ISSN 1870-4069

# Un método para simular lluvia mediante Motion Blur y ruido sal y pimienta

Roberto-Carlos Romero-Olmos [1], Sebastián Salazar-Colores[2], Hugo Moreno-Jiménez[2], Gerardo Flores[2]

[1] Instituto Tecnológico de México campus León, Departamento de Ingeniería Eléctrica, México

[2] Centro de Investigaciones en Óptica, Guanajuato, México

robertocarlosromeroolmos@gmail.com

**Resumen.** La eliminación de los efectos climáticos adversos en imágenes de exteriores tales como neblina, nieve o lluvia es un tema de profundo interés dado el número de posibles aplicaciones en áreas como la conducción asistida y autónoma de autos y drones, teledetección y vigilancia, entre otras. En el desarrollo de métodos heurísticos y de Inteligencia Artificial son empleados datos simulados para diseñar y entrenar sus modelos, por lo que la calidad de la simulación repercute en el desempeño final de los métodos desarrollados. Mientras existen muchos trabajos de simulación de neblina, la simulación de lluvia y nieve han sido poco estudiadas. Por lo que, en este trabajo se propone un método para la generación de lluvia sintética en imágenes, con el fin de crear bases de datos realistas en un extenso número de escenarios, con el cual se podrán entrenar diferentes algoritmos de eliminación de lluvia (deraining) basados en Deep Learning. En los resultados mostrados se observa un mejor desempeño cualitativo en términos de realismo en relación con otros métodos similares, al incluir velos atmosféricos generados por la acumulación de trazas de lluvia, así como diferentes formas y direcciones de las trazas de lluvia.

**Palabras clave:** Simulación eventos climáticos, lluvia, deraining.

# A Method for Simulation of Rain Using Motion Blur and Salt and Pepper Noise

**Abstract.** The elimination of adverse weather effects in outdoor images such as haze, snow or rain is a very interesting topic given the number of possible applications in areas such as assisted and autonomous driving of cars and drones, remote sensing and surveillance, among others. In the development of heuristic and Artificial Intelligence methods, simulated data are used to design and train their models, so the quality of the simulation has an impact on the final performance of the developed methods. There are many works on fog simulation,

but rain and snow simulation have been poorly studied. Therefore, in this work we propose a method for the generation of synthetic rain in images, in order to create realistic databases in a wide number of scenarios, with which different rain elimination algorithms (deraining) based on Deep Learning can be trained. The results presented, show a better qualitative performance in terms of realism in relation to other similar methods, by including atmospheric veils generated by the accumulation of rainfall traces as well as different shapes and directions of the rainfall traces.

**Keywords:** Weather events simulation, rain, deraining.

## 1. Introducción

La simulación de fenómenos naturales como la lluvia, nieve o neblina, es un problema complejo y de gran interés, debido a las aplicaciones que estas pueden tener tanto en la industria, fotografía, juegos, exploración, procesamiento de imágenes, entre otras áreas. Actualmente, la simulación realista de lluvia ha sido poco explorada a pesar del impacto que podría tener para la generación de bases de datos extensas y realistas que impulsen a los algoritmos de Deep Learning para eliminación de lluvia (deraining), estos métodos se verán beneficiados al ser entrenados con grandes bases de datos, que contemplen diferentes escenarios y características, debido a que estas propiedades harán más sencillo que la red aprenda a generalizar el problema.

La lluvia es uno de los fenómenos meteorológicos más comunes, y está perjudica en gran medida la eficiencia de muchos algoritmos de visión, ya que disminuye la visibilidad de la imagen e introduce interferencias indeseables que pueden afectar gravemente el rendimiento de los algoritmos de visión por computador [1, 2, 3, 4]. Para abordar este problema se han propuesto varios métodos para la eliminación de los efectos de la lluvia en imágenes, entre los cuales se encuentran los algoritmos de aprendizaje automático basados en redes neuronales, como ya se mencionó estos métodos mejoran su desempeño al utilizar una base de datos extensa, ya que la precisión del modelo depende en gran medida de la cantidad de datos utilizados y la generalización del problema en diferentes escenarios, para el entrenamiento de la red.

Los métodos de deraining en imágenes singulares se pueden dividir en dos enfoques básicos: basados en modelos físicos (sin aprendizaje profundo) y basados en datos (aprendizaje profundo) [9]:

Los métodos basados en modelos emplean marcos de optimización para el deraining, En general estos métodos solo se ocupan de las trazas de lluvia e ignoran la presencia de acumulación de lluvia, el deraining se realiza mediante la descomposición de la imagen mediante un análisis de componentes morfológicos como en [10, 11, 12].

Los métodos basados en Deep Learning para deraining, emplean redes conjuntas que se encargan de la detección y eliminación de lluvia, estos métodos pueden soportar lluvias intensas, trazas de gotas superpuestas y acumulación de lluvia. En general estas redes pueden detectar las ubicaciones de lluvia al predecir una máscara de lluvia binaria y tomar un marco recurrente para eliminar las trazas de las gotas y aclarar progresivamente la acumulación de lluvia.

**Fig. 1.** Aplicación de lluvia sintética con Photoshop.

Este método logra buenos resultados en casos de lluvia intensa. Sin embargo, podría eliminar texturas verticales falsamente y generar subexposición [6,13,14]. Estos métodos se ven beneficiados al ser entrenados con bases de datos extensas y que contemplen diferentes escenarios, ya que esto permite generalizar el problema.

Actualmente, el método utilizado para la generación de lluvia en imágenes se realiza en gran medida con herramientas de Photoshop, este se basa en añadir una capa blanca y rellenarla con ruido uniforme monocromático, a este se le aplica un filtro de blur gaussiano, tras lo cual se usan niveles para reducir la cantidad de ruido, y aumentar el contraste en el ruido aplicado, al resultado de este proceso se le aplica un MotionBlur para ajustar la dirección en la que la lluvia caerá, al tener esto se reajustan los niveles de blancos y negros según sea necesario, finalmente se utiliza esta capa como “Pantalla” sobre la imagen original obteniendo una imagen con lluvia sintética, también se puede usar el "modo de fusión de pantalla" no lineal que incluye Photoshop en sus librerías. Este método es empleado en la creación de bases de datos de lluvia sintética con variación de la cantidad de lluvia [5].

El método actualmente utilizado, para generar bases de datos sintéticas, no es del todo realista y carece de efectos propios de la lluvia, ver Fig.1, este método claramente genera una superposición de líneas en la imagen original, además si el usuario quisiera tener trazas de lluvia en distintos ángulos tendría que realizar nuevamente el proceso debido a que este solo aplica líneas en un ángulo definido y con un tamaño promedio establecido.

En este trabajo se muestra un primer acercamiento y resultados iniciales al aplicar capas de trazado de lluvia con un método novedoso. El algoritmo propuesto se encarga de crear múltiples capas de lluvia e integrarlas con la imagen a la cual se aplique.

El resto del presente trabajo está distribuido de la siguiente forma. En la sección 2 se muestra la metodología empleada para realizar el proceso propuesto.

En la sección 3 y 4 se expone el uso del proceso y los resultados obtenidos. Por último, en la sección 5 se presentan las conclusiones y perspectivas de investigaciones futuras. El código fuente de este método se encuentra disponible[1] .

[1] https://github.com/RobbRom1206/Rain-Layer

**a)** **b)**

**Fig. 2.** Comparativa entre el método con Photoshop (a) y el método propuesto (b), al simular lluvia moderada.

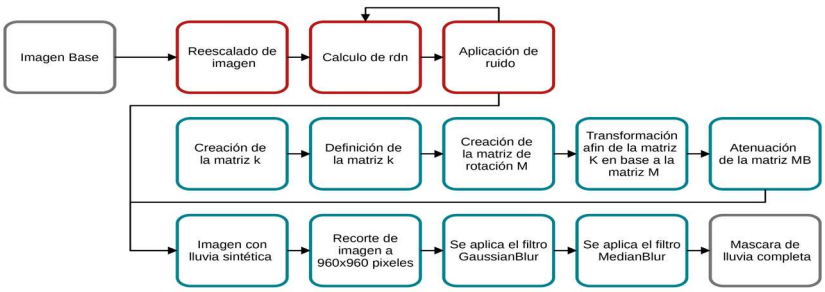



**Fig. 3.** Proceso de creación de máscara.

## 2. Método propuesto

Capturar las características de un fenómeno tan complejo como la lluvia es difícil y los métodos existentes generan imágenes que no lucen naturales, a continuación, se describe un método de creación de imágenes con lluvia sintética de manera automática. Este método está programado en Python, se encarga del procesamiento de imágenes para aplicar a estas una máscara de lluvia con la cual se podrán crear bases de datos que posteriormente se podrían usar en el entrenamiento de redes neuronales para la eliminación de los efectos de la lluvia (Deraining), ver Fig. 2.

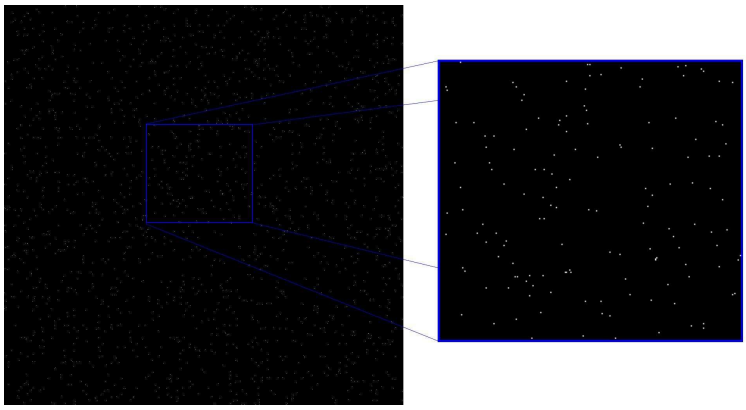


**Fig. 4.** Ruido de sal para la máscara de lluvia.

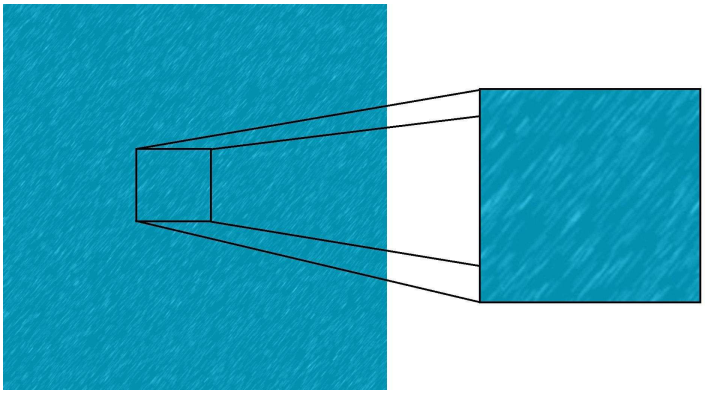


**Fig. 5.** MotionBlur aplicado a la imagen con ruido de sal (se cambió el fondo a color azul para una mejor muestra del efecto).

El algoritmo propuesto usa como base la aplicación de ruido de sal y pimienta, así como la aplicación de un efecto Motion Blur para simular el movimiento en una dirección de las gotas de lluvia, con ello se generaron capas (máscaras) con lluvia sintética, las cuales se combinarán con la imagen original, estableciendo con una menor intensidad el peso de la capa de lluvia, esto aplicará un efecto parecido a la transparencia en las trazas de lluvia.

Para ello se tuvieron en cuenta varias fases las cuales serán detalladas a continuación:

1 A partir de una imagen base (img) se crea la imagen lluvia y se genera ruido blanco para establecer la cantidad de gotas de lluvia que habrá en esta.

   a. Escalar "img" a 1024 x 1024 píxeles.

   b. Establecer el umbral " $u$ ", valor dado por el usuario donde $\{u \in \mathbb{R}: 0 < u < 1\}$.

   c. Calcular un número aleatorio (" $rdn$ ") pixel a pixel, donde $\{rdn \in \mathbb{R}: 0 < rdn < 1\}$.

**Fig. 6.** Aplicación de máscara con el método propuesto.

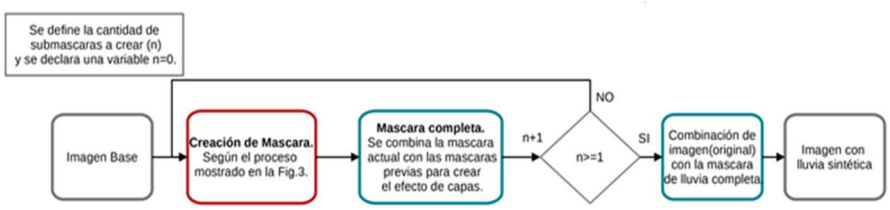



**Fig. 7.** Proceso de aplicación de máscara de lluvia.

d. Aplicar ruido de tipo sal (este se caracteriza principalmente por cubrir de forma dispersa toda la imagen con una serie de píxeles blancos), se genera una imagen "lluvia" llena de ceros con las mismas dimensiones de img[i,j], y realiza un barrido de la misma, se compara $rdn$ y $u$ pixel a pixel, si $rdn < u$ el valor del pixel se establece como lluvia[i,j]=255.

2 Aplicar *motion blur (*el cual se define como: "El rastro dejado por los objetos en movimiento en una fotografía o en una secuencia de imágenes" en este caso se aplica para simular el movimiento de la lluvia sintética) a la imagen "lluvia" con ruido de sal, y finalmente, se guarda este modelo como máscara de Lluvia.

a. Creación de una matriz *k* de ceros con el tamaño que el usuario estipulo "*size*" para la gota.

b. Se introducen unos en la fila central, -1 para asignar el centro de rotación de la imagen.

c. Se obtiene la matriz de rotación *M* a partir del tamaño de gota "*size*", el ángulo y su escala que en este caso será 1.

d. Aplicamos una transformación afín a la matriz *k* con base en la matriz de rotación M y el tamaño de "size" dándonos como resultado la matriz MB.

e. Se atenúa el filtro de MotionBlur al hacer una división entre la matriz MB y 1 sobre la suma de los valores de esta.

f. Se aplica el filtro MotionBlur como un filtro 2D a la imagen previamente modificada con ruido de sal "lluvia", estipulando que esta tendrá la misma profundidad que la imagen original.

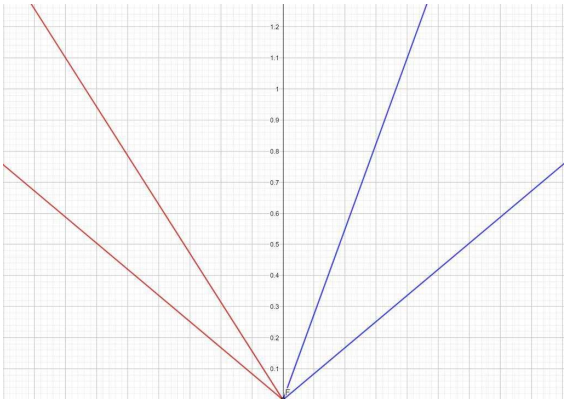
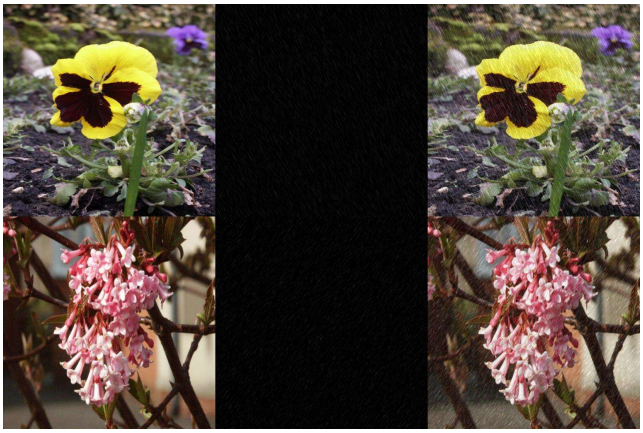


**Fig. 8.** Apertura del ángulo de selección para aplicación de MotionBlur.



a) b) c)

**Fig. 9.** Muestra de imagen. (a) GT(Ground-Truth) b) Máscara de lluvia c) Máscara de lluvia aplicada.

g. Se recorta un área de la máscara para usarse como base (con un tamaño total de 960 x 960 píxeles).

h. Se aplica un filtro GaussianBlur para eliminar un poco el ruido Gaussiano de la imagen.

i. Se aplica después un filtro MedianBlur para realizar un difuminado de imagen y conseguir un efecto más realista.

## 3. Funcionamiento

En esta sección se muestra el funcionamiento del método, como primer paso se aplicará ruido de sal (puntos blancos en una imagen de fondo negro) para definir la cantidad de gotas de lluvia que nuestra mascará tendrá al final del proceso, ver Fig. 4.

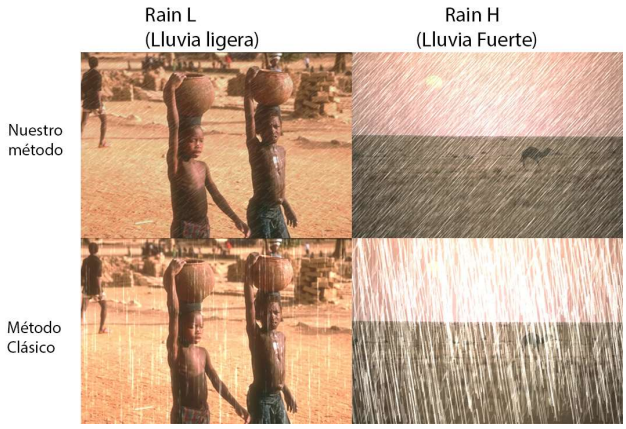



**Fig. 10.** Comparativa entre imágenes creadas con nuestro método y creadas con métodos clásicos, en los 2 tipos de imágenes más usadas en bases de datos de Deraining.

La cantidad de gotas depende del umbral definido por el usuario, con base en el cual se establece la probabilidad de que un píxel de esta se convierta en los puntos blancos antes vistos. Después de ello se aplica un filtro motion blur con el cual se expanden los puntos blancos anteriores hacia un ángulo definido dándonos como resultado la máscara de lluvia que usaremos para la imagen final Fig. 5.

Después de esto se hace una adición de la máscara a la imagen original, realizando una suma pixel a pixel y limitando los valores a 255, para dar el efecto de falsa lluvia, ver Fig. 6, esto se puede realizar múltiples veces con ángulos ligeramente diferentes para crear imágenes más realistas.

## 4. Uso

Para llevar a cabo este trabajo se realizó un programa que hace uso de este nuevo método, teniendo como input algunos parámetros que son necesarios, tamaño de la gota (size), umbral (u). ángulo (A), peso de la capa de lluvia (Beta), imagen original (img) y número de capas, este método de manera simple crea una máscara de lluvia con un solo tamaño de raya/gota, a continuación, se muestra un ejemplo el cual se encarga de realizar 3 máscaras de lluvia y las combina en una sola para una emulación más realista y con mejor efecto práctico.

El proceso mostrado anteriormente está diseñado para unir 3 capas de máscara de lluvia cada una con un valor de tamaño diferente, pero respetando el ángulo de estas el cual toma un valor de 50° a 70° y de 110° a 140° con respecto a la horizontal esto para dar una simulación de que las gotas caen de la parte superior de la imagen dándonos como base la siguiente apertura de ángulos para la caída de lluvia:

Ahora utilizando este algoritmo, se tomarán imágenes de pruebas de las cuales usaremos como base las imágenes aleatorias del apartado Ground-Truth del dataset Rain1400 [8] para tener un repertorio inicial de imágenes con las cuales trabajar y posteriormente comparar con su base de imágenes de lluvia sintéticas.

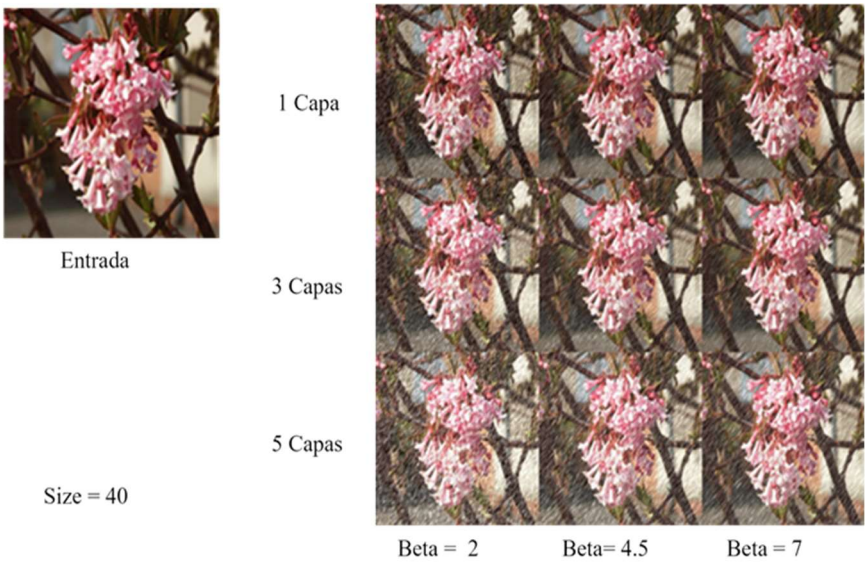



**Fig. 11.** imágenes modificadas por el método propuesto. En este ejemplo se muestran cambios de Beta que se refiere al peso que se usará en la capa de lluvia al unir las imágenes, Size (Tamaño en el cual se estira el ruido de sal aplicado), y una variación de capas lluvia aplicadas en el proceso.

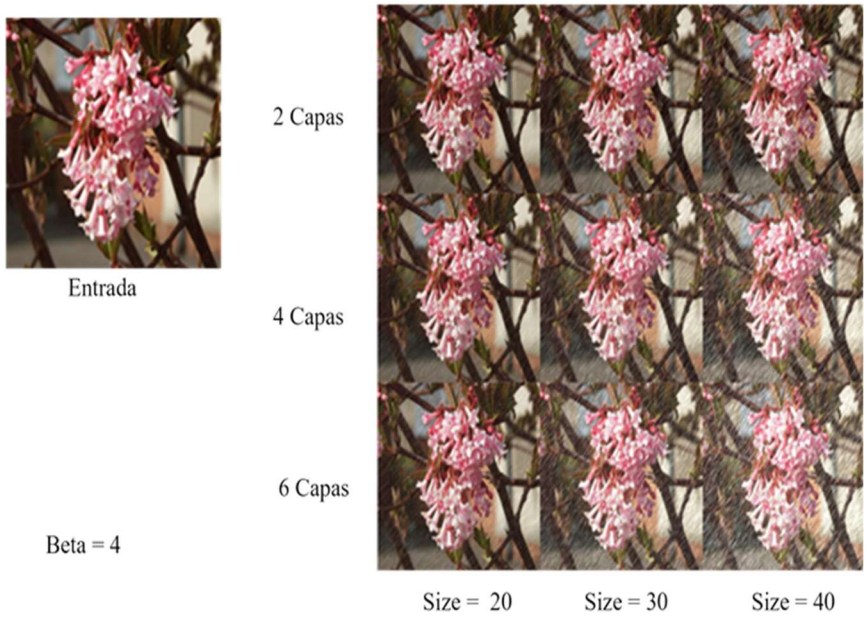



**Fig. 12.** imágenes modificadas por el método propuesto. En este ejemplo se muestran cambios de Beta que se refiere al peso que se usará en la capa de lluvia al unir las imágenes, Size (Tamaño en el cual se estira el ruido de sal aplicado), y una variación de capas lluvia aplicadas en el proceso.

El programa mostrado con anterioridad solo trabajará una imagen por ejecución debido a que fue realizado con el propósito de realizar pruebas, a continuación, se muestran algunas aplicaciones de la máscara a imágenes aleatorias.

Con estas imágenes podemos observar el potencial de la aplicación de un software de este estilo, el cual tiene un tiempo de procesamiento de alrededor de 2 segundos por imagen para la creación de datasets específicos para Deraining, también debe ser tomado en cuenta el ahorro de esfuerzo y atención que el usuario necesita dar a la creación de estos en comparación a otros métodos los cuales necesitan que un usuario esté aplicando las máscaras el mismo una a una.

## 5. Resultados

Durante el desarrollo de este programa se tomaron en cuenta 3 cosas, la imagen Ground-Truth a la cual se le aplicará una máscara de lluvia, la máscara de lluvia y la imagen resultante, En la figura 9 se muestra la transición de estas 3 fases realizadas durante el procesamiento.

Para terminar de comprobar la calidad de este proceso se comparará la aplicación de la máscara de lluvia contra el procesamiento de imágenes realizado con métodos anteriores basados en Photoshop para la aplicación de diferentes tipos de lluvia, que son usados en bases de datos actuales.

A continuación, se muestran algunos resultados variando los parámetros del método desarrollado Fig. 11 y 12.

## 6. Conclusiones

En este trabajo se propuso un algoritmo de creación de lluvia sintética para la asistencia en el proceso de producción de bases de datos y simulación de lluvia en imágenes singulares. Obteniendo satisfactorios resultados de este primer acercamiento, lo que muestra la factibilidad del proceso propuesto, ya que el algoritmo es capaz de trabajar y elegir un tamaño y dirección de manera semiautomática y se reduce el tiempo empleado en la creación de estas máscaras.

Hemos observado que en términos generales la emulación de lluvia a través de este método se aproxima más a las características de lluvia real, sin embargo actualmente no se ha realizado un experimento cualitativo extenso sobre la percepción de las personas ante estas imágenes sintéticas, lo cual se trabajará en un futuro, a la vez que se refina el procedimiento y se implementaran algunas mejoras debido a que hemos detectado que las imágenes generadas presentan ciertas incongruencias al trabajar con imágenes con variados puntos de profundidad, por lo que como trabajo futuro esto será mejorado agregando un sistema de identificación de profundidad, también se implementará neblina en las imágenes, así como mejorar la selección de transparencia y tamaño de gota en la imagen, se contemplaran los efectos de la lluvia en diferentes entornos como lo son las marcas generadas en suelo seco o las ondulaciones en espejos de agua. Una vez conseguido lo anterior, se comenzará a realizar pruebas en imágenes más variadas, para un trabajo próximo de deraining en redes neuronales.

## Referencias


1. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
2. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C.: SSD: Single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, Springer, Cham, vol. 9905 (2016) doi: 10.1007/978-3-319-46448-0_2
3. Ren, S., He, K., Girshick, R., Sun. J.: Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, vol. 28, pp. 91–99 (2015)
4. Chen, Y., Li, W., Sakaridis, C., Dai, D., Van-Gool, L.: Domain adaptive faster r-cnn for object detection the wild. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3339–3348 (2018)
5. Zhang, H.,Patel, V. I.: Density-aware single image de-raining using a multi-stream dense net-work. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 695–704 (2018)
6. Zhang, H., Sindagi, V., Patel, V. M.: Image de-raining using a conditional generative adversarial network. IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 3943–3956 (2020) doi: 10.1109/TCS VT.2019.2920407
7. Zhu, J. Y., Park, T., Isola, P., Efros, A. E.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision (ICCV), pp. 2223–2232 (2017)
8. Ren, D., Zuo, W., Hu, Q., Zhu, P., Meng, D.: Progressive image deraining networks: A better and simpler baseline. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3937–3946 (2019)
9. Yang, W., Tan, R., Wang, S., Fang, Y., Liu, J.: Single image deraining: From model-based to data-driven and beyond. ransactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 11, pp. 4059–4077 (2020). doi: 10.1109/TPA MI.2020.2995190
10. Kang, L. W., Lin, C. W., Fu, Y. H.: Automatic single image-based rain streaks removal via image decomposition. IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 1742–1755 (2012) doi: 10.1109/TIP.2011.2179057
11. Luo, Y., Xu, Y., Ji, H.: Removing rain from a single image via discriminative sparse coding. In: Proc. IEEE Int'l Conf. Computer Vision, 2015, pp. 3397–3405
12. L. Zhu, C. Fu, D. Lischinski, and P. Heng.: Joint bilayer optimization for single-image rain streak removal. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2545–2553 (2017)
13. Qian, R., Tan, R. T., Yang, W., Su, J., Liu, J. Attentive generative adversarial network for raindrop removal from a single image. In. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2482–2491 (2018)
14. Li, R., Cheong, L. F., Tan, R. T.: Heavy rain image restoration: Integrating physics model and conditional adversarial learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1633–1642 (2019)