Appendix

–Supplementary Material–

A Further Discussions about Sec. 3.1

A.1 More Insights behind Benchmark Dataset Design

Table 1 presents the case study that highlights a potential bias in person search models when the same type of corruption is applied to both query and gallery images. The table compares two scenarios: the case where corruption is applied only to the gallery images ('Gallery corrupted'), and the other case where the same corruption is applied to both query and gallery images ('Both corrupted'). Four different types of corruptions are examined: Brightness, Contrast, Saturate, and Spatter. Interestingly, for four corruption types, we observe higher performance when the same corruption is applied to both query and gallery images. This pattern suggests that when both query and gallery images are subjected to the same type of corruption, the model's ability to match them could improve. We consider this potential bias when designing our benchmark dataset, which is to randomly apply corruption types and severity levels to query and gallery images.

Table 1. **Case studies when the same corruption is applied to both query and gallery images**, it could lead to another bias, such that the similarity of the two images increases. CUHK-SYSU-C, severity level 5 are used for the case studies, R@1 is used as an evaluation metric.

Corruptions	Brightness	Contrast	Saturate	Spatter
Gallery corrupted	71.5	11.7	55.9	60.9
Both corrupted	77.8	12.8	62.8	63.6

A.2 Model Explanation in Sec. 3.1

In this section, we describe person search models that we examine in Sec 3.1. These state-of-the-art works have contributed to various aspects of the person search field. OIMNet [18] firstly proposes an end-to-end person search framework by jointly training the detection and reID head. NAE [2] addresses the issue of conflicting learning objectives, a common challenge in person search and related fields [14, 19, 21], that arise during the joint learning of the detector and reID head. SeqNet [13] improves the quality of detection results through a stronger detection head by considering that the detection result influences the training of the reID head. This simple yet effective concept has prompted subsequent studies to adopt its design [7, 11, 12]. OIMNet++ [9] improves the widely used OIM [18] loss and considers the quality of detection results in the training of the reID head. COAT [20], PSTR [1] and SAT [5] enable recent one-step person search frameworks to leverage the advantages of the Transformer [17].

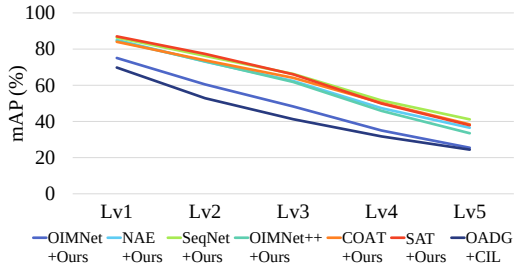
B Further Experiments and Analysis

B.1 Scalability to Larger-Scale Database

Our approach to constructing corruption scenarios is designed to be scalable and applicable to private person databases. This scalability stems from our approach which applies corruptions to existing images rather than collecting new data in various corruption scenarios.

While our primary evaluations are conducted on CUHK-SYSU and PRW—the two standard benchmarks in person search research—we also investigate our method's scalability to larger-scale datasets. Our corruption robustness evaluation framework is specifically designed to be adaptable to various datasets without requiring fundamental modifications.

To validate this scalability, we extend our experiments to PoseTrack21 [4], a large-scale dataset that has recently gained adoption in person search literature [16]. As shown in Figure 1, our method shows consistent performance improvements when applied to both OIMNet++ and SeqNet on this



Method	PoseTrack21-C				PoseTrack21	
	rR@1	rmAP	R@1	mAP	R@1	mAP
OIMNet++	71.3	27.4	63.3	16.6	88.7	60.6
+Ours	**78.4**	**39.9**	**70.7**	**24.5**	**90.1**	**61.4**
SeqNet	69.6	28.7	59.9	17.0	86.1	59.2
+Ours	**81.0**	**46.0**	**70.0**	**27.5**	**86.4**	**59.7**

Figure 1. **Verification of the proposed approach across multiple testing conditions (different severity levels and larger benchmark) Left:** Evaluation of our method across various severities. We apply our method to existing six different person search models. OADG+CIL refers to the combination of existing works mentioned previously. **Right:** Our proposed approach to constructing corruption scenarios is scalable to different database, and our proposed method shows scalable performance on larger benchmark PoseTrack21 [4] as well.

larger benchmark. Specifically, when applied to OIMNet++, our approach improves rR@1 by 7.1% and rmAP by 12.5% on PoseTrack21-C. Similarly, with SeqNet as the baseline, we observe more substantial gains of 11.4% in rR@1 and 17.3% in rmAP.

These results confirm that our proposed method is not limited to smaller-scale benchmarks but can effectively enhance corruption robustness across different dataset scales and characteristics. Furthermore, our approach maintains or slightly improves performance on clean data, demonstrating that robustness can be achieved without sacrificing accuracy in ideal conditions.

B.2 Performance Comparison across Various Severity Levels

We conduct the experiment to evaluate the robustness of our proposed method across various severity levels when applied to six different person search models. The graph in Figure 1 illustrates the mAP performance of five person search models enhanced with our approach, compared against the OADG+CIL combination, across five levels of corruption severity. The graph shows a general downward trend in performance as severity increases. SeqNet+ours and COAT+ours show strong resilience, maintaining high mAP scores even at the most severe corruption levels. We observe that even the basic OIMNet model, when enhanced with our method, consistently surpasses the performance of the OADG+CIL combination. This underscores the effectiveness of our approach, even when applied to simpler baseline models.

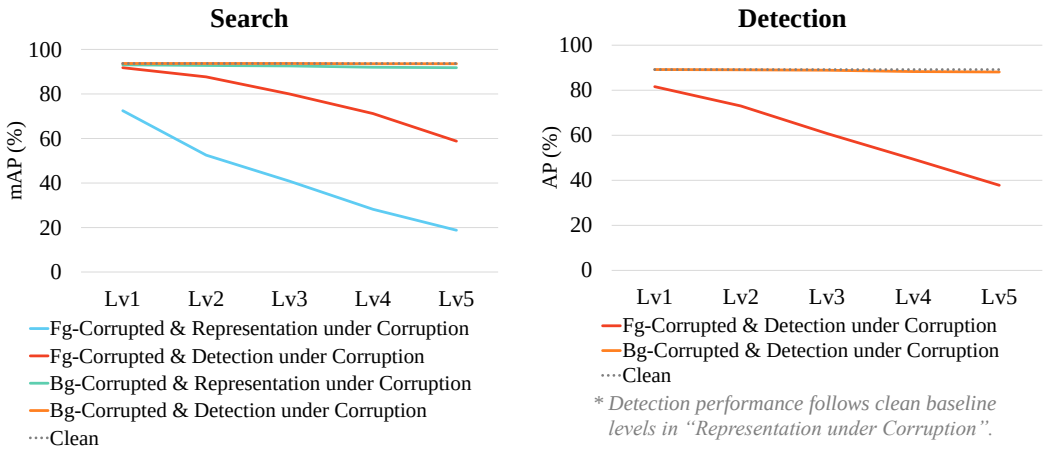


Figure 2. **Analysis of corruption effects on detection and representation stages under foreground and background corruptions.** The terms 'Bg-Corrupted', 'Fg-Corrupted', 'Representation under Corruption', and 'Detection under Corruption' have the same meanings as those used in Tab. 2 and Fig. 3.

B.3 Two-Way Cross-Validation of Person Search Corruption Susceptibility

In Sec. 3.2 and Sec. 3.3 of the main paper, we analyzed which stage (representation or detection) is more susceptible to corruption and examined the influence of foreground and background regions on robust person representation. To cross-validate these analytical findings, we conduct experiments that simultaneously examine both aspects. We analyze four scenarios: 'Bg-Corrupted & Representation under Corruption', 'Bg-Corrupted & Detection under Corruption', 'Fg-Corrupted & Representation under Corruption', and 'Fg-Corrupted & Detection under Corruption'. This experimental design allows us to examine the relative susceptibility of detection and representation stages under both background-only and foreground-only corruptions.

Figure 2 presents our experimental results, revealing several key findings: In the case of 'Fg-Corrupted & Representation under Corruption', search performances align with the trends observed in both Fig. 3's (main paper) 'Fg-Corrupted' and Tab. 2's (main paper) 'Representation under Corruption' cases; The performance from 'Fg-Corrupted & Detection under Corruption' shows similar the patterns seen in Fig. 3's (main paper) 'Fg-Corrupted' and Tab. 2's (main paper) 'Detection under Corruption'; Both search and detection performances under background corruption maintain tendencies similar to those obtained with clean set. These results are consistent with previous findings, validating our observations.

Table 2. **Analysis for τ_{iou} impact.** The number indicates the value set for τ_{iou}, *uniform* denotes not to apply the concept of IoU score, *reverse* indicates applying it in reverse.

τ_{iou}	CUHK-SYSU-C	
	R@1	mAP
reverse	64.2	62.4
uniform	65.0	63.1
0.4	66.2	64.2
0.6	**66.7**	**65.0**
0.8	66.1	64.4

Table 3. **Analysis for impacts of $\mathcal{T}$, τ, and α.**

Type	Settings	CUHK-SYSU-C		Type	Settings	CUHK-SYSU-C		Type	Settings	CUHK-SYSU-C	
		R@1	mAP			R@1	mAP			R@1	mAP
$\mathcal{T}$	color	54.5	52.3	τ	0.1	63.6	61.6	α	0.5	64.8	63.2
	geometric	57.5	55.3		0.2	**66.7**	**65.0**		0.6	**66.7**	**65.0**
	both	**66.7**	**65.0**		0.3	63.2	62.0		0.7	64.2	62.0

B.4 Hyperparameter Analysis

We conduct experiments to analyze four hyperparameters: the temperature for IoU score τ_{iou}, types of augmentation used in $\mathcal{T}$, τ, and the threshold α. Table 2 shows the results of our analysis on the effect of τ_{iou}. In this analysis, 'reverse' refers to the effect of our τ_{iou} applied in reverse. We implement this by applying $1 - \tau_{iou}$, which inverts the shape of the IoU distribution, before applying τ_{iou}. The 'uniform' case represents an uniform IoU distribution, equivalent to not using the IoU score at all. Our results show that performance improves when using the IoU score (with τ_{iou} values of 0.4, 0.6, and 0.8) compared to the 'uniform' case where IoU is not used. Moreover, the 'uniform' case outperforms the 'reverse' case, which aligns with our intuition. The performance remains relatively stable across different hyperparameter values when using the IoU score.

Table 3 presents our analysis of the three hyperparameters: types of augmentation used in $\mathcal{T}$, τ, and α. For the analysis of augmentation $\mathcal{T}$, 'color' represents the evaluation result of a model trained using solarization, autocontrast, equalization, and posterization, while 'geometric' refers to the result of a model trained using rotation, shearing, and translation. The results indicate that the 'both' setting, which combines color and geometric augmentations, yields the best performance. This suggests that both types of augmentation contribute positively, with geometric augmentations showing more effectiveness. Regarding τ and α, we set these parameters as values of 0.2 and 0.6, respectively, which shows the best performance.

Figure 3. **Comparison of proposed corruptions with real-world corruption scenarios** for dark, blur, fog, and rain conditions.

B.5 Qualitative Comparison Between Proposed and Real-World Corruptions

In this section, we compare our proposed corruptions with real-world corruptions gathered as described in Sec. 5. Figure 3 illustrates four corruption scenarios: dark, blur, rain, and fog. Our proposed corruption can capture distinct features in real-world corruption scenes. For example, the proposed dark reflects the reduced overall brightness observed in real-world dark scenes, the proposed blur presents decreased sharpness, and motion traces observed in real-world blur scenes. While our corruption benchmark assumes the presence of only one type of corruption in a scene, corruption in real-world images can be more complex. As shown in Figure 3, the real-world fog image contains some raindrops as well as a fog effect, while the real-world rain image exhibits blurriness. We aim to investigate these multiple corruption scenarios in our future research endeavors.

B.6 Qualitative Results

We present the qualitative results of evaluating different models on corruption scenarios. The results on PRW-C are shown in Figure 5. Each row represents the results of different models, and each column shows different query examples and the search results of each model accordingly. Blue indicates the location of the query person, red represents incorrect detection results by the model, and green represents correct detection results. By looking at examples 1 to 3, where our model successfully retrieves a person while other models fail, we can see the efficacy of our method in extracting robust representations of the person when corruption is applied. We also provide the result of the real-world corruption data (introduced in Sec. 5) in Figure 6.

C Benchmark Details

In this section, we provide comprehensive details about our proposed CUHK-SYSU-C and PRW-C benchmarks, including dataset statistics, implementation specifics for each corruption type, and performance analysis across corruption scenarios. As illustrated in Figure 4, we implement 18 distinct corruption types that represent various real-world scenarios, showing how each corruption transforms the original images with varying visual characteristics. Table 5 shows how these different corruption types affect person search performance.

C.1 Benchmark Statistics

Table 4 presents the statistics of CUHK-SYSU-C and PRW-C. The statistics regarding images, identities, and pedestrians are the same as those for the test split of CUHK-SYSU and PRW. As mentioned in the main paper, each image in CUHK-SYSU-C and PRW-C is randomly assigned to one of the 18 corruptions and randomly assigned to one of the five severity levels. We repeat this process several times (5) and report the average performance.

Table 4. **Statistics for CUHK-SYSU-C and PRW-C benchmarks.**

	# images	# identities	# pedestrians	# images per corruption	# images per severity
CUHK-SYSU-C	6,978	2,900	40,871	388.6	1,395.6
PRW-C	6,112	544	25,062	339.5	1,222.4

C.2 Bias

CUHK-SYSU was collected from various locations in Chinese urban cities and movie scenes, while PRW was filmed at a Chinese university. Both datasets contain a sufficiently large number of individuals representing both genders. While PRW features a relatively younger demographic, CUHK-SYSU includes people of various ages. Both datasets have an ethnicity bias, predominantly featuring Asian individuals.

C.3 Corruption Implementation Details

In creating the benchmarks for the person search, we consider the typical attributes of scenes used in person search, where both prominent subjects and multiple small background figures coexist in the same scene. We conduct the data quality check to ensure that the persons in the images are still detectable and re-identifiable by humans, even after corruption is applied. The implementation details of the corruption in CUHK-SYSU-C and PRW-C are as follows:

Snow. We use the method proposed in [6] to implement the snow corruption. To represent the diverse features of the snowy scene in the real world, we express various attributes of snowy scenes and adjust their intensity for different severity levels. ***Flake Size:*** Determines the average size or thickness of the snowflakes. As this value increases, the size of the snowflakes increases. We set the parameters (0.1, 0.2, 0.55, 0.55, 0.55) to adjust its intensity. ***Size Variation***: Represents the standard deviation of the size distribution of snowflakes. A larger value results in greater variation in the sizes of snowflakes. We set 0.3 as a parameter for the intensity. ***Snowfall Intensity***: Indicates the degree of snowfall intensity applied to the image. Higher values simulate heavier snowfall with more dense snow particles, while lower values create lighter snowfall conditions. We set the parameters (3, 2, 4, 4.5, 2.5) to adjust its intensity. ***Snow Coverage Threshold***: Sets the minimum value for snow generation. Snow will not be generated below this threshold, simulating areas with less snow coverage. We set the parameters (10, 12, 12, 12, 12) to adjust its intensity. ***Wind Effect***: Determines the radius of motion blur. This simulates the direction and speed of falling snow, affected by wind. We set the parameters (0.5, 0.5, 0.9, 0.85, 0.85) to adjust its intensity. ***Blurriness***: Determines the intensity of motion blur. Higher values make the snowflakes appear more blurred, simulating faster-falling snow or stronger wind. We set the parameters (4, 4, 8, 8, 12) to adjust its intensity. ***Snow Opacity***: Determines the mixing ratio between the original image and the snow effect layer. Values closer to 1 show more of the original image, while values closer to 0 intensify the snow effect, simulating denser snowfall. We set the parameters (0.8, 0.7, 0.7, 0.65, 0.65) to adjust its intensity.

Frost. We use the method proposed in [6] to implement the frost corruption. This simulates the effect of frost or ice forming on the surface of the image, giving the appearance of a cold and frosty environment. ***Frost Intensity:*** This determines the strength of the frost effect applied to the image. Higher values result in more pronounced frost, simulating thick ice or frost on the surface. We set the parameters (1, 0.8, 0.7, 0.65, 0.6) to adjust its intensity. ***Blending Ratio:*** Controls how much the frost image is blended with the original image. A lower value results in more frost coverage, while a higher value reveals more of the original image beneath the frost layer. We set the parameters (0.4, 0.6, 0.7, 0.7, 0.75) to adjust the blending ratio.

Fog. We use the method proposed in [6] to implement the fog corruption. This simulates the effect of fog or mist, reducing visibility and softening the details in the image. ***Fog Density:*** This determines the density of the fog applied to the image. Higher values result in denser fog, which obscures more of the image. We set the parameters (1.5, 2.0, 2.5, 2.5, 3.0) to adjust the density. ***Fog Smoothness:*** Controls the rate of decay for the fractal noise used to generate the fog. Lower values

create fog with sharper, more defined transitions, while higher values produce smoother fog with more gradual transitions. We set the parameters (2, 2, 1.7, 1.5, 1.4) to adjust its smoothness.

Rain. We use the method proposed in [15] to implement the rain corruption. To simulate the rain effect on images, we introduce various attributes related to rain and adjust their intensity at different severity levels. ***Slope of Rain Droplets:*** Determines the inclination of rain droplets. This simulates how much the rain is tilted by the wind. At lower intensities, droplets are lighter and more easily tilted by wind, forming steeper slopes. At higher intensities, droplets are heavier, causing rain to fall more vertically and be less affected by wind. We randomly select the slope from within the range (-20, 20) to adjust its intensity. ***Color of Rain Droplets:*** Determines how the rain droplets reflect light. As the severity increases, the color of the droplets changes from light gray to dark gray, simulating the visual effect of increasing rain density. This change reflects the reduction of light reflection and transmission through the air due to heavier rain. The default color is set to light gray (200, 200, 200), while for heavy rain, it is defined as medium gray (150, 150, 150), and for torrential rain, it is represented as dark gray (80, 80, 80). ***Blur Value:*** Indicates the degree of blur effect applied to the image. Higher values make the image blurry, simulating reduced visibility during heavy rainfall. We set the parameters (2, 3, 4, 5, 6) to adjust its intensity. ***Drop Length:*** Sets the length of rain droplets. At higher intensities, the length of droplets increases, creating a more dramatic rain effect. We set the parameters (10, 20, 30, 40, 50) to adjust its intensity. ***Drop Width:*** Determines the width of rain droplets. At higher intensities, the width of droplets increases, enhancing the more distinct rain effect. We set the parameters (1, 2, 3, 4, 5) to adjust its intensity. ***Brightness Adjustment:*** Adjusts the overall brightness of the image to simulate the dark environment of a rainy day. Lower values decrease the overall image brightness, creating a gloomy atmosphere. We set the parameters (0.7, 0.6, 0.5, 0.4, 0.3) to adjust its intensity.

Dark. We use the method proposed in [8] to implement the dark corruption. To simulate darkness, we reduce the overall brightness of the image, making it appear darker. The lower the value, the darker the image becomes. We set the parameters (0.60, 0.54, 0.48, 0.42, 0.36) to adjust its intensity.

Contrast. We use the method proposed in [6] to implement the contrast corruption. To simulate contrast, we adjust the difference in brightness between pixels based on the average brightness of the image. Lower values result in a reduction of contrast, causing the image to appear blurrier and colors to become more uniform. We set the parameters (0.4, 0.33, 0.26, 0.18, 0.1) to adjust its intensity.

Gaussian Noise. We use the method proposed in [6] to implement the gaussian noise corruption. To simulate the gaussian noise, we adjust the noise intensity, which controls the standard deviation of the gaussian noise distribution applied to the image. The higher the value, the more pronounced the noise becomes. We set the parameters (0.05, 0.07, 0.09, 0.12, 0.15) to adjust its intensity.

Speckle Noise. We use the method proposed in [6] to implement the speckle noise corruption. To simulate the speckle noise, we adjust the noise intensity that is multiplied by the pixel values themselves. The higher the value, the more the image is disrupted and appears to have a grainy mixture. We set the parameters (0.1, 0.2, 0.3, 0.4, 0.5) to adjust its intensity.

Gaussian Blur. We use the method proposed in [6] to implement the gaussian blur corruption. To simulate the gaussian blur, we adjust the standard deviation of the Gaussian filter applied to the image. The higher the value, the more blurred the image appears. We set the parameters (1, 2, 3, 4, 5) to adjust its intensity.

Motion Blur. We use the method proposed in [6] to implement the motion blur corruption. This simulates the blur caused by the movement of the camera or objects in the scene during exposure, producing a streaking effect. ***Motion Trace Length:*** This controls the length of the motion blur, representing how far objects have moved during the exposure. A higher radius results in a more pronounced blur effect, simulating faster movement. We set the parameters (10, 15, 15, 15, 20) to adjust its intensity. ***Blur Sharpness:*** Determines the sharpness of the motion blur. Higher values result in a smoother blur, while lower values retain more definition along the motion streak. We set the parameters (3, 5, 8, 12, 15) to adjust its sharpness. ***Angle Direction:*** Controls the angle at which

the motion blur is applied, simulating motion in different directions. The angle is randomized within a certain range to simulate natural motion blur effects caused by different movements.

Defocus Blur. We use the method proposed in [6] to implement the defocus blur corruption. To represent the diverse features of defocus blur in real-world photography, we express two attributes of defocus and adjust their intensity for different severity levels. ***Blur Kernel Size:*** Determines the size of the blur kernel. As the radius increases, the blurring spreads over a larger area, causing details to fade into the background. We set the parameters (3, 4, 5, 7, 9) to adjust its intensity. ***Blur Smoothness:*** Controls the smoothness or sharpness of the blur effect. Higher values produce a smoother, more gradual blur, while lower values retain sharper transitions at the edges of the blur. This affects the overall softness of the defocus effect. We set the parameters (0.1, 0.5, 0.5, 0.5, 0.5) to adjust its intensity.

Glass Blur. We use the method proposed in [6] to implement the glass blur corruption. To represent the diverse features of the glass blur scene in the real world, we express various attributes of glass blur and adjust their intensity for different severity levels. ***Blur Strength:*** Determines the strength of the glass blur applied to the image, simulating distortion as seen through the glass. Higher values result in a more blurred and diffused image, where details become softer and less defined. As sigma increases, the overall smoothness of the blur effect intensifies. We set the parameters (0.7, 0.9, 1, 1.1, 1.5) to adjust its intensity. ***Glass Distortion Magnitude:*** Represents the degree of distortion caused by imperfections in the glass. As this value increases, the image appears more warped, simulating the effect of viewing through glass with varying thickness or composition. We set the parameters (1, 2, 2, 3, 4) to adjust its intensity. ***Distortion Repetitions (Iterations):*** Determines how many times the displacement effect is applied, creating multiple layers of distortion. More iterations result in a more pronounced and complex glass-like effect. We set the parameters (2, 1, 3, 2, 2) to adjust its intensity.

Elastic Transform. We use the method proposed in [6] to implement the elastic transform corruption. To simulate elastic distortion, we apply random, small-scale deformations to the image pixels, adding a flexible, rubber-like warping effect. The higher the value, the more pronounced the distortions become, making the image appear more heavily warped. We set the parameters (12.5, 16.25, 21.25, 25, 30) to adjust its intensity.

Spatter. We use the method proposed in [6] to implement the spatter corruption. This method simulates liquid splashes or mud spatters on the image, which can occur in outdoor or dirty environments, distorting visibility and adding a natural effect of environmental interference. ***Liquid Amount:*** This determines the average amount of liquid spattered on the image. Higher values simulate heavier splashes or more liquid, resulting in larger and more widespread spatter areas. We set the parameters (0.65, 0.65, 0.65, 0.65, 0.67) to adjust its intensity. ***Size Variation:*** Represents the standard deviation of the size of spatter drops. A larger value results in more variation in the size of splatter particles, simulating irregular drops. We set the parameters (0.3, 0.3, 0.3, 0.3, 0.4) to adjust its intensity. ***Blur Radius:*** Determines the size of the gaussian blur applied to the liquid layer. Higher values create more diffused spatter, simulating less defined edges and softer splashes. We set the parameters (4, 3, 2, 1, 1) to adjust the blurriness. ***Coverage Threshold:*** This sets the minimum intensity threshold for spatter formation. Spatter below this threshold is not visible, simulating splatters that did not adhere to the surface. We set the parameters (0.69, 0.68, 0.68, 0.65, 0.65) to adjust its coverage. ***Opacity:*** This controls the opacity of the spatter layer. Higher values create more opaque spatter, making it more prominent on the image. Lower values create more transparent splatter effects, simulating thinner layers of liquid. We set the parameters (0.6, 0.6, 0.5, 1.5, 1.5) to adjust its visibility. ***Mud vs. Water Effect:*** This determines whether the spatter simulates water (0) or mud (1). When set to 1, the spatter is brown and more opaque, simulating thick mud. When set to 0, the spatter simulates pale water splashes. We set the parameters (0, 0, 0, 1, 1) to adjust its effect.

Saturate. We use the method proposed in [6] to implement the saturate corruption. To represent varying levels of color saturation in real-world scenarios, we express different attributes that influence color intensity and adjust them for different severity levels. ***Saturation Scale:*** This determines the strength of the color saturation applied to the image. Higher values result in more vivid and intense colors, while lower values make the image appear more desaturated or washed out. Both

high or low saturation scales can lead to distortion and degradation of the clean image. We set the parameters (2, 0.2, 0.1, 0.3, 5) to adjust its intensity. ***Offset:*** Represents a constant value added to the saturation scale. It controls the base level of saturation applied uniformly across the image, ensuring that even low saturation images retain some color vibrance. We set the parameter (0, 0, 0, 0, 0.1) to adjust the base intensity.

Pixelate. We use the method proposed in [6] to implement the pixelate corruption. To simulate pixelation, we reduce the resolution of the image, converting it into larger blocks of pixels, then resizing it back to the original resolution, removing details. The lower the value, the stronger the pixelation effect, making the image appear more blocky. We set the parameters (0.6, 0.5, 0.4, 0.3, 0.25) to adjust its intensity.

JPEG Compression. We use the method proposed in [6] to implement the JPEG compression corruption. To simulate JPEG compression, we adjust the compression quality level to observe its effects on image quality. The lower the value, the more the image quality degrades, leading to more artifacts and damage. We set the parameters (25, 18, 15, 10, 7) to adjust its intensity.

Brightness. We use the method proposed in [6] to implement the brightness corruption. To simulate brightness, we adjust the overall brightness of the image by adding a constant value to the pixel values. The higher the value, the brighter the image becomes, and the lower the value, the darker the image appears. We set the parameters (0.1, 0.2, 0.3, 0.4, 0.5) to adjust its intensity.

Figure 4. **All types of corruption scenarios in our proposed benchmarks.**

Table 5. **Performance by different types of corruptions.**

Type	**Snow**	**Frost**	**Fog**	**Rain**	**Dark**	**Contrast**
R@1/mAP	59.9 / 55.5	64.6 / 63.1	84.8 / 83.3	72.5 / 70.0	83.5 / 82.3	62.9 / 58.5
Type	**Gaussian Noise**	**Speckle Noise**	**Gaussian Blur**	**Motion Blur**	**Defocus Blur**	**Glass Blur**
R@1/mAP	32.3 / 27.8	71.0 / 68.9	79.2 / 75.6	79.1 / 74.5	79.3 / 76.5	81.5 / 79.0
Type	**Elastic Transform**	**Spatter**	**Saturate**	**Pixelate**	**JPEG Compression**	**Brightness**
R@1/mAP	92.6 / 91.7	79.5 / 77.7	54.2 / 45.0	78.8 / 76.7	78.3 / 73.2	90.3 / 89.3

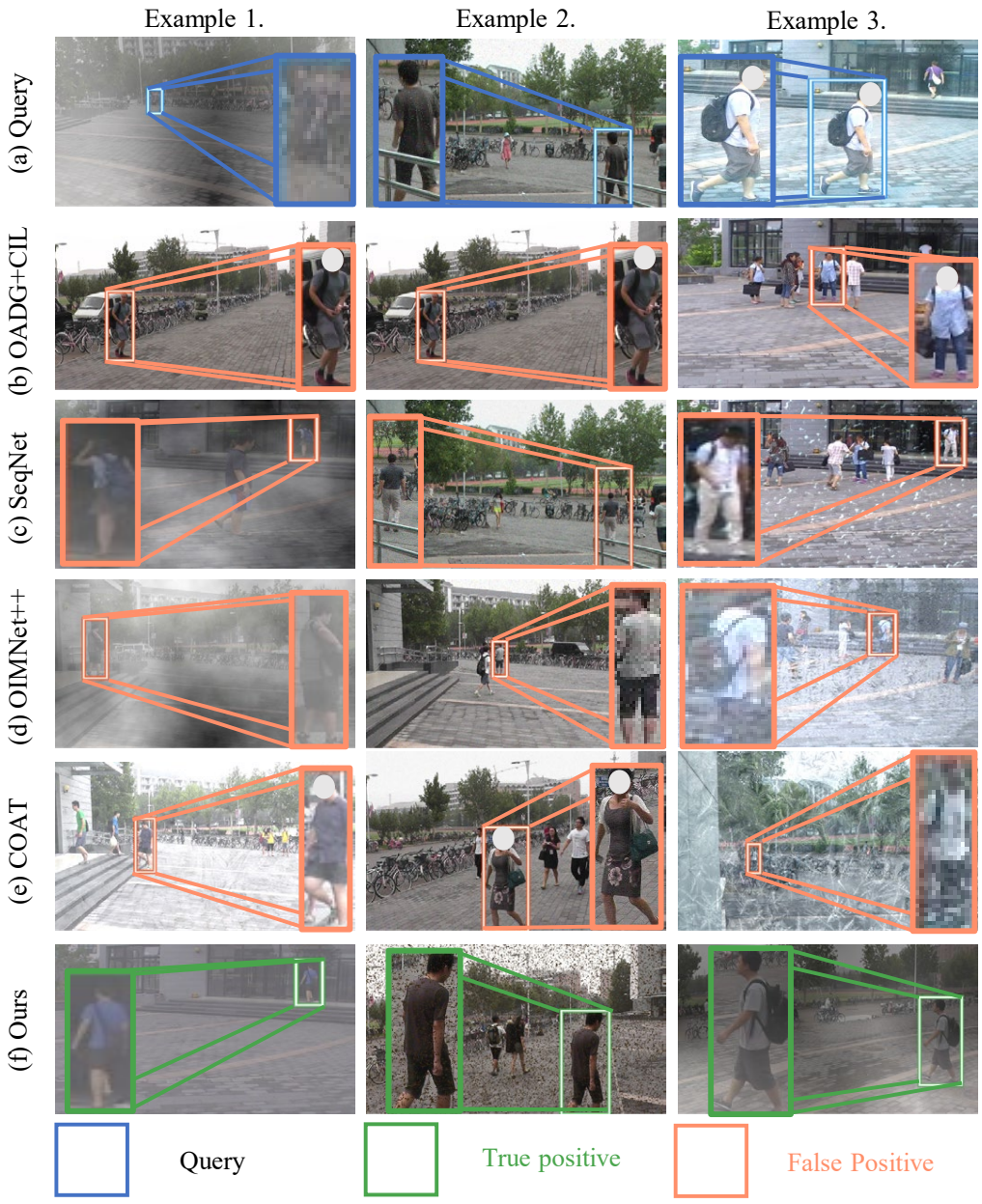


Figure 5. **Qualitative results of person search** on the PRW-C Dataset. The first row displays query images, while the second, third, fourth, and fifth rows show the results from OADG+CIL [3, 10], SeqNet [13], OIMNet++ [9], COAT [20], and Ours with SeqNet, respectively. Each column indicates the different query and the corresponding retrieval results of various models. The blue color denotes the box for a query, the red color indicates the box for failure cases, and the green color represents the box for success cases. A total of 18 types of corruption and 5 levels of severity are involved in establishing the PRW-C dataset.

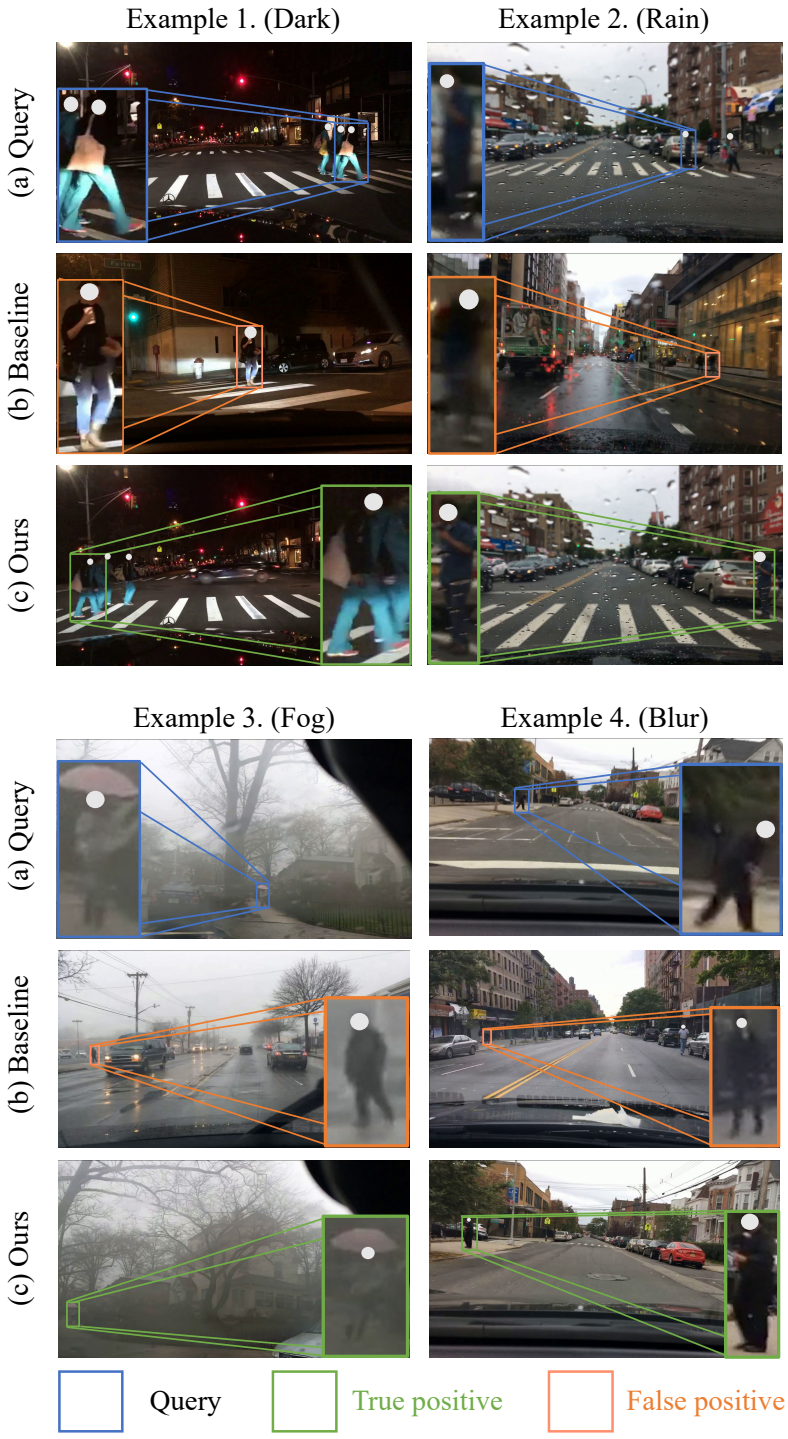


Figure 6. **Qualitative results of baseline and our method on real-world corruptions.** Baseline refers to SeqNet [13].

Ethics Statement

–Supplementary Material–

D Social Impact

The development and deployment of person search systems require careful consideration due to their potential impact on individual privacy and societal norms. While these systems offer substantial benefits for security and surveillance by enhancing the ability to locate and identify individuals across diverse environments, they simultaneously pose a profound risk to personal privacy. The ability to track and identify individuals without their consent can lead to a range of privacy violations, from the unwarranted monitoring of public movements to the potential for misuse in stalking and harassment. It is, therefore, imperative that developers, implementers, and policymakers involved in the creation and use of person search systems consider these ethical implications from the outset. The development of these systems should be guided by ethical principles that prioritize the well-being and privacy of individuals, incorporating mechanisms for accountability and oversight to prevent misuse. It is our collective responsibility to balance innovation with the imperative to safeguard human dignity and privacy.

By doing the above, we can harness the benefits of person search systems while mitigating the risks, ensuring these technologies enhance societal welfare without compromising individual freedoms. For instance, person search technology can expedite victim identification and rescue efforts in natural disasters or accidents, improving the effectiveness of emergency responses and potentially saving lives. Disaster environments often involve challenges like heavy rainfall or snowfall, while accident scenes may present issues such as varying lighting conditions or spatter-covered camera lenses. Our study aims to optimize person search techniques to function effectively under these varied and adverse conditions.

E Further Discussions about Datasets

Privacy. Testing person search models in diverse situations and environments requires collecting new data for various situations with people in them, which can raise ethical issues. Since we provide 18 different scenarios for evaluating the person search models, our benchmarks serve as a good proxy for evaluating models across diverse environments without collecting new data. The parent datasets CUHK-SYSU [18], PRW [22] expose people's faces, and our datasets (CUHK-SYSU-C, PRW-C) inevitably inherit this issue. To mitigate the issue, we mask faces that are captured prominently and distinctly visible for the figures used in the paper to ensure their identities are not recognizable. Future researchers using the proposed CUHK-SYSU-C and PRW-C must be aware of this and take precautionary measures. When using them in research papers, we strongly recommend to de-identify individuals when their faces are clearly recognizable. Regarding the subset collected from BDD100K for Sec. 5, we ensure that the samples we utilize are from a data source that has already been publicly available, carefully adhering to its licensing terms.

License. We will release them in the form of code to generate our benchmarks from the parent datasets(CUHK-SYSU, PRW) rather than as raw images. The authors of the parent datasets provide guidelines for dataset usage on their respective websites, which are similar to the terms of CC BY-NC: For CUHK-SYSU, users are permitted to use the data only for non-commercial research and educational purposes. Users are not permitted to distribute the data. For PRW, they emphasize the purpose of non-commercial research applications. The dataset requires citation. For these reasons, we will also release CUHK-SYSU-C and PRW-C under CC BY-NC.

References

[1] Jiale Cao, Yanwei Pang, Rao Muhammad Anwer, Hisham Cholakkal, Jin Xie, Mubarak Shah, and Fahad Shahbaz Khan. Pstr: End-to-end one-step person search with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9458–9467, 2022.

[2] Di Chen, Shanshan Zhang, Jian Yang, and Bernt Schiele. Norm-aware embedding for efficient person search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12615–12624, 2020.

[3] Minghui Chen, Zhiqiang Wang, and Feng Zheng. Benchmarks for corruption invariant person re-identification. *Advances in Neural Information Processing Systems*, 2021.

[4] Andreas Döring, Di Chen, Shanshan Zhang, Bernt Schiele, and Jürgen Gall. Posetrack21: A dataset for person search, multi-object tracking and multi-person pose tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20963–20972, June 2022.

[5] Mustansar Fiaz, Hisham Cholakkal, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Sat: scale-augmented transformer for person search. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 4820–4829, 2023.

[6] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[7] Lucas Jaffe and Avideh Zakhor. Gallery filter network for person search. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1684–1693, 2023.

[8] Lingdong Kong, Shaoyuan Xie, Hanjiang Hu, Lai Xing Ng, Benoit Cottereau, and Wei Tsang Ooi. Robodepth: Robust out-of-distribution depth estimation under corruptions. *Advances in Neural Information Processing Systems*, 36, 2024.

[9] Sanghoon Lee, Youngmin Oh, Donghyeon Baek, Junghyup Lee, and Bumsub Ham. Oimnet++: Prototypical normalization and localization-aware learning for person search. In *European Conference on Computer Vision*, pp. 621–637. Springer, 2022.

[10] Wooju Lee, Dasol Hong, Hyungtae Lim, and Hyun Myung. Object-aware domain generalization for object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 2947–2955, 2024.

[11] Junjie Li, Yichao Yan, Guanshuo Wang, Fufu Yu, Qiong Jia, and Shouhong Ding. Domain adaptive person search. In *European Conference on Computer Vision*, pp. 302–318. Springer, 2022.

[12] Junjie Li, Guanshuo Wang, Yichao Yan, Fufu Yu, Qiong Jia, Jie Qin, Shouhong Ding, and Xiaokang Yang. Generalizable person search on open-world user-generated video content. *arXiv preprint arXiv:2310.10068*, 2023.

[13] Zhengjia Li and Duoqian Miao. Sequential end-to-end network for efficient person search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 2011–2019, 2021.

[14] Xiangtan Lin, Pengzhen Ren, Yun Xiao, Xiaojun Chang, and Alex Hauptmann. Person search challenges and solutions: A survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

[15] Ujjwal Saxena. Automold–road augmentation library. `https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library`, 2023.

[16] Yanling Tian, Di Chen, Yunan Liu, Jian Yang, and Shanshan Zhang. Divide and conquer: Hybrid pre-training for person search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 5224–5232, 2024.

[17] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[18] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3415–3424, 2017.

[19] Yuanlu Xu, Bingpeng Ma, Rui Huang, and Liang Lin. Person search in a scene by jointly modeling people commonness and person uniqueness. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 937–940, 2014.

[20] Rui Yu, Dawei Du, Rodney LaLonde, Daniel Davila, Christopher Funk, Anthony Hoogs, and Brian Clipp. Cascade transformers for end-to-end person search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7267–7276, 2022.

[21] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International journal of computer vision*, 129:3069–3087, 2021.

[22] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, Yi Yang, and Qi Tian. Person re-identification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1367–1376, 2017.