A APPENDIX

A.1 DRONEBIRD DATASET

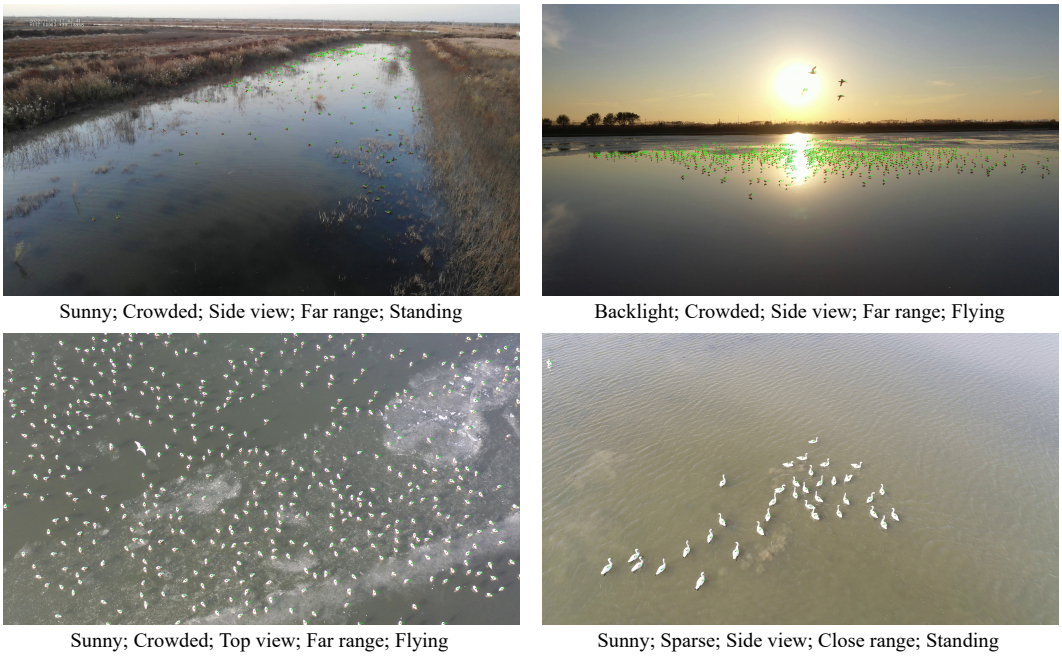

Figure 6: Visualization of partial examples of DroneBird.

Our DroneBird dataset is captured by cameras mounted on drones using consumer drones such as the DJI Mavic 2 Pro, Phantom 4 Pro, etc. DroneBird captures a wide range of scenarios, including rivers, wetlands, lakes, ice, and other common bird habitats. The data captured in DroneBird is primarily obtained by the drone from 30 meters or 60 meters in the air, with a small portion of the data captured close to the ground. DroneBird's collection time is during the daytime or in the early evening when the weather conditions are favorable, and the view is relatively clear.

Table 4: Existing bird datasets and our proposed DroneBird dataset.

Dataset	Type	Trajectory	Highest Resolution	Frames	Ave count	Total count
Penguin (Arteta et al., 2016)	Image	×	1536×2048	$33,405$	178.4	$5,970,899$
Bird-Count (Wang et al., 2023)	Image	×	768×1024	$1,372$	131.1	$173,458$
DroneBird	Video	✓	2160×4096	$21,500$	171.5	$3,686,409$

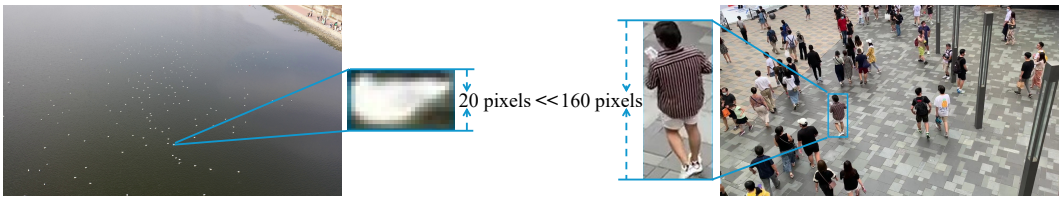

Figure 7: Visualization of pixel occupied by the target in DroneBird and existing crowd dataset.

DroneBird captured 50 videos of migratory birds and segmented them. Specifically, we used 30 of the video data as a $train$ set, 10 of the remaining 20 videos as a $test$ set, and 10 as a $validate$ set. We cut the 40 videos in the $train$ and $test$ sets to 500 frames per video (around 17s), and cut the 10 videos in the $validate$ set to 150 frames per video (around 5s) to accomplish a reasonable data division. The $train$ set, $test$ set and $validate$ set after the division is completed contain $15,000$ frames, $5,000$ frames and $1,500$ frames, respectively. It is worth noting that the data scenarios for each of the two divisions in the $train$, $test$ and $validate$ sets are different, as a way to ensure that the data will not be leaked during the training process.

Table 5: Transfer performance on bird data.

Exp.	Original dataset	Target dataset	**MAE** ↓	**RMSE** ↓
1	VSCrowd	DroneBird	183.31	217.42
2	DroneBird	DroneBird	38.72	42.92

We have compiled the dataset's statistics and compared them with existing bird datasets, with the results shown in Table 4. A few examples of DroneBird are demonstrated in Fig. 6. Most of the targets in DroneBird are small in size. We compared the pixel height occupied by individual bird targets in DroneBird with that occupied by individuals in existing crowd data, and visualized this comparison in Fig. 7. The pixel height occupied by individual bird targets is significantly smaller than that of crowd individuals, posing a significant challenge for the target counting task.

We summarize the contributions and new challenges brought by DroneBird into the following four points:

- **Small target**: We compared the number of foreground pixels occupied by independent targets in the DroneBird data and the existing crowd data at the same resolution, and the independent targets in DroneBird occupy fewer pixels (less than 20 pixels height) than the independent targets in the existing dataset (more than 160 pixels height). Note that although the counting target in the crowd counting task is the human head, the human body still provides important information, and the human body occupies more pixels compared to birds, so our dataset provides data on small targets for counting. On the other hand, to avoid drone interference with bird activity, DroneBird's drone was photographed farther away from the flock, resulting in a further reduction of pixels occupied by the photographed individual birds.
- **Sparse distribution**: Unlike humans, birds in nature need enough space to move around, which means birds are more sparsely distributed in space. Coupled with the smaller target size of birds, the sparseness of the foreground portion is even more pronounced in DroneBird compared to the already existing dataset.
- **Fast movement**: Unlike existing video crowd-sourced data, the unique flight movements of birds allow them to do fast movements, posing a challenge for better temporal modeling.
- **New counting category**: DroneBird fills the gap of missing video bird data in the field. The models trained on crowd data fail to conduct bird counting directly (See Table 5). We evaluated the performance of our method trained on the VSCrowd dataset (Exp. 1) and the DroneBird training set (Exp. 2) on the DroneBird testing set. The results show that the model trained on the VSCrowd dataset failed to perform well on the bird data, which indicates the large gap between crowd and bird data.

A.2 Detailed Method

We detail the training process of our E-MAC method in Algorithm 1. The training process of `DEMO` is represented in Algorithm 2.

As shown in Algorithm 1, our model requires two frames (I_t, I_{t-1}) and their corresponding density maps (D_t, D_{t-1}) as input. The image and density map pairs $\mathbf{S_t} = \{I_t, D_t\}$ and $\mathbf{S_{t-1}} = \{I_{t-1}, D_{t-1}\}$ are fed to the `DEMO` to predict the density map $\hat{D}_t$ and $\hat{D}_{t-1}$. Meanwhile, a pretrained optical flow estimation network (Sun et al., 2018) is performed on $\hat{D}_t$ and $\hat{D}_{t-1}$ to generate optical flow $\mathcal{M}$, which is used to *warp* the $\hat{D}_{t-1}$ to $\hat{D}_{t-1}^{\text{warp}}$. Then the $\hat{D}_{t-1}^{\text{warp}}$ and $\hat{D}_t$ are fed into a cross-attention layer to calculate the residual $\hat{D}_t^{\text{res}}$ of adjacent predicted density map. The final predicted density map $\hat{D}_{\text{fuse}}$ of I_t is then calculated by a pixel-wise add of $\hat{D}_t$ and $\hat{D}_t^{\text{res}}$.

Algorithm 1 Framework Workflow in Training Phase

Ensure: $\{I_{\text{t}}, D_{\text{t}}\}, \{I_{\text{t-1}}, D_{\text{t-1}}\}$
Require: $\hat{D}_{\text{fuse}}$
for all epoch **do**
  $\hat{D}_{\text{t}} \leftarrow \phi_{\texttt{DEMO}}(I_t, D_t)$
  $\hat{D}_{\text{t-1}} \leftarrow \phi_{\texttt{DEMO}}(I_{t-1}, D_{t-1})$
  $\mathcal{M} \leftarrow \phi_{\text{OpticalFlow}}(I_{\text{t}}, I_{\text{t-1}})$
  $\hat{D}_{\text{t-1}}^{\text{warp}} \leftarrow \phi_{\text{warp}}(\mathcal{M}, \hat{D}_{\text{t-1}})$
  $\hat{D}_{\text{t}}^{\text{res}} \leftarrow \phi_{\text{ca}}(\hat{D}_{\text{t-1}}^{\text{warp}}, \hat{D}_{\text{t}})$
  $\hat{D}_{\text{fuse}} \leftarrow \hat{D}_{\text{t}} + \hat{D}_{\text{t}}^{\text{res}}$
  $\mathcal{L}_{\text{fuse}}, \mathcal{L}_{\text{cur}}, \mathcal{L}_{\text{opt}}, \mathcal{L}_{\text{TV}} \leftarrow \phi_{\text{Loss}}$
  $\mathcal{L} \leftarrow \lambda_1 \mathcal{L}_{\text{fuse}} + \lambda_2 \mathcal{L}_{\text{cur}} + \lambda_3 \mathcal{L}_{\text{opt}} + \lambda_4 \mathcal{L}_{\text{TV}}$
end for

Algorithm 2 `DEMO` workflow in training phase

Ensure: I, D
Require: $\hat{D}$
$\mathbf{T}_I, \mathbf{T}_D, \mathbf{I}_{\text{patch}}, \mathbf{D}_{\text{patch}} \leftarrow \phi_{\text{patchify}}(I, D)$
$\mathbf{V}_D \leftarrow \phi_{\text{sum}}(\mathbf{D}_{\text{patch}})$
$\mathbb{N} = \texttt{random}(0, 1)$
if $\mathbb{N} \leq 1 - \mathcal{P}$ **then**
  $\mathcal{K} \leftarrow \texttt{argsort}_{des}(\mathbf{V}_D)$
else
  $\mathcal{K} \leftarrow \texttt{argsort}_{asc}(\mathbf{V}_D)$
end if
for all $t_i \in \mathbf{T}_I$ **do**
  if $i \in \mathcal{K}$ **then**
    $M^i \leftarrow 0$
  else
    $M^i \leftarrow 1$
  end if
end for
$\mathbf{T}_I^{\text{ret}}, \mathbf{T}_D^{\text{ret}} \leftarrow \texttt{mask}(\mathbf{T}_I, M_{\text{adaptive}}), \texttt{mask}(\mathbf{T}_D, M_{\text{random}})$
$\mathbf{T}^{\text{ret}} \leftarrow \phi_{\text{concate}}(\mathbf{T}_I^{\text{ret}}, \mathbf{T}_D^{\text{ret}})$
$\mathbf{T}^{\text{ret}} \leftarrow \phi_{\text{encoder}}(\mathbf{T}^{\text{ret}})$
$\mathbf{T}_I^{\text{ret}}, \mathbf{T}_D^{\text{ret}} \leftarrow \text{split}(\mathbf{T}^{\text{ret}})$
$\text{Mask} = \texttt{random}$
$\hat{\mathbf{T}_D} \leftarrow \phi_{\text{fill}}(\mathbf{T}_D^{\text{ret}}, \text{Mask})$
$\hat{\mathbf{T}_D} \leftarrow \phi_{\text{ca}}(\hat{\mathbf{T}_D}, \mathbf{T}^{\text{ret}})$
$\hat{D} \leftarrow \phi_{\text{decoder}}(\hat{\mathbf{T}_D})$

In the `DEMO`, a video frame I and its corresponding density map D performed a patchify and a projection operation to get the token set $\mathbf{T}_I, \mathbf{T}_D$ and patch set $\mathbf{I}_{\text{patch}}, \mathbf{D}_{\text{patch}}$. Then we perform mask operation to get the retained tokens $\mathbf{T}_I^{\text{ret}}, \mathbf{T}_D^{\text{ret}}$. The number of retained tokens for the image and density map modalities are generated by the Dirichlet distribution, denoted as $\mathcal{N}_I^{\text{ret}}$ and $\mathcal{N}_D^{\text{ret}}$, respectively. Specifically, for image modality, we use an adaptive mask to obtain $\mathbf{T}_I^{\text{ret}}$ from $\mathbf{T}_I$. Firstly, we calculate the number of targets $\mathbf{V}_D^i$ in the i-th density map patch $\mathbf{D}_{\text{patch}}^i$, and $\mathbf{V}_D^i = \phi_{\text{sum}}(\mathbf{D}_{\text{patch}}^i)$, where ϕ_{sum} represents the pixel-wise sum operation in each density map patch, and the result represents the number of targets in the corresponding patch. To focus on the foreground, we sort the image tokens according to the number of targets $\mathbf{V}_D^i$ in the corresponding density map patch. Although the foreground provides more valid information, the background should not be completely ignored. Therefore, to introduce background information, we set a background retention probability (BRP) $\mathcal{P}$ to control the sorting manner. The tokens are sorted in ascending order with a probability of $\mathcal{P}$ (focus on background) or in descending order with a probability of $1 - \mathcal{P}$ (focus on

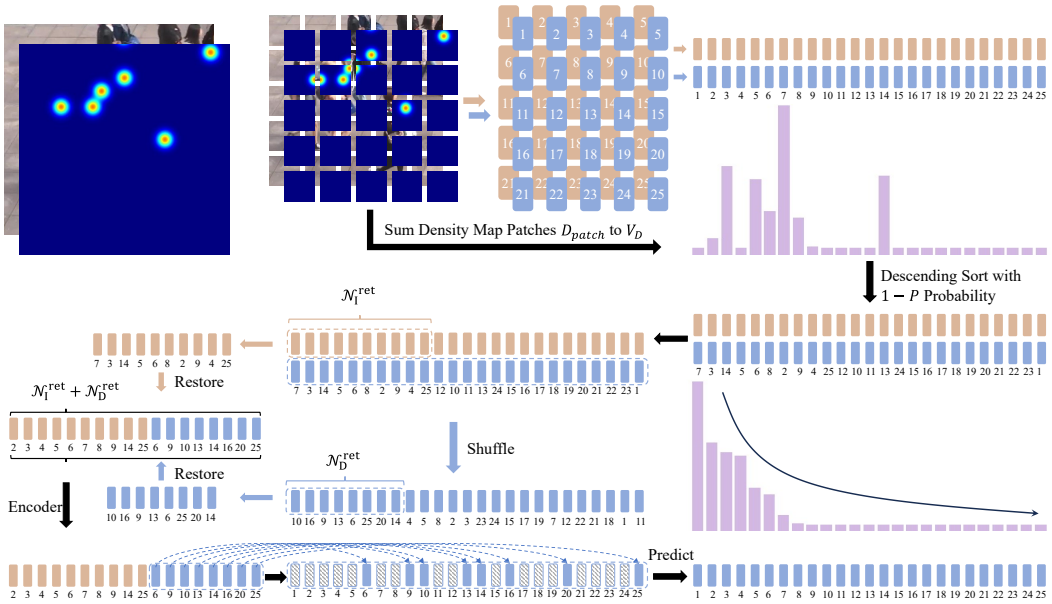

Figure 8: The detailed process of SAM. For ease of expression, we crop the image and the density map into 25 patches, as shown in the figure, this number varies according to the size of input images (each patch is set to 16×16). We first calculate the $\mathbf{V}_D^i$ of each density map patch $\mathbf{D}_{patch}^i$, and sort the token according to the number of targets $\mathbf{V}_D^i$. To balance the fore- and back-ground information, we set a background retention probability (BRP) $\mathcal{P}$ to determine the sorting manner, which is detailed in Sec. 4.3. For tokens from image modality, we keep the first $\mathcal{N}_I^{\mathrm{ret}}$ tokens after sorting. For tokens from density map modality, we randomly **shuffle** their order and keep the first $\mathcal{N}_I^{\mathrm{ret}}$ tokens, i.e., randomly select $\mathcal{N}_I^{\mathrm{ret}}$ density tokens. Since the masked tokens are filled by learnable tokens in the decoder, we first restore the retained image and density tokens to their original order before sorting. Then, we concatenate and feed them into the encoder. Note that the restoration follows the same setting with Bachmann et al. (2022), which can be performed in the decoder as well. The full set of retained tokens and filled density map tokens are then fed into the decoder to predict the density map. Specifically, the full set of retained tokens are treated as key and $value$ vectors, and the filled density map tokens are treated as $query$ vector in the cross-attention layer in the decoder. Then, the output of the cross-attention layer is fed into two self-attention layers to perform the final prediction.

foreground). The first $\mathcal{N}_I^{ret}$ tokens after sorting is then retained as $\mathbf{T}_I^{\mathrm{ret}}$. For density map modality, we randomly shuffle the order of $\mathbf{T}_D$ and retain the first $\mathcal{N}_D^{\mathrm{ret}}$ tokens (randomly mask) as $\mathbf{T}_D^{\mathrm{ret}}$. The retained tokens $\mathbf{T}_I^{\mathrm{ret}}$ and $\mathbf{T}_D^{\mathrm{ret}}$ are jointly fed into the encoder, while the remaining tokens are discarded and not passed into the model. The output tokens $\mathbf{T}^{\mathrm{ref}}$ of encoder are then split to $\mathbf{T}_D^{\mathrm{ret}}$ and $\mathbf{T}_I^{\mathrm{ret}}$. The learnable random mask tokens are filled at the masked positions in $\mathbf{T}_D^{\mathrm{ret}}$ as placeholders, and the filled tokens set is denoted as $\hat{\mathbf{T}}_D$. $\hat{\mathbf{T}}_D$ and $\mathbf{T}^{\mathrm{ref}}$ are then fed in a cross-attention layer, in which $\hat{\mathbf{T}}_D$ servers as *query* and $\mathbf{T}^{\mathrm{ref}}$ servers as *key* and *value*. Then, the reconstructed density map $\hat{D}$ is regressed by two transformer layers.

We have illustrated the detailed SAM process in Fig. 8. For ease of expression, we crop the image and density map into 25 patches. Each patch from the image and the density map at the same position is paired, and we have noted an index for each pair of patches in Fig. 8. For each pair of patches ($\mathbf{I}_{\mathrm{patch}}^i$, $\mathbf{D}_{\mathrm{patch}}^i$), we calculate the sum of the density map patch $\mathbf{V}_D^i$, sort the token pair ($\mathbf{T}_I, \mathbf{T}_D$) according to the number of targets $\mathbf{V}_D^i$. We set a background retention probability (BRP) $\mathcal{P}$ to determine the sorting manner, which is detailed in Sec. 4.3. For tokens from image modality, we retain the first $\mathcal{N}_I^{\mathrm{ret}}$ tokens in the sorted $\mathbf{T}_I$. For tokens from density map modality, we randomly **shuffle** their order and retain the first $\mathcal{N}_D^{\mathrm{ret}}$ tokens. Since the masked tokens will be filled by learnable tokens in the decoder, we first restore the retained image and density tokens to their original order

Table 6: Quantitative comparison between the proposed method and existing methods on the Mall dataset with metrics MAE and RMSE.

Method	Type	MAE↓	RMSE↓
CSRNet (Li et al., 2018)	Image	2.46	4.70
RPNet (Yang et al., 2020)	Image	2.20	3.60
TAN (Wu et al., 2020)	Image	2.03	2.60
HMoDE (Du et al., 2023)	Image	2.82	3.41
PET (Liu et al., 2023)	Image	1.89	2.46
Gramformer (Lin et al., 2024)	Image	1.69	2.14
ConvLSTM (Xiong et al., 2017)	Video	2.24	8.50
LSTN (Fang et al., 2019)	Video	2.00	2.50
E3D (Zou et al., 2019)	Video	1.64	2.13
MLSTN (Fang et al., 2020)	Video	1.80	2.42
MOPN (Hossain et al., 2020)	Video	1.78	2.25
Monet (Bai & Chan, 2021)	Video	1.54	2.02
PFTF (Avvenuti et al., 2022)	Video	2.99	3.72
FRVCC (Hou et al., 2023)	Video	<u>1.41</u>	<u>1.79</u>
STGN (Wu et al., 2023)	Video	1.53	1.97
Ours	Video	**1.35**	**1.76**

Table 7: Quantitative comparison between our proposed method and existing methods on the FDST dataset with metrics MAE and RMSE, lower metrics better.

Method	Type	MAE↓	RMSE↓
MCNN (Zhang et al., 2016)	Image	3.77	4.88
CSRNet (Li et al., 2018)	Image	2.56	3.12
ChfL (Shu et al., 2022)	Image	3.33	4.38
MAN (Lin et al., 2022)	Image	2.79	4.21
HMoDE (Du et al., 2023)	Image	2.49	3.51
PET (Liu et al., 2023)	Image	1.73	2.27
Gramformer (Lin et al., 2024)	Image	5.15	6.32
ConvLSTM (Xiong et al., 2017)	Video	4.48	5.82
LSTN (Fang et al., 2019)	Video	3.35	4.45
MLSTN Fang et al. (2020)	Video	2.35	3.02
EPF (Liu et al., 2020)	Video	2.17	2.62
MOPN (Hossain et al., 2020)	Video	1.76	2.25
PHNet Meng et al. (2021)	Video	1.65	2.16
GNANet (Li et al., 2022)	Video	2.10	2.90
PFTF (Avvenuti et al., 2022)	Video	2.07	2.69
FRVCC (Hou et al., 2023)	Video	1.88	2.45
STGN (Wu et al., 2023)	Video	<u>1.38</u>	<u>1.82</u>
Ours	Video	**1.29**	**1.69**

before sorting. Then, we concatenate and feed them into the encoder. The full set of retained tokens and filled density map tokens are then fed into the decoder to predict the density map. Specifically, the full set of retained tokens are treated as key and $value$ vectors, and the filled density map tokens are treated as $query$ vector in the cross-attention layer in the decoder. Then, the output of the cross-attention layer is fed into two self-attention layers to perform the final prediction.

A.3 More Experiment Results

Additional results on the Mall, FDST, and VSCrowd datasets are provided in Tabs. 6, 7, and 8. In an extensive survey, our method consistently achieved competitive results.

Table 8: Quantitative comparison between our proposed method and existing methods on the VSCrowd dataset with metrics MAE and RMSE, lower metrics better.

Method	Type	MAE↓	RMSE↓
MCNN (Zhang et al., 2016)	Image	27.1	46.9
CSRNet (Li et al., 2018)	Image	13.8	21.1
Bayesian (Ma et al., 2019)	Image	8.7	11.8
MAN (Lin et al., 2022)	Image	8.3	10.4
HMoDE (Du et al., 2023)	Image	19.8	39.5
PET (Liu et al., 2023)	Image	<u>6.6</u>	11.0
Gramformer (Lin et al., 2024)	Image	8.09	15.65
EPF (Liu et al., 2020)	Video	10.4	14.6
GNANet (Li et al., 2022)	Video	8.2	**10.2**
STGN (Wu et al., 2023)	Video	9.6	12.5
Ours	Video	**6.0**	<u>10.3</u>

Table 9: Effect of TCF.

Exp.	Model	**MAE**↓	**RMSE**↓
3	ViT	2.45	3.22
4	ViT w/ TCF	2.32	2.69
5	E-MAC	1.51	2.03
6	E-MAC TCF	**1.29**	**1.69**

Table 10: Effect of E-MAC architecture.

Exp.	Model	**MAE** ↓	**RMSE** ↓
7	vanilla ViT	2.63	3.31
8	MultiMAE	2.45	3.22
9	E-MAC w/ vanilla ViT weight	1.49	1.93
10	E-MAC w/ MultiMAE weight	1.29	1.69

A.4 Impact of Image Size

We conducted experiments on the FDST dataset to explore the effect of image size on the performance of our E-MAC module. We tried a variety of image input sizes and compared their experimental results to validate the effect of input image size. When we increased the image size used for training from 224×224 to 480×480, the final MAE showed a downward trend in the figure and decreased from 1.93 to 1.47, which improved by 24%. However, the computational cost of the model increases exponentially as the size of the image increases (Dosovitskiy et al., 2020). Integrating temporal information further increases the computational cost, thereby affecting the overall performance. Therefore, we set the input image size to 320×320 on the FDST dataset, balancing the performance and computational cost.

A.5 Impact of TCF

Additionally, we have conducted additional ablation studies on the TCF module, as shown in Table 9. We compare the performance of the original ViT architecture (Exp. 3 in Table 9), the ViT framework augmented with TCF (Exp. 4 in Table 9), and E-MAC with or without TCF (Exp. 5 and 6 in Table 9). The results demonstrate that TCF provides performance improvements of 5% and 15% on the two architectures, respectively. To further demonstrate how the fusion module works, we visualized the input $\hat{D}_t$, and output $\hat{D}_{\text{fuse}}$ of the fusion module as well as the intermediate variable $\hat{D}_t^{\text{res}}$ under three datasets, as shown in Fig 9, we provide the original image I_t and corresponding ground truth D_t in the first two rows of the figure for reference. Compared with $\hat{D}_t$, the fusion module can well realize the correction of the current density map by correlating the residuals obtained from the previous and current predicted density map, which makes the integration of the fused density map $\hat{D}_{\text{fuse}}$ closer to the ground truth. We zoom in on some areas, and we notice that the fusion module can remove some of the background interference by correlating the front and back predicted density maps, which makes the final predicted density map better.

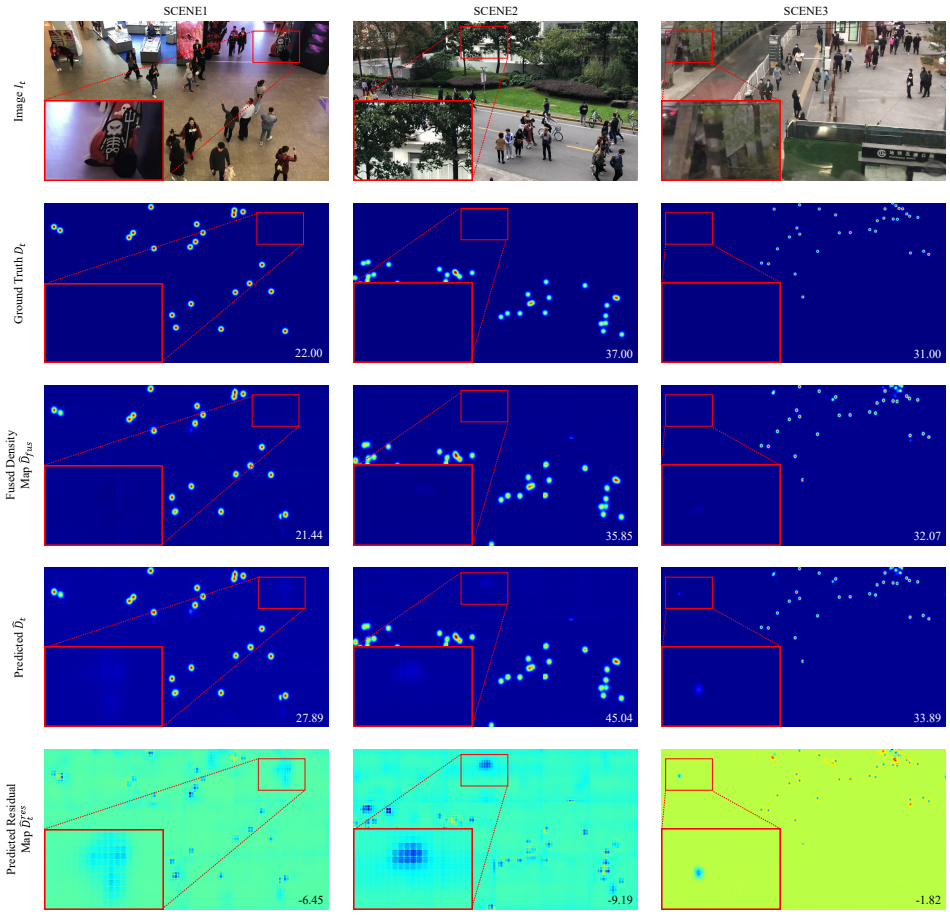

Figure 9: Visualization of the output and intermediate variables in the Fusion Module.

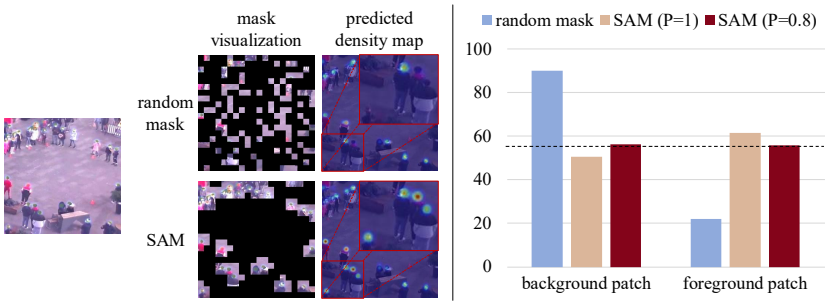

Figure 10: Visualization of the predicted density map w/ and w/o SAM, and statistical results of the number of foreground and background patches w/ and w/o SAM.

A.6 VISUALIZATION OF SAM

We visualized the differences between the density maps predicted from the same image *w/* and *w/o* SAM and the ground truth, as presented in Fig. 10. The zoom-in regions display intuitive and significant visual differences. E-MAC trained *w/* SAM effectively counts the targets, while E-MAC using random masking fails to count some targets. Besides, we conducted a statistical comparison of the number of foreground and background patches in the images after applying random masks and different $\mathcal{P}$ (BRP) settings with SAM, as presented in Fig. 10. Obviously, SAM significantly reduces the proportion of background regions, thereby balancing the positive and negative samples.

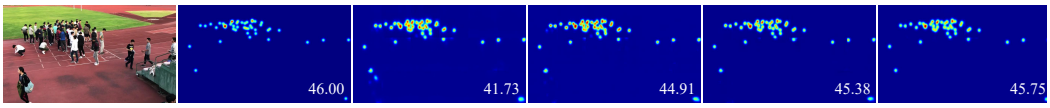

Figure 11: Visualization of the predicted results of our key components (TCF, DEMO, SAM).

Table 12: Comparison on FLOPs, FPS, and the number of Parameters.

Method	FLOPs(G) ↓	FPS ↑	Parameters(M) ↓	MAE (DroneBird) ↓	RMSE (DroneBird) ↓
EPF (Liu et al., 2020)	815	19	20	97.22	133.01
PFTF (Avvenuti et al., 2022)	1075	13	23	89.76	101.02
STGN (Wu et al., 2023)	742	30	13	92.38	124.67
Ours	811	16	98	38.72	42,92

We have also visualized some of the predicted results of our key components (TCF, DEMO, SAM) with our baseline, as shown in Fig. 11. We gradually add the components (TCF, DEMO, SAM) to the baseline, resulting in increasingly accurate predictions and better visual results.

Table 11: Result with different random seeds.

Metrics	1	2	3	Average
MAE	1.29	1.28	1.33	$1.30_{\pm 0.022}$
RMSE	1.69	1.66	1.69	$1.68_{\pm 0.015}$

A.7 Effect of E-MAC architecture.

We provide additional quantitative experiments to demonstrate the performance sources of our proposed framework. As shown in Table 10, we provide a comparison of the performance of the vanilla ViT architecture using different pre-training weights (Exp. 7 and 8 in Table 10) as well as our framework using different pre-training weights (Exp. 9 and 10 in Table 10). As can be seen from the table: using a larger pre-training weight on ViT does provide better performance, but the improvement is very limited (6% improvement between Exp. 7 and 8). In contrast, our unique design tailored to the data and video counting tasks significantly enhances performance (43% between Exp. 7 and 9 and 47% between Exp. 8 and 10 respectively).

A.8 Impact of Random Seed.

To explore the impact of different random seeds on the overall performance of our model, we have also conducted more experiments on the FDST dataset, which is a relatively small dataset for quick validation. The average MAE and RMSE scores are $1.30_{\pm 0.022}$ and $1.68_{\pm 0.015}$. The results in Table 11 show that different random seeds do not significantly affect the performance of our model.

A.9 Model Efficiency Discussion

We have compared FLOPs, FPS, and the number of parameters with some other video counting models and reported the results in Table 12. All the experiments were conducted on an NVIDIA RTX 3090 GPU. It is worth noting that the FLOPs and FPS of our method and the competing methods are comparable, although we first adopted the vision foundation model. Compared to EPF (Liu et al., 2020) and PFTF (Avvenuti et al., 2022), our E-MAC achieves superior performance with less required computations. STGN (Wu et al., 2023) achieves higher FPS with fewer parameters, but its performance is limited. Compared to STGN (Wu et al., 2023), our method achieved over 58% performance improvement with only 9% more FLOPs.

A.10 Dynamics of Data

Here, we explore the intra-frame dynamics of the foreground regions in video data. We processed data in the FDST dataset and made analyses. We first utilized a 60×60 window to crop the images into patches and then counted the number of people in each patch.

Statistical result is shown in Fig. 12, the horizontal axis coordinates are the density of people in each patch, calculated from the corresponding density map. The left vertical coordinate is the number of patches for each crowd density. We fold a portion of the axis as the number gap is too large. The heterogeneous density distribution across image patches indicates inherent dynamism in the intra-frame density characteristics. Additionally, due to the presence of large background areas, the model should focus more on the foreground regions of the samples to extract the most informative and relevant information. The utilization of a fixed focus region across different samples may result in the loss of critical information about the foreground areas. Similarly, the adoption of completely random focus regions is unable to consistently capture the salient information within the foreground regions. Thus, we suggest the employment of a dynamic masking mechanism to obtain the foreground focus regions for different samples.

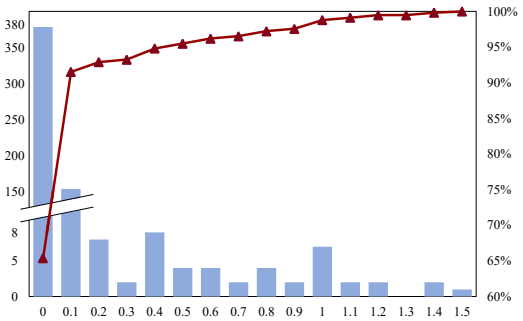

Figure 12: Density distribution. The bar graph portion (blue) represents the number of patches corresponding to the crowd density. The line graph portion (red) represents the percentage of the number of patches whose density is less than the current density.