DeepLabCut-based Multi-Objects Tracking Framework

Tabet Ehsainieh[1] *

[1]Student of Computer Science - University of Freiburg

Understanding flies movements is crucial for many neuroscience-related applications in the Straw Lab. Despite, the recent remarkable progress in object detection deep-learning-based-systems, many of them do not track across time, but rather process each frame individually. Furthermore, the absence of computational efficiency and important latencies inspired taking local motion estimation and thus, achieving higher performance and lower latency. DeepLabCut took lately huge attention in the neuroscience community by presenting an efficient method for markerless pose estimation based on transfer learning with deep neural networks that achieved excellent results with minimal training data. However, the framework was designed for single object detection and did not track over time. The goal of this project was to modify DeepLabCut inference to extract information not only about single fly location but also about similar looking objects hence, sieving flies from random clutter and taking local motion estimates to track multiple flies.

Image Processing | Kalman Filter | Hungarian Algorithm | Tracking By Detection| DeepLabCut| Deep Learning

Introduction. Automated tracking of animal movement allows analysis that would not be possible only by providing great quantities of data. Moreover, the capability of tracking in real-time allows detailed explorations of the neural basis for control of behavior in animals. Remarkably, DeepLabCut - an efficient method for markerless pose estimation based on transfer learning with deep neural networks - achieved excellent detection performance with minimal training data-set on test frames that is comparable to human accuracy. However, due to batch processing, it is not applicable in online scenarios where a target identity must be available at each time step, which is of high importance for some of the Straw Lab applications. In addition, the algorithm extracts the argument maximum per score-map (likelihood matrix per frame), as a consequence *tracking* only one animal per frame is possible. Online multi-object tracking is a fundamental problem in time-critical video analysis applications. The problem can be divided into 3 sub-problems: (i) objects detection, (ii) data association and (iii) objects tracking. Here we present a simple, computationally efficient real-time tracking framework that exploits the excellent DeepLabCut detectors performance, extending it to extract the local maxima using a non-maximum suppression-based filter and thus, solve the multi-object detection problem.

To deal with the other two sub-problems respectively, the framework performs Kalman filtering in image space and frame-by-frame data association using the Hungarian method with an association metric that measures the distance between the *Innovation*[25] *matrix* and observation.

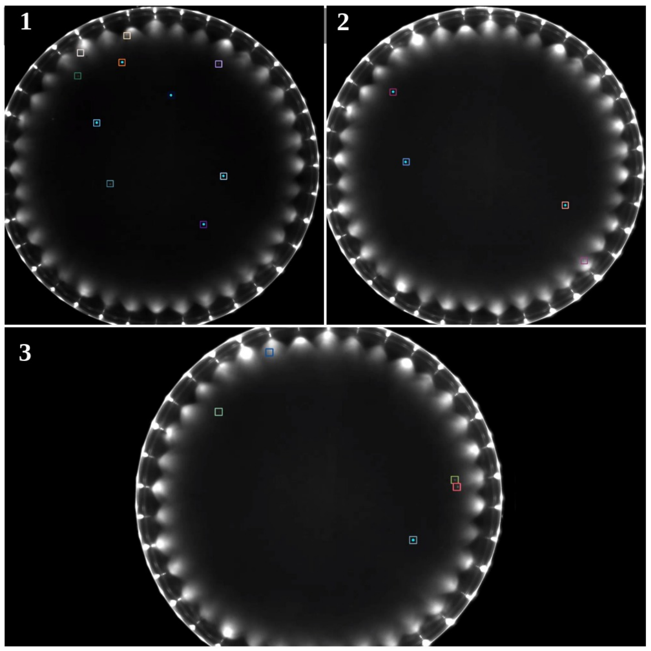


Fig. 1. Multiple object tracking .3 images taken from 3 different videos showing the end-results of the complete framework process starting with the DeepLabCut detection and ending with Kalman filtering local motion estimation. While the blue circles represent the DeepLabCut detection of the flies, the bounding boxes constitute the prediction, where each fly has a unique color symbolising its identity. All Figures demonstrate flies tracking in a darkened arena. These images extracted randomly out of the video. In **Fig 1.1** we experimented on 10 flies, one can see that mostly our predictions were highly accurate, even with missing measurements (missing blue circles). Notice that the one detected fly in has a dark-colored associated bounding box. In **Fig 1.2** a small number of flies were tracked, precisely 3, however, we notice 4 bounding boxes, due to false-alarm continuous measurements. **Fig 1.3** demonstrate a situation where the flies were close to each other. Although in this particular frame measurements were missing, the framework was able to track both of them with no change of identity (same colors).

Framework overview. In this work, the goal was to integrate the mentioned groundwork into a traditional tracking-by-detection framework. Therefore, it is essential to clarify briefly the functionality of DeepLabCut inference process. DeepLabCut is a deep convolutional network which relays on two algorithms for object recognition and semantic segmentation: pre-trained ResNets and deconvolutional layers. The network embodies a version of ResNets, whose weights were trained on that popular ImageNet, on which it achieves excellent performance according to their paper. The deconvolutional layers are used to increase the sampling rate of the visual information and produce spatial probability densities, representing "evidence" that an animal or body part is in

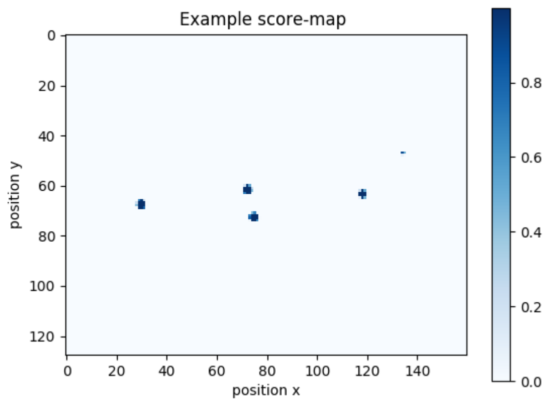


Fig. 2. Local maximun extraction. Example of a score-map extracted from the DeepLabCut inference. Network applied labels marked with darkened blue colour, while low confidence pixels marked the snow white colour.

a particular location. The probability densities are assigned during the training, the weights are adjusted iteratively such that for a given frame the network assigns high probabilities to labeled animal location and low probabilities elsewhere (Fig. 2). Thereby, the network is rewired and 'learns' feature detectors for the labeled animals. The combination between the initialized ResNet and the used data-set for training assure that this rewiring is robust and data-efficient.
The network detects presumably a scale of high probabilities representing other animals or look-alike objects, which allows modeling the probability map as a function with multiple local maxima. Afterwards, candidates - *detections* - will be collected from the score-maps, all of the candidates will be assigned as new births *tracks*, each having a unique identity, Kalman filter instance initialized with the first observation and a constant signifying the amount of skipped frames because of no available observation. After that, a common conventional way to solve the association between the predicted Kalman states and newly arrived measurements is used. Into this problem formulation we integrate only motion information because of the lack of appearance information in the tested videos (Fig. 1). After solving the assignment problem, the *tracks* will be maintained and updated using the matched detection, only if their motion metric value is unreasonable (e.g. too high), otherwise, they will be reassigned. Measurements with no existing *tracks* will be assigned as new *births*. In the following section we describe the core components of the tracking system in greater detail.

Local maxima extraction. After looking attentively at a variety of probability densities maps produced by the inference algorithm of DeepLabCut, with a random number of animals, it turned out that they do possess information about the likelihood of the different objects with similar features to the labeled object (fly) in the training phase.

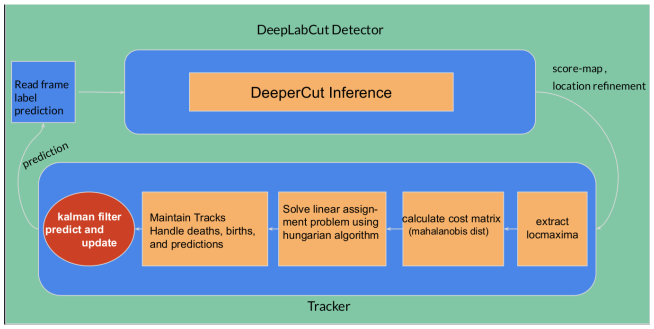

DeepLabCut Detector
Read frame label prediction
DeeperCut Inference
score-map , location refinement
prediction
kalman filter predict and update
Maintain Tracks Handle deaths, births, and predictions
Solve linear assignment problem using hungarian algorithm
calculate cost matrix (mahalanobis dist)
extract locmaxima
Tracker

Fig. 3. MOT Framework. Illustration of the Framework pipeline. This figure describe a typical cycle of the framwrok.

To exploit this fact, one can model the problem as follows:
Let (X, d_x) be a metric space[1] and the score-map a function $f : X \to [0,1]_{\subset\mathbb{R}}$, then $x_0 \in X$ is a local maximum point of the function f if $(\exists \varepsilon > 0)$ such that : $(\forall x \in X) d_x(x, x_0) < \varepsilon \; f(x_0) \geq f(x)$.

The only challenging problem with this mathematical formation is, choosing the adequate metric.
The flies trajectories in the score-maps are represented by a chunk of pixels with a higher probability than the surrounding background (Fig. 4). Thus, it is crucial for a successful tracking to get one *detection* per chunk of pixels - with a higher probability (fly) - otherwise, there would be more bounding boxes (tracks) per fly. After a series of experiments using different local maxima extraction methods[2], only one showed promising results. By plotting the score-map, we concluded that the pixels with the higher probability have always the brightest colors, on the opposite side the background has the darkest color, hence that, using the Maximum multidimensional filter and the Minimum multidimensional filter can achieve the desired results.
The used method: Using the Maximum filter, it is possible to enhance bright values in the score-map by increasing its area. Similar to a dilate function, each 5x5 is processed for the brightest surrounding pixel. That brightest pixel then becomes the new pixel value at the center of the window. On the other hand, the Minimum filter enhances dark values in the image by increasing its area, therefore, the difference between them up to a certain threshold showed a better local maxima extraction. To reduce the extractions error rate, the irrelevant values in the score-maps was changed to 0. By finding the connected components in the modified score-map, we will have the actual flies-*detections*. Note: the implemented algorithm, the *centroid* of those was extracted.

Cost Metric. A traditional way to solve the association between the predicted Kalman states and newly arrived measurements is to build an assignment problem that can be

[1]a **metric space** is a set together with a metric on the set. The metric is a function that defines a concept of distance between any two members of the set, which are usually called points.
[2]Implementations of the different methods

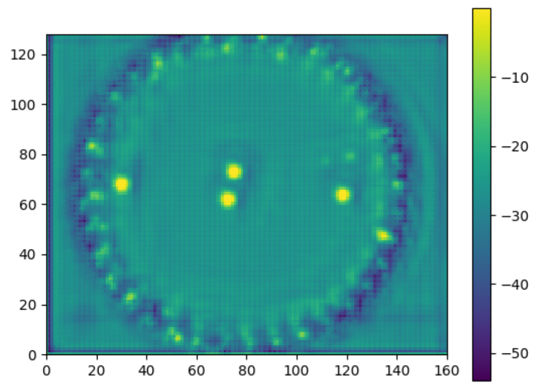


Fig. 4. Score-map analysis. Log transformation of a score-map extracted from a video of for flies in the arena we showed in Fig 1.

solved by using the Hungarian algorithm. Because of the lack of appearance information in the tested environment, the only information that could be integrated into this problem formulation is motion information. At first glance, the euclidean distance seemed to be a simple useful assignment problem, yet the experiments showed unreliability in situations where two measurements have the same distance. To incorporate motion information, a better motion metric was necessary, here, the Mahalanobis distance comes to light.

The MD is a statistical distance between probability distribution D and a measurement m. Basically, it takes for each of those components *state estimation uncertainty* into account when determining its distance to the corresponding center *mean*. The main idea lays in measuring how many standard deviations is m away from the mean of D. So components with a higher variability receive less weight than components with low variability, in other words: the closer m to the mean of D, the smaller the distance will be.

Metric formulation: Mahalanobis distance can also be defined as a dissimilarity measure between two vectors $\vec{x}$ and $\vec{y}$ of the same distribution with the covariance matrix S:

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}. \tag{1}$$

This can be applied in our model. Let

$$z_j = (x_j, y_j) \tag{2}$$

be a *detection* from a set of *detections* $z_k \in Z$,

$$\beta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3}$$

a a helper matrix to project pixel space information,

$$a_i = p_i \cdot y \text{ where } p_i = (\tilde{x}, \tilde{y}, v_{xi}, v_{yi}) \tag{4}$$

is a Kalman prediction projected to the pixel space $(\tilde{x}, \tilde{y})$ out of a set of *predictions* $p_i \in P$ and

$$\hat{S} = \beta^T \cdot S \cdot \beta \tag{5}$$

Then the Mahlahonis distance between z_j and a_i is:

$$d(i,j) = \sqrt{(z_j - a_i)^T \hat{S}^{-1}(z_j - a_i)} \tag{6}$$

where S^{-1} is the inverse *Innovation matrix* S which, gives information about filter tuning and model accuracy (will be discussed later).

For tracking along image sequences, the Mahalanobis distance is a preferred assignment formulation, however this step is one of the most time-consuming, because it involves inverting a matrix and other arithmetic operations. To minimize the computational cost, one can exploit the fact that for a Gaussian distribution, the probability density of an observation is uniquely determined by the Mahalanobis distance d , precisely, d^2 is chi-squared distributed. For our four dimensional measurement space the corresponding Mahalanobis threshold is 9.4877 according to (1). Using this metric it is possible to exclude unlikely associations at a 95% confidence.

Note: the validation using the chi-squared was not published in the code for stability problems. Finally, while the Mahalanobis distance is a good association metric when dealing with low motion uncertainty, in our problem formulation the predicted state distribution obtained from the Kalman filter provides only a rough estimate of the object location. In some situations, the Mahalanobis will uninformed metric for tracking through occlusions. Therefore, contentious research will be taken in finding appearance metric.

Assignment Problem. One way to define object tracking is the act of associating the data representing an object in each time step in consecutive frames. Data association is evidently the key issue in tracking approaches, especially when talking about multiple object tracking. What makes this problem difficult is that it involves false alarms, detection uncertainty (occlusions, detector failure) and ambiguities (several possible measurements). There are two ways to formulate the problem: as a matrix or as a bipartite graph, here is the bipartite graph formulation.

Problem formulation:

Let

$$G = (P_{k-1}, Z_k, E_k) \tag{7}$$

be a bipartite Graph[3] with

$$P_{k-1} = \{p_i \mid 0 \le i < n, n \in \mathbb{N}\} \tag{8}$$

the set of predicted Kalman states vertices in frame $k-1$,

$$Z_k = \{z_j \mid 0 \le j < m, m \in \mathbb{N}\} \tag{9}$$

measurements vertices in frame k, $n \le m$,

$$E_k = \{e_{ij} \mid \text{between } p_i \in P_{k-1} \wedge z_j \in Z_k\} \tag{10}$$

[3] A bipartite graph, also called a bi-graph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.

is an edge set and $f\colon E_k \to \mathbb{R}$ is a cost function where $\forall e_{ij} \in E_k$ has a non negative cost

$$f(e_{ij}) = \sqrt{(z_j - a_i)^T \hat{S}^{-1}(z_j - a_i)} \qquad \textbf{(11)}$$

where $y_i = H \cdot p_i$ (see Kalman Filter).

The goal is to find a minimum weight matching [4].
Let $M_{n\times m}$ be a matrix representation of G, where $M(i,j) = c_k(i,j)$ the cost between vertex $p_i \in P_{k-1}$ and vertex $z_j \in Z_k$ and $A_{n\times n}$ be a helper matrix which holds information about the assignment e.g:

$$A(i,j) = \begin{cases} 1 & \text{iff } p_i \text{ is assigned to } z_j \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(12)}$$

Notice that each row is assigned to at most one column and each column is assigned to at most one row.
Then the optimal assignment has cost :

$$\gamma = \min \sum_{(i,j)\in A\times M} m_{ij}x_{ij} \qquad \textbf{(13)}$$

Also :

$$= \min_{\gamma \in Symm_n} Tr(\gamma(A_{ij}) \cdot M_{ij}^T) \qquad \textbf{(14)}$$

where $Symm_n$ the symmetric group over A_{ij} and $\gamma(A_{ij})$ has the form of the identity matrix E_n. Notice that in cases of $m > n$ the matrices will be is padded further with dummy values to be square matrices and at the end the dummy values will be include in the summation.

Further Data Association - Track Management. Data Association must consider that new measurements can show up in the field of view, or that a track can leave the field of view. In other words, in any given frame, some number of new *tracks* might need to be created and some number of existing *tracks* might need to be discarded. As we saw multiple objects tracking data association problem reduces to a minimum weight bipartite matching problem which is solved using the described method in the last section. However, because of the constraint that each row is assigned to at most one column, and each column is assigned to at most one row, we are left off with $m - n$ unmatched measurements, and in the worst-case scenario no adequate for the existing *tracks*. One way of handling unmatched measurements is to create a new track - *births* - from each of them after solving the linear assignment problem. Finally the matched *tracks* Kalman filter instances will be updated using their correspondent's measurements and use this information to predict (see Kalman Filter) To deal with the unmatched *tracks*, we keep track using the *a priori*[5] information that their Kalman filter instances hold. If they remain unmatched for a certain number of frames they will be deleted *deaths*.

[4]Given a bepartite graph $G = (V, K, E)$, a matching M in G is a set of pairwise non-adjacent edges $e \in E$, none of which are loops; that is, no two edges share a common vertex

Kalman Filter. Sensors are noisy, ML-based detectors are still imperfect and therefore, cannot be relayed on to output perfect information. Prediction enables better estimate, but it is also subjects to noise. In statistics, Kalman filter is a method that takes an input a series of observations over time, containing statistical noise and other uncertainties and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time step. Kalman filter is a Bayesian filter, Bayesian probability determines the truth-likelihood based on past information. Kalman filter uses a systems dynamic model, a dynamic system is a physical system whose state (position, temperature, etc) evolves over time.
The Kalman filter consists of two steps: predict and update. The predict step uses the state estimate from the previous frame $k-1$ to produce an estimate of the state at the current frame k (*a priori* state [5]). On the other hand, in the update step, the current *a priori* prediction is merely a combination of the current measurement and the prediction, which is to refine the state estimate.
Build a Model Kalman filter showed intriguing results in different applications.
Luckily, object tracking in 2D images can be modelled as a process of a physical system. Hence, Kalman filtering conditions fit our problem.

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \qquad \textbf{(15)}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \qquad \textbf{(16)}$$

It is easy to perceive that equation (15) is recursive equation. As a consequence it, can be evaluated by using a *linear stochastic equation*. Any x_k is a linear combination of its previous value plus a control signal B_k and a process noise w_k.
This can be applied to our model:

$$p_k = F_k \cdot \begin{bmatrix} x \\ y \\ \hat{x} \\ \hat{y} \end{bmatrix}_{k-1} + w_k \qquad \textbf{(17)}$$

where

- F_k: state transition matrix x_{k-1}, its is discrete, and represents a set of linear equations which transitions x_{k1} to x_k over a discrete time step δt.
- x the x dimension in the image, y the y dimension in the image and $\hat{x}$, $\hat{y}$
- w_k: process noise , white noise assumed
- u_k: control vector, which is initializing to the $\vec{0}$

[5] the estimation of the state for the current frame k does not include measurements from the current frame k

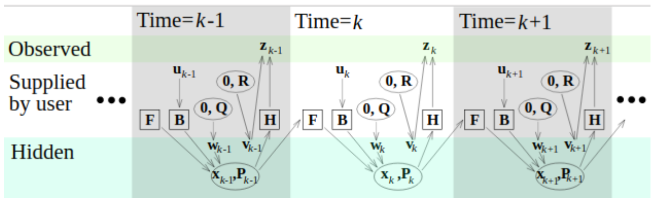


Fig. 5. Kalman Filter. Model underlying the Kalman filter. Squares represent matrices. Ellipses represent multivariate normal distributions (with the mean and covariance matrix enclosed). Unenclosed values are vectors. In the simple case, the various matrices are constant with time, and thus the subscripts are dropped, but the Kalman filter allows any of them to change each time step. source: Wikipedia

Finding this matrix is often quite difficult, however from simple physics equation it can be derived that:

$$F = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

As mentioned before, DeepLabCut detectors, on our trained Data-set (200 frames) could result in false-alarms or missing measurements in some frames, therefore, we need to consider measurement noise.

Equation (16) shows that any measurement value is a linear combination of the signal value p_k and the measurement noise.

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{p}_k + \mathbf{v}_k \tag{19}$$

The entities H_k is the observation model which maps the true state space into the observed space and v_k is the observation noise which is assumed to be *zero mean Gaussian white noise*[6] with covariance R_k: $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$

These entities are just numeric values. Also as an additional ease, those entities could change over time, but we can assume that they're constant and will be initializing as following:

$$H = E_4, R = \lambda \cdot E_4 \tag{20}$$

After initializing the state estimate, the mentioned two-step process will be discussed explicitly.

Typically, the two phases alternate (Fig. 5), with the prediction advancing the state until the next scheduled observation, and then correct our prediction using an observation:

Phase 1: Frame Update (prediction)

$$\hat{\mathbf{x}}_k = \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \tag{21}$$

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k \tag{22}$$

Phase 2: Measurement Update (correction)

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k \tag{23}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k \tag{24}$$

[6]In signal processing, white noise is a random signal having equal intensity at different frequencies, giving it a constant power spectral density

$$\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^{\mathsf{T}} \mathbf{S}_k^{-1} \tag{25}$$

$$\tilde{\mathbf{x}}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{26}$$

$$\tilde{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k \tag{27}$$

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k \tag{28}$$

The intuition behind equation (21) is straight-forward, it is a rough estimation before the measurement update correction also called *a prior estimate*[5]. Additionally, equation (22) is a metric to predict-measure the estimated state accuracy, which allows us to adjust our belief to account for the uncertainty in prediction, mathematically $\mathbf{P_k} = \operatorname{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k)$.

In contrast to phase 1, phase 2 is nontrivial, therefore, here is a derivation of phase 2 equation system. **Derivation of Phase 2**

The main goal of the Kalman filter is to minimize the Variance of $\mathbf{P}_k$:

$$\begin{gathered} \min \mathbf{Tr}(\mathbf{P}_k) = \min \sum_{i=1}^{n} p_{ii} \\ \text{Because: } \mathbf{P}_{init} = \begin{bmatrix} \varphi^2 & 0 & 0 & 0 \\ 0 & \varphi^2 & 0 & 0 \\ 0 & 0 & \varphi^2 & 0 \\ 0 & 0 & 0 & \varphi^2 \end{bmatrix} \end{gathered} \tag{29}$$

$\tilde{\mathbf{P_k}}$ is defined as the coveriance between the actual state x_k and the updated state $\tilde{x_k}$:

$$\begin{aligned} \tilde{\mathbf{P_k}} &= \operatorname{cov}(\mathbf{x}_k - \tilde{\mathbf{x}}_k) \\ &= \operatorname{cov}(\mathbf{x}_k - (\hat{\mathbf{x}}_k + \mathbf{K}_k \cdot \tilde{\mathbf{y}}_k))\ {}_{23} \\ &= \operatorname{cov}(\mathbf{x}_k - (\hat{\mathbf{x}}_k + \mathbf{K}_k \cdot (\mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k \cdot \hat{\mathbf{x}}_k))) \\ &= \operatorname{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k - \mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{x}_k - \mathbf{K}_k \cdot \mathbf{v}_k + \mathbf{K}_k \cdot \mathbf{H}_k \hat{\mathbf{x}}_k))) \\ &= \operatorname{cov}((\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k) - \mathbf{K}_k \cdot \mathbf{v}_k) \\ &= \operatorname{cov}((\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)) + \operatorname{cov}(\mathbf{K}_k \cdot \mathbf{v}_k) \\ &= (\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \operatorname{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k) \cdot (\mathbf{I_4} - \mathbf{K}_k \mathbf{H}_k)^T \\ &+ \mathbf{K}_k \cdot \operatorname{cov}(\mathbf{v}_k) \cdot \mathbf{K}_k^T \\ &= (\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \operatorname{cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k) \cdot (\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k)^T \\ &+ \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \\ &= (\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k) \cdot \mathbf{P}_k \cdot (\mathbf{I_4} - \mathbf{K}_k \cdot \mathbf{H}_k)^T \\ &+ \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \\ &= \mathbf{P}_k - \mathbf{P}_k \cdot \mathbf{K}_k^T \cdot \mathbf{H}_k^T - \mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{P}_k \\ &+ \mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{K}_k^T \cdot \mathbf{H}_k^T \cdot \mathbf{P}_k \\ &+ \mathbf{K}_k \cdot \mathbf{R}_k \cdot \mathbf{K}_k^T \\ &= \mathbf{P}_k - \mathbf{P}_k \cdot \mathbf{K}_k^T \cdot \mathbf{H}_k^T - \mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{P}_k \\ &+ \mathbf{K}_k \cdot (\mathbf{H}_k \cdot \mathbf{P}_k \cdot \mathbf{H}_k^T + \mathbf{R}_k) \cdot \mathbf{K}_k^T \\ &= \mathbf{P}_k - \mathbf{P}_k \cdot \mathbf{K}_k^T \cdot \mathbf{H}_k^T - \mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{P}_k \\ &+ \mathbf{K}_k \cdot \mathbf{S}_k \cdot \mathbf{K}_k^T\ {}_{24} \end{aligned} \tag{30}$$

Now we want to minimize the $\mathbf{P}_k$:

$$\begin{aligned}\frac{\nabla \mathbf{Tr}(\mathbf{P_k})}{\nabla \mathbf{K}_k} &= \frac{\nabla(\mathbf{P})}{\nabla \mathbf{K}} - \frac{\nabla(\mathbf{P}_k \cdot \mathbf{K}_k^T \cdot \mathbf{H}_k^T)}{\nabla \mathbf{K}} - \frac{\nabla(\mathbf{K}_k \cdot \mathbf{H}_k \cdot \mathbf{P}_k)}{\nabla \mathbf{K}} \\ &+ \frac{\nabla(\mathbf{K}_k \cdot \mathbf{S}_k \cdot \mathbf{K}_k^T)}{\nabla \mathbf{K}} \\ 0 &= -2 \cdot (\mathbf{H}_k \cdot \mathbf{P}_k) + 2 \cdot (\mathbf{K}_k \cdot \mathbf{S_k}) \\ \mathbf{K}_k &= \mathbf{P}_k \mathbf{H}_k^{\intercal} \mathbf{S}_k^{-1} {}_{25}\end{aligned} \tag{31}$$

Further intuitive explanations. ***S*** *matrix* is the *sensitivity matrix* that inversely weighs measurement errors. Further, equation (24) is called the *Kalman gain* is the relative weight given to the measurements and current state estimate, high gain, means high prediction error and thus the filter places more weight on the most recent measurements, alternatively the filter follows the model predictions more closely. The importance of the noise models in the Kalman filter was deliberately emphasized, it was showed in our initialisation of the observation noise model $\mathbf{R}$ and its derivation. Equation (22) consists of another noise matrix, the process noise matrix. In general the design of the Q matrix is among the most difficult aspects of Kalman filter design. Here we will show only rough its initialization in our model:

$$Q = \begin{bmatrix} \frac{dt^4}{4} & 0 & \frac{dt^3}{2} & 0 \\ 0 & \frac{dt^4}{4} & 0 & \frac{dt^3}{2} \\ \frac{dt^3}{2} & 0 & dt^2 & 0 \\ 0 & \frac{dt^3}{2} & 0 & dt^2 \end{bmatrix} \tag{32}$$

Implementation. In this framework, each *track* is initiated with a Kalman filter instance and it takes as an input a value $\mathbf{p}_0$ which is the center of the initial detected object (measurement) of the *track* object, other configurations are numeric constants that are initialized similarly for all the *tracks* as described in the last section. Other than that, in the last stage of the data association, matched *tracks* will be updated using the assigned detection as described in the Further Data Association - Track Management section.

Performance of the Kalman filter. Primarily, the framework was not tested enough to be able to conclude a steady-state performance, yet our few experiments showed promising results. In the next section, few plot diagrams will be presented to illustrate the performance of Kalman Filter on our framework.The framework was tested offline and on different compressed videos. In few videos similar objects were placed in the arena.

Speed Performance and accuracy performance: In this section two important features for the laboratory application will be evaluated: time complexity and accuracy. Since the end-goal is to integrate the framework into a live tracking system, time complexity and accuracy are essential. Of course the time complexity of the framework is not only affected by the Kalman filter, consequently, other component will be evaluated in the latter sections. In essence, the performance of Kalman filter will be demonstrated on single object target prediction and then on to end on multiple object tracking. For the single target prediction task, we are given the ground truth coordinates in each frame and we compare our predictions to the DeepLabCut detectors.

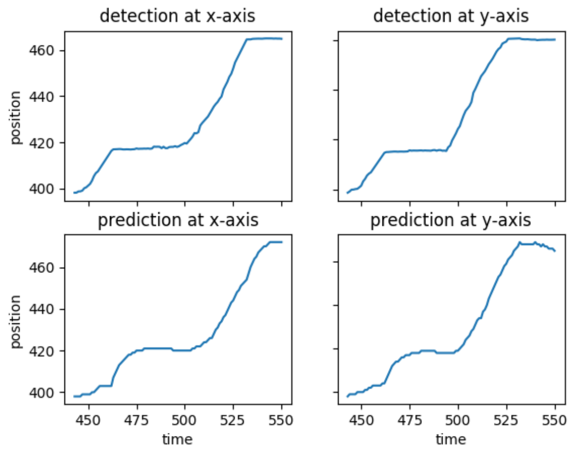
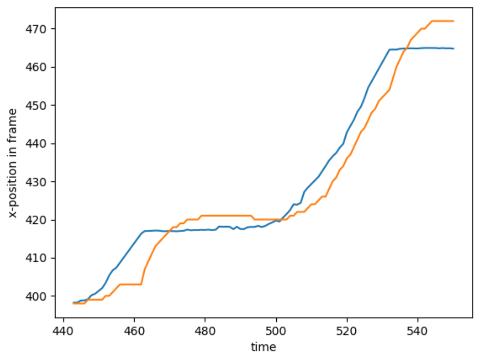


Fig. 6. Trajectories plots of tracking process of one fly. The 4 subplots diagrams demonstrate the accuracy performs of Kalman filter prediction against the DeepLabCut detectors.while the top two subplots illustrate the detectors performance over 20 second video, the other two subplots shows how accurate the Kalman filtering was

Evaluation results: Figure 6, and Figure 7 demonstrate the results of one fly tracking over 20 seconds recorded videos, here we show the accuracy performance of the framework and compare it with the detectors accuracy performance.

Framework evaluation and results. Here we demonstrate the practical value of the proposed framework by evaluating the framework as a unit. Figure 8 demonstrate the time complexity of the framework. Here we tested the framework on cluttered videos with different numbers of flies. Although, we tested the the time complexity with a limited data, we conclude that the amount of time it takes to run the algorithm is linear to the number of tracked objects (under the assumption that we are tracking small number of flies).

Conclusion. In this report, we have proposed and implemented an effective approach for multiple moving object tracking. Specifically, incorporating deep learning based detectors with local motion estimation to track multiple flies showed promising results, although DeepLabCut was designed to take pose estimation highly relaying on appearance information, which was absent in our target environment. Despite of the mentioned drawbacks, we can track through periods of occlusion, making the framework a suitable online tracking algorithms and fulfill the experiments needs.



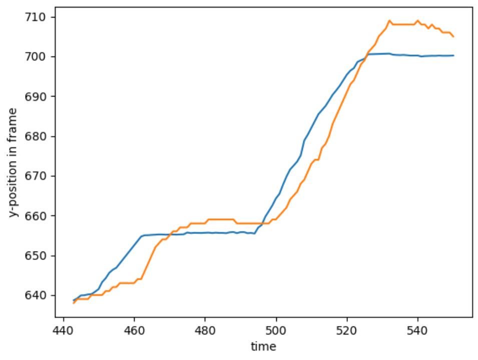


Fig. 7. Kalman filter vs. DeepLabCut detectors Trajectories comparison between the predictions (orange line) and the DeepLabCut detection (blue line) over 20 seconds video are shown. x, y trajectories of one fly match accurate with corresponding prediction.

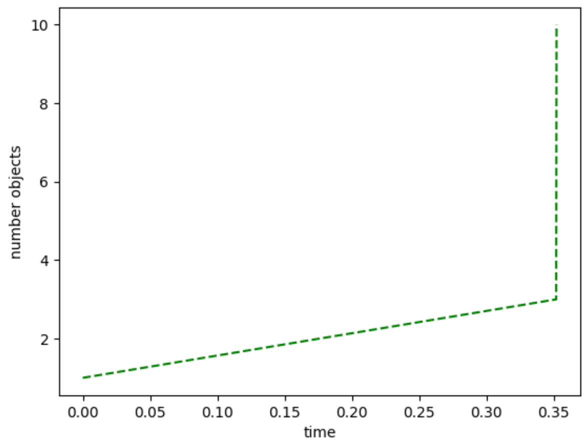


Fig. 8. Illustration of the framework time complexity Visualisation of framework's time complexity. The x-axis represent the time in second and the y-axis the number of tracked objects. Due to technical difficulties, the frameworks time complexity was evaluated only with a limited number of videos with several objects

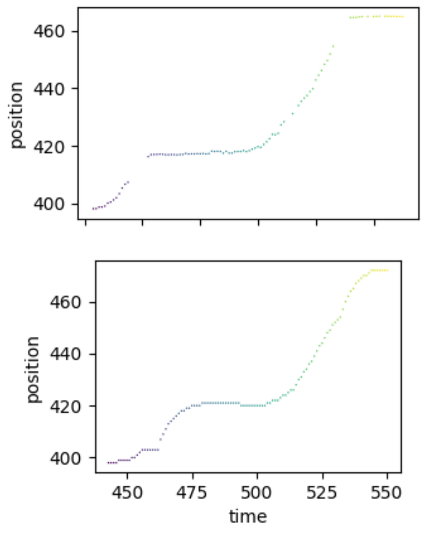


Fig. 9. Illustration of the prediction performance of the Kalman filter upon missing measurements over a 10 frames sequence.

Reference

1. Mahalanobis' distance-table of critical chi-square values.
2. Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Björn Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4929-4937:8, 2016.
3. Kevin M. Cury Taiga Abe Venkatesh N. Murthy Mackenzie Weygandt Mathis Matthias Bethge Alexander Mathis, Pranav Mamidanna. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neurosciencevolume*, 21:1281–1289, 2018.
4. N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, Sep. 2017.
5. Dietrich Paulus Nicolai Wojke, Alex Bewley. Simple online and realtime tracking with a deep association metric. 2017.
6. *Estimation with applications to tracking and navigation Bar-Shalom (Book).*
7. Titus R. Neumann Andrew D. Straw, Kristin Branson and Michael H. Dickinson. Multi-camera real-time three-dimensional tracking of multiple flying animals. *royal society publishing*, 2010.
8. Alexey Sharonov and Robin M Hochstrasser. Wide-field subdiffraction imaging by accumulated binding of diffusing probes. *Proceedings of the National Academy of Sciences*, 103(50):18911–18916, 2006.
9. Ralf Jungmann, Maier S Avendaño, Johannes B Woehrstein, Mingjie Dai, William M Shih, and Peng Yin. Multiplexed 3d cellular super-resolution imaging with dna-paint and exchange-paint. *Nature methods*, 11(3):313, 2014.
10. Jean Ponce David A. Forsyth. *Computer vision a modern approach (2nd edition)*. 2002.
11. Jonathan Kuck. Target tracking with kalman filtering. 2016.
12. Jianhua Xuang Xi Chen, Xiao Wang. Tracking multiple moving objects using unscented kalman filtering techniques. 2018.
13. Kalman filter.
14. James Munkres. Algorithms for the assignment and transportation problems, 1957.
15. Roger R Labbe Jr. Kalman and bayesian filters in python.
16. H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.
17. Ramsey Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal Process. Mag.*, 29(5):128–132, 2012.
18. Stefano Messelodi, Carla Maria Modena, Nicola Segata, and Michele Zanin. A kalman filter based background updating algorithm robust to sharp illumination changes. In Fabio Roli and Sergio Vitulano, editors, *Image Analysis and Processing – ICIAP 2005*, pages 163–170, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
19. R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.
20. T. Babb. *How a Kalman filter works, in pictures*, Bzarg, 2015.
21. Vivek Yadav. *Kalman filter: Intuition and discrete case derivation*, Towards Data Science, 2017.
22. Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. 07 2018.
23. Gabriel Martos, Alberto Muñoz, and Javier González. On the generalization of the mahalanobis distance. In José Ruiz-Shulcloper and Gabriella Sanniti di Baja, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 125–132, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
24. interactive online program and spreadsheet computation.

25. intuitive, illustrated explanation, from rick wicklin on blogs.sas.com.
26. Github: Multi object tracking with kalman-filter.
27. MathWorks. Multiple object tracking.