Acquire Fix-Point Images from Internet Video Stream Captured by Moving Cameras under Bad Weather

Ping Juei Liu[1*], Yu-Cheng Wang[2]

ABSTRACT

In this study, we propose and evaluate a fixed-point image-acquiring framework to overcome adverse situations caused by camera activities. Our method integrates optical flow algorithms and random sample consensus (RANSAC) to detect the orientation and focal center of the camera's motion and zoom-in, respectively. Image restoration study is a typical example facing data collection issues because the samples and the ground truth must be captured at the same positions in different situations, necessitating substantial human resources. Therefore, automation and pre-processing are essential skills in corresponding preliminary works. Our method helps collect image restoration datasets in the experiments by analyzing the 24-hour online video stream. Corresponding results indicate our method has satisfactory performance, resulting in only 21% accuracy loss in unfavorable natural and artificial conditions compared with tasks processing on sunny days.

Keywords: Haze, Dehaze, Haze Removal, Coefficient Optimization, Self-adaptive Coefficient, Contrastive Learning

I. INTRODUCTION

In the recent world, supervised machine learning has become an emerging task; therefore, data involving particular ontology were priceless for those emerging tasks and vendible products. Data acquisition plays a vital role in developing today's technology. Conventionally, data acquisition relied on much staffing. Automation of data acquisition reduced costs and eventually facilitated the progress of machine learning. However, acquiring data from the real world was challenging, even in the modern world with advanced equipment. Image restoration was a vital field studying the recovery of impaired images [1]. Image restoration was distinct from enhancement tasks, aiming to eliminate adverse effects imposed on the scene instead of enhancing brightness, contrast, or detailed features; for example, the dehazing task proposed in [2]-[5]. The comprehensive hazy dataset should involve impaired images and corresponding ground truth captured in fixed positions [6][7]. With the ground truth, researchers could train a model by minimizing the error between the dehazed image and ground truth; however, acquiring the comprehensive hazy dataset remains a case-by-case challenge; it is usually time-consuming and needs various techniques.

Lacking the comprehensive hazy dataset, some researchers used synthesized datasets instead to train their models. Li et al. proposed a hazy dataset [8] composed of real-world images collected from the internet and used the real-world images to synthesize hazy images. Zhang et al. proposed a method to simulate various weather conditions; in this way, they synthesized impaired images, including hazy and sandstorm conditions [9]. Synthesized hazy datasets were the remedy for the lack of a comprehensive hazy dataset with sacrifices of proper performance because the simulation of the natural environment was not promising.

There are two ways to build a comprehensive hazy dataset. First, collecting images using the camera in fixed positions with staffing is highly time-consuming because machine learning takes many samples; meanwhile, adverse weather only happens sometimes. Second, images can be collected online; real-time video streaming is a common technique in developed countries. The second method is relatively convenient because some cameras are in fixed positions. For example, the Taiwan government has conducted the open government platform (OGP) [11] and deployed many cameras at tourist spots [12][13], vital traffic places [14], and air-quality monitoring stations [15]; these resources allow us to acquire data indoors with computer programs; moreover, the cameras are deployed in fixed positions and are capable of collecting images with the change of weather.

Moving cameras was the first challenging problem in building the comprehensive hazy dataset based on OGP and caused many problems because the scene content, lighting, and natural-environment features changed; this resulted in issues when adjusting the camera parameters to adapt to the changes. Therefore, moving camera issues were widely studied in various research fields, such as online tracking [16], object detection [17][18], and semantic parsing [19]. As previously discussed, the comprehensive hazy dataset must contain hazy and clear images (the ground truth), meaning cameras should be static and have a fixed orientation; therefore, moving cameras are problematic for capturing samples. Feature matching is a fundamental technique that allows researchers to find corresponding points between two images and calculate

*Corresponding Author: Ping Juei Liu (E-mail:immich0716@gmail.com).
[1] Department of Artificial Intelligence and Engineer, National Chin-Yi University of Technology, No.57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411030, Taiwan (R.O.C.).
[2] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, No.57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411030, Taiwan (R.O.C.).

the movement of cameras. However, feature-matching algorithms are various, and most take time; therefore, optical flow algorithms [20][21] were widely used to avoid heavy computation.

Gibson [20] provided the fundamentals of calculating the object's shifting; various derivative algorithms were proposed afterward. In computer vision, the features used to calculate the optical flow comprised keypoints and their feature descriptors. Keypoints pointed out where the features were, while their feature descriptors described how to extract them. From the perspective of feature descriptors, optical flow algorithms based on low-level properties might save execution time because feature descriptors take time, such as SIFT [22]; however, this resulted in another problem: the keypoints used to calculate the optical flow might have very dense distributions or take much time to calculate. Like other algorithms, the optical flow algorithms needed help with a trade-off between time complexity and accuracy; generally, the gap in time complexity lay on feature descriptors and keypoint distributions. As a result, optical flow algorithms were distinct as dense, semi-dense, and sparse methods according to keypoint distributions. The dense method considered all pixels as keypoints. The semi-dense method involves only those pixels having a gradient, and the sparse method extracts keypoints using light-workload methods, such as corner detection algorithms.

The most famous sparse method was called the Lucas-Kanade optical flow (LKOF) [23]. LKOF assumed that keypoints in the world were rigid or deformable, elastically moving together coherently; therefore, LKOF looked for a constant motion for a limited number of keypoints. Similarly, the Kanade-Lucas-Tomasi feature tracker (KLT) [24][25] was a semi-dense method, calculating ShiTomasi corner points [25] and taking them as keypoints. Specifically, KLT pursued a balance between execution time and accuracy. On the contrary, Horn-Schunck optical flow (HSOF) was a dense method that considered optical flow a smooth flow within a very short time [26]; therefore, HSOF looked forward to a stable movement within every pixel and its neighbors. HSOF was a robust method that defined a smooth regularization term. With a similar concept, G. Farneback et al. proposed an algorithm (GFOF) [27] computing every pixel's optical flow, which results in a considerable computational workload but might generate accurate estimations. The above-discussed algorithms enabled the motion calculation from low-level (points or edges) or high-level (features, objects) features in images.

On the other hand, the optical flow was based on the assumption that the brightness of objects was invariant everywhere; this means an assumption of uniform illumination regardless of scene depth. As a result, changeable weather was the second challenging problem, resulting in non-uniform illumination and various disturbances. For example, the weather in Taiwan's mountain area was famous for being rapidly changed and unpredictable. Besides the haze and fog, changeable weather was widely discussed in various study fields, such as in the study of vehicle and saliency feature detection [28][29]. Furthermore, rain could disturb optical flow algorithms, causing rain streaks, veiling effects, and the len droplet [30]. Collecting data in the natural environment was challenging in complicated adverse situations.

II. RELATED WORKS

A. Optical Flow

Under the assumption of invariant brightness in the video stream, the brightness of a pixel in a frame is as follows:

$$I(X,t) = I(X + \delta(X, t + \Delta t)), \quad (1)$$

where I denotes frames in the same video stream, X denotes the x and y coordinates of the center of the region of interest (ROI), and δ was called the replacement, meaning the coordinate change and was related to previous coordinates X and time changes Δt. The problem of the model is that the coordinate was self-correlated and sometimes violated because of occluding boundaries, hugely changed color, or surface brightness changes; this resulted in different flows within ROI. Moreover, when a tracked object went out of the windows and returned, the accuracy of LKOF was doubtful as reported in [31] by Shi et al. Shi et al. discussed the situation and proposed a new model, including an affine matrix in the replacement, as follows:

$$\delta = DX + d, \quad (2)$$

where D was the affine matrix describing the deformability as follows:

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}, \quad (3)$$

and d was the translation. Therefore, the motion was simplified as a simple matrix form as follows:

$$I(X) = I(AX + d), \quad (4)$$

where A was defined as follows:

$$A = 1 + D. \quad (5)$$

Note that $\mathbf{1}$ denoted an identical matrix of $R^{2\times 2}$. In this way, Shi et al. successfully improve the performance of tracking objects. However, this model needs a solver to resolve a 6x6 matrix system iteratively, which takes much time. To balance the accuracy and execution time, Shi et al. conducted experiments to decide the best combination of the two models; eventually suggested using (4) to track objects and LKOF to estimate the camera motion.

LKOF model simplified δ to the fixed translation of all keypoints to achieve faster execution and flexibility. The LKOF model was as follows:

$$I(X) = I(X + d), \quad (6)$$

where d denoted the translation. The LKOF model was widely used because it worked accurately with a very small ROI or even a keypoint. To analyze the relation associated with the motion, the Taylor series of (6) is necessary, as follows:

$$I(X + d) \approx I(X) + \frac{\partial I(X)}{\partial x} d_x + \frac{\partial I(X)}{\partial y} d_y + \frac{\partial I(X)}{\partial t} d_t. \quad (7)$$

The above equation was truncated by ignoring the higher oder terms, resulting in a linear model. By the assumption of invariant brightness, (7) should lead to the following:

$$\frac{\partial I(X)}{\partial x} d_x + \frac{\partial I(X)}{\partial y} d_y + \frac{\partial I(X)}{\partial t} d_t \approx 0. \quad (8)$$

To obtain the velocity V, both side in (8) were divided by d_t, generating the following:

$$\frac{\partial I(X)}{\partial x}V_x + \frac{\partial I(X)}{\partial y}V_y \approx -\frac{\partial I(X)}{\partial t}. \quad (9)$$

In an image, the derivative could be replaced with the gradient operator; therefore, the optical flow was written in the matrix form as follows:

$$[\nabla_x, \nabla_y]\begin{bmatrix}V_x\\V_y\end{bmatrix} \approx -\nabla_t, \quad (10)$$

where ∇_x, ∇_y, and ∇_t respectively indicated the gradient of x axis, y axis, and time. In video frames, (10) usually led to the overdetermined situation if keypoints were sufficient; therefore, using the least square method could quickly solve the optimal estimation of V_x and V_y. Specifically, LKOF suggested using weighted least square and iteratively selecting keypoints to optimize V_x and V_y, and the details of (10) were as follows:

$$\begin{bmatrix}\nabla_{x_1} & \nabla_{y_1}\\ \dots & \dots\\ \nabla_{x_n} & \nabla_{y_n}\end{bmatrix}\begin{bmatrix}w_1 & \dots & 0\\ \dots & \dots & \dots\\ 0 & \dots & w_n\end{bmatrix}\begin{bmatrix}V_{x_1} & \dots & V_{x_n}\\ V_{y_1} & \dots & V_{y_n}\end{bmatrix} \approx -[\nabla_{t_1} \quad \dots \quad \nabla_{t_n}], \quad (11)$$

where n denoted the number of keypoints, and w_n was the weight. Note that w_n formed a diagonal matrix and was usually negatively proportional to the distance from the ROI center.

B. Shi-Tomas Corner

Shi-Tomas corner detector [24][25] was an evolution version of the Harris corner detector [32]. Harris et al. figured out that compared with other situations, patches containing any corner structure had considerable dissimilarity when they shifted to any orientation; this resulted in a formula to detect corner points by calculating the dissimilarity, as follows:

$$\epsilon = \sum_X w_s(I(X) - I(X-d))^2, \quad (12)$$

where w_s was the weight. Similar to the derivation from (6) to (7), we could linearize the later part of the equation based on the Taylor series, and the result was as follows:

$$I(X-d) \approx I(X) + \frac{\partial I(X)}{\partial x}d_x + \frac{\partial I(X)}{\partial y}d_y. \quad (13)$$

Note that there was no d_y because this happened in a single image, and the derivative could be replaced with the gradient operator in images. Therefore, substituting (13) to (12) generated the following:

$$\epsilon \approx \sum_X w_s\left(\nabla_x d_x + \nabla_y d_y\right)^2. \quad (14)$$

Therefore, this generated the quadratic equation of matrix form, as follows:

$$\epsilon \approx [d_x, d_y]M\begin{bmatrix}d_x\\d_y\end{bmatrix}, \quad (15)$$

where the matrix M was a self-correlated matrix, defined as follows:

$$M = \begin{bmatrix}(\nabla_x)^2 & \nabla_x\nabla_y\\ \nabla_y\nabla_x & (\nabla_y)^2\end{bmatrix}. \quad (16)$$

Harris et al. suggest using a surrogate benchmark to estimate corner points as follows:

$$R = detM - \alpha * trace(M)^2, \quad (17)$$

where R is the surrogate benchmark, and α is a coefficient to weight the trace.

However, the surrogate benchmark sometimes caused problems, resulting in accuracy decay. Shi et al. explained that Harris's equation in (17) is highly related to λ_n because of the following:

$$detM = \lambda_1 * \lambda_2, \quad (18)$$

and the trace was as follows:

$$trace(M) = \lambda_1 + \lambda_2. \quad (19)$$

However, the quadratic equation was the elliptic function, and the eigenvalue λ_n of the self-correlated matrix M denoted the texture changes in different axes of the elliptic. Shi et al. correlated λ_n to the texture and concluded three cases. Two small λ_n indicated the constant intensity profile of texture patterns, and a large λ_n and a small λ_n show the unidirectional texture pattern. Two large λ_n strongly suggested corner points, salt-and-pepper noises, or other reliable features; therefore, good features were highly associated with λ_n. Accordingly, they defined the following to obtain R:

$$R = min(\lambda_1, \lambda_2). \quad (20)$$

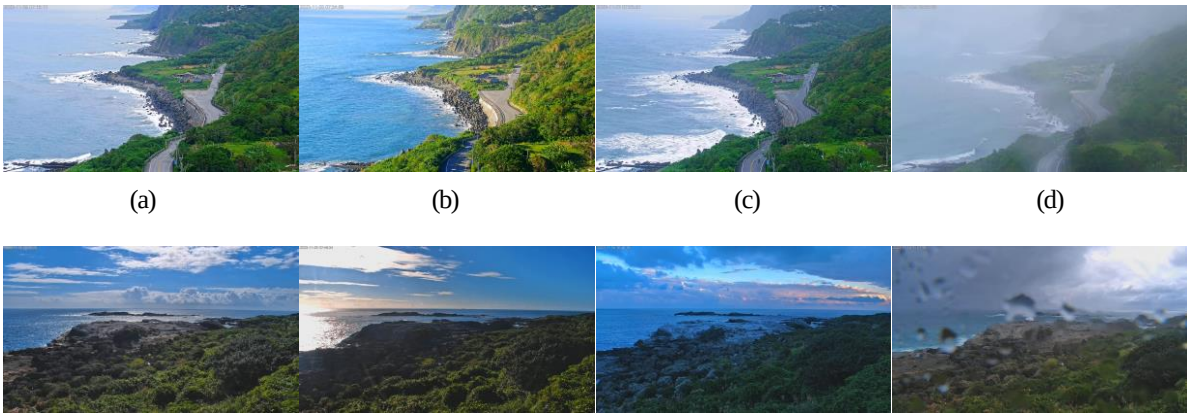
This way, when R was larger than a given thresholding, it meant there was a reliable corner point.

C. Video Stream

The OGP system is built on the YouTube platform based on various streaming techniques. The HTTP live streaming (HLS) [35] is the most popular streaming protocol owing to Apple's support; however, it only supports H.264 and H.265. On the other hand, real-time messaging protocol (RTMP) is also a widely used protocol [34], which Adobe provides. Besides, the dynamic adaptive streaming over HTTP (DASH) accepts various media formats. Unlike the progressive stream, HLS, RTMP, and DASH allow clients to extract a single type of multiple video streams with better quality at a fixed bit rate; therefore, techniques to transfer the media information are essential. HLS clients acquire media information through a descriptor called the MP3 URL (M3U); this is a text format file, and M3U8 supports UTF-8. On the other hand, DASH uses the media presentation description (MPD), and MPD is built in XML format (also with UTF-8 support). MPD specification is defined by ISO/IEC 23009-1 [35]; therefore, DASH is also known as MPEG-DASH.

M3U8 and MPD are crucial techniques; precisely, after requesting a resource existing in the HTTP server, the HTTP server separates the resource into segments and writes the metadata as M3U8 and MPD. Clients can acquire the resource by appropriate HTTP-URLs. The most important part of the M3U8 and MPD is the MIME information, which tells clients how to process the file according to the type. This study used the Vidgear package to analyze MPD and acquire the necessary video stream from YouTube. Afterward, we use Python to detect the camera motion, address specific scenes, collect image samples, and construct a comprehensive hazy dataset for further study.

III. THE PROPOSED METHOD



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1. The situation related to the camera, scenes, weather, and sun conditions. (a). distant-view image captured at 7:00 with clear air; (b). distant-view image captured at 7:40 with clear air; (c). distant-view image captured under haze conditions at 7:00; (d) distant-view image captured under thick fog at 8:00; (e). sea-view image captured at 10:00 with clear air; (f). sea-view image captured in the morning with backlight; (h). sea-view image captured at 15:00 and covered with the color cast; (g). sea-view image captured at 11:00, covered with rainlets, and suffered from camera motions.

A. Problem Address and Our Goal

After some surveys and two months of field operation, we figured out that the difficulty of building a comprehensive hazy dataset based on OGP was the changeable natural environment and camera activities; among them, the natural environment issues include the following:

- Noises
- Hazy and Fog Effects
- Rain Effects
- Weather Changes
- Backlight

We illustrated some cases related to the abovementioned problems in Fig. 1. Figs. 1(a) and 1(b) demonstrate images captured under good weather at different times, and the color tones of the two images are different because of the sun's position. Figs 1(c) and 1(d) show images captured at similar times but are captured under haze and fog conditions, respectively. The gradient of the two images is relatively weak, and distant objects in the two images are unclear or missing. Moreover, the camera activities also caused some problems, including:

- Camera Motion
- Camera Zoom-in
- Len Rainlets
- Len condensation

Fig 1(e) illustrates a sea-view image captured under good weather, and the color tone is much closer to the canonical illumination. In the evening, the scattered sunlight seriously affected the sea view; therefore, the color tone is hugely biased, as illustrated in Fig. 1(f). On the other hand, Fig 1(g) is an example of the camera facing backlight problems in the morning; the backlight is too strong in some places, so the algorithm hardly works. Finally, we illustrated an image captured when the camera lens was covered with rainlets; this results in the distortion of objects, especially when the camera is zooming in or taking a long shot.

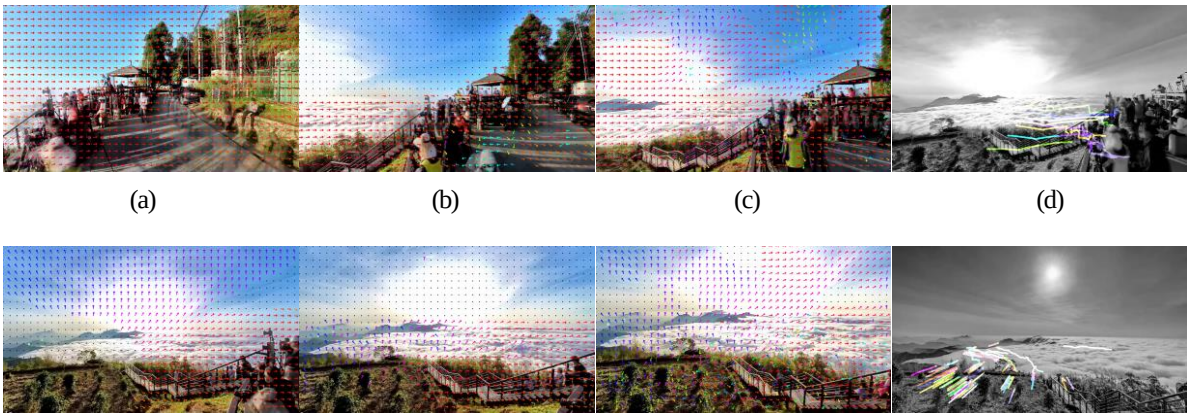
Natural environments and camera activities can hinder the estimation of optical flows; this is the first issue we must address. Besides, according to the documents officially given by YouTube [36], the recommended bitrate of the 1080p 30fps image is 10 Mbps. Empirically, video in this format can work smoothly with a bitrate from 5 to 10 Mbps; therefore, the optical flow algorithm should take no longer than 0.033 seconds to deal with two frames in the real-time video stream; however, the fast implementation might result in accuracy degradation. To balance the execution time and accuracy, we discard traditional benchmarks, trying to design a method that can fetch the maximal correct image (meaning the camera faces the same direction with a fixed focal length and angles).

Meanwhile, OGP comprises various scenes, including natural (sea, mountain, lake, …) and artificial (building, highway, bridge, …) scenes. The texture of distinct objects differs, meaning the keypoints extraction is essential. The problem is difficult to resolve by a single algorithm; therefore, we must test and integrate various algorithms to succeed in our task.

B. Detect Camera's Motion

Note that our goal is to get images captured at the same position with the same camera angles to succeed in image restoration training. Unlike conventional optical flow frameworks, we aim to detect camera activities like motion and zoom-in. We used the CamGear library in the VidGear package [37] to fetch the video stream from YouTube. To accelerate the process, we wrote a multithreading program to initialize a new thread to make the request and awaited the response from YouTube while the main thread dealt with the optical flow algorithms.

Camera motion is much easier to detect because all the keypoints should have the same optical flow at a short period except for those moving keypoints, as illustrated at the top in Fig. 2. Fig. 2(a)-(d) illustrates the rest ledge scene. At that time, the camera moved from the right to the left, resulting in objects' optical flows to the right, as demonstrated in red arrows. In this case, a simple consensus election voted by all keypoints can quickly reveal the camera's motion. First, we calculate the



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2. (a)(b)(c) images of a rest ledge scene captured with the camera's motion; the arrow indicates the orientation of the d-LKOF, GFOF, and RLOF; (d) image of the same scene; the color lines indicate the optical flow generated by st-LKOF, and the keypoints are less than those sparse to dense methods; (e)(f)(g) images of a mountain view captured with camera zoom-in; the arrow indicates the orientation of the d-LKOF, GFOF, and RLOF; (h) image of the same scene; the color lines indicates the optical flow generated by st-LKOF, and the optical flow are related to both camera zoom-in and cloud's movements.

magnitudes of all optical flow to collect a keypoint set composed of keypoints whose optical flow magnitudes exceed thresholding, as follows:

$$k = \{ n \mid \|\vec{v}_n\| > t_m \}, \tag{21}$$

where k is the keypoint set, $\vec{v}_n$ denotes the optical flow of keypoint n, and t_m is the thresholding associated with the magnitude of $\vec{v}_n$. After that, we calculate the orientation of $\vec{v}_k$ and denote the orientation as $\theta_{\vec{v}_k}$; after that, we form a histogram based on $\theta_{\vec{v}_k}$ to find the bin having the maximal number of keypoints, as follows:

$$arg\,max_c \; H(\theta_{\vec{v}_k}, c), \tag{22}$$

where H denotes the histogram returning the number of elements in the bin c. In this way, we find the consensus most extensive supported by V and ensure the camera's motion by giving lower-bound thresholding to $H(\theta_{\vec{v}_k}, c)$, and the lower-bound thresholding t_n is defined as follows:

$$t_n = 0.5 * Card(k). \tag{23}$$

We tested corresponding scenes with LKOF based on dense keypoints (d-LKOF) and Shi-Tomas corner detector (st-LKOF), figuring out that st-LKOF seemed weak to deal with the camera motion based on (22). Fig. 2(a)-2(c) illustrates d-LKOF results. The number of valid keypoints is sufficient to implement the consensus election which generates accurate results; on the other hand, st-LKOF generates fewer valid keypoints, making the result relatively unstable, as illustrated in Fig. 2(d). The color arrows in the images point out the direction related to $\theta_{\vec{v}_k}$. Meanwhile, we represent the orientation with the HSV Hue channel, and the definition is as follows:

$$H_{hsv} = 360° * \theta_{\vec{v}_k}/\pi. \tag{24}$$

C. Detect Camera Zoom-in

We looked into a more complicated case concerning camera zoom-in, as illustrated at the bottom of Fig 2. The vector field of $\vec{v}_k$ in Figs. 2(e)-2(g) distribute tidily, and all of $\vec{v}_k$ come from the same source located at the left-bottom of the scene; this leads to the fact that lines forming by $\vec{v}_k$ are concurrent, meaning that the lines share a common point. We denote the lines formed by $\vec{v}_k$ and the common point p as $L_{\vec{v}_k,p}$. Accordingly, $L_{\vec{v}_k,p}$ is defined as follows:

$$y_k - p_y = \frac{a_k}{b_k}(x_k - p_x), \tag{25}$$

where x_k and y_k respectively denote the coordinator of $L_{\vec{v}_k,p}$ concerning x and y axes; p_x and p_y denote the coordinator of the common point p; a_k and b_k denote the components of $\vec{v}_k$ concerning x and y axes, respectively. However, the formula represents the ideal concurrent system; we can check the optimal common points $\tilde{p}$ from lines formed by $\vec{v}_k$ (denoted as $L_{\vec{v}_k}$), and the generic formula of $L_{\vec{v}_k}$ is as follows:

$$b_k y_k - a_k x_k + c_k = 0, \tag{26}$$

where c_k denotes a constant. Since $\tilde{p}$ is not fully compatible to the ideal concurrent system, we can calculate the distance between $\tilde{p}$ and $L_{\vec{v}_k}$ to estimate errors, as follows:

$$d(\tilde{p}, L_{\vec{v}_k}) = \frac{|b_k \tilde{p}_y - a_k \tilde{p}_x + c_k|}{({a_k}^2 + {b_k}^2)^{1/2}}, \tag{27}$$

where $d()$ denotes a function returning the distance between two inputs. Therefore, we conclude that $\tilde{p}$ can be solved by minimizing an energy term defined as follows:

$$arg\,min_{\tilde{p}} \sum_k d(\tilde{p}, L_{\vec{v}_k}). \tag{28}$$

However, optimizing (28) directly is unrealistic because of the absolute value symbol; this makes the equation non-linear and cannot be solved based on quadratic optimization. To eliminate the effect of the absolute value symbol, we rewrite (28) as follows:

$$arg\,min_{\tilde{p}} \sum_k d(\tilde{p}, L_{\vec{v}_k})^2. \tag{29}$$

Therefore, we obtain the following:

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3. (a)(b)(c)(d) images of Eryanping Trail, Taiping Suspension Bridge, Sanxiantai, and Duoliang Station, respectively; (e)(f)(g)(h) images of unfavorable scene conditions, including extra moving objects (human), thick fog, rainlets, and night with artificial illluminator.

$$arg\,min_{\tilde{p}} \sum_{k} \frac{(-b_k \tilde{p}_y + a_k \tilde{p}_x - c_k)^2}{{a_k}^2 + {b_k}^2}. \quad (30)$$

Note that this means fitting a linear model concerning $\tilde{p}$ with the output c_k while c_k can be computed from $\vec{v}_k$ with a_k, b_k, and one of its endpoints. Unfortunately, finding optimal common points based on the regression remains unrealistic. Moving objects can affect some optical flows, making these optical flows irrelevant to camera zoom-in and becoming outliers. For example, the cloud moves because of the wind, as shown in Figs. 2(e)-2(g), and the corresponding optical flows have distinct orientations compared with those only affected by the camera's motion. Note the left-top region in Figs. 2(e)-2(g) exhibits many outliners; this could disable any regression estimator.

Concerning the above-mentioned issues, we use the random sample consensus (RANSAC) to estimate $\tilde{p}$. First, we choose several hypothetical inliers from keypoints in V and fit the hypothetical inliers based on (30). Second, we evaluate other keypoints with the fitted model to form the consensus set composed of keypoints whose errors are acceptable. Finally, if the number of the consensus set is enough, the model is accepted; otherwise, we restart from the first step. In this way, we can estimate the focal center of the camera zoom-in activity with $\tilde{p}$. The necessary number of iterations of RANSAC can be calculated with the empirical inlier probability and the expected accuracy, defined as follows:

$$k = \frac{log(1 - p_e)}{log(1 - w^N)}, \quad (31)$$

where k is the necessary number of iterations, p_e is the expected accuracy, w is the inlier probability, and N denotes the number of hypothetical inliers. To accelerate the execution, we down-sample the input image to 640*480 to locate the relative region of the focal center.

After some experiments, we found that st-LKOF works fine in detecting moving objects, especially for artificial objects and humans. St-LKOF was fast and accurate in detecting if the camera was moving; however, st-LKOF generated few optical flows and, as discussed previously, an image might contain optical flows associated with camera and object activities, making st-LKOF inaccurate in estimating the details of the camera's motion and zoom-in; in these cases, we suggest using d-LKOF instead. Empirically, the camera zoom-in resulted in many corresponding optical flows in cases based on d-LKOF, as illustrated in Figs. 2(a)-2(c). Therefore, w and N is respectively defined as 0.8 and 5 in this work, and the expected accuracy p_e is set as 0.8.

IV. EXPERIMENT AND ANALYSIS

The experiment was conducted with Python 3.9 and Ubuntu 22.04. We used NVIDIA GTX 3060 12G under an environment based on CUDA 11.8 and CUDNN 8.6.0 with a desktop computer based on Intel i-7 12700 and 32G RAM. In the experiments, we used three datasets of live video streams captured in Taiwan's famous scenic spots. The images of Shitiping and Dashshibi Hill in Hualien have been illustrated in Fig. 1. The images of Eryanping Trail and Taiping Suspension Bridge in Chiayi are provided by the Ministry of the Interior of Taiwan (MIoT) and are illustrated in Figs. 3(a) and 3(b), respectively. Sanxiantai and Duoliang Station are also famous scenic spots in Taitung; we respectively illustrated two images provided by MIoT in Figs. 3(c) and 3(d). Figs. 3(e)-3(h) illustrate examples of unfavorable scene situations; in this experiment, we classified these situations into rain, fog, haze, and bad illumination. The bad illumination included backlights, artificial illuminators, and color casts. We also introduced video streams captured on sunny days for comparison. *The ratio between sunny days and each unfavorable situation was 3:2.* Besides, all the tested video streams were captured under the camera's activities, including camera motion (CM) and zoom-in (CZ).

Note that each of the tested video streams exceeds 20 hours, but the duration of the camera activity loops is quite different and can be changed anytime. Calculating the number of correct images that could be captured in the tested video streams would be very difficult; therefore, *we focused on the ability to collect*

TABLE I
FULL-REFERENCE BENCHMARK RESULTS

Location	Dataset	d-LKOF	s-LKOF	st-LKOF	GFOF	RLOF	Average
Taitung – Sanxiantai, Duoliang Station.	Rain	42	38	31	22	16	29.8
	Fog and Haze	45	41	18	25	13	28.4
	Bad Illumination	28	29	15	28	15	23.0
	Sunny Day	114	121	101	78	58	94.4
Hualien - Shitiping, Dashshibi Hill.	Rain	37	38	32	18	13	27.6
	Fog and Haze	48	42	23	22	14	29.8
	Bad Illumination	32	35	21	27	9	24.8
	Sunny Day	135	141	97	74	46	98.6
Chiayi - Eryanping Trail, Taiping Suspension Bridge.	Rain	38	37	21	19	11	25.2
	Fog and Haze	41	42	17	18	13	26.2
	Bad Illumination	26	31	11	21	10	19.8
	Sunny Day	107	111	85	79	31	82.6
Overall Ability	At Correct Position	693	706	472	431	249	510.2
	CM detected	408	431	391	372	203	361.0
	CZ Center detected	297	282	97	65	48	157.8
	Mis-estimated	12	7	16	6	2	8.6

The numbers of each field are the number of images either captured at the correct positions or meet the goals we demand.

images when the camera stopped. We counted the "correct images" obtained during days with several artificial or natural conditions. More specifically, tested video streams included rain, fog, haze, and sunny situations recorded on different days. We counted the correct images generated by tested methods to determine which were better to defend against unfavorable situations. Note also that we empirically set a time limitation of 45 seconds between two shots to prevent capturing multiple images during camera stops. The tested algorithm included d-LKOF, st-LKOF, GFOF, sparse Lucas-Kanade optical flow (s-LKOF), and robust local optical flow (RLOF) [38]. s-LKOF is a sparse version of d-LKOF aiming at fast implementation, and RLOF focuses on tracing long-range optical flows based on local features.

All video streams associated with different natural environments were processed in the first step using optical flow algorithms. We calculated the number of images captured at the correct positions (when the camera stopped); this was performed according to (23). TABLE I demonstrates the experimental results. Accordingly, unfavorable environments degrade the performance of the tested optical algorithms, resulting in a decrease in correct images compared with situations on sunny days. Among tested unfavorable situations, inadequate illumination is the most critical obstacle when collecting our dataset. GFOF and RLOF are relatively weak in helping collect images at fixed positions for the tested optical flow algorithms. According to the results, GFOF and RLOF detect optical flows in a higher resolution, as illustrated in Figs. 2(b), 2(c), 2(f), and 2(g). For example, note the grass regions at the left-bottom in Fig. 2(g); optical flows in the regions correctly reflect the movement of planets; however, this hinders the accuracy in estimating the camera zoom-in center because they are irrelevant to the camera zoom-in. d-LKOF and s-LKOF perform with the best accuracy in defending unfavorable situations with a slightly higher computational cost; fortunately, this is acceptable because real-time processes are unnecessary when collecting image datasets. Last, the computational time of st-LKOF is speedy, making this algorithm capable of real-time processing; meanwhile, the performance is also satisfactory. Overall, when our method collaborates with d-LKOF, it produces correct images with only 21% losses in unfavorable situations compared with sunny days; this is a satisfactory ratio.

We also evaluated the ability to estimate the camera's motion. At the bottom of TABLE I, we demonstrated the number of images captured at the correct positions. Note that the correct images were captured when the camera stopped from its motion or zoom-in. We compiled statistics to analyze camera activities before capturing these images. According to the results, our method is accurate; it can detect suitable camera activities in most cases because only a few instances were wrongly detected as the camera's motion and zoom-in simultaneously.

V. CONCLUSIONS

Data collection is an essential task of machine learning and artificial intelligence; however, this faces difficulty owing to natural or artificial issues, and fewer lectures focus on this. In this study, we proposed a framework to collect images at a fixed position; this is a beneficial technique for collecting image restoration datasets in which the ground truth and samples must be captured at fixed positions in different situations. In our framework, the orientation and focal center of the camera's motion and zoom-in can be accurately estimated; therefore, users can collect image samples with any online video streams at an efficient level.

Our framework's worst case is collecting samples on rainy days, especially when the camera lens is full of rainlets, as demonstrated in Fig. 3(g). We figured out that the rainlet seriously interferes with our algorithm in long-shot situations (the camera uses a long focal to capture images). The main reason is that the rainlet and a long focal physically result in image blur, and the point spread function is hardly estimated. Therefore, we are currently focusing on analyzing the blueness pattern of the input image to improve accuracy on rainy days.

REFERENCES

[1] P. Narayanan et al., "A Multi-Purpose Realistic Haze Benchmark With Quantifiable Haze Levels and Ground Truth," in IEEE Transactions on Image Processing, vol. 32, pp. 3481-3492, 2023, doi: 10.1109/TIP.2023.3245994.

[2] H. Dong et al., "Multi-Scale Boosted Dehazing Network With Dense Feature Fusion," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 2154-2164, doi: 10.1109/CVPR42600.2020.00223.

[3] X. Liu, H. Li and C. Zhu, "Joint Contrast Enhancement and Exposure Fusion for Real-World Image Dehazing," in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2021.3110483.

[4] P. L. Suárez, D. Carpio, A. D. Sappa and H. O. Velesaca, "Transformer based Image Dehazing," 2022 16th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Dijon, France, 2022, pp. 148-154, doi: 10.1109/SITIS57111.2022.00037.

[5] S. C. Agrawal and A. S. Jalal, "Dense Haze Removal by Nonlinear Transformation," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 2, pp. 593-607, Feb. 2022, doi: 10.1109/TCSVT.2021.3068625.

[6] C. O. Ancuti, C. Ancuti, R. Timofte and C. D. Vleeschouwer, "I-HAZE: a dehazing benchmark with real hazy and haze-free indoor images," arXiv:1804.05091v1, 2018

[7] Ancuti, C.O., Ancuti, C., Timofte, R., & Vleeschouwer, C.D. (2018). O-HAZE: A Dehazing Benchmark with Real Hazy and Haze-Free Outdoor Images. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 867-8678.

[8] B. Li, W. Ren, D. Fu, D. Tao, D. Feng, W. Zeng, and Z. Wang, "Benchmarking single-image dehazing and beyond," in IEEE Transactions on Image Processing, vol. 28, no. 1, pp. 492-505, Jan. 2019.

[9] L. Zhang, A. Zhu, S. Zhao and Y. Zhou, "Simulation of Atmospheric Visibility Impairment," in IEEE Transactions on Image Processing, vol. 30, pp. 8713-8726, 2021, doi: 10.1109/TIP.2021.3120044.

[10] Y. Zhou, Y. Han and P. Zhou, "Rain removal in videos based on optical flow and hybrid properties constraint," 2015 Seventh International Conference on Advanced Computational Intelligence (ICACI), Wuyi, China, 2015, pp. 143-147, doi: 10.1109/ICACI.2015.7184765.

[11] https://data.gov.tw/

[12] https://monitor.wfuapp.com/

[13] https://www.gov.tw/taiwan/

[14] https://ocam.live/index.php?route=product/category&path=122

[15] https://airtw.epa.gov.tw/cht/EnvMonitoring/Central/SitePhoto.aspx

[16] Y. Liu and X. Granier, "Online Tracking of Outdoor Lighting Variations for Augmented Reality with Moving Cameras," in IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 4, pp. 573-580, April 2012, doi: 10.1109/TVCG.2012.53.

[17] Y. Wu, X. He and T. Q. Nguyen, "Moving Object Detection With a Freely Moving Camera via Background Motion Subtraction," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 27, no. 2, pp. 236-248, Feb. 2017, doi: 10.1109/TCSVT.2015.2493499.

[18] J. Chen, K. Zhang, B. Jia and Y. Gao, "Identification of a Moving Object's Velocity and Range With a Static-Moving Camera System," in IEEE Transactions on Automatic Control, vol. 63, no. 7, pp. 2168-2175, July 2018, doi: 10.1109/TAC.2017.2755988.

[19] A. López-Cifuentes, M. Escudero-Viñolo and J. Bescós, "Automatic Semantic Parsing of the Ground Plane in Scenarios Recorded With Multiple Moving Cameras," in IEEE Signal Processing Letters, vol. 25, no. 10, pp. 1495-1499, Oct. 2018, doi: 10.1109/LSP.2018.2865833.

[20] J.J. Gibson, "The Perception of the Visual World. Houghton Mifflin, " 1950 .

[21] K.P. Horn and B. Sunck, "Determining optical flow," Artificial Intelligence, 1981, doi:10.1016/0004-3702(81)90024-2. hdl:1721.1/6337.

[22] D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.

[23] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Proceedings of Imaging Understanding Workshop, 1981, pages 121-130

[24] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features, " Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[25] J. Shi and C. Tomasi, "Good Features to Track, " IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.

[26] B.K.P. Horn and B.G. Schunck, "Determining optical flow." Artificial Intelligence, vol 17, pp 185–203, 1981.

[27] G. Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion, " J. Lect. Notes Comput. Sci, 363–370, 2003, https:// doi. org/ 10. 1007/3- 540-45103-X_ 50.

[28] Y. Wang, "Joint Random Field Model for All-Weather Moving Vehicle Detection," in IEEE Transactions on Image Processing, vol. 19, no. 9, pp. 2491-2501, Sept. 2010, doi: 10.1109/TIP.2010.2048970.

[29] A. Singha and M. K. Bhowmik, "Salient Features for Moving Object Detection in Adverse Weather Conditions During Night Time," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 10, pp. 3317-3331, Oct. 2020, doi: 10.1109/TCSVT.2019.2926164.

[30] R. Li, R. Tan, L. F. Cheong, A. Aviles-Rivero, Q. Fan and C. Schoenlieb, "RainFlow: Optical Flow Under Rain Streaks and Rain Veiling Effect," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 7303-7312, doi: 10.1109/ICCV.2019.00740.

[31] J. Shi and C. Tomasi, "Good Features to Track, " TR93-1399 Cornell University, 1993.
[32] C. Harris and M. Stephens, "A Combined Corner and Edge Detector, " Alvey Vision Conference, 15, 1988.
[33] R Pantos; W. May, "Playlists". HTTP Live Streaming, IETF. p. 9. sec. 4. doi:10.17487/RFC8216. ISSN 2070-1721. RFC 8216. Retrieved Jan 15, 2020.
[34] https://helpx.adobe.com/adobe-media-server/dev/stream-live-media-rtmp.html
[35] https://www.iso.org/standard/83314.html
[36] https://support.google.com/youtube/answer/2853702?hl=en
[37] https://abhitronix.github.io/vidgear/v0.3.2-stable/
[38] J. Geistert, T. Senst and T. Sikora, "Robust local optical flow: Dense motion vector field interpolation," 2016 Picture Coding Symposium (PCS), Nuremberg, Germany, 2016, pp. 1-5, doi: 10.1109/PCS.2016.7906352.

Ping Juei Liu[1*] received a Ph.D. degree in Computer Science and Information Engineering from the National Taiwan University of Science and Technology. He is an Assistant Professor in the Department of Artificial Intelligence and Computer Engineering at the National Chin-Yi University of Technology in Taiwan. His research interests include computational photography, image processing, and machine learning.

Yu-Cheng Wang[2] is looking forward to a Master's degree in Computer Science and Information Engineering at the National Chin-Yi University of Technology in Taiwan. His research interests include image processing and machine learning.