) + c \tag{7}$$

where T_j are the support template vectors taken from an ensemble of offline positive (aligned) and negative (misaligned) object examples (see Figure 2), γ_j are their support weights, b_j are their sign, and l are the number of support vectors. $k()$ is the kernel we choose to use, $Y(\mathcal{W}(\mathbf{z}; \mathbf{p}))$ is the candidate region in the source image Y we are evaluating and c is a bias term learnt during the SVM optimization process (the constant c, however, can be dropped as it does not affect the solution). In a similar approach to the LK algorithm, Avidan proposed a method to solving Equation 7 through a continuous optimization (rather than exhaustively searching over $\mathbf{p}$). Again, we take the first order Taylor series approximation around $Y(\mathbf{z}')$ so that we obtain a new linear optimization problem,

$$\arg\max_{\Delta\mathbf{p}} \sum_{j=1}^{l} \alpha_j k(Y(\mathbf{z}') + \mathbf{J}^T\Delta\mathbf{p}, T_j(\mathbf{z})) \tag{8}$$

where $\alpha_j = b_j \gamma_j$ combines the support weight and sign into a single value which we shall refer to herein as *alpha weights*. Avidan demonstrated that if a homogeneous quadratic polynomial given by the kernel $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T\mathbf{b})^2$ is employed it is possible to solve for $\Delta\mathbf{p}$ explicitly. In this particular case the equations *resemble* the standard LK solution seen in Equation 4 with the support vectors replacing the role of the template $T(\mathbf{z})$. In Appendix A we demonstrate that the two solutions are actually identical given that,

$$\mathbf{D} = \sum_{j=1}^{l} -\alpha_j T_j(\mathbf{z}) T_j(\mathbf{z})^T \tag{9}$$

and assuming our object template $T(\mathbf{z})$ (not the support vectors denoted by $T_j(\mathbf{z})$) is zero in Equation 4. We can now view SVT as minimizing the weighted error in Equation 1 rather than maximizing the classification score in Equation 8. As we shall discuss in subsequent sections this insight leads to a number of interesting avenues for improving the behavior of Avidan's SVT algorithm.

Problems with the Positive Alpha Weights: As discussed previously in Section 2 for the weighted LK algorithm to perform effectively the pseudo-Hessian matrix $\mathbf{H}$ *must* at least be positive semidefinite. This is because each iteration of the LK algorithm performs a type of curve fitting which is quadratic. When $\mathbf{H}$ is positive semidefinite it ensures that this surface is a convex quadratic and thus has a unique minimum. If $\mathbf{H}$ is negative semidefinite or indefinite then the convexity requirement [6] for the unconstrained quadratic optimization problem being iteratively solved is violated. In the special situation when $\mathbf{H}$ is positive semidefinite but not positive definite (i.e., singular) we assume a weighted identity matrix can be added to $\mathbf{H}$ to enforce convexity (e.g., Levenberg-Marquardt optimization).

When alpha weights are allowed to become positive it is possible for the Hessian $\mathbf{H}$ to violate our positive semidefinite assumption and turn our convex quadratic optimization into a non-convex problem. Combining Equations 5 and 9 to form the pseudo-Hessian matrix,

$$\mathbf{H} = \mathbf{J}\mathbf{D}\mathbf{J}^T = -\sum_{j=1}^{l} \alpha_j \mathbf{J}\left[T_j(\mathbf{z})T_j(\mathbf{z})^T\right]\mathbf{J}^T \tag{10}$$

it is clear that we must have only non-positive alpha weights to ensure $\mathbf{H}$ is positive semidefinite (since the non-zero weighted sum of a set of convex functions is strictly convex [6]). Inspecting Equation 10 further it becomes clear that if $\mathbf{D}$ is positive semidefinite then the Hessian matrix $\mathbf{H}$ is guaranteed of being positive semidefinite.

Unfortunately, by definition [7, 4] the alpha weights for an SVM must sum to zero (i.e., $\sum_{j=1}^{l} \alpha_j = 0$) thus implying there will always be positive and negative alpha weights. We should emphasize, however, that the zero sum constraint on the SVM alpha weights does not prohibit $\mathbf{D}$ from being positive semidefinite which explains why in many circumstances the conventional SVT framework performs well.

4. Our Approach

As discussed in Section 3 major problems can occur if we do not enforce our weight matrix $\mathbf{D}$ to be positive semidefinite within the LK framework. A naïve way to solve this problem is to learn a least-squares convex homogeneous quadratic classifier directly. This optimization problem can be posed as a convex quadratically constrained quadratic programming (QCQP) problem [6] (as forcing $\mathbf{D}$ to be semidefinite involves quadratic constraints). While this convex QCQP can be solved by standard online packages, general-purpose solvers tend to scale poorly in the number of constraints.

By imposing additional structure on $\mathbf{D}$ (e.g., enforcing $\mathbf{D}$ to be diagonal) so all constraints are linear this QCQP problem can reduce to a quadratic programming (QP) problem[3]. Unfortunately, even in this instance for reasonable values of M the amount of memory required to store this QP problem is far too large (i.e., $(M+1)(M/2) \times (M+1)(M/2)$) to be stored in memory making such an optimization untenable in practice. Another option is to borrow the "kernel" trick employed by Avidan [1] from within the SVT framework and attempt to optimize,

$$\begin{aligned}
\arg\min_{\boldsymbol{\alpha}} &= \frac{1}{2}||\mathbf{b} - \mathbf{K}\boldsymbol{\alpha}||^2 \\
&= \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \qquad (11)\\
\text{given,} & \\
& \mathbf{Q} = \mathbf{K}^T\mathbf{K}, \mathbf{e} = \mathbf{K}^T\mathbf{b} \\
& \mathbf{K}_{i,j} = \boldsymbol{\psi}_i^T\boldsymbol{\psi}_j = (T_j(\mathbf{z})^T T_i(\mathbf{z}))^2 \\
& i = j = 1, \ldots, N \\
& \mathbf{b} = [b_1, \ldots, b_N] \\
\text{subject to,} \quad & -C \leq \alpha_i \leq 0, i = 1, \ldots, N
\end{aligned}$$

where $b_j \in \{+1, -1\}$ are the class labels for the N offline M dimensional training vectors $T_j(\mathbf{z})$ and $\boldsymbol{\psi}_j = \text{vec}\left[T_j(\mathbf{z})T_j(\mathbf{z})^T\right]$. Attempting to solve for $\mathbf{D}$ using a least-squares classifier, rather than through a SVM, is advantageous as we no longer have the constraint that the alpha weights α_j must sum to zero. By posing Equation 11 as a QP problem we can actually enforce the alpha weights α_j all to be negative or zero. The solution to this problem can also be solved using QP. Additionally, the matrix $\mathbf{Q}$, in Equation 11, can be stored feasibly in memory (i.e., $N \times N$) which is now a function of the number of training images N. This kernel solution is not equivalent

[3]Readers are encouraged to read [11] to gain an insight for how least-squares fitting of a polynomial can be constrained to be convex.

to the naïve solution as we are just ensuring that the alpha weights α_j are negative (which implies *indirectly* that $\mathbf{D}$ is positive semidefinite) rather than *directly* ensuring $\mathbf{D}$ is positive semidefinite. This stronger constraint, however, is acceptable as it allows us to solve for $\mathbf{D}$ in a practical manner.

We shall refer to this classifier from here on as a *non-positive support kernel machine* (NSKM). The NSKM shares many similarities with conventional SVMs with the exception that we employ a "least-squares" error-function rather than the "hinge" error-function used in the canonical SVM formulation [7, 4]. In a similar manner to SVMs, we also have a regularization parameter C that controls the magnitude of the alpha weights which indirectly controls overfitting. As with the SVMs used in this paper we select C through a cross-validation procedure.

Optimization through Projected Gradients: Even though we can now store the matrix $\mathbf{Q}$ in memory, the optimization of Equation 11 using conventional quadratic programming packages can be extremely time consuming, especially if N is large and $\mathbf{Q}$ is dense. In recent work by Lin [12] on the related problem of non-negative matrix factorization (NMF) it was demonstrated that good results could be achieved for the optimization problem described in Equation 11 when $\mathbf{Q}$ is large using projected gradient methods. In this work Lin demonstrated that this approach exhibited good results and quick convergence for large scale problems. More details on this approach can be found in [12].

5. Experiments

All experiments in this paper were conducted on the frontal portion of the MultiPIE face database [9]. A total of 1128 images were employed in this subset of the database taken across 141 subjects. Face images varied substantially in expression (see Figure 2 for examples) making detection and alignment quite challenging. This data set was separated into subject independent training and testing sets with 560 and 568 images in each set respectively. In our experiments we assume that all warped images are referenced as an 80×80 image. We obtained negative face examples by synthetically adding affine noise to the ground-truth coordinates of the face (located at the eyes and nose). We randomly generated affine warps $\mathcal{W}(\mathbf{z};\mathbf{p})$ in the following manner. We used the top left corner $(0,0)$, the top right corner $(79,0)$ and the center bottom pixel $(39,79)$. We then perturbed these points with a vector generated from white Gaussian noise. The magnitude of this perturbation was controlled to give a desired root mean squared (RMS) pixel error (PE) from the ground-truth coordinates. During learning, the negative/misaligned class was defined to have between $5-10$ RMS-PE. This range of perturbation was chosen as it approximately reflected the range of alignment error seen when employing the freely available OpenCV face detector (see Figure 1 for examples of this perturbation). In all experiments we conducted in this paper our warped training and testing examples (both positive and negative) are assumed to have zero bias and unit gain.

Detection Experiments: Before employing our proposed NSKM classifier on a gradient-descent alignment task, it is of use to analyze how this classifier performs in terms of detection. In our first set of experiments we evaluated the detection performance of three classifiers each having the same quadratic form,

$$E(Y(\mathbf{z})) = ||Y(\mathbf{z}) - T(\mathbf{z})||^2_{\mathbf{D}} \tag{12}$$

where $Y(\mathbf{z})$ is the input image to the classifier, $T(\mathbf{z})$ is the bias and $\mathbf{D}$ is the weighting matrix. Note that Equation 12 has the same parametric form as Equation 3 except we are now obtaining an error score instead of solving for a warp displacement. Note the error score in Equation 12 departs from how a canonical classifier behaves as when $Y(\mathbf{z})$ belongs to a positive class $E(Y(\mathbf{z}))$ is aiming to be low (rather than high) and when $Y(\mathbf{z})$ belongs to a negative class $E(Y(\mathbf{z}))$ is aiming to be high (rather than low). All the quadratic classifiers used in our experiments were trained on 560 positive face examples and 5000 negative examples (if the approach required negative examples during learning). We evaluated these classifiers initially for a detection task with a test set of 568 positive face examples and $10,000$ negative examples. The train and test sets used to learn and evaluate the quadratic classifiers were completely independent from one another. To ensure that there was no bias, subject identities were also completely independent between train and test sets. Detection performance for the three quadratic classifiers can be seen in Figure 3 in terms of a detection error tradeoff (DET) curve. A short description of each classifier follows:-

DFFS: This classifier was inspired by the "project-out" algorithm of Hager and Belhumeur [10] which was formalized by Baker et al. [2] where we define a quadratic classifier to output the distance from feature space (DFFS) measure. PCA was employed to model 95% of the appearance variation in the positive/aligned face class. The ensemble of eigenvectors $\mathbf{A}$ stemming from the PCA process was then used to define $\mathbf{D} = \text{null}(\mathbf{A})\text{null}(\mathbf{A})^T$. As a result $\mathbf{D}$ is *always* positive semidefinite for a DFFS classifier.

SVM: This classifier was trained according to the description in Section 3 such that a canonical SVM was trained using a homogeneous quadratic kernel. The weight matrix $\mathbf{D}$ was found by applying Equation 9.

NSKM: This classifier was trained according to the description in Section 4 such that a homogeneous quadratic

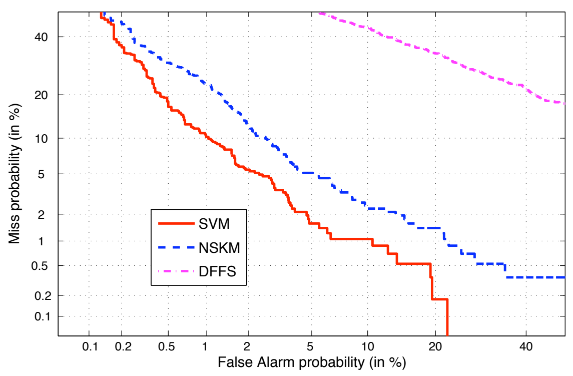

Figure 3. Detection performance on the test-set for different quadratic classifiers. In this figure we demonstrate that the canonical SVM and our NSKM extension receive the best detection performance. Next, comes the DFFS classifier based based on the principal components of the positive object. All quadratic classifiers shown have guarantees on $\mathbf{D}$ being positive semidefinite except the SVM classifier.

kernel machine is learnt with alpha weights that are constrained to be non-positive. Again, the weight matrix $\mathbf{D}$ was found by applying Equation 9.

For all classifiers the bias $T(\mathbf{z})$ was estimated as the average image across all images stemming from the positive face class[4]. As expected the SVM classifier outperformed the other three classifiers. Our proposed NSKM classifier, although not performing as well as the SVM classifier, exhibited superior performance to the DFFS classifier thus validating the utility of this type of quadratic classifier.

Alignment Experiments: In our next set of experiments we analyze the error score and alignment convergence properties of the SVM, NSKM and DFFS classifiers within the weighted LK algorithm. To test the ability of our algorithms to correctly register a previously unseen source image we again synthetically generated random initial affine warps to range between $5 - 10$ RMS-PE. In Figure 4 one can see the normalized average error score taken across all trials of the SVM and NSKM approaches as a function of the number of iterations performed within the weighted LK algorithm. The DFFS classifier was not considered in these experiments as the convergence properties of this quadratic classifier within the LK algorithm is already well understood [10, 2]. The error scores for both techniques were normalized so the maximum average error score at any iteration is *not* above unity. The normalization was performed as the magnitude of the error scores between the two approaches varied considerably. Inspecting Figure 4 we can see that the NSKM error score is monotonically reducing at each iteration. One can additionally see that the error score for the NSKM approach reaches convergence on average within a couple of iterations which is consistent with what one would expect in a Gauss-Newton optimization. The SVM approach, however, has dramatically different performance to the NSKM result. Most noticeably, the error score is not reducing at each iteration, but instead increasing in an erratic manner.

[4]Employing a non-zero bias within the SVM learning framework departs slightly from Avidan's original approach. In our experiments employing a zero or non-zero bias had almost no effect on the quadratic SVM's detection performance. However, employing a non-zero bias for the other two classifiers did effect performance substantially. For consistency we chose to use the same bias for all the classifiers.

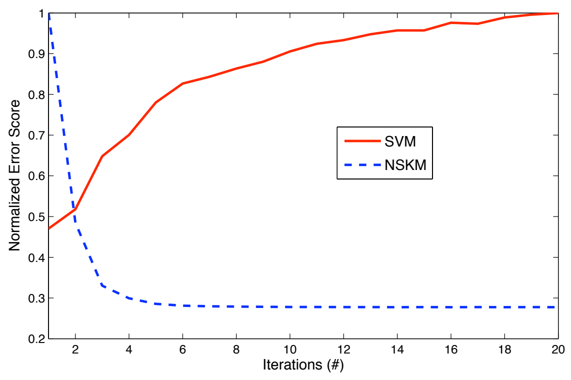

Figure 4. Convergence of the average error scores, as a function of the number of iterations, within the weighted LK algorithm for the SVM and NSKM algorithms. The error scores for both classifiers were normalized so the maximum average value at any iteration is not above unity. Results demonstrate that the NSKM classifier behaves as we expect with fast convergence on average within a couple of iterations. The SVM approach, however, has dramatically different performance with the error score diverging. The poor behaviour of the SVM approach can be linked to the fact that $\mathbf{D}$, for the SVM approach, is not positive semidefinite (see Figure 5) so the quadratic being solved in many of the iterations of the weighted LK algorithm is not unique.

These differing error score convergence curves can be best explained by inspecting the signs of the eigenspectrum of $\mathbf{D}$ in Figure 5. This figure depicts whether the eigenvalues of $\mathbf{D}$ are positive, negative or sparse (i.e., equal to zero). One can see that a sizable portion of the eigenvalues for the SVM classifier are both positive and negative indicating that $\mathbf{D}$ is indefinite. Conversely, the eigenvalues for the NSKM classifier are all positive or sparse indicating that $\mathbf{D}$ is positive semidefinite. As discussed in Section 4, by enforcing $\mathbf{D}$ to be positive semidefinite within the NSKM approach we can ensure that the quadratic being minimized at each iteration has a unique minimum. When this assumption is violated poor performance often occurs as can be seen in the error score convergence curves in Figure 4. We should emphasize here that Figure 5 depicts the nature of the eigenvalues for a specific SVM and NSKM classifier and is not a general result for SVM classifiers. A major contribution we are trying to make in this paper, however, is that the conventional SVT framework to object alignment/tracking provides no *guarantee* on the nature of $\mathbf{D}$ which can have dire consequences when attempting to use this framework generically. In fact, in Avidan's original

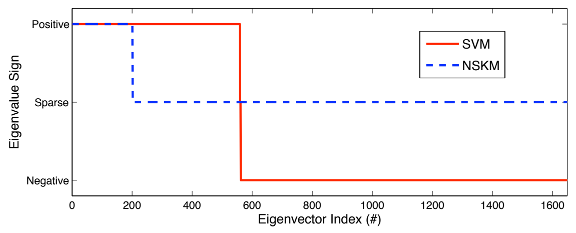

Figure 5. Depicting how many eigenvalues of $\mathbf{D}$ are positive, negative or sparse (i.e., zero). For the SVM classifier trained on the MultiPIE face data in our experiments the matrix $\mathbf{D}$ has a sizable number of eigenvalues that are both positive and negative indicating the matrix is indefinite. The NSKM classifier, however, has only positive or sparse eigenvalues indicating that $\mathbf{D}$ is positive semidefinite.

work he demonstrated that the SVT framework performed extremely well for the task of car tracking. The nature of his results would indicate that the $\mathbf{D}$ learnt for his specific application was positive semidefinite.

Inspecting Figure 5 further we can see that the NSKM classifier has 198 non-zero eigenvalues and is noticeably more sparse than the SVM classifier with 1638 non-zero eigenvalues. This is another highly desirable attribute within our proposed framework as the application of our NSKM approach within the weighted LK algorithm is considerably more efficient than the conventional SVM approach. In Avidan's original approach [1] he employed a *reduce set method* to reduce the number of support vectors stemming from the SVM considerably. This approach worked off the basis that the number of support vectors should be bounded by the dimensionality of the input space. Since we are dealing with image templates of dimensionality 80×80 such a method will have no benefit in our experiments.

To gauge the performance of the three described approaches for the task of alignment within the weighted LK algorithm we also present an *alignment convergence curve* (ACC) in Figure 6. These curves have a threshold distance in RMS-PE on the x-axis and the percentage of trials that achieved convergence (i.e., final alignment RMS-PE below the threshold) on the y-axis. A perfect alignment algorithm would receive an ACC that has 100% convergence for all threshold values. We can see in Figure 6 that our proposed NSKM approach outperforms both the DFFS [5] and SVM [5] strategies. The drastic difference in performance between the NSKM and SVM approaches is consistent with the error score convergence curves seen in Figure 4 depicting the stable and fast convergence of the NSKM approach in comparison to the erratic convergence curves produced by the SVM approach.

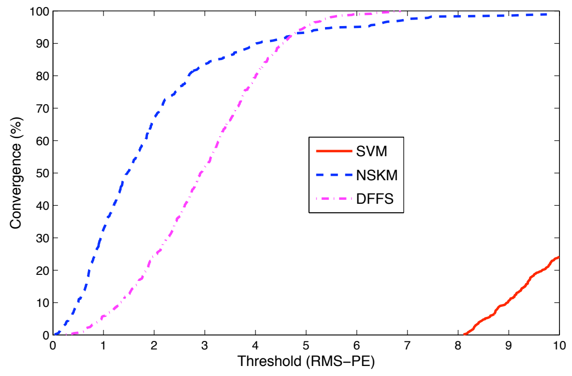

Figure 6. Alignment results demonstrating the benefit of our proposed NSKM approach over the canonical DFFS [5] and SVM [1] approaches for gradient descent object alignment in the presence of unseen appearance variation.

6. Discussion and Conclusions

In this paper, we have demonstrated that the support vector tracking (SVT) framework proposed by Avidan [1], under specific constraints, is equivalent to the weighted LK algorithm. From this equivalence we have also demonstrated that, in certain circumstances, the SVT framework is theoretically prone to unstable behaviour. This behaviour was also confirmed empirically and can be directly attributed to the nature of the weight matrix being employed within the weighted LK algorithm. Specifically, if this weight matrix is *not* positive semidefinite we have no guarantees that the quadratic being iteratively solved is "close" to convex. In these circumstances the quadratic, being solved iteratively, may have multiple or no minima resulting in erratic behaviour.

To circumvent this problem we have presented a novel framework for performing discriminative alignment of an object with "unseen" appearance variation using the weighted LK algorithm. The approach employs what we refer to as a *non-positive support kernel machine* (NSKM). This machine learns a weight matrix, that is suitable for use within the weighted LK algorithm, that is guaranteed of being positive semidefinite. This guarantee is important as it ensures theoretically that the weighted LK algorithm behaves in a stable manner during alignment/tracking. For the specific object-class evaluated (i.e., faces) the NSKM has demonstrated superior convergence properties to the canonical SVT approach [1] and is considerably more computationally efficient due its sparser nature. Our approach also exhibits superior alignment performance when compared to leading generative approaches [5] for dealing with appearance variation within the weighted LK algorithm.

Our approach enjoys advantages over existing approaches [13, 16, 15] to discriminative gradient descent object alignment/tracking as it: (i) is based on the well understood Gauss-Newton optimization framework, (ii) enjoys fast convergence without the requirement for specifying a step-size, and (iii) is adaptive to the current iterative warp estimate. Future work shall try and extend our work to other more complex warps (e.g., piece-wise affine) and also extend our current method for learning a discriminative positive semidefinite weight matrix.

References

[1] S. Avidan. Support vector tracking. *IEEE Trans. PAMI*, 26(8):1064–1072, August 2004.

[2] S. Baker, R. Gross, and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 3. Technical Report CMU-RI-TR-03-01, Robotics Institute, Carnegie Mellon University, 2003.

[3] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1: The quantity approximated, the warp update rule, and the gradient descent approximation. *International Journal of Computer Vision*, 2004.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] 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.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[8] R. Gross, S. Baker, and I. Matthews. Generic vs. person specific active appearance models. *Image and Vision Computing*, 23(11):1080–1093, November 2005.

[9] R. Gross, I. M. S. Baker, and T. Kanade. The CMU Multiple pose, illumination and expression (MultiPIE) database. Technical Report CMU-RI-TR-07-08, Robotics Institute, Carnegie Mellon University, 2007.

[10] 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.

[11] D. J. Hudson. Least-squares fitting of a polynomial constrained to be either non-negative, non-decreasing or convex. *Journal of the Royal Statistical Society. Series B*, 31(1):113–118, 1969.

[12] C. J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19(10):2756–2779, October 2007.

[13] X. Liu. Generic face alignment using boosted appearance model. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

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

[15] J. Saragih and R. Goecke. A nonlinear discriminative approach to AAM fitting. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[16] O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Trans. PAMI*, 27(8):1292–1304, August 2005.

A. Equating the SVT and LK Equations

In Avidan's original work [1] (section 4.1.2., Equation 6) he demonstrated that for a translational warp $\mathcal{W}(\mathbf{x};\mathbf{p}) = [x + p_1, y + p_2]^T$ we arrive at the following equation for aligning the source image Y with an SVM employing a homogeneous quadratic kernel $k(\mathbf{a},\mathbf{b}) = (\mathbf{a}^T\mathbf{b})^2$,

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \Delta p_1 \\ \Delta p_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{13}$$

where,

$$\begin{aligned}
A_{11} &= \sum_{j=1}^{l} \alpha_j (T_j(\mathbf{z})^T Y_x(\mathbf{z}'))^2 \\
A_{12} &= A_{21} = \sum_{j=1}^{l} \alpha_j T_j(\mathbf{z})^T Y_x(\mathbf{z}') T_j(\mathbf{z})^T Y_y(\mathbf{z}') \\
A_{22} &= \sum_{j=1}^{l} \alpha_j (T_j(\mathbf{z})^T Y_y(\mathbf{z}'))^2 \\
b_1 &= -\sum_{j=1}^{l} \alpha_j T_j(\mathbf{z})^T Y_x(\mathbf{z}') T_j(\mathbf{z})^T Y(\mathbf{z}') \\
b_2 &= -\sum_{j=1}^{l} \alpha_j T_j(\mathbf{z})^T Y_y(\mathbf{z}') T_j(\mathbf{z})^T Y(\mathbf{z}')
\end{aligned}$$

given $Y(\mathbf{z}')$ is short notation for the source image vector $Y(\mathcal{W}(\mathbf{z};\mathbf{p}))$ and $Y_x(\mathbf{z}')$ and $Y_y(\mathbf{z}')$ are the x- and y-derivatives of the source image at warp $\mathbf{p}$. $T_j(\mathbf{z})$ are the support vectors from the SVM, α_j are the alpha weights and l is the number of support vectors. Δp_1 and Δp_2 are the translation warp parameters we are attempting to estimate in the x- and y-directions respectively.

It is trivial to show that Equation 13 is just a rearrangement of the standard LK solution shown previously in Equation 4 and for completeness shown here,

$$\mathbf{H}\Delta\mathbf{p} = \mathbf{J}\mathbf{D}\left[T(\mathbf{z}) - Y(\mathcal{W}(\mathbf{z};\mathbf{p}))\right] \tag{14}$$

where,

$$\mathbf{H} = \mathbf{J}^T\mathbf{D}\mathbf{J} = -\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \tag{15}$$

$$\mathbf{J}\mathbf{D}\left[\mathbf{0} - Y(\mathcal{W}(\mathbf{z};\mathbf{p}))\right] = -\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{16}$$

and $\Delta\mathbf{p} = [\Delta p_1, \Delta p_2]^T$. The Jacobian matrix $\mathbf{J}$ and weight matrix $\mathbf{D}$ are defined previously in Equations 6 and 9 respectively. Note, that we are subtracting $Y(\mathcal{W}(\mathbf{z};\mathbf{p}))$ by a template $T(\mathbf{z})$ in Equation 14 which is considered to be zero in Avidan's SVT formulation (see Equation 16). An advantage of our reformulation over the one originally proposed by Avidan is that it becomes trivial to employ more complex warps (e.g., affine) and we can study the SVT algorithm from the perspective of the weighted LK algorithm.