Efficient spring system optimization for part-based visual tracking

Alan Lukežič , Luka Čehovin , Matej Kristan

Faculty of Computer and Information Science, University of Ljubljana
alan.lukezic@gmail.com, {luka.cehovin, matej.kristan}@fri.uni-lj.si

Abstract

Part-based trackers typically use visual and geometric constraints to find the most optimal positions of the parts in the constellation. Recently, spring systems was successfully applied to model these constraints. In this paper we propose an optimization method developed for multi-dimensional spring systems, which can be integrated in the part-based tracking model. The experimental analysis shows that our optimization method outperforms the conjugated gradient descend optimization in terms of convergence speed, accuracy and numerical stability.

1 Introduction

Visual object tracking is a fundamental computer vision problem where a trajectory of the object is estimated through the sequence of frames. The problem is challenging for numerous reasons, including appearance change (e.g., object deformation), occlusion (or self-occlusion), object or camera motion, illumination change of the scene and size change of the object. Visual trackers can be divided into two groups, depending on how they model the target, e.g., holistic and part-based trackers. Holistic trackers present the target as a whole, while part-based trackers present the target with the constellation of several patches. Part-based trackers are known by the robust performance and they are typically efficient in handling the occlusion or target deformation [3, 2] and have demonstrated top performance on recent visual tracking challenges [5, 8]. Part-based trackers typically localize the target by optimizing a tradeoff between the visual and geometric agreement between the model and image. For tractability, most of the recent trackers use star-based topology, e.g. [6, 2, 9, 4], or local connectivity, e.g. [3], instead of a fully-connected constellation [1], but at a cost of a reduced power of the geometric model.

Recently, we presented a deformable-part tracking as the optimization of the spring system [7]. Optimization was formulated as the energy minimization of the spring system, by the conjugated gradient descend. In this paper we propose a novel optimization for a multi-dimensional spring system energy minimization, called the iterative direct approach (IDA).

2 Problem formulation

Part-based trackers represent the target as a geometrically constrained constellation of N_p parts $\mathbf{X}_t = \{\mathbf{x}_t^{(i)}\}_{i=1:N_p}$. Each part $\mathbf{x}_t^{(i)}$ is specified by the part visual model (template) $\mathbf{z}_t^{(i)}$. The probability of a constellation being at state $\mathbf{X}_t$ conditioned on the measurements $\mathbf{Y}_t$ and parameters of the deformation model Θ is decomposed into

$$p(\mathbf{X}_t|\mathbf{Y}_t,\Theta) \propto p(\mathbf{Y}_t|\mathbf{X}_t,\Theta)p(\mathbf{X}_t|\Theta). \quad (1)$$

The density $p(\mathbf{Y}_t|\mathbf{X}_t,\Theta)$ is the *measurement constraint* term, reflecting the agreement of measurements with the current state $\mathbf{X}_t$ of constellation, whereas the second term, $p(\mathbf{X}_t|\Theta)$, reflects the agreement of the constellation with the *geometric constraints*.

Geometric constraints. The constellation is specified by a set of links $(i,j) \in \mathcal{L}$ indexing the connected pairs of parts (Figure 1). The parts and links form an undirected graph, and the joint pdf over the part states can be factored over the links as

$$p(\mathbf{X}_t|\Theta) = \prod_{(i,j)\in\mathcal{L}} \phi(||d_t^{(i,j)}||; \mu^{(i,j)}, k^{(i,j)}), \quad (2)$$

where $d_t^{(i,j)} = \mathbf{x}_t^{(i)} - \mathbf{x}_t^{(j)}$ is a difference in positions of the linked parts, $\mu^{(i,j)}$ is the preferred distance between the pair of parts and $k^{(i,j)}$ is the intensity of this constraint. The factors in (2) are defined as Gaussians $\phi(\cdot;\mu,k)$ with mean μ and variance k to reflect the property that deviations from the preferred distances should decrease the probability (2).

Measurement constraints. Given a fixed part state, $\mathbf{x}_t^{(i)}$, the measurement at that part is independent from the states of other parts and the measurement probability decomposes into a product of per-part visual likelihoods

$$p(\mathbf{Y}_t|\mathbf{X}_t,\Theta) = \prod_{i=1:N_p} p(\mathbf{y}_t^{(i)}|\mathbf{x}_t^{(i)},\Theta). \quad (3)$$

The visual likelihoods are chosen from the same family of pdfs as factors in (2) for tractability. Let $\mathbf{x}_{tA}^{(i)}$ be the position in vicinity of $\mathbf{x}_t^{(i)}$ that maximizes the similarity of the visual model $\mathbf{z}_t^{(i)}$ and the measurement $\mathbf{y}_t^{(i)}$ (see Figure 1, left). The visual likelihood can then be defined as a Gaussian $p(\mathbf{y}^{(i)}, |\mathbf{x}^{(i)},\Theta) = \phi(||d_t^{(i)}||; 0, k^{(i)})$ where $d_t^{(i)} = \mathbf{x}_t^{(i)} - \mathbf{x}_{tA}^{(i)}$ is the difference of the part current state

and its visually-ideal position, and $k^{(i)}$ is the intensity of this constraint. The visual likelihood thus reflects the agreement of the visual model $\mathbf{z}^{(i)}$ with the measurement $\mathbf{y}^{(i)}$ at position $\mathbf{x}_t^{(i)}$.

The geometric and measurement constraints (2 and 3) represent lead to an exponential posterior

$$p(\mathbf{X}_t|\mathbf{Y}_t, \Theta) \propto \exp(-E). \tag{4}$$

Maximization of the posterior (4) is equivalent to the minimization of the spring system energy, defined as

$$E = \frac{1}{2} \sum_{i=1:N_p} k_t^{(i)} \left\| d_t^{(i)} \right\|^2 + \sum_{i,j \in \mathcal{L}} k_t^{(i,j)} (\mu_t^{(i,j)} - \left\| d_t^{(i,j)} \right\|)^2, \tag{5}$$

where N_P represents the number of static (anchor) parts in a spring system, $k_t^{(i)}$ and $\left\| d_t^{(i)} \right\|$ represent the stiffness and length of i-th static spring, respectively. Set of dynamic nodes connected to i-th dynamic node is denoted as $\mathcal{L}$ and stiffness of the spring connecting i-th and j-th dynamic nodes is denoted as $k_t^{(i,j)}$. The nominal length of that spring is denoted as $\mu_t^{(i,j)}$ and current length as $\left\| d_t^{(i,j)} \right\|$. The described terms are illustrated on a spring system in Figure 1. Note that the static nodes represent the measurements constraints and they remain on fixed positions in the image during the optimization. Dynamic nodes represent the geometric constraints and they change their positions during the optimization. In Figure 1 dynamic nodes are denoted as circles, while static nodes are denoted as black squares.

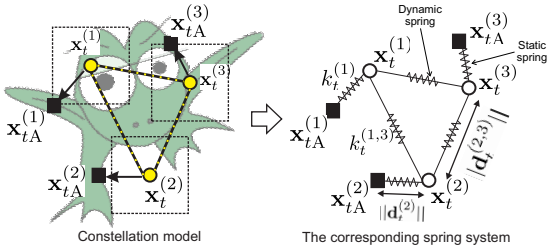

Figure 1: Example of a constellation model with rectangular parts and arrows pointing to most visually similar positions (left) and the dual form corresponding to a spring system (right).

3 Efficient spring system optimization

Our approach proceeds iteratively minimizes the energy of a spring system by decomposing the 2D spring system into two 1D independent systems (Figure 2), minimizing the separate energies in a closed form and re-composing the 2D system. This procedure is iterated until convergence. In the following an efficient algebraic closed-form solution for a 1D spring system is derived.

The forces at springs of a 1D system are defined as

$$\mathbf{F}_{\text{springs}} = -\mathbf{K}(\mathbf{A}\mathbf{x} - \mathbf{L}), \tag{6}$$

where $\mathbf{K} = \text{diag}([k_1, \cdots, k_{N_{\text{springs}}}])$ is a diagonal matrix of spring stiffness coefficients. The terms in the parenthesis in (6) represent the spring displacements, where

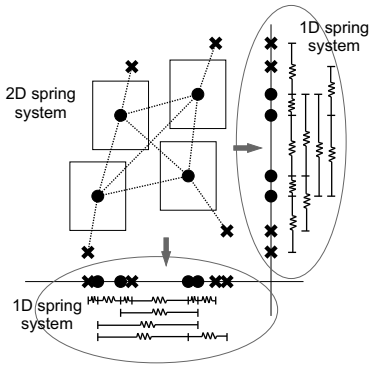

Figure 2: Example of decomposition of a 2-D spring system with 4 dynamic (circles) and 4 static nodes (crosses) on two 1-D spring systems. Each 1-D spring system can be solved in a closed-form.

$\mathbf{x}$ is a vector of the 1-D positions of the nodes, $\mathbf{L} = [l_1, \cdots, l_N]^T$ is the current-lenght vector , where l_i is the length of the i-ith spring. Elements in the vector of positions $\mathbf{x}$ are arranged in the following form

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\text{dyn}} \\ \mathbf{x}_{\text{stat}} \end{bmatrix}, \tag{7}$$

where $\mathbf{x}_{\text{dyn}}$ and $\mathbf{x}_{\text{stat}}$ are 1D positions of the dynamic and static nodes, respectively. The connectivity matrix $\mathbf{A}$ represents the directed connections between the nodes with the dimensionality $N_{\text{springs}} \times N_{\text{nodes}}$. If i-th spring connects a pair of nodes $\{n_{i1}, n_{i2}\}$, than the element a_{ij} of the matrix $\mathbf{A}$ is defined as

$$a_{ij} = \begin{cases} 1 & ; j \equiv n_{i1} \\ -1 & ; j \equiv n_{i2} \\ 0 & ; \text{otherwise} \end{cases} \tag{8}$$

Equation (6) can be rewritten in to the form $\mathbf{F}_{\text{springs}} = -\mathbf{K}\mathbf{A}\mathbf{x} + \mathbf{K}\mathbf{L}$. The forces in the nodes are given by left-multiplying by $\mathbf{A}^T$, obtaining an equation

$$\mathbf{F}_{\text{nodes}} = -\mathbf{A}^T\mathbf{K}\mathbf{A}\mathbf{x} + \mathbf{A}^T\mathbf{K}\mathbf{L}. \tag{9}$$

Equilibrium of the system is reached when forces in nodes vanish, resulting in the following linear system

$$\mathbf{A}^T\mathbf{K}\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{K}\mathbf{L}. \tag{10}$$

For efficient calculation of $\mathbf{x}$ two more matrices are defined, $\widehat{\mathbf{K}} = \mathbf{A}^T\mathbf{K}\mathbf{A}$ and $\mathbf{C} = \mathbf{A}^T\mathbf{K}$. Since the nodes position vector $\mathbf{x}$ is defined in a special form (7), the matrix $\widehat{\mathbf{K}}$ can be written as

$$\widehat{\mathbf{K}} = \left[\begin{array}{cc} \widehat{\mathbf{K}}_{\text{dyn}} & \widehat{\mathbf{K}}_{\text{stat}} \\ & \widehat{\mathbf{K}}_{\text{rem}} \end{array}\right]. \tag{11}$$

The remainder matrix $\widehat{\mathbf{K}}_{\text{rem}}$ does not affect computation of static nodes and is ignored. The matrix $\widehat{\mathbf{K}}$ has the dimensionality $N_{\text{nodes}} \times N_{\text{nodes}}$ and matrix $\widehat{\mathbf{K}}_{\text{dyn}}$ is defined as

$$\widehat{\mathbf{K}}_{\text{dyn}} = \begin{bmatrix} \widehat{k}_{1,1} & \cdots & \widehat{k}_{1,N_{\text{dyn}}} \\ \vdots & \cdots & \vdots \\ \widehat{k}_{N_{\text{dyn}},1} & \cdots & \widehat{k}_{N_{\text{dyn}},N_{\text{dyn}}} \end{bmatrix}, \tag{12}$$

where element $\widehat{k}_{i,j}$ represents an element from $\widehat{\mathbf{K}}$ and N_{dyn} is the number of dynamic nodes. The matrix $\widehat{\mathbf{K}}_{\text{stat}}$ is defined in the same manner as $\widehat{\mathbf{K}}_{\text{dyn}}$,

$$\widehat{\mathbf{K}}_{\text{stat}} = \begin{bmatrix} \widehat{k}_{1,N_{\text{dyn}+1}} & \cdots & \widehat{k}_{1,N_{\text{nodes}}} \\ \vdots & \cdots & \vdots \\ \widehat{k}_{N_{\text{dyn}},N_{\text{dyn}+1}} & \cdots & \widehat{k}_{N_{\text{dyn}},N_{\text{nodes}}} \end{bmatrix}. \quad (13)$$

Note that the dimensions of the matrix $\widehat{\mathbf{K}}_{\text{stat}}$ are $N_{\text{dyn}} \times N_{\text{stat}}$, where N_{stat} is the number of static nodes in the spring system. The matrix $\mathbf{C}$ has the dimensionallity $N_{\text{nodes}} \times N_{\text{springs}}$ and it can be, similarly as $\widehat{\mathbf{K}}$, written in to the form $\mathbf{C} = \left[\mathbf{C}_{\text{dyn}}^T, \mathbf{C}_{\text{stat}}^T\right]^T$. Following the notation in (12) and (13) the elements of the matrix $\mathbf{C}$ are denoted as $c_{i,j}$ and $\mathbf{C}_{\text{dyn}}$ is defined as

$$\mathbf{C}_{\text{dyn}} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,N_{\text{springs}}} \\ \vdots & \cdots & \vdots \\ c_{N_{\text{dyn}},1} & \cdots & c_{N_{\text{dyn}},N_{\text{springs}}} \end{bmatrix}. \quad (14)$$

As before, we can ignore $\mathbf{C}_{\text{stat}}$ since it does not enter computation of the dynamic nodes positions. Using the matrices $\widehat{\mathbf{K}}$ and $\mathbf{C}$, the spring system from (10) can be written as $\widehat{\mathbf{K}}\mathbf{x} = \mathbf{C}\mathbf{L}$. If we take into account that static nodes do not change their positions during the optimization the system can be decomposed into the form

$$\widehat{\mathbf{K}}_{\text{dyn}}\mathbf{x}_{\text{dyn}} + \widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}} = \mathbf{C}_{\text{dyn}}\mathbf{L}. \quad (15)$$

The closed-form solution for the dynamic nodes is therefore

$$\mathbf{x}_{\text{dyn}} = \widehat{\mathbf{K}}_{\text{dyn}}^{-1}(\mathbf{C}_{\text{dyn}}\mathbf{L} - \widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}). \quad (16)$$

To summarize our iterative direct approach (IDA): At each iteration, 2D system is decomposed into two 1D systems, each system is solved by (16) and then the 2D system is re-assembled. This process is iterated until convergence (Algorithm 1). Note that the term $\widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}$ is independent from the dynamic nodes and can be precomputed before entering the iterations.

Algorithm 1 : Optimization of a 2-D spring system.

Require:
Positions of the dynamic $\mathbf{x}_{\text{dyn}}$ and static nodes $\mathbf{x}_{\text{stat}}$, stiffness vector $\mathbf{k}$ and adjacency matrix $\mathbf{A}$, defined as in (8).
Ensure:
New positions of the dynamic nodes $\mathbf{x}_{\text{dyn}}$.
Procedure:
1: Construct stiffness matrix $\mathbf{K} = diag(\mathbf{k})$.
2: Calculate matrices $\widehat{\mathbf{K}} = \mathbf{A}^T\mathbf{K}\mathbf{A}$ and $\mathbf{C} = \mathbf{A}^T\mathbf{K}$.
3: Compose $\widehat{\mathbf{K}}_{\text{dyn}}$, $\widehat{\mathbf{K}}_{\text{stat}}$, and $\mathbf{C}_{\text{dyn}}$ according to (12), (13), (14).
4: Calculate the product $\widehat{\mathbf{K}}_{\text{stat}}\mathbf{x}_{\text{stat}}$ for each dimension.
5: **while** stop condition **do**
6: For each dimension do:
7: Extract positions of dynamic nodes for a selected dimension from $\mathbf{x}_{\text{dyn}}$.
8: Calculate vector of current spring lengths $\mathbf{L}$ (the distance between connected nodes) along selected dimension.
9: Solve (16) and update $\mathbf{x}_{\text{dyn}}$.
10: **end while**

4 Experimental analysis

This section compares the standard baseline method, i.e., the conjugated gradient descend (CGD) with the proposed iterated direct approach (IDA). To allow controlled experimental analysis of the optimization approach, a simulated fully-connected spring system with four dynamic nodes and four anchor nodes was used. We have analyzed the performance of the optimization methods by averaging over 100,000 randomly generated spring systems and their displacements. The positions of all nodes were each time randomly perturbed around the initial positions. Four dynamic nodes were initialized on the positions (0,0), (0,1), (1,0) and (1,1). Each node was displaced by the vector $\mathbf{d} = [d_x, d_y]$, where d_x and d_y were sampled from uniform distribution $\mathcal{U}([-0.5; 0.5])$. Each anchor node was set by displacing the corresponding dynamic node by the vector $\mathbf{b} = [b_x, b_y]$, where b_x and b_y were sampled from uniform distribution $\mathcal{U}([-0.25; 0.25])$. The stiffness of i-th dynamic spring was set as $k_i = (\sigma d_i)^{-2}$, where d_i represents the length of the spring and $\sigma = 0.1$ is the size change parameter. The stiffness of j-th static spring was set as $k_j = \frac{1}{2} + u_j\overline{d}_{\text{dyn}}$, where $\overline{d}_{\text{dyn}}$ represents the average stiffness of the dynamic springs and u_j is the average number sampled from the uniform distribution $\mathcal{U}([0; 1])$. The performance of the optimization method was measured by: (i) the number of iterations and time needed to reach the stable state and (ii) the remaining energy of the optimized spring system.

Table 1 summarizes the performance analysis of CGD and IDA. The first and second columns represent average, standard deviation and median of the number of iterations needed to reach the stable state and the amount of energy that remains in the spring system after the optimization, respectively. The last column shows average time (in milliseconds) needed for convergence of a method (both methods were implemented in Matlab). The results show that the proposed IDA converges is half as many iterations compared to CGD. The remaining energy in a spring system does not change much between methods, which means the both methods reach a global minimum. This is not surprising due to convexity of the optimization function. Figure 3 shows how energy of the spring system drops in each iteration for both methods. It is clear that energy at IDA drops much faster in the beginning of the optimization which is key reason for fast performance. The proposed IDA is in average approximately more than three-times faster than the CGD. With less than 3ms needed for optimization a single spring system, this method can be used in tracking tasks without entailing a significant overhead.

	Number of iterations			**Energy**			**Time**
Method	Avg	Std	Median	Avg	Std	Median	Avg [ms]
IDA	12.75	9.32	10	1.22	3.34	0.83	2.92
CGD	28.01	9.90	28	1.28	3.34	0.83	9.72

Table 1: Comparison of the proposed IDA and CGD.

We have also analyzed the average time required for the initialization and for iterating phases of both meth-

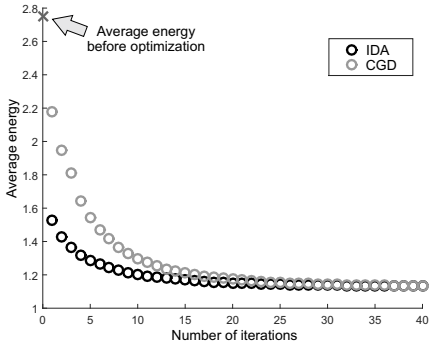

Figure 3: Spring system energy for 40 iteration. Experiment is averaged over 10,000 spring systems.

ods. The first column in Table 2 shows average time in milliseconds for each phase. The results show that initialization of the optimization method is faster at IDA. The last column shows the time needed for a single iteration, which is also faster at IDA. Note that the Table 2 was obtained with the same experiment settings as the Table 1.

	Average time [ms]				
Method	Init	Iterations	Total	Avg. Iterations	ms per-iteration
IDA	0.18	2.74	2.92	12.75	0.23 (0.215)
CGD	0.35	9.37	9.72	28.01	0.35 (0.347)

Table 2: Average time for method initialization, iterations and sum of both is given in the table. Last column represent the average time for one iteration. Number in parenthesis is the average time for one iteration without considering the initialization phase.

We have also observed that the proposed IDA is numerically more stable than the CGD. Figure 4 shows an example of a spring system for which CGD does not reach the optimal state. The spring system consists of four static and four dynamic nodes. Black lines represent dynamic springs of an initial spring system and the gray lines represent dynamic springs of spring system after optimization. Small circles denote the dynamic nodes. Centers of the larger circles (crosses) represent anchor nodes and the radius of circles represent the variance of each node. The dotted lines are static springs before and after optimization. The energy of the final optimized spring system is also given in the figures. Note that the IDA converged to a stable state with much lower energy, while CGD method did not reach that state. The poor performance of CGD can be explained with the initial positions of the nodes. Since there are two nodes very close to each other, the CGD becomes numerically unstable. There is also a huge difference in number of steps needed for terminating the optimization e.g., the proposed IDA converged in only 5 iterations whereas the CGD required 471 iterations. Note that the unstable behavior of CGD was automatically detected during the experiment and these measurement were not considered in the calculation in Table 1.

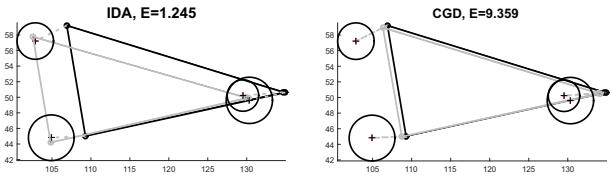

Figure 4: Comparison of both optimization methods on a numerically unstable initial spring system. The black color represents initial and the gray color represents optimized spring system.

5 Conclusion

An efficient optimization of a deformable parts model was proposed. The cost function is formulated as an equivalent spring system and an iterative direct approach (IDA) is derived for minimizing the energy of such a system. Experimental analysis showed that IDA is more than three-times faster and numerically more stable than the conjugate gradient descent on randomly generated, fully connected spring systems. In average, it took less than 3ms for the optimization, which is important for real-time performance in modern trackers.

References

[1] N. M. Artner, A. Ion, and W. G. Kropatsch. Multi-scale 2d tracking of articulated objects using hierarchical spring systems. *Patt. Recogn.*, 44(4):800–810, 2011.

[2] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Li. Robust deformable and occluded object tracking with dynamic graph. *IEEE Trans. Image Proc.*, PP(99):1–1, 2014.

[3] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(4):941–953, Apr. 2013.

[4] M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *Comp. Vis. Image Understanding*, 117(10):1245–1256, 2013.

[5] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojir, and G. et al. Fernandez. The visual object tracking vot2014 challenge results. In *Proc. European Conf. Computer Vision*, 2014.

[6] J. Kwon and K. M. Lee. Tracking by sampling and integratingmultiple trackers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1428–1441, July 2014.

[7] A. Lukežič, L. Čehovin, and M. Kristan. A two-layer spring-system-based deformable parts model for visual tracking. Comp. Vis. Winter Workshop, February 2015.

[8] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Comp. Vis. Patt. Recognition*, 2013.

[9] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel. Part-based visual tracking with online latent structural learning. In *Comp. Vis. Patt. Recognition*, pages 2363–2370, June 2013.