Appendix

Codes available:
https://github.com/SchroeterJulien/ACCV-2020-Subpixel-Point-Localization

A Derivation of the Loss Function and its Gradients

$$
\begin{aligned}
&\mathscr{L}_{\text{HM}}(\mathcal{P}_\theta, \mathcal{L}) \\
&= \iint_{\mathbb{R}^2} D(x_0, y_0 \mid \mathcal{P}_\theta, \mathcal{L}) dx_0 dy_0 \\
&= \iint_{\mathbb{R}^2} \left[S(x_0, y_0 \mid \mathcal{L}) - \hat{S}(x_0, y_0 \mid \mathcal{P}_\theta)\right]^2 dx_0 dy_0 \\
&= \iint_{\mathbb{R}^2} \Bigg[\sum_j \exp\left(-\frac{(x_j - x_0)^2}{\lambda^2} - \frac{(y_j - y_0)^2}{\lambda^2}\right) \\
&\qquad\qquad - \sum_i \hat{p}_i \exp\left(-\frac{(\hat{x}_i - x_0)^2}{\lambda^2} - \frac{(\hat{y}_i - y_0)^2}{\lambda^2}\right)\Bigg]^2 dx_0 dy_0.
\end{aligned}
\tag{1}
$$

Distributivity $(a-b)^2 = a^2 + b^2 - 2ab$

$$
\begin{aligned}
&= \iint_{\mathbb{R}^2} \left[\sum_j \exp\left(-\frac{(x_j - x_0)^2 + (y_j - y_0)^2}{\lambda^2}\right)\right]^2 dx_0 dy_0 \\
&\quad + \iint_{\mathbb{R}^2} \left[\sum_i \hat{p}_i \exp\left(-\frac{(\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2}{\lambda^2}\right)\right]^2 dx_0 dy_0 \\
&\quad - 2 \iint_{\mathbb{R}^2} \left[\sum_j \exp\left(-\frac{(x_j - x_0)^2 + (y_j - y_0)^2}{\lambda^2}\right)\right] \\
&\qquad\qquad \cdot \left[\sum_i \hat{p}_i \exp\left(-\frac{(\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2}{\lambda^2}\right)\right] dx_0 dy_0
\end{aligned}
\tag{2}
$$

Distributivity 2 $\left(\sum_i a_i\right) \cdot \left(\sum_j b_j\right) = \sum_i \sum_j a_i \cdot b_j$

$$
\begin{aligned}
= &\iint_{\mathbb{R}^2} \sum_i \sum_j \exp\left(-\frac{(x_i - x_0)^2 + (y_j - y_0)^2 + (x_j - x_0)^2 + (y_j - y_0)^2}{\lambda^2}\right) dx_0 dy_0 \\
&+ \iint_{\mathbb{R}^2} \sum_i \sum_j \hat{p}_i \hat{p}_j \exp\left(-\frac{(\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2 + (\hat{x}_j - x_0)^2 + (\hat{y}_j - y_0)^2}{\lambda^2}\right) dx_0 dy_0 \\
&- 2 \iint_{\mathbb{R}^2} \sum_i \sum_j \hat{p}_i \exp\left(-\frac{(x_j - x_0)^2 + (y_j - y_0)^2 + (\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2}{\lambda^2}\right) dx_0 dy_0
\end{aligned} \tag{3}
$$

Fubini's Theorem If $\sum_i \int |f_i(x)dx| < \infty$ and $\int \sum_i |f_i(x)dx| < \infty$, then $\sum_i \int f_i(x)dx = \int \sum_i f_i(x)dx$

$$
\begin{aligned}
= &\sum_i \sum_j \iint_{\mathbb{R}^2} \exp\left(-\frac{(x_i - x_0)^2 + (y_i - y_0)^2 + (x_j - x_0)^2 + (y_j - y_0)^2}{\lambda^2}\right) dx_0 dy_0 \\
&+ \sum_i \sum_j \iint_{\mathbb{R}^2} \hat{p}_i \hat{p}_j \exp\left(-\frac{(\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2 + (\hat{x}_j - x_0)^2 + (\hat{y}_j - y_0)^2}{\lambda^2}\right) dx_0 dy_0 \\
&- 2 \sum_i \sum_j \hat{p}_i \iint_{\mathbb{R}^2} \exp\left(-\frac{(x_j - x_0)^2 + (y_j - y_0)^2 + (\hat{x}_i - x_0)^2 + (\hat{y}_i - y_0)^2}{\lambda^2}\right) dx_0 dy_0
\end{aligned} \tag{4}
$$

Integration

$$
\begin{aligned}
&\iint_{\mathbb{R}^2} \exp\left(-\frac{(a - x_0)^2 + (b - y_0)^2 + (c - x_0)^2 + (d - y_0)^2}{\lambda^2}\right) dx_0 dy_0 \\
&\overset{[13]}{=} \int_{\mathbb{R}} -\sqrt{\frac{\pi\lambda^2}{8}} \operatorname{erf}\left(\frac{a + c - 2x_0}{\sqrt{2\lambda^2}}\right) \\
&\qquad \cdot \exp\left(-\frac{a^2 - 2ac + 2b^2 - 4by_0 + c^2 + 2d^2 - 4dy_0 + 4y_0^2}{2\lambda^2}\right) \Bigg|_{x_0=-\infty}^{\infty} dy_0 \\
&= \int_{\mathbb{R}} 2\sqrt{\frac{\pi\lambda^2}{8}} \exp\left(-\frac{a^2 - 2ac + 2b^2 - 4by_0 + c^2 + 2d^2 - 4dy_0 + 4y_0^2}{2\lambda^2}\right) dy_0 \\
&= \sqrt{\frac{\pi\lambda^2}{2}} \exp\left(-\frac{a^2 - 2ac + 2b^2 + c^2 + 2d^2}{2\lambda^2}\right) \int_{\mathbb{R}} \exp\left(-\frac{-4by_0 - 4dy_0 + 4y_0^2}{2\lambda^2}\right) dy_0 \\
&\overset{[13]}{=} \sqrt{\frac{\pi\lambda^2}{2}} \exp\left(-\frac{a^2 - 2ac + 2b^2 + c^2 + 2d^2}{2\lambda^2}\right) \\
&\qquad \cdot \left(-\sqrt{\frac{\pi\lambda^2}{8}} \exp\left(\frac{(b + d)^2}{2\lambda^2}\right) \operatorname{erf}\left(\frac{b + d - 2y_0}{\sqrt{2\lambda^2}}\right)\right) \Bigg|_{y_0=-\infty}^{\infty}
\end{aligned}
$$

$$
\begin{aligned}
&= \sqrt{\frac{\pi\lambda^2}{2}} \exp\left(-\frac{a^2 - 2ac + 2b^2 + c^2 + 2d^2}{2\lambda^2}\right) \left(2\sqrt{\frac{\pi\lambda^2}{8}} \exp\left(\frac{(b+d)^2}{2\lambda^2}\right)\right) \\
&= \frac{\pi\lambda^2}{2} \exp\left(-\frac{a^2 - 2ac + 2b^2 + c^2 + 2d^2 - b^2 - d^2 - 2bd}{2\lambda^2}\right) \\
&= \frac{\pi\lambda^2}{2} \exp\left(-\frac{(a-c)^2 + (b-d)^2}{2\lambda^2}\right).
\end{aligned}
\tag{5}
$$

Plugging Eq.5 into Eq.4

$$
\begin{aligned}
\mathscr{L}_{\text{HM}}(\mathcal{P}_\theta, \mathcal{L}) = &\sum_i \sum_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2\lambda^2}\right) \\
&+ \sum_i \sum_j \hat{p}_i \hat{p}_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - \hat{x}_j)^2 + (\hat{y}_i - \hat{y}_j)^2}{2\lambda^2}\right) \\
&- 2 \sum_i \sum_j \hat{p}_i \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2}{2\lambda^2}\right)
\end{aligned}
\tag{6}
$$

Derivative by $\hat{p}_k$

$$
\begin{aligned}
\frac{\partial}{\partial \hat{p}_k} \mathscr{L}_{\text{HM}}(\mathcal{P}_\theta, \mathcal{L}) = &\frac{\partial}{\partial \hat{p}_k} \sum_i \sum_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2\lambda^2}\right) \\
&+ \frac{\partial}{\partial \hat{p}_k} \sum_i \sum_j \hat{p}_i \hat{p}_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - \hat{x}_j)^2 + (\hat{y}_i - \hat{y}_j)^2}{2\lambda^2}\right) \\
&- \frac{\partial}{\partial \hat{p}_k} 2 \sum_i \sum_j \hat{p}_i \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2}{2\lambda^2}\right) \\
= 0 + &\sum_i \sum_j \frac{\partial}{\partial \hat{p}_k} \hat{p}_i \hat{p}_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - \hat{x}_j)^2 + (\hat{y}_i - \hat{y}_j)^2}{2\lambda^2}\right) \\
&- 2 \sum_i \sum_j \frac{\partial}{\partial \hat{p}_k} \hat{p}_i \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2}{2\lambda^2}\right) \\
= 0 + &\sum_i \sum_j \frac{\partial}{\partial \hat{p}_k} \hat{p}_i \hat{p}_j \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - \hat{x}_j)^2 + (\hat{y}_i - \hat{y}_j)^2}{2\lambda^2}\right) \\
&- 2 \sum_i \sum_j \frac{\partial}{\partial \hat{p}_k} \hat{p}_i \frac{\pi\lambda^2}{2} \exp\left(-\frac{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2}{2\lambda^2}\right) \\
= &\pi\lambda^2 \sum_i \hat{p}_i \exp\left(-\frac{(\hat{x}_i - \hat{x}_k)^2 + (\hat{y}_i - \hat{y}_k)^2}{2\lambda^2}\right) \\
&- \pi\lambda^2 \sum_j \exp\left(-\frac{(\hat{x}_k - x_j)^2 + (\hat{y}_k - y_j)^2}{2\lambda^2}\right)
\end{aligned}
\tag{7}
$$

Derivative by $\hat{x}_k$

$$
\begin{aligned}
\frac{\partial}{\partial \hat{x}_k}\mathscr{L}_{\text{HM}}(\mathcal{P}_\theta, \mathcal{L}) &= \frac{\partial}{\partial \hat{x}_k}\sum_i\sum_j \frac{\pi\lambda^2}{2}\exp\left(-\frac{(x_i-x_j)^2+(y_i-y_j)^2}{2\lambda^2}\right) \\
&\quad + \frac{\partial}{\partial \hat{x}_k}\sum_i\sum_j \hat{p}_i\hat{p}_j\frac{\pi\lambda^2}{2}\exp\left(-\frac{(\hat{x}_i-\hat{x}_j)^2+(\hat{y}_i-\hat{y}_j)^2}{2\lambda^2}\right) \\
&\quad - \frac{\partial}{\partial \hat{x}_k}2\sum_i\sum_j \hat{p}_i\frac{\pi\lambda^2}{2}\exp\left(-\frac{(\hat{x}_i-x_j)^2+(\hat{y}_i-y_j)^2}{2\lambda^2}\right) \\
&= 0 + \sum_{i\neq k}\hat{p}_i\hat{p}_k\frac{\pi\lambda^2}{2}\exp\left(-\frac{(\hat{x}_i-\hat{x}_k)^2+(\hat{y}_i-\hat{y}_k)^2}{2\lambda^2}\right)\left(\frac{2\,(\hat{x}_i-\hat{x}_k)}{2\lambda^2}\right) \\
&\quad + \sum_{j\neq k}\hat{p}_k\hat{p}_j\frac{\pi\lambda^2}{2}\exp\left(-\frac{(\hat{x}_k-\hat{x}_j)^2+(\hat{y}_k-\hat{y}_j)^2}{2\lambda^2}\right)\left(\frac{-2\,(\hat{x}_k-\hat{x}_j)}{2\lambda^2}\right) \\
&\quad - 2\sum_j \hat{p}_k\frac{\pi\lambda^2}{2}\exp\left(-\frac{(\hat{x}_k-x_j)^2+(\hat{y}_k-y_j)^2}{2\lambda^2}\right)\left(\frac{-2\,(\hat{x}_k-x_j)}{2\lambda^2}\right) \\
&= \boxed{\begin{aligned}&\hat{p}_k\pi\sum_j \exp\left(-\frac{(\hat{x}_k-x_j)^2+(\hat{y}_k-y_j)^2}{2\lambda^2}\right)(\hat{x}_k-x_j) \\ &\quad - \hat{p}_k\pi\sum_i \hat{p}_i\exp\left(-\frac{(\hat{x}_i-\hat{x}_k)^2+(\hat{y}_i-\hat{y}_k)^2}{2\lambda^2}\right)(\hat{x}_k-\hat{x}_i)\end{aligned}}
\end{aligned}
\tag{8}
$$

B Importance of Counting Regularization

In this section, we introduce a few examples that further illustrate the usefulness of counting regularization.

B.1 Prediction Sparsity

Let us consider a unique object in *1-dimension* at location (x) and two point predictions $(x, \hat{p}_0)$ and $(x, \hat{p}_1)$.

Without Regularization In this scenario,

$$
\begin{aligned}
\mathscr{L}_{\text{HM}}(\mathcal{P}_\theta, \mathcal{L}) &\propto 1 + 2\hat{p}_0\hat{p}_1 + \hat{p}_0^2 + \hat{p}_1^2 - 2\hat{p}_0 - 2\hat{p}_1 \\
&= (\hat{p}_0 + \hat{p}_1 - 1)^2.
\end{aligned}
\tag{9}
$$

Thus, the loss function is minimized when

$$
\hat{p}_0 + \hat{p}_1 = 1. \tag{10}
$$

This result is trivial when considering that the loss is the integrated squared difference between sums of Gaussians. However, it confirms that all combinations of $\hat{p}_0$ and $\hat{p}_1$ that satisfy this condition are stable solutions to the optimization problem.

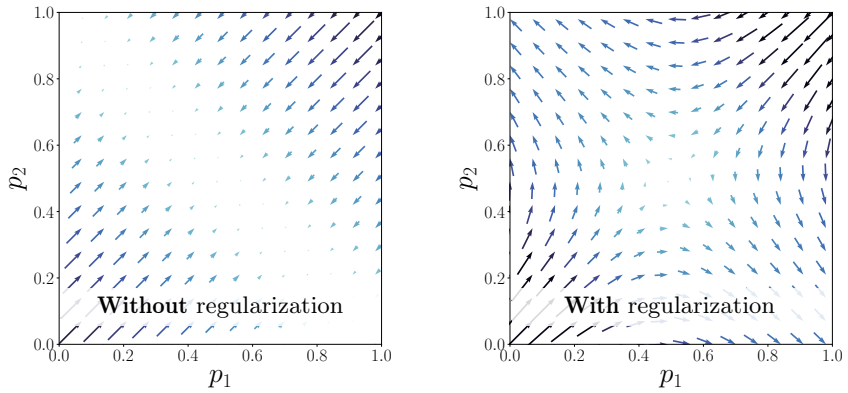


Fig. 1: Gradients of the loss with respect to probability estimates when we consider a unique object at location (x) and two point predictions $(x, \hat{p}_0)$ and $(x, \hat{p}_1)$.

With Regularization In contrast, when adding the counting regularization the loss function is proportional to

$$\mathscr{L} \propto (\hat{p}_0 + \hat{p}_1 - 1)^2 - \beta \underbrace{\log\left(\hat{p}_0(1-\hat{p}_1) + \hat{p}_1(1-\hat{p}_0)\right)}_{\mathscr{L}_{\text{MC}}}. \tag{11}$$

Fig. 1 shows the value of the gradients of the loss function (with and without regularization) with respect to probability estimates $\hat{p}_0$ and $\hat{p}_1$. This confirms that, without regularization, the optimization problem can have as a stable solution any combination of $\hat{p}_0$ and $\hat{p}_1$ which satisfies $\hat{p}_0 + \hat{p}_1 = 1$. In contrast, with counting regularization, it can be observed that—depending on the starting point—the only two stable solutions to the optimization problem are $(\hat{p}_0, \hat{p}_1) = (1, 0)$ and $(\hat{p}_0, \hat{p}_1) = (0, 1)$. While there is a saddle point at $(\hat{p}_0, \hat{p}_1) = (1/2, 1/2)$, this solution is highly unstable and any deviation from that middle point will cause the predictions to converge towards one of the true minima.

While the situation becomes increasingly more complex as the number of predictions increases, the main ideas and results remain identical. Overall, the counting regularization acts as a means to achieve prediction sparsity—a useful feature in practical applications as it alleviates the need for additional heuristics to obtain precise point localization.

B.2 Faster Location Convergence

In this section, let us once again consider a unique object in *1-dimension* at location (x), but this time the two point predictions are $(\hat{x}_1 = x - \Delta, \hat{p})$ and $(\hat{x}_2 = x + \Delta, \hat{p})$.

Without Regularization In this case, the optimal probability estimate $\hat{p}$ is

$$\begin{aligned}
&\frac{\partial}{\partial \hat{p}} \mathscr{L}_{\mathrm{HM}}(\mathcal{P}_\theta, \mathcal{L}) = \pi\lambda^2 \hat{p}_{opt} \exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + \pi\lambda^2 \hat{p}_{opt} - \pi\lambda^2 \exp\left(-\frac{\Delta^2}{2\lambda^2}\right) = 0 \\
&\Leftrightarrow \hat{p}_{opt}\left(\exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + 1\right) = \exp\left(-\frac{\Delta^2}{2\lambda^2}\right) \\
&\Leftrightarrow \hat{p}_{opt} = \exp\left(-\frac{\Delta^2}{2\lambda^2}\right) / \left(\exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + 1\right).
\end{aligned} \tag{12}$$

With Regularization Similarly, the optimal probability estimate $\hat{p}$ is given by the following equation—which can easily be solved using a standard root-finding algorithm (e.g., Brent's method [3]):

$$\begin{aligned}
&\frac{\partial}{\partial \hat{p}}\{\mathscr{L}_{\mathrm{HM}}(\mathcal{P}_\theta, \mathcal{L}) + \beta \mathscr{L}_{\mathrm{MC}}(\mathcal{P}_\theta, \mathcal{L})\} \\
&\quad = \pi\lambda^2 \hat{p}_{opt} \exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + \pi\lambda^2 \hat{p}_{opt} - \pi\lambda^2 \exp\left(-\frac{\Delta^2}{2\lambda^2}\right) - \beta \underbrace{\frac{1 - 2\hat{p}_{opt}}{(\hat{p}_{opt} - 1)\hat{p}_{opt}}}_{\frac{\partial}{\partial \hat{p}} \mathscr{L}_{\mathrm{MC}}} = 0 \\
&\Leftrightarrow \hat{p}_{opt} \exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + \hat{p}_{opt} - \exp\left(-\frac{\Delta^2}{2\lambda^2}\right) - \tilde{\beta} \frac{1 - 2\hat{p}_{opt}}{(\hat{p}_{opt} - 1)\hat{p}_{opt}} = 0 \\
&\overset{\hat{p}_{opt} \neq 0,1}{\Leftrightarrow} \hat{p}_{opt}(\hat{p}_{opt} - 1)\left[\hat{p}_{opt} \exp\left(-\frac{4\Delta^2}{2\lambda^2}\right) + \hat{p}_{opt} - \exp\left(-\frac{\Delta^2}{2\lambda^2}\right)\right] - \tilde{\beta}(1 - 2\hat{p}_{opt}) = 0.
\end{aligned} \tag{13}$$

The location gradient, assuming $\hat{p} = \hat{p}_{opt}$, is

$$\frac{\partial}{\partial \hat{x}_2} \mathscr{L}_{\mathrm{HM}}(\mathcal{P}_\theta, \mathcal{L}) = \hat{p}_{opt} \pi \exp\left(-\frac{\Delta^2}{2\lambda^2}\right)(\Delta) - \hat{p}_{opt}^2 \pi \exp\left(-\frac{4\Delta^2}{2\lambda^2}\right)(2\Delta). \tag{14}$$

Fig. 2 displays the location gradient as a function of Δ for $\lambda = 1$, assuming that $\hat{p}$ is set optimally. It can be observed that the gradient gets larger as the counting regularization is increased. Thus, in this scenario, the regularization acts as a means for faster convergence of the location predictions towards the true object location.

While this effect is hard to quantify in higher dimensions (i.e., more than two point predictions), the counting regularization is still expected to improve location convergence in more complex settings.

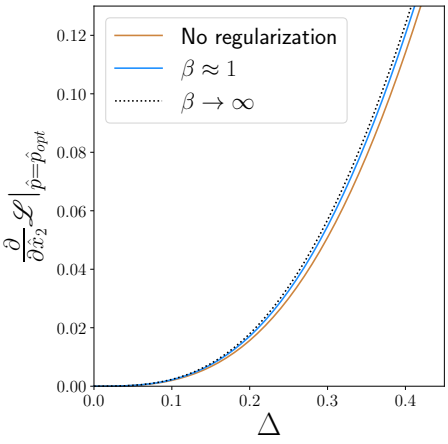


Fig. 2: Location gradient

Table 1: Single molecule localization microscopy λ-sensitivity analysis based on the experiment proposed in [9]. The Jaccard index [and F_1 score] are computed with the software from [11]. The setting $\lambda = 0.2$ is the one reported—without any hyperparameter optimization—as Ours in the main text (see →).

		Jaccard Index [F_1]	
	Method	$\tau = 25$nm	$\tau = 50$nm
	Deep-Storm [9]	0.153 [0.266]	0.416 [0.588]
	Upsampling	0.171 [0.292]	0.448 [0.618]
	Refinement	0.195 [0.326]	0.448 [0.619]
	Ours ($\lambda = 0.1$)	0.219 [0.359]	0.464 [0.634]
→	Ours ($\lambda = 0.2$)	0.234 [0.379]	0.517 [0.681]
	Ours ($\lambda = 0.3$)	0.233 [0.378]	0.519 [0.678]
	Ours ($\lambda = 0.4$)	0.229 [0.373]	0.519 [0.684]
	Ours ($\lambda = 0.5$)	0.226 [0.368]	0.524 [0.687]
	Ours ($\lambda = 0.6$)	0.234 [0.379]	**0.531** [**0.694**]
	Ours ($\lambda = 0.7$)	**0.235** [**0.381**]	0.528 [0.691]
	Ours ($\lambda = 0.8$)	0.234 [0.379]	0.526 [0.690]

C Molecule Localization Microscopy Experiment

The architectures used for the experiments are depicted in Fig. 3.

C.1 Results Sensitivity to Smoothing Parameter λ

In order to assess the sensitivity of the results to changes in the model softness parameter λ, we replicate the single molecule localization microscopy experiments with a wide array of potential λ values. The results—reported in Table 1—highlight the remarkable robustness of the model to changes in this key hyperparameter. Indeed, the model achieves extremely consistent results with near constant performance metrics for λ between 0.2 and 0.8. Thus, the model's success is almost independent of the value of this hyperparameter.

D Checkerboard Corner Detection Experiment

D.1 Additional Results

Additional experiments have been conducted for the checkerboard corner localization experiment.

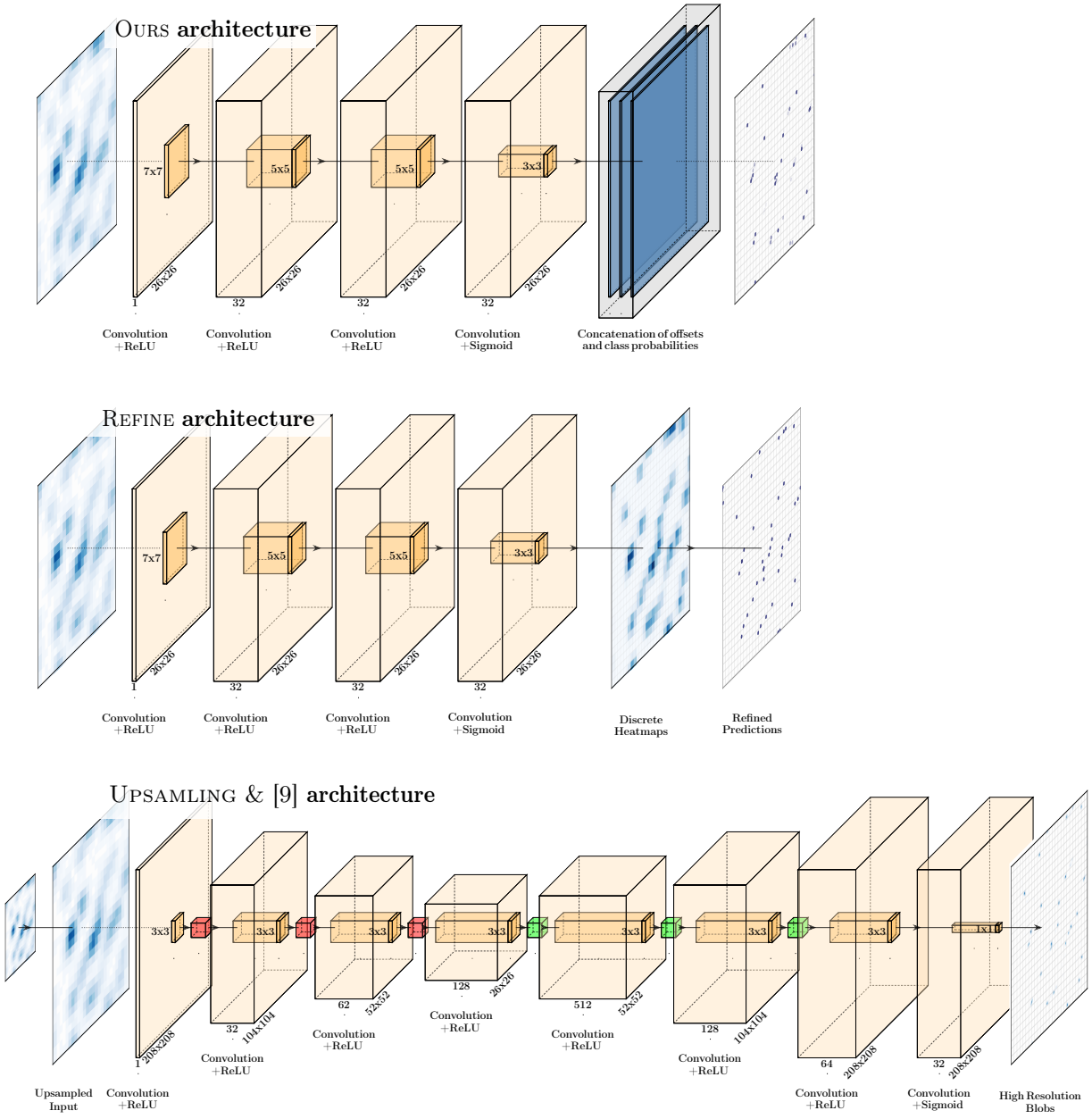


Fig. 3: Model Architecture for the Molecule Localization Microscopy Experiments. Since our approach can directly operate on the original image resolution, its model architecture does not need to include any downsampling—nor upsampling—layer to learn meaningful representations—-in contrast to Nehme et al. [9] and the UPSAMPLING benchmark.

Table 2: Corner localization performance on our synthetic test set. The mean-absolute deviation (MAD) from ground-truth (in units of original pixel size) as well as precision, recall, and F_1-scores (with a tolerance of 3 pixels) are reported.

	Methods	MAD (px)	Recall	Precision	F_1
Classic	OCamCalib [12]	0.362	**97.0**	99.8	**98.4**
	Rochade [10]	0.147	59.1	**99.9**	74.3
	OpenCV [2]	0.137	45.6	89.5	60.4
	MATLAB [6]	**0.086**	65.8	96.4	78.2
Learn.	DL-Heatmap (sim. [5, 4])	0.488	98.1	99.7	98.9
	+ Refinement (sim. [7])	0.130	98.1	99.7	98.9
	OURS	**0.105**	**99.3**	**99.9**	**99.6**

Experiment: sub-pixel accuracy on synthetic test data To evaluate the absolute sub-pixel accuracy of our method, we test it on a synthetic test dataset generated analogously to the training dataset described in the main text. (Appendix D.3 illustrates the variety of the data generated.) The exact ground-truth corner locations are thus known by construction. For all benchmarks, 1000 synthetic images are used for testing. The results are summarized in Table 2. Overall, our method consistently outperforms state-of-the-art algorithms both in terms of absolute spatial precision — with typical errors in the order of $\approx 1/10$th of a pixel — as well as in terms of detection rates. Especially noteworthy is the fact that the excellent spatial precision of our approach is not a result of a low recall rate. In contrast, the lower recall values achieved by other methods, show that they often fail to detect challenging corners (due to distortions, noise, low contrast), and hence only the most easily detectable ones are taken into account when computing the spatial error for these methods. For instance, the remarkable mean absolute error of 0.086 pixels achieved by MATLAB [6] results from only 65.8% of the corners it was able to detect in the first place. Finally, our method does not require additional information about the grid structure of the calibration board and its size—in contrast to [2, 10, 12] which leverage this information to refine the predictions.

Further, we note that the above state-of-the-art methods, with the exception of deep learning-based ones, rely on traditional image processing techniques that have been hand-crafted specifically for this task. Hence, they do not generalize well to other applications. In contrast, we hypothesize that our approach can be straightforwardly applied to any sub-pixel localization task, such as the accurate analysis of medical images.

Finally, in terms of inference efficiency, our point prediction approach is significantly faster (3.43 images/second on a 4.3 GHz CPU, and 100+ with an NVidia TitanXp GPU) than OpenCV (2.04 images/s), the saddle point-based ROCHADE (1.10 images/s), and the heatmap-based approach (1.15 images/s) which is slowed down by the corner-refinement step and the lack of spatial downsampling.

Table 3: Full results of localization performance in low-resolution settings on the **GoPro** dataset [10]. **Consistency**: mean-absolute displacement (and the 90th quantile) between predictions on high and low-resolution images downsampled by δ. **Reprojection Error:** corresponding errors in corner reprojection [and the number of fully detected boards]. In units of original pixel size.

		Consistency		
Methods		$\delta = 2$	4	6
Classic	OCamCalib [12]	0.660 (1.12)	1.389 (2.50)	1.989 (3.61)
	Rochade [10]	0.380 (0.67)	0.467 (0.81)	1.125 (2.07)
	OpenCV [2]	**0.111** (**0.20**)	**0.179** (**0.31**)	0.336 (0.50)
	MATLAB [6]	0.129 (0.20)	0.198 (0.32)	**0.314** (**0.50**)
Learn.	DL-Heatmap (sim. $[5, 4]$)	0.900 (1.41)	1.629 (2.24)	2.395 (3.61)
	+ Refinement (sim. [7])	0.153 (0.28)	0.279 (0.50)	0.428 (0.76)
	OURS	**0.133** (**0.24**)	**0.244** (**0.43**)	**0.378** (**0.66**)

		Reprojection Error		
Methods		$\delta = 2$	4	6
Classic	OCamCalib [12]	0.107 [100]	0.390 [73]	—— [18]
	Rochade [10]	0.085 [100]	0.321 [100]	1.716 [71]
	OpenCV [2]	**0.045** [100]	0.256 [98]	0.994 [73]
	MATLAB [6]	**0.045** [99]	**0.205** [100]	**0.325** [100]
Learn.	DL-Heatmap (sim. $[5, 4]$)	0.146 [100]	0.363 [100]	0.797 [77]
	+ Refinement (sim. [7])	0.054 [100]	0.336 [100]	0.531 [100]
	OURS	**0.046** [100]	**0.198** [82]	**0.417** [100]

Additional Results for experiments in main text In the main text, only the lowest downsampling factors δ have been presented. Therefore, for the sake of completeness, Table 4 and Table 3 show the results of the corner detection experiments for several other downsampling factors δ (i.e., higher resolution inputs).

Our method displays strong overall results for all downsampling levels and outperforms the other deep learning benchmarks on almost all measures. More precisely, on the GoPro dataset, it achieves state-of-the-art reprojection errors on lower-resolution settings, while being competitive on higher-resolution inputs. In terms of detection consistency across resolutions, our approach is only outperformed by the OpenCV [2] and MATLAB [6] implementation. However, while OpenCV performs slightly better on the consistency measure, its relatively high number of false positives and false negatives have a clear impact on its camera calibration performance (i.e., reprojection error). Overall, our approach appears to be extremely suitable for high precision calibration in low-resolution settings.

Table 4: Full results of localization performance in low-resolution settings on the uEye dataset [10]. **Consistency**: mean-absolute displacement (and the 90th quantile) between predictions on high and low-resolution images downsampled by δ. **Reprojection Error:** corresponding errors in corner reprojection [and the number of fully detected boards]. In units of original pixel size

		Consistency		Reprojection Error	
	Methods	$\delta = 2$	4	$\delta = 2$	4
Classic	OCamCalib [12]	0.783 (1.58)	1.447 (2.92)	0.129 [200]	0.197 [114]
	Rochade [10]	0.176 (0.29)	0.587 (1.05)	0.057 [206]	0.107 [197]
	OpenCV [2]	0.126 (0.21)	0.889 (2.66)	0.057 [197]	——— [0]
	MATLAB [6]	**0.090** (**0.15**)	**0.174** (**0.29**)	**0.048** [206]	**0.059** [204]
Learn.	DL-Heatmap (sim. $[5, 4]$)	0.955 (1.41)	1.666 (2.24)	0.126 [206]	0.230 [175]
	+ Refinement (sim. [7])	**0.077** (**0.14**)	0.562 (1.20)	**0.052** [206]	0.086 [162]
	OURS	0.134 (0.23)	**0.348** (**0.64**)	0.055 [200]	**0.073** [187]

D.2 Regularization Ablation Study

As done in the main text for the single molecule localization microscopy experiment, we perform an ablation study to measure the impact of the counting-based regularization on the performance of the checkerboard corner detection model. The results—summarized in Tables 5 and 6—are in line with the findings of the single molecule localization microscopy experiment. Indeed, adding the counting loss as a regularizer to the soft localization learning loss consistently improves the performance of the trained model. For instance, there is only one metric on which the approach without regularization outperforms its regularized counterpart (i.e., reprojection error on the GoPro dataset with downsampling factor $\delta = 4$). However, this unique favourable outcome for the approach without regularization

Table 5: Regularization ablation study for the experiment on the **GoPro** dataset [10]. **Consistency**: mean-absolute displacement (and the 90th quantile) between predictions on high and low-resolution images downsampled by δ. **Reprojection Error:** corresponding errors in corner reprojection [and the number of fully detected boards]. In units of original pixel size

	Consistency		
Methods	$\delta = 2$	4	6
Without Count Regularization	0.199 (0.37)	0.369 (0.66)	0.562 (0.99)
With Count Regularization	0.133 (0.24)	0.244 (0.43)	0.378 (0.66)

	Reprojection Error		
Methods	$\delta = 2$	4	6
Without Count Regularization	0.052 [100]	0.172 [48]	0.460 [100]
With Count Regularization	0.046 [100]	0.198 [82]	0.417 [100]

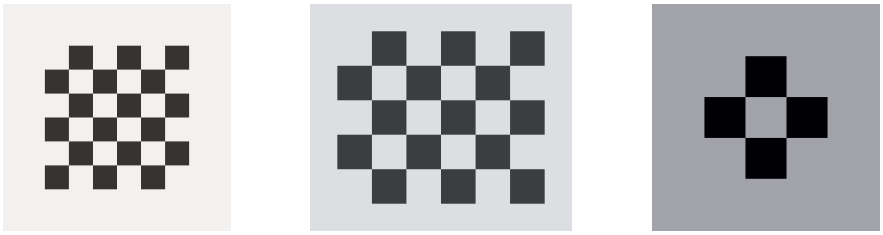
is merely due to the much lower recall it reports; indeed, the metric is thus computed on the easiest samples only which positively biases the outcome. Overall, counting-based regularization undoubtedly improves the learning of sub-pixel point localization.

Table 6: Regularization ablation study for the the **uEye** dataset [10]. **Consistency**: mean-absolute displacement (and the 90th quantile) between predictions on high and low-resolution images downsampled by δ. **Reprojection Error:** corresponding errors in corner reprojection [and the number of fully detected boards]. In units of original pixel size.

	Consistency		Reprojection Error	
Methods	$\delta = 2$	4	$\delta = 2$	4
Without Count Regularization	0.192 (0.36)	0.501 (0.90)	0.059 [148]	0.085 [137]
With Count Regularization	0.134 (0.23)	0.348 (0.64)	0.055 [200]	0.073 [187]

D.3 Synthetic Dataset

The codes for the generation of the synthetic checkerboard dataset are provided[1] (c.f. `create_dataset.py`).



(6x6) board (6x5) board (3x3) board

Fig. 5: Initial checkerboard of various size, shape and coloration

Overall, initial checkerboards are first generated by randomly sampling their size, shape, and coloration (see Fig. 5). Then, some of these checkerboards are projected onto textures such as wood, paper, and stone (see the first row of Fig. 4). Finally, between one and eight (sampled uniformly at random) of the following eight transformations are applied to these checkerboards: blurring, lighting, sharpening, contrast change, scaling, distortion, perspective transform,

[1] https://github.com/SchroeterJulien/ACCV-2020-Subpixel-Point-Localization

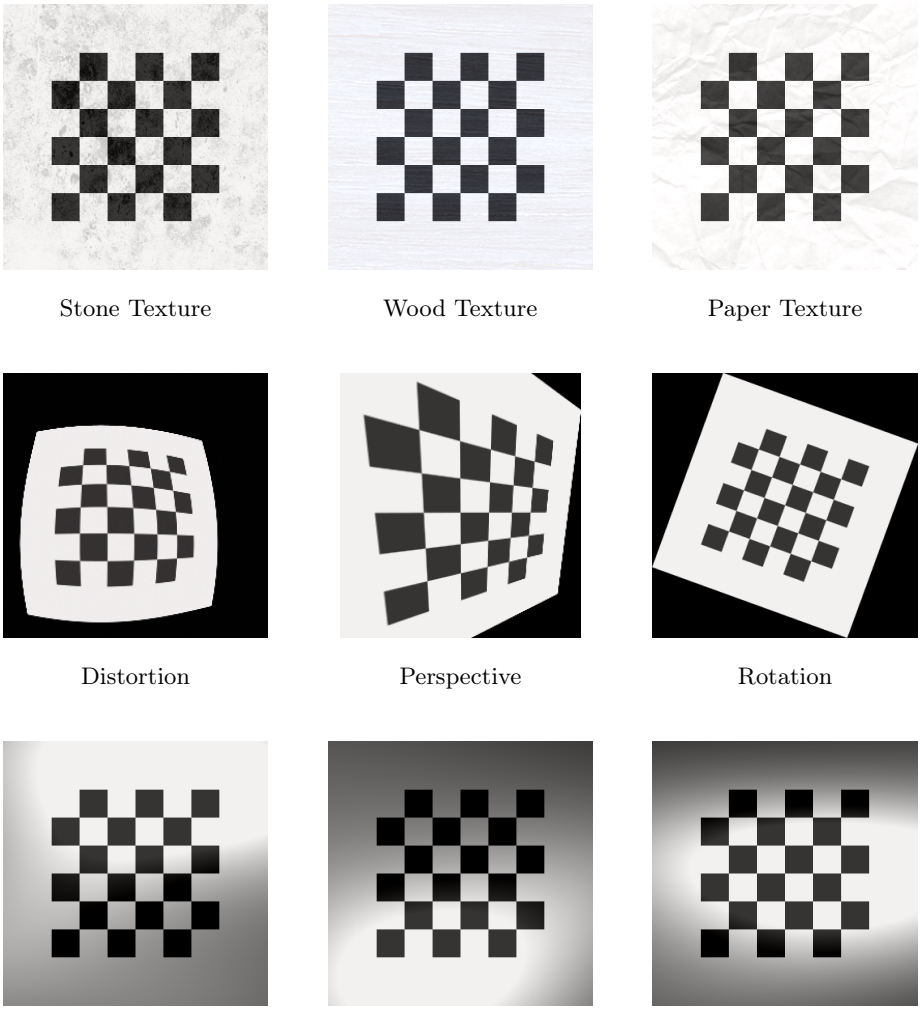


Fig. 4: Examples of transformations applied to the initial checkerboard images

and rotation (see Fig. 4 for examples). All of these transformations have hyper-parameters that are also sampled at random, such as the level of distortion or the angle of the rotation.

While this process allows for a rich variety of checkerboards to be generated (see Fig. 6), most importantly it allows to track the location of the corners with high levels of precision. Indeed, spatial transformations (e.g., rotation and perspective transform) can be applied to both checkerboard images and corner locations without any significant approximation. Thus, we are able to leverage these precise labels as the ground truth to train our sub-pixel precision model.

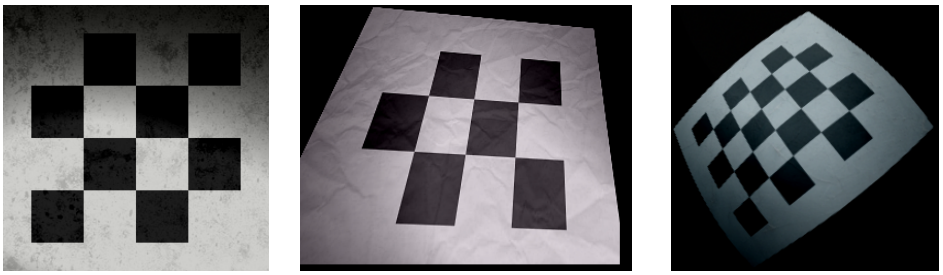


Fig. 6: Example checkerboard training images from our synthetic dataset

E Golf Swing Event Localization Experiment

E.1 Ablation Study

Table 7 reveals that the counting regularization consistently improves the detection performance of our model on the golf swing event sequencing experiment. Indeed, training with this regularizer yields at least a 1% improvement in accuracy on all decimation levels, when compared to the results obtained when training with $\mathscr{L}_{\text{HM}}$ alone.

E.2 Additional Results

Table 5 (in the main text) reports the mean golf swing event detection accuracy over all event classes, i.e. address (A), toe-up (TU), mid-backswing (MB), top (T), mid-downswing (MD), impact (I), mid-follow-through (MFT), and finish (F) [8]. However, each of these classes differs drastically from one another, especially in terms of temporal ambiguity and detection difficulty. Therefore, in order to assess whether the performance improvement achieved by our model can be attributed to a few event classes only or whether the improvement is consistent across all classes, we provide a detailed report of per class detection accuracy in Table 8.

Overall, our method displays consistent improvement on most event classes and decimation rates. (Given the relatively moderate size of the testing splits and

Table 7: Ablation study. Golf swing event detection accuracy (within a ±1 frame tolerance) as a function of decimation factor δ. Averages and standard deviations (in brackets) are reported over 4 folds. The architecture is from [8]

Loss	$\delta = 1$ frame	2 frames	4 frames	8 frames	16 frames
Ours ($\mathscr{L}_{\text{HM}}$+$\mathscr{L}_{\text{MC}}$)	**70.9** (1.4)	**70.4** (1.2)	**70.7** (1.3)	**69.8** (1.4)	**60.6** (1.6)
Ours ($\mathscr{L}_{\text{HM}}$ only)	69.7 (1.9)	69.3 (1.8)	69.7 (1.0)	68.0 (1.6)	59.3 (0.8)
Diff.	+1.2	+1.1	+1.0	+1.8	+1.6

the stochastic nature of the learning process, a few outliers are to be expected.) Our approach not only improves the detection accuracy of temporally ambiguous events (e.g., A and F) but also pushes further the detection capabilities on more easily detectable classes (e.g., MD and MFT).

Table 8: Golf swing event detection accuracy (within a ±1 frame tolerance) per class as a function of decimation factor δ. Averages are reported over 4 folds. The architecture is from [8]

	Loss	$\delta = 1$ frame	2 frames	4 frames	8 frames	16 frames
A	Naïve upsampling	18.2	18.4	20.9	22.1	20.5
	Frame interpolation [1]	— " —	19.4	23.1	19.0	15.6
	Dense classification	— " —	19.7	22.9	21.3	21.2
	Ours	**23.9**	**24.6**	**23.4**	**25.1**	**22.4**
TU	Naïve upsampling	79.0	80.5	68.7	47.7	28.1
	Frame interpolation [1]	— " —	73.3	71.2	60.2	42.5
	Dense classification	— " —	**81.8**	78.8	75.6	63.1
	Ours	**80.1**	77.9	**79.0**	**76.5**	**69.6**
MB	Naïve upsampling	81.3	83.7	68.9	46.6	30.4
	Frame interpolation [1]	— " —	82.3	78.6	66.3	44.2
	Dense classification	— " —	82.9	**84.4**	78.6	63.6
	Ours	**86.6**	**86.4**	84.3	**81.9**	**68.8**
T	Naïve upsampling	62.3	62.8	60.6	43.1	25.1
	Frame interpolation [1]	— " —	64.5	64.6	62.4	45.3
	Dense classification	— " —	69.4	72.8	69.3	67.4
	Ours	**70.4**	**70.5**	**75.7**	**78.0**	**70.5**
MD	Naïve upsampling	95.7	95.3	83.9	52.0	30.3
	Frame interpolation [1]	— " —	95.1	94.0	85.8	58.6
	Dense classification	— " —	**96.3**	94.9	89.1	**77.8**
	Ours	**96.2**	92.9	**95.0**	**90.4**	75.7
I	Naïve upsampling	94.7	94.6	80.4	57.8	14.1
	Frame interpolation [1]	— " —	94.7	93.5	88.8	60.2
	Dense classification	— " —	**96.3**	**94.6**	91.2	79.3
	Ours	**95.3**	96.1	**94.6**	**92.1**	**80.4**
MFT	Naïve upsampling	94.4	94.6	80.6	71.5	31.6
	Frame interpolation [1]	— " —	92.9	92.0	84.6	55.4
	Dense classification	— " —	**94.8**	92.9	87.2	75.8
	Ours	**94.9**	94.3	**93.3**	**91.3**	**77.7**
F	Naïve upsampling	15.9	17.8	15.9	15.8	10.8
	Frame interpolation [1]	— " —	17.1	20.1	16.9	13.8
	Dense classification	— " —	15.8	18.1	18.4	14.4
	Ours	**20.0**	**20.2**	**20.4**	**20.9**	**21.5**

References

1. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3703–3712 (2019)
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools **120**, 122–125 (2000)
3. Brent, R.P.: Algorithms for minimization without derivatives (1973)
4. Chen, B., Xiong, C., Zhang, Q.: CCDN: Checkerboard corner detection network for robust camera calibration. In: International Conference on Intelligent Robotics and Applications. pp. 324–334. Springer (2018)
5. Donné, S., De Vylder, J., Goossens, B., Philips, W.: MATE: Machine learning for adaptive calibration template detection. Sensors **16**(11), 1858 (2016)
6. Geiger, A., Moosmann, F., Car, Ö., Schuster, B.: Automatic camera and range sensor calibration using a single shot. In: 2012 IEEE International Conference on Robotics and Automation. pp. 3936–3943. IEEE (2012)
7. Graving, J.M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B.R., Couzin, I.D.: DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. eLife **8**, e47994 (2019)
8. McNally, W., Vats, K., Pinto, T., Dulhanty, C., McPhee, J., Wong, A.: GolfDB: A video database for golf swing sequencing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2019)
9. Nehme, E., Weiss, L.E., Michaeli, T., Shechtman, Y.: Deep-storm: super-resolution single-molecule microscopy by deep learning. Optica **5**(4), 458–464 (2018)
10. Placht, S., Fürsattel, P., Mengue, E.A., Hofmann, H., Schaller, C., Balda, M., Angelopoulou, E.: ROCHADE: Robust checkerboard advanced detection for camera calibration. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 766–779. Springer (2014)
11. Sage, D., Kirshner, H., Pengo, T., Stuurman, N., Min, J., Manley, S., Unser, M.: Quantitative evaluation of software packages for single-molecule localization microscopy. Nature methods **12**(8), 717–724 (2015)
12. Scaramuzza, D., Martinelli, A., Siegwart, R.: A toolbox for easily calibrating omnidirectional cameras. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5695–5701. IEEE (2006)
13. Wolfram Research, Inc.: Wolfram Alpha, https://www.wolframalpha.com/