1. Synthesis

1.1. Objective:

In this process, we aim to improve the quality of the frame(s) of the given video through the known methods, such as gamma correction, deblurring using filters, reducing the noise and magnification of the point of interest. The final goal is to check if

- we can read the license plate of the truck in the video, completely or partially, and
- improve the processed frame(s) quality by denoising, deblurring and adjusting the lighting of the scenery.

1.2. Summary of the activities:

The procedure that was followed is as the list below:

- Extracting the frames of the video
- Denoising, gamma correction, and deblurring the frame(s)
- License plate extraction and magnification

1.3. Conclusions:

The process that we have carried on made the frame quality higher and as a result, the license plate more readable. In other words, we can state:

- The frame(s) quality can be noticeably improved
- The green truck's license plate's letters and numbers couldn't be clearly read from a single frame so we had to process multiple frames to conclude the plate's information with a higher confidence

2. Relation

2.1. Received material

File: video1.mov

Format:	MOV
Resolution:	1024x768
Duration:	00:00:08
Frame rate:	30.00 fps
MD5:	f34f9f86238f7d916bf6d8c95c533aff
SHA-1:	e87d43261b1b448a0eb13b908f10e76a8d70039d

2.2. Adopted software and hardware

During the process, we used the sources listed below:

- Laptop Lenovo Legion 5i (2021, 15.6") – Used Google colab service on the device (https://colab.research.google.com/)
- "certutil -hashfile Example.txt MD5" and "certutil -hashfile Example.txt SHA1" commands in the terminal to acquire the MD5 and SHA-1 values of our file (Windows 11 Pro, build 22621.1778)
- Python version 3.10.12 (https://www.python.org/downloads/release/python-31012/)
- Library opencv-python version 4.7.0 (https://opencv.org/blog/2022/12/29/opencv-4-7-0/)
- Library numpy (https://numpy.org/)
- Library matplotlib (https://matplotlib.org/)
- Script *DF_Final.ipynb,* created to analyze the image (in the notebook format to see the output without re-running the code)

2.3. Handed-in material

- /Legal Analysis: *The folder containing all the materials handed in*
- /Legal Analysis /Original_Frames: *The folder containing the frames that were used in the process*
- /Legal Analysis /Original_Frames/frame_180.jpg
- /Legal Analysis /Original_Frames/frame_185.jpg
- /Legal Analysis /Original_Frames/frame_190.jpg
- /Legal Analysis /Processed_Frames: *The folder containing the images that were produced in the process*
- /Legal Analysis /Processed_Frames/frame_180_deblurred.jpg
- /Legal Analysis /Processed_Frames/frame_180_denoised.jpg
- /Legal Analysis /Processed_Frames/frame_180_gamma_corrected.jpg
- /Legal Analysis /Processed_Frames/frame_185_deblurred.jpg
- /Legal Analysis /Processed_Frames/frame_185_denoised.jpg
- /Legal Analysis /Processed_Frames/frame_185_gamma_corrected.jpg
- /Legal Analysis /Processed_Frames/frame_190_deblurred.jpg
- /Legal Analysis /Processed_Frames/frame_190_denoised.jpg
- /Legal Analysis /Processed_Frames/frame_190_gamma_corrected.jpg
- /Legal Analysis/md5-files.txt *The text file containing the MD5 hash values of all the files in the main folder*
- /Legal Analysis/sha1-files.txt *The text file containing the SHA-1 hash values of all the files in the main folder*

- /Legal Analysis/relation_copy.txt *The text file containing a copy of the relations section*
- /Legal Analysis/video1.mov *The original video given*
- /Legal Analysis/Final_Report.pdf *The final report file*
- /Legal Analysis/DF_Final.ipynb *The notebook file containing the main script used to edit and enhance the frames, and also the function used to extract the frames of the video file*

2.4. Appendixes

The relation is made of 2 pages, it is linked to a digital archive and includes a final appendix.

2.5. Reproducibility

The analysis follows the standard for handling digital evidences ISO/IEC 27037 and all the results can be replicated using the mentioned software.

3. Acquisition process

We know that videos are a sequence of frames put together, so in order to read the license plate or any other information in a video file we need to process the frame(s) of that file. To do so, we first need to extract the frames properly. To achieve that, I wrote a script (included in the *DF_Final.ipynb),* that extracts all the frames of the video and saves them in the given directory.

After we have access to all the frames, we can start choosing which ones to process. One obvious option is enhancing all the frames of the video which is absolutely unnecessary since many of the frames are actually useless to be processed, for example, there are dozens of frames without the car plate in the video frame.

After carefully observing the extracted frames, I first chose one to process with the best angle of the car plate (frame_190.jpg) but the problem was one of the letters came out blurred. As a result I chose two other frames that has the car plate in them and processed them as well to, first, understand the mysterious letter and also verify the already visible ones in other frames to make sure it is not read by mistake.

4. Analysis

In this section, we are going to go through the steps taken in the whole process that resulted in an enhancement in our frame(s). The main steps taken are denoising, gamma enhancement and deblurring. In order to achieve this, I made a function containing all these modifiers to make it easier applying this procedure in multiple frames. Because the scene (except for the truck moving) had the same features, we didn't need different parameters for each frame so using a function with fixed parameters was the efficient solution. Other than this function, I used two other function in the given notebook. First was a function that simply plots the given image (used to make the code more readable and less redundant). Second was the frame extraction function that was called once at the very beginning of the process. Now we are going to get into our main function and see how it is working.

4.1. Denoising

Since the frames were noisy, the first step we wanted to take was making the frames less noisy (denoising). Through a small research done, I found that for colored images we can use a function in *opencv* called *fastNlMeansDenoisingColored.*

This function has the following parameters:

1. src: The image that we are working on → "image"
2. dst: destination for the denoised image → "None" we save it in variable Idenoised
3. h: It simply indicates the filter strength. If we set this too high we will lose more noise but also some parts of our original image which is very important when we need to read the plate. → 4
4. hColor: It is the filter intensity for the color component of the image. We set it a bit higher than the h value but could be the same value. → 5
5. templateWindowSize: If we set this patch size too high, our image might be less noisy but it might also be blurry. We should be careful not to set it too high to have a readable plate. → 3
6. searchWindowSize: One common value for this parameter is 21. It should be odd and the higher value means less noise but also more computational demand which in this case wasn't very restrictive. → 21

It is noteworthy that choosing the parameters is not something definite most of the time and we must find the optimal parameters through trial and error. These values might not be the most optimal for this solution but based on the approach chosen it gives us the expected results.

4.2. Gamma Correction

The only parameter we need to set in this section is the gamma correction value. If it is 1 it doesn't change the image. If we go lower than 1, the image would be darker and if we set the value above 1, the image will be brighter. Based on the image settings, we must choose if the value must be set to make the image darker or brighter. In this case I chose to set a value of 2/3 (in the code we have inverted of the gamma so invGamma = 1/(2/3) = 1.5) to make the image a little dimmer and as a result easier to read the car plate.
Like denoising, the best value that we can use for this parameter is acquired through trial and error and it could be rather a matter of personal preference as well (in this case, some people might have sensitive eyes and might prefer a darker setting to read the car plate).

4.3. Deblurring or sharpening

There are multiple ways to deblur/sharpen an image. One common way is achieved by first blurring the image and then try to add these two images (original and blurred) with weights. For example, the blurred image would be given a negative weight so we can basically subtract the blurriness out of the original image.
In this case, I used a slightly blurred version of our frame and added it with the weight of -0.5 to our original frame with the weight of +1.5 with the functions available in opencv. *GaussianBlur* was used to make the blurry image (with kernel size (0,0) and std = 3) and *addWeighted* was used to add the two images and produce a deblurred version of our original image (with 1.5 being the weight of our image, -0.5 the weight of the blurry image and 0 as the additional value).

Then in the rest of the function, we plot the images produced in each step and finally save them on our drive to be handed out.

One other thing that we had to do was to detect the car plate and magnify that. This was done by acquiring the coordination to a subsection in our main frame that has the whole plate. For each frame we had to find the coordinates and extract them then magnify that achieved new image to make it even more visible. In this part we set the window size to 100 x 50.

After having the resulting images, we can conclude the analysis by reading the plate.

5. Conclusion

The images that we wanted to analyze were as below (images were resized to fit the document):

** frame_180.jpg*

** frame_185.jpg*

** frame_190.jpg*

These were the original frames, and the final results, with all the procedures mentioned in analysis part, are as below:

** frame_180_deblurred.jpg*

** frame_185_deblurred.jpg*

** frame_190_deblurred.jpg*

In the first two images, we can clearly see the first two letters being “DE”, followed by a space and number “1”. Then we have number “3” followed by number “5” which was a little bit similar to number three but having multiple frames analyzed, we can say it has it differences and it is more likely number five. Then we have letter “C” that is followed by a not so visible letter/number. To solve the last part. I switched to frame 190 analysis that clearly shows letter “X” but unfortunately the letter “C” is blurred out and not very visible.

In conclusion, the plate number I could read through this analysis was “DE 135CX”. It is noteworthy that this conclusion has a high reliability because it was done through analysis of multiple frames that were chosen based on the visibility of each letter and number on the plate.

Padova
09/06/2023

Mohammad Hossein Bagheri