NGUYEN Hien Minh
ET4 Informatique

INTERNSHIP REPORT

Research internship at the CARE Lab (Cybernetics and Reality Engineering Laboratory)

21/05/2018 – 10/08/2018

From night to day: a real-time image translation project using machine learning

A symbolic painting made for the project. It depicts a witch, embodying the magic of machine learning, contemplating the twilight before dawn.

CARE Lab tutors: Professor Kiyoshi Kiyokawa & Associate Professor Nobuchika Sakata
Polytech Paris-Sud tutor: Associate Professor Emmanuelle Frenoux

Polytech Paris-Sud
Maison de l'Ingénieur - bât. 620 - Centre Scientifique d'Orsay - 91405 Orsay - France
te : +33 (0)1 69 33 86 00 - fax : +33 (0)1 69 41 99 58 - www.polytech.u-psud.fr

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and respect to Professor Kiyoshi Kiyokawa and Assistant Professor Nobuchika Sakata, who followed me and helped me during and even before the internship. This project would not have been possible without their precious assistance.

I also would like to thank Ms Mina Nakamura as well as every members of the CARE Lab for their warm welcome and continuous assistance: their presence have made my stay in Japan one of the most incredible experience of my life.

Last but not least, I would like to thank Mohamed Beldi who spent this internship with me on the same project. Through good and bad times, we have learnt together to never give up and keep on working until the end.

CONTENTS

INTRODUCTION

During the 2018 summer, I conducted a 3-months research internship in Japan, as part of my Computer Science engineering curriculum at Polytech Paris-Sud. I had the chance to be accepted by Professor Kiyokawa of the CARE Lab[1], a laboratory dealing with various fields from Augmented Reality to wearable technologies.

CARE Lab mostly focuses on the interactions between humans and future technologies, such as AR or VR. Multiple projects are being conducted, but all of them are emphasizing on developing convenient, comfortable and secure tools, helping everyone's lives and ultimately conciliate the human and machine worlds.

Motivated by my passion for computer vision, I have chosen to work on an image conversion project. Along with another student from Polytech, Mohamed Beldi, I have developed an application which can convert a night video into a day video using a pre-trained neural network. As we are creating our own datasets on the go, other conversions are also possible. For example, a rainy video can turn into a sunny one thanks to a dataset captured during the typhoon season.

This project is broadly divided into two main parts: in order to train the neural network, I created a dataset composed of various kind of images. Some of them are pairs of day and night images captured by a 360 camera. Some others, like the rain dataset, have been captured by a smartphone and are all unpaired images. Mohamed mostly took care of the second part which is developing and training the neural network.

The process and results of this project are explained in this report. Even if this is an experimental project which does not have immediate industry applications, we have learned a lot during the process. One of the main teaching has been about the importance of a dataset: without a good one, even the best neural network will not be able to produce good results.

This internship has been a huge opportunity for me to learn about datasets, machine learning, and more generally the working methods of a computer engineering laboratory. However, not all knowledge should be technical. This is why this report is followed by a personal report, describing what I learned about the life of an engineer.

1 https://carelab.info/en/

This report will sum up our work done on the project as well as the valuable experience I gained during those three months. In the first part, I will describe the laboratory and the university I joined. The project's motivations and objectives will be explained in the second part. The third part will be a summary of the work done. In the last part will be presented the final application and a retrospective analysis of the results achieved during the internship as well as the internship itself.

The project's complete code and dataset is available here: https://github.com/minh-n/NightfallProject.

1 AN OVERVIEW OF CARE LAB

1.1 NAIST, ONE OF THE TOP JAPANESE UNIVERSITY

Japan's academic context seems to be unique in the world: although being a relatively young country in terms of research, it is known to have contributed to many fields, such as electronics, robotics and industrial processes. Japanese brands are indeed famous around the world in nearly every industrial sector, with household names such as Sony, Hitachi, Toyota, Panasonic, etc.

Thanks to its heavy investments in research (more than 3% of its GDP, placing Japan on the third place)[1] and education, the country has become a first-class destination for researchers and students alike, looking for a top-of-the-line scientific community and well-funded academic infrastructures. This effort is mostly driven by the private industry (roughly 75% of the total research budget) but the country's affordable educational system and high quality teaching also play their part.

NAIST (Nara Institute of Science and Technology) is the university I joined for this internship. Easily accessible by train from the cities of Nara and Osaka, it is located in the middle of the Japanese countryside, in the Takayama Science Town. It was funded in 1991 with the aim of contributing to Japan's economy and society by promoting research, education and international and industrial collaboration.

NAIST is home to more than a thousand M1, M2 and PhD students, including about 25% international students. According to the official website, NAIST has been achieving top 3 ranking among Japanese universities and even the highest evaluation in many fields, including educational scores, intellectual property, business ventures and research methods. The university receives more than 9 billion yens (about 70 million euros at current rates) each year in fundings and revenues[2].

The university provides students and teachers extensive facilities with equipments, allowing them to study and work in optimal conditions. The students are assessed continuously through reports and interviews, instead of traditional planned tests. This allow a rather flexible system which allows them to conduct their projects at their own pace while being able to schedule their classes. To further encourage personal initiatives, students can choose to join any lab among the 21 available and even work on interdisciplinary projects with the other departments[3].

Through projects associating industry, government and academia, as well as numerous internships and partnerships, students are well prepared for their careers.

1 UNESCO, 2017
2 Asahi Shimbun, 2010
3 NAIST, 2014

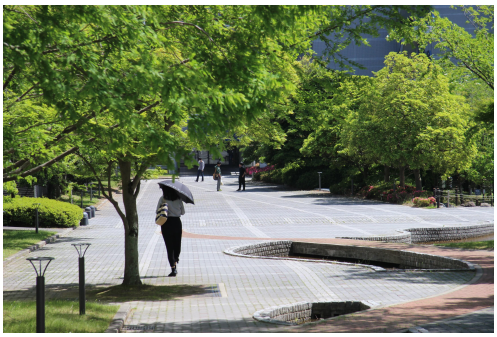
They are also able to pursue their research with a PhD or join the university staff as an assistant professor.

NAIST is divided into three main branches : the division of Biological Science, the division of Materials Science and the division of Information Science, in which my lab is located.

(a) Sunset over NAIST

(b) An university surrounded by countryside

(c) A deer, the poster animal for the city of Nara



(d) A summer day

Figure 1: The picturesque campus of NAIST (photos by me).

1.2 THE LABORATORY

Cybernetics and Reality Engineering Laboratory (CARE Lab) is a relatively new lab, formed in 2017 on the legacy of the former Vision and Media Computing lab. It is a part of the Information Science division of NAIST, and focuses on future technologies, the way they interact with their environment and the way users interact with them.

The lab's main concept is the 'personalized reality' (which is represented in the lab's logo): each individual deserves consideration in a technology-filled world. The goal of such technologies is to improve each person's skills or perception of the world; generally speaking, helping them with their daily lives.

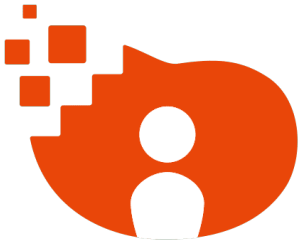


Figure 2: The lab's logo.

Various projects are conducted by the M1, M2, PhD and intern students in the lab. Many of them are working on AR and VR technologies using head-mounted displays like the Oculus Rift or the HTC Vive, whereas others are developing their own systems using 3D printers and electronic components.

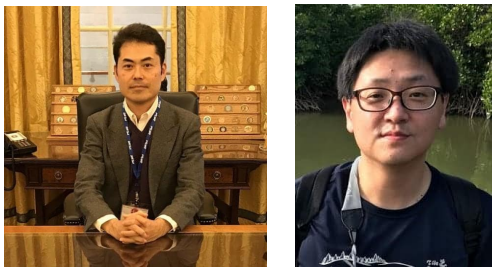
Here are some of the most singular projects, all conducted by M1 and M2 students:

- The ARamen project aims to study if AR is able to influence taste perception, by projecting a ramen bowl (traditional japanese noodle) onto an industrial cup noodle, through an HMD.
- A study of transitions between real and VR spaces. This project aims to ease the transition by various means, such as fading or animations, to make the VR space more believable and realistic. A good transition would have positive effects in learning or entertainment.
- Another project aims to develop a wearable pouch which inflates when the user eats. This is based on the assumption that when a person's shirt tightens, they feel they have eaten too much and feel full, allowing them to keep up with their diet.

These projects, touching various field of AR, VR and wearable research, should give a good idea of the interests of the laboratory in terms of objectives and expected achievements.

1.2.1 The staff

The laboratory is home to 10 students, 6 interns and 2 professors. The lab is overseen by Professor Kiyoshi Kiyokawa (PhD graduate from NAIST), a researcher specialized in virtual and augmented reality, human augmentation, 3D user interfaces, computer-supported cooperative work and context awareness[4]. The other professor responsible for running the lab is Associate Professor Nobuchika Sakata. His research interests are wearable technologies, search queries and remote collaboration[5]. Both professors are responsible for many publications and contributions in their research field as well as multiple conferences, such as IEEE or SIGGRAPH.



(a) Prof. Kiyokawa **(b)** Prof. Sakata

Figure 3: The lab's professors.

1.2.2 Working at the lab

The lab has a very laid-back atmosphere compared to the other working environments of NAIST. There is no separation between the students' desks, encouraging exchanges between them. Very soft carpets and cheap snacks and food are the reasons students often stay overnight at the lab.

The lab encourages personal responsibilities by allowing everyone to come and go at any time of the day, any day of the week. This allows very flexible schedules: I was able to conduct my dataset capture during the night, after 1AM without having any difficulties. Working during weekends is also possible if one needs to focus in a noise-free environment. Thanks to this system, I have been able to work on my project and still discovering life in Japan even in weekdays.

Students usually have classes during the morning and come to the lab by the afternoon. Both professors are usually present and ready to help students in need. Students also has to meet the professors to assess their project and receive some guidance if needed, through regular individual meetings. A research meeting, where all of the lab members gather, is held weekly, allowing each students to present their project progression during the past two weeks. After each presentation, students and professors discuss about the work achieved and the resulting course of action.

4 http://carelab.info/en/kiyoshi-kiyokawa/, as of 02/08/18.
5 https://sites.google.com/view/nobuchikasakata/research?authuser=0, as of 02/08/18, and https://scholar.google.com/citations?user=bEXv1VYAAAAJ&hl=en&oi=ao as of 02/08/18.

2 THE PROJECT

This chapter will explain the details of the project, its objectives and potential applications.

2.1 OBJECTIVES

This project (dubbed **Nightfall Project**) originally aims to convert a night video into a day video, using an image generating neural network. Transforming night to day has several applications in automated vehicles (to have a clear image of the night, allowing better visibility), surveillance and even tourism or AR attractions. For example, a future tourist will be able to clear the rain from a landscape in order to take the perfect picture.

The project is based on the work of Nvidia researchers who developed the main ideas behind the Pix2Pix and CycleGAN machine learning algorithms. We have mostly worked on the Pix2Pix algorithm which will be explained later in this chapter.

Most of the existing applications which has been released to the general public has been about converting a single image for demonstration purposes or entertainment. So far, they have been relying on cloud or browser computing, thus making real-time output very difficult to attain. Machine learning would be much more useful and close to real-life applications when applied to a live video feed. This is why one of the main objective of the project is to develop a standalone, local video processing application using a neural network model, to ensure maximum performances and eventually real-time neural network output on a wearable device, e.g. AR glasses.

Moreover, creating a night-to-day model is considerably more difficult than the existing day-to-night projects: night-to-day is about creating extra informations from an image composed of shades of black, whereas day-to-night simply strips information from a normal image.

The project is divided into roughly three parts; the first two running in parallel:

- First, a dataset of paired night and day images needs to be gathered,
- Then, we can train the neural network and produce a model,
- Finally, we can develop a real-time application using this model.

This project tackles the basics of machine learning and computer vision and will serve as an introduction about both fields, before allowing us to tackle more complex problems.

2.2 THE CURRENT STATE: A FUNCTIONING APPLICATION

The final product of this project is a Python application displaying a live webcam and the processed output in real-time of the night-to-day neural network using the webcam's feed. A video demo is available here.

It is only able to convert the laboratory's environment because of the very specific training dataset I created. However, other datasets can be used for training as the application is easily customized.

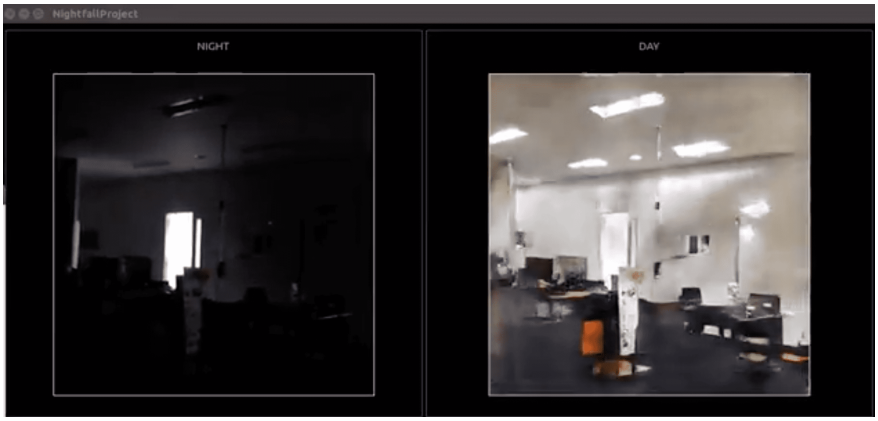


Figure 4: A screenshot of the final application featuring a night to day transformation.

Figure 4 is a screenshot of the application as it is running at night, in the lab, using a Logitech webcam. On the left, the raw webcam can be seen. On the right, the same feed with the neural network applied. The application can run at approximately at 30fps. The lack of good framerate is due to the fact that the computer needs to process each frame through the 122 000 parameters of the neural network. Other than the webcam feed, the application is also able to process a night video into a day video, to compensate the fact that no live demonstration would be possible outside of night time with only the webcam option.

The user is also able to enter the path to their own dataset when launching the application from the shell. They can also use their own dataset for training to produce the model they need, and then use it for the application.

Further informations about the application can be found in the third part and annex of this report, as well as on the project's GitHub platform.

2.3 THE THEORY BEHIND THE PROJECT

This section goes more in details into the project's three main parts:

- Part one: a dataset of paired night and day images needs to be gathered
- Part two: the neural network is trained with this dataset and produce a model
- Part three: a real-time application is developed, using this model

It will also include a description of the tools and technologies used for the dataset gathering and machine learning part.

2.3.1 Part one: the dataset

What is a dataset?

My main work during this internship has been about gathering a dataset.

thedictionary.com defines a dataset as "a collection of data records for computer processing"[1].Datasets are thus not only limited to machine learning. They are used extensively in various research and industry fields such as natural language, big data or metadata processing, and more traditionally, in mathematical and statistical models.

Data science is in fact a field on its own. According to techopedia.com [2], data science is:

> A broad field that refers to the collective processes, theories, concepts, tools and technologies that enable the review, analysis and extraction of valuable knowledge and information from raw data. It is geared toward helping individuals and organizations make better decisions from stored, consumed and managed data.

Being such an important task, creating a dataset is usually left to a data scientist who manages the production, extraction, processing and formatting of the required informations.

A dataset is always created with an intention in mind. Our objective was specific and using a pre-existing dataset would have limited the interest and scope of this project. For example, a more general dataset that include outdoor and indoor images would not be as efficient when used inside the lab. We could not find any existing night and day datasets (captured from the same position) available freely on the internet. Either the datasets were simulations (used for automated driving computer vision), or were very small samples captured from a single fixed webcam.

1 http://www.dictionary.com/browse/data-set
2 https://www.techopedia.com/definition/30202/data-science

Datasets can be presented in many shapes such as .csv files for mathematical, surveys and statistical data, or in our case, series of images. Some image datasets are labelled, such as the Cat dataset where the cats' eyes and other facial features has been annotated by hand on 10 000 images.

Labelling is not needed for our project where the images are simply separated into two categories, as it would add a lot of complexity and tedious, repetitive work. However, labelling could be beneficial for our project in order to process elements differently. For example, for a both outdoor and indoor night-to-day algorithm, the sky could be differentiated from the ceiling thanks to a labelled dataset.

Thus, the objective of the first part of the project is to learn the ways of a data scientist. I will have to create the largest and highest quality dataset possible by myself and prepare it in a way it could be used for an efficient neural network training.

On the quality of a dataset

As said before, machine learning quality depends on the quality of the input data. Since this internship does not aim to produce an application useable in any kind of situation; the quality of the dataset does not matter as much as a commercial project.

However, I have identified several points used to judge the quality of a dataset:

- A dataset large and varied enough,
- A consistency in formatting and data quality,
- Good results when used as training data,
- Can be loaded as is into a neural network without much processing,
- Can be re-used and is well-documented.

Focusing on those points have been my main objective through this internship; some may be easier than others but I believe only the first point has not been fully reached yet. Further reading about datasets and dataset quality can be found here, here, and here.

Known expectations and limitations

Gathering our own dataset certainly has certain advantages. The main one is to be able to freely shape and adapt the dataset depending on the needs of the project. For example, by developing my own dataset preprocessing algorithms, I was able to quickly refactor any new data I captured. The datasets I have produced are thus very consistent between each other: the images are the same size, captured by the same hardware and using the same jpg format. I was also able to adjust the kind of data captured (zoomed, randomly flipped, panoramic, unpaired...) to the needs of the machine learning part.

The main disadvantage is obviously the quantity of data I gathered. As an intern and not a data scientist, I was only able to do so much in terms of capture. Furthermore, I have probably made some mistakes methodologically-wise. For example, writing the image processing code so late into the project has considerably slowed me down when I had to use a plethora of scripts in different softwares, instead of one Python script.

Furthermore, an ideal dataset will include a great variety of environments, lighting conditions, and subjects, in order to adapt to any kind of situation the user would find themselves in. My dataset will essentially focus on the environment immediately available to me, i.e. the lab's main and auxiliary rooms and the NAIST campus.

2.3.2 Part two: the machine learning

In this part, I will broadly explain the basis of machine learning, Pytorch and the ideas behind the two Pix2pix and CycleGAN algorithms.

I had the chance to learn about Pytorch and machine learning as well, by following the Udemy course[3]. However, the machine learning part was mostly left to Mohamed since capturing the dataset took most of my time. Further information about the machine learning theory will be presented in his version of the report.

What is machine learning?

Machine learning is a field of artificial intelligence that focuses on training a computer to predict the outcome of a known or unknown situation. It involves a lot of input data and training time: the data is used to statistically predict the output, which can be used in turn to improve the prediction algorithm.

Machine learning is used in many sides of our digital life: speech recognition and generation (smartphone personal assistants), image recognition (automated vehicles, face recognition), search algorithms (web or mail search), GPS navigation... Other 'hidden' applications involves data from health, industrial, governmental or financial sources where machine learning is critical for pattern recognition and data analysis.

All those fields have in common the fact that without machine learning, they would have taken a lot more computational resources and programming man-hours to accomplish, and would be deemed impossible under today's budget-restricted economies.

3 Deep Learning Wizard, 2017

Neural networks

Artificial neural networks (ANN) are a specific part of machine learning. The idea originated from natural neural networks, composed by millions of neurons, which can take an input electrical signal and give an output based on this signal.

A neural network is in fact a function, composed of thousands of neurons, which are its parameters. It takes a matrix of data as an input, then applies each neuron's parameter on the data to produce an output.

The neural network can also make use of a loss function (or cost function), which is a way to evaluate the quality of the output. If the output produces a high loss number, this information can be used to optimize the neurons' parameters and tweak them to obtain a lower loss number (back-propagation). This learning process, where the computer improves the algorithm by itself, can take hours or days, depending on the complexity of the neuron layers and the quantity of inputs.

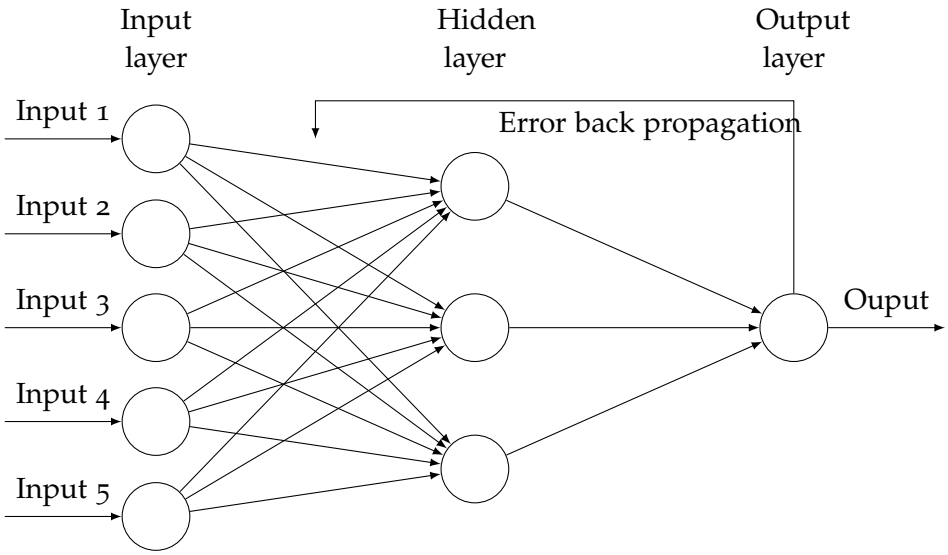


Figure 5: A diagram of a simple neural network involving back-propagation (based on Gonzalo Medina's work.)

Different types of neural networks

Neural networks are usually used to either classify (identifying a set of features and sorting items into categories) or to generate. Generative networks are the one that interest us for this project. Generative Adversiual Networks (GANs) are a very clever way to do so: it is based on a set of two neural networks working together. The first one, the **generator**, or forger, generates fake data, whereas the other one, the **discriminator**, is fed with either a set of real or fake data. The discriminator then returns a probability of the data being fake (figure 6).

As it is connected to the generator, a feedback is sent at each iteration, improving the generator network one epoch at a time. If the probability of the fake image being

fake is high, then the generating parameters needs to be changed. If the fakeness is detected as low, it means the generator is doing a good job fooling the discriminator.

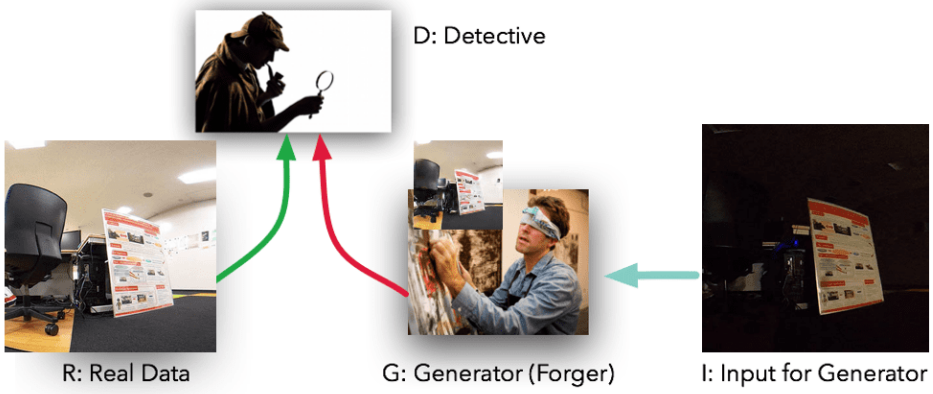


Figure 6: A diagram of a GAN. Based on the work of Dev Nag, 2017.

Pix2pix and CycleGANs

The two algorithms used in this project are Pix2Pix and CycleGANs. Both papers can be read here and here.

Pix2pix, also known as Image-to-image translation with Conditional Adversarial Networks, is an algorithm which can generates a fake image from a real one. Pix2pix needs paired images taken at the exact same location to work. The algorithm is very powerful and conversions quality is high, but the dataset creation process is much more tedious. In fact, it is the main limiting factor preventing a Pix2pix widespread use.

Pix2pix relies on the GANs principle I explained earlier: in our case, a night image (input) is *encoded* through several layers into hidden, simpler representations of the original 2D picture, then *decoded* into a day image (output). The discriminator then guesses if the output is fake or real. The generator is then adjusted depending on the output's fakeness. Fun demos of Pix2pix can be found here, including the well-known *edges2cats*.

The CycleGANs algorithm is slightly more complicated than Pix2pix, but doesn't need paired images to be trained. In CycleGANs, there is not only one but two generators working in parallel in the opposite direction (GenAB and GenBA).

In our case, GenAB transforms a night image into a fake day image. GenBA does the reverse transformation: it transforms that fake day image into a fake night image. The two generators both get better at the task and produce more believable results through the training epochs (figure 7). The CycleGAN name comes from this cyclic process.

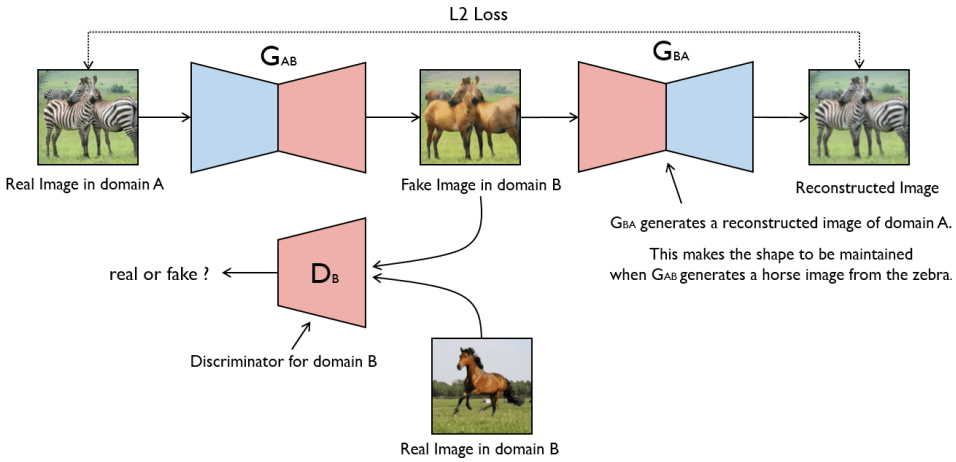


Figure 7: A diagram of a CycleGANs algorithm, from Yunjey, 2017.

Pytorch

PyTorch is one of the many machine learning libraries available. Developed by Facebook on top of Torch (a Lua library), it is extensively used across the research world and the industry.

PyTorch has two advantages for our project:

- Its syntax is very similar to traditional Python and numpy, making the learning process easier,
- It makes use of CUDA, the GPU acceleration API developed by Nvidia, making the calculation times lower.

That is why we have chosen to work with PyTorch over other libraries like Tensorflow. Furthermore, the 1.0 version of PyTorch will bring a lot of features, including C++ and mobile exports. Those two features are most interesting for our project: a consumer mobile version is the ultimate goal and C++ would only increase its performances.

2.3.3 Part three: the real-time application

This project would not be of interest if it was not able to process a real-time webcam. To some extent, processing a single image is already possible with existing technologies. That is why we have chosen to work on this side when the first two parts were done.

We have thought about two modes : direct webcam conversion and video conversion. Both are using Python and OpenCV2, as well as our pre-trained neural network, but the principles behind them are a bit different.

The direct webcam mode allows the user to directly process the scene he is witnessing in real life. This mode is close to what we wanted to achieve and works relatively well within our testing parameters (the lab room with very low light). It runs around 30 frames per second, which is very acceptable for an application pretending to be in real-time.

The video conversion mode is much slower: it extracts each frame of the video, converts them to the correct size (350px) and bundles everything again. This mode takes roughly as much time as the running time of the video. It is not optimal at all and only serves for demonstration purposes, since presentations are usually done during daytime.

So far, only an output with a size of 350x350 pixels can be produced, due to the processing time required for larger resolutions. The project currently runs on a rather powerful desktop PC and is not producing optimal performances due to the code being in Python.

We are confident that computational power will increase to a point where good quality is feasible with customer hardware. The objective of such kind of application is to run on AR glasses or smartphone and deliver a low-latency, real-time image feed to the user.

3 MY WORK

During most of the internship, I have spent my time capturing images and further improving my capture technique. This is a very experimental process: I was able to witness the machine learning results in the following days after capturing a dataset and adjust my methods accordingly.

This section will first explain the methods I used to create the three datasets, and then show the final datasets and their features.

3.1 CREATING THE DATASETS

Let's focus now on the creation of the datasets. The process can be divided into four main parts :

- Choose the capture tools,
- Move those tools,
- Use a combination of the two first parts to capture the images,
- Processing the images.

3.1.1 How to capture the images?

The three first parts are considered similar enough to be placed in the same category: the *image capturing process*.

The capture tools

I have planned the process according to the tools I was given access to (table 1). The first few days at the lab were spent on experimenting on the various cameras at my disposal. On the next page is a table summing up the difference between those tools. I tried to use the first three (webcam, 360 camera and smartphone) for this project. The DSLR was deemed too impractical and unrealistic, mostly because of its size and the inability to wirelessly control it.

	Price	*Portability*	*Image quality*	*Automation*	*Other*
PC webcam	Cheapest	Low: Needs to be tethered to a laptop	Very low: No night images possible	High: must be controlled with a laptop, either program or human	Not waterproof, open firmware
360 camera	Expensive	High: lightweight and self-sufficient	Medium: Some night images possible	Medium: can be controlled over Wifi with a proprietary app. Can stream a feed to PC	Very wide angle = more image capture. Waterproof, closed firmware
Smartphone	Expensive	High: lightweight and self-sufficient	High: good range of luminosity. Night image possible	Medium: can run an app, but mostly needs an human operator	Waterproof, proprietary but customizable software to an extent
DSLR	Very expensive	Low: very bulky and heavy	Very high. Every kind of images possible	Very low: always need an human operator	Not waterproof, closed firmware
Human eye	Priceless	Very high, always available	Highest range of lighting, highest image quality	None (or fully automated?)	Computer vision should aim to reach this quality!

Table 1: Table summing up the different capture tools available at the lab

How to move the capture tools

Now that we have seen that nearly all of the capture tools can be used, let's study how to move them. One of the specificity of this project was the use of a programmable robot to capture the dataset. The lab had two iRobot Create 2 units (figure 8), based on the Roomba cleaner robot but intended for an academic or experimental use.

Figure 8: The iRobot Create2 robot.

Finding how to guide the robot was the first difficulty I encountered. I used a robot guidance program which allowed control of the robot using a tethered computer. An API was available on the official website, giving access not only to the robot's movement functions but also its front sensors and music capabilities.

This API allowed me to avoid researching the electronics and electrical parts of the robot (the amount of bauds used to send a certain command, etc.). However, it was very badly documented and prone to crashes. I spent a lot of time making it work and run without human intervention.

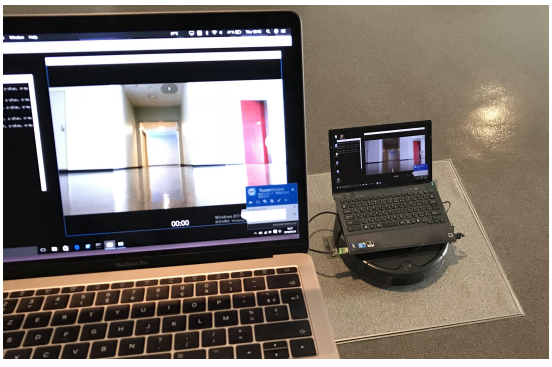


Figure 9: The robot-computer system.

With the help of a Python script (available on the project's Github), I was able to manually control the robot through a system of two computers.

One computer is directly tethered onto the robot, using a USB-to-serial cable. Since the robot is flat and very slow, the computer can be stabilized over it without any fixature. The other computer controls the first computer via TeamViewer, as the two machines share the same wifi network.

This setup is intended to be used indoors in a controlled environment. However, tests were made and the robot was able to move inside the university building – as long as it was in range of a Wifi repeater. The robot performed very badly outside because of the bumpy road tiles used on the campus' grounds. The image was very shaky. The cameras tried to compensate that effect using embedded software stabilization, which made the output even worse.

Guiding the robot manually revealed a serious issue: the robot was not calibrated to go straight. Since it was a bit old and not intended for precise trajectory (a cleaner robot only needs random movements), its trajectory was unstable and it always end up in a wall, despite a "go straight" command. This issue was one of the most determining finding of the first week of work. Before that, I assumed that the robot could follow a predetermined trajectory with little or no deviation at all and that the dataset capture will be simple.

In order to capture the images at the same location, the robot needs to follow a very precise path. If the robot's path is wrong even by 10 centimeters, the difference could be seen on the produced pair of images and mess up the machine learning results.

Several methods were tried to guide the robot.

Relative positioning

Using the number of wheel rotation is one of the way to obtain the position of the robot. For example, when knowing the speed of the motor, one can predict the robot's travelling distance after a certain number of wheel rotations. This method only works in ideal conditions: different floor materials can effectively change the properties of the wheels and relay false informations to the sensors.

Since the robot's path cannot stay straight, this method cannot be used.

Markers

Markers are very effective tools to give orders to a robot and to help it determine its own position in a room. In this project, markers are detected by a webcam plugged into a laptop, but an Arduino board could be used in more robotic-focused projects.

The Aruco library provides a marker generator which can generates thousands of different markers of different resolutions. I only needed 5 as I was only interested in giving 5 commands to the robot: go straight, go left, go right, go back, stop. Those markers were printed. When presented to the webcam, the code corresponding to each marker was executed and the robot reacted accordingly (figure 10).

However, this solution was overly complicated and ultimately scrapped, that is why no demo videos are available. Fine tuning was impossible since the markers' position were only approximately known by the robot, judging by the marker's size.

Only a limited set of orders could be given and severely reduced the interest of this method.

Figure 10: The robot was able to recognize the five type of markers. It will receive different commands depending on the type of marker seen.

Infrared markers were suggested by the professors, as the images had to be captured at night. Those IR markers were different from the Aruco ones (the library I used) and would have required me to use an IR camera with different properties than the normal webcams.

Moreover, using markers would have added a layer of complexity to the already-overloaded robot (the laptop had to control two cameras and the robot at the time, taking three USB ports).

Line-following

After the markers failure, I tried something much simpler: **a line-following robot**. Inspired by an electronic students project I have seen at Polytech, a robot following a black tape line, I attempted to implement a similar solution on the cleaner robot.

The line-following algorithm is based on the OpenCV library and the Project Robo Cup Junior 2014 by Arne Baeyens. Mr Baeyens' project also used OpenCV but was destined to run a Raspberry Pi-based robot. I learned a lot reading his algorithm and created my own version of it using part of his, with a standard webcam and the iRobot's API.

The Python script first creates an infinite loop. At each iteration of the loop, a grey image of the scene is created. The tape is thus made visible and identified, allowing a vector to be traced from the center of the screen to the point where we want the tape to stop being recognized. Using the coordinates of this vector, we can give orders to the robot to correct its trajectory (figure 11).

The main drawback of this method was again, night vision. By installing a smartphone LED light on the robot pointing toward the tape, I was able to illuminate the path and making the robot follow the line, without hampering the night capture of the 360 camera. An explanatory video of the robot in action can be found here.

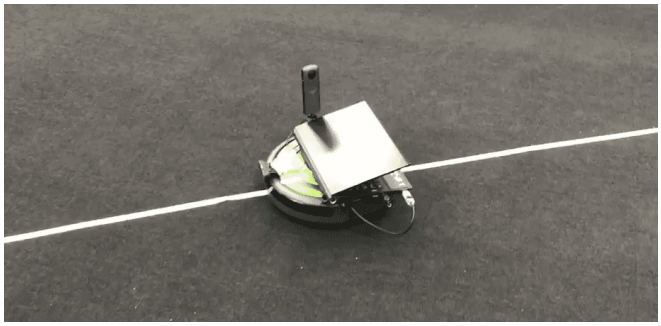


Figure 11: The robot following the white tape lines. Demo: https://youtu.be/vov4H4KSB8A

In the end, I was able to capture part of the dataset with the line-following algorithm. I have learned that software trajectory correction is at best imprecise and at worst impossible and having a correctly calibrated robot is crucial for a task such as dataset capture.

Pushing by hand

A physical solution would be preferred: model train tracks, for example. As I was unable to install train tracks around the lab or the campus, I had to improvise and used a cart pushed by hand (figure 12), on which I put a webcam and a laptop. This solution was also used with a 360 camera for the third dataset (approximate pix2pix). Unfortunately, the cart was as unstable as the robot on the outside because of the small wheels and the bumpy tiles.

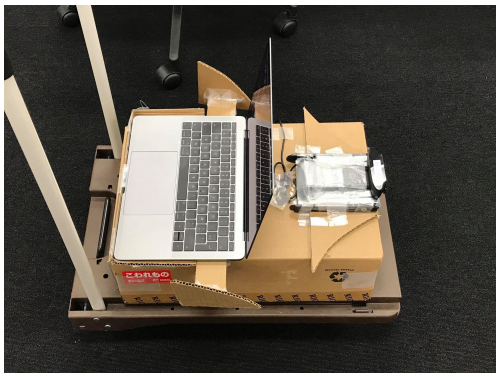


Figure 12: The cart used to move the webcam. No computer science is involved at all.

For the CycleGANs algorithm, I was able to use a smartphone held by hand to capture the videos. This is by far the best method since the smartphone is a very fast and high quality tool (see table 1). Using a smartphone would be the best way to create a dataset, since it is available everywhere and easy to use. Unfortunately, since the CycleGANs algorithm idea has been scrapped for our project, this method needs to be forgotten as well.

As a conclusion for this section, there is no optimal way to capture images. Depending on the needs of a project, the methods used to move the cameras can vary wildly and the best way to figure it out is through testing and developing one's own methods.

The capture

After choosing the capture tools and movement system, we can properly step into the final part of the dataset capture: the capture process itself.

A time constraint soon appeared : for the night images, I had to wait until the lab was empty, limiting my capturing hours to after midnight or during weekends. As for the rain to day dataset, I had to wait for a typhoon, as weather prediction was very inconsistent during the Japanese rainy season. It was clear that only a small-scale dataset could be produced during this three-months internship. My speed of dataset creation has also been unequal due to the changing tools and parameters behind each series, and the evolving needs of the machine learning part.

Although I encountered those difficulties, I carried on with the capture and achieved the results described earlier.

Capturing the first dataset

At first, I tried to capture night images using a desktop Logitech webcam (Logitech C270, 720p). However, it was clear that nothing could be captured as the webcam's sensor wasn't powerful enough or the software not intended for night vision (fig. 13).

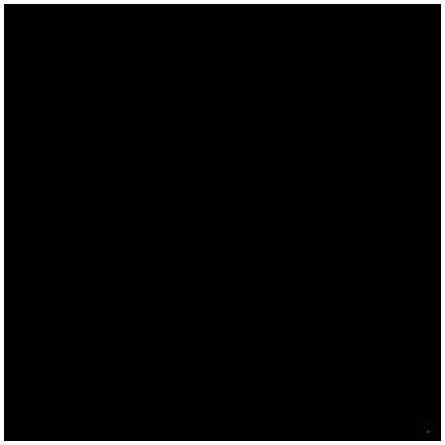


Figure 13: Nothing much can be seen using a webcam in the dark, except for light sources such as computer screens LEDs.

This behavior made me think about the future of this project: the original objective was to deliver a real-time feedback for a night worker for example, but since the human eye does better in the dark, a head-mounted display would be useless. The project would be more useful for photography or AR games, tasks that generally rely more on a display and need less precision.

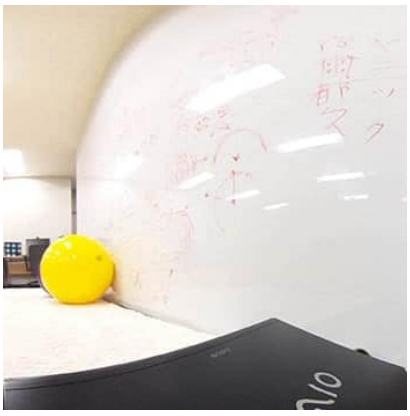
I then switched to a 360 camera which had better a sensor, and thus better night vision capabilities. The 360 camera also allowed me to capture more images because of the field of view which cover the entirety of the surrounding environment.

Figure 14: A Ricoh Theta 360 camera (actually in wireless mode).

In good conditions (straight lines), the robot can be guided using the line-following algorithm (figures 12 & 15). In other conditions, the robot needs to be manually guided. A picture is taken with the light on. The light is turned off and a picture is taken again. This ensures that the pairs are taken at the exact same position.

Figure 15: The line-following robot capturing images with the 360 camera.

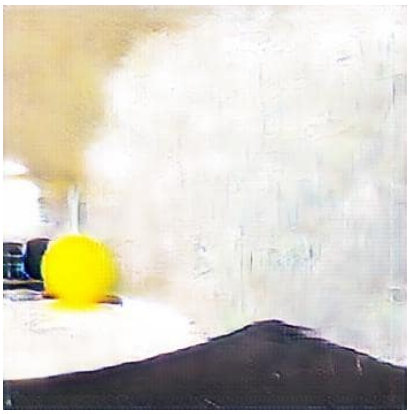
The images captured with the 360 camera can seem to be completely dark and unusable, but in reality, there is more informations in them than in the webcam's dark images. When I pushed the luminosity to the maximum level in an image editing software, there was nothing but artifacts in the webcam image (of the same scene). In the 360 camera image, the outlines of the room could be seen. Thus, we can consider this image exploitable by the neural network. This proved to be right (figure 16c).



(a) The day image captured with the 360 camera. Note: the computer guiding the robot can be seen at the bottom of the image.

(b) The corresponding night image. Nothing much can be seen except for the yellow ball.

(c) The night image with artificially increased luminosity.



(d) The night image used in the Pix2pix neural network for training (around 100 epochs).

Figure 16: Different results from one image from the 1st dataset.

Capturing the second dataset: new horizons using CycleGANs

Trying a CycleGAN approach freed me from the constraint of having to guide the robot, as I didn't have to make pairs anymore. I immediately tried to capture videos and images with the third proposed tool: the smartphone. It was the most efficient device to create a dataset, if paired images were not involved.

The smartphone is the most easy tool to set up, use and extract data from. It provides efficient software and hardware stabilization, a live feedback and video editing is even possible on the device itself. It allowed me to just point and shoot. Thus, the capturing process was straightforward and does not need further explanation, in my opinion.

Capturing the third dataset: CycleGANs has failed, let's go back to Pix2pix

I finally attempted to capture a paired images dataset during the last part of the internship. Knowing that the robot and the CycleGAN-based datasets were not perfect, I decided to try to make a Pix2Pix dataset one more time.

Using the hand-pushed cart (figure 17), I walked around the campus during (the very rare) rainy and sunny days, trying to trace along the same lines. However, the 360 camera I used was not properly oriented. As no direct visual feedback was possible on the 360 camera (I would have been forced to use a laptop under the rain, and those are not waterproof), I was not capable of aligning the sunny and rainy capture angles. The results are mediocre and not very diverse but useable nonetheless.

Figure 17: Pushing the cart around the campus with the 360 camera.

3.1.2 How to process a series of captured photos?

After capturing a series of images, post-processing is necessary in order to make the dataset useable by the neural network. The cameras used to capture the images produce either a video or a series of photos. We will call these *raw data*.

Raw videos and raw photos have both advantages and drawbacks. The video allows for much more data: it can be started at the beginning of the capture session and stopped at the end, capturing hundred of thousands of frames at 60fps, whereas the individual photos needs to be taken by hand at precise moments.

However, when capturing a video, a lot of dynamic range and image quality is lost. That means a night video will be of a lesser quality than a night photo. In the end, it can be summed up into the following table (table 2):

Dataset produced from videos	**Dataset produced from photos**
More data, less quality	More quality, less data

Table 2: Comparison between data capture types

All of these assumptions is based on my observations during this internship. It could be different with other kind of video and photo hardware but the principles behind it should stay the same.

The dataset standards

I wrote a script in Python in order to pre-process videos and photos alike. First, the script takes a folder of videos or photos. If it is a video, one every two frames are extracted into a temporary folder by a script (frame.py). This folder is then cropped by another script (crop.py) into 350x350px jpg images, which can be directly used by the neural network.

This allows the produced dataset to be very consistent, following one of the quality guidelines explained in the second chapter. All the dataset images' size are 350x350, are formatted into jpeg and follows the same naming convention.

Side note: is real-time video processing achievable with my algorithm?

My script has been used a lot to process a batch of videos or photos into a dataset. This processing is not time-constrained since it happens even before the machine learning. I have developed this script with this use in mind, solely as an dataset input script and not as a user input one.

However, the same script could be used for the "output" part: producing a day video from a night video. For this case, my script is not optimal: the pre-processing needs to happen before the neural network's main processing. This process cannot happen when the video is played or converted, since each frame of the video has to be loaded into the memory. Only when all frames have been loaded, the neural network can begin its work on each frame.

At the end, a 24mn video can be pre-processed in about 8mn. Even before the conversion from day to night, roughly one-third of the video time is spent on extracting and cropping the images.

We have also made the choice of using Python as our main language, as real-time was not the primary focus at the beginning of the project. Python effectively covered our needs as it contains many libraries ready to be used, both in image processing, robot driving or machine learning. However, another language like C++ (up to 50 less processing time) would be much more optimized for a real-time application.

With the constraints of needing to load every frames before the conversion and the slowness of Python, real-time is currently not possible with my method, even with better or futuristic hardware. The only way to achieve real-time is to use the webcam feed system developed by Mohamed.

3.2 THE DATASETS' FEATURES

Now that we know more about the process of making each of the three datasets, from the choice of the tools to the capture of the images, here are the details about those datasets.

3.2.1 First batch of dataset

The images of this dataset are all paired and taken from a static point of view. They were created by the originally planned tethered cleaner robot and a 360 camera, using partially a line-following algorithm and partially manually controlled.

I was able to produce about 450 pairs of distinct images we trained the neural networks with to create our model. Combined with a dataset created by another CARE Lab student last year (the Koga dataset), we had about 600 paired images available for training. Here are some samples of this dataset:

(a) A day image captured with the 360 camera.

(b) The corresponding night image.

(c) Another day image.

(d) The corresponding night image.

Figure 18: Different samples from the 1st dataset.

3.2.2 Second batch of dataset

This dataset consist of unpaired images captured by a smartphone. The process was much simpler since I didn't have to take pictures from the same locations. This dataset is by far the biggest and highest quality, because of the smartphone's superior image capture capabilities and the removed constraint or paired images.

This dataset is composed of about 30 000 images divided into several categories: Rain, Typhoon, Night, Day. The four categories can be used interchangeably for training, in order to convert various environmental settings. The raindrops in the first two categories have an advantage: as they do not exist in the sunny data, the model will learn to remove those raindrops which could be a hindrance for real life uses. An automated car driving in the rain will not want to have its visual sensors covered by water.

(a) A rain image captued by a smartphone.

(b) A night image captured by a smartphone.

(c) Another rain image during a typhoon, showing raindrops.

(d) A day image.

Figure 19: Different samples from the 2nd dataset captured from various places in the campus.

As said before, this dataset is used for CycleGAN but did not yield sufficiently realistic results. More results can be seen in Appendix 5.1. However, it is completely exploitable for non-commercial purposes such as artistic creations.

3.2.3 Third batch of dataset

This dataset is entirely experimental and not very diverse. It is composed of about 1500 approximate paired images of sunny and rainy weather, captured by the 360 camera. The raw footage was then synchronized using Adobe Premiere Pro, by placing the sunny footage on top of the rainy one.

(a) A rain image captured by the 360 camera.

(b) The corresponding sun image by the same camera.

(c) Another rain image.

(d) The corresponding sun image.

Figure 20: Different samples from the 3rd dataset.

3.3 OTHER WORK

My other work involves some project documentation and presentation writing, including the two readme (appendix 5.3 and appendix 5.4), as well as a comprehensive video destined to show the main achievements of this project (appendix 5.2). I also wrote a 4 pages guide for NAIST newcomers with essential informations such as transportations, food and administrative tasks.

4 RESULTS AND CONCLUSION

4.1 RESULTS

In the end, we believe we have achieved our initial objectives, despite the difficulties encountered on the way.

As said before, the application is able to convert a night scene into a day scene, both in real-time and with existing data:

Figure 21: A generated day image of the lab.

The scene is effectively translated into the right mapping, illumination and colors, but a lot of artifacts remain on the image and immediately gives away the fact that it is a machine learning-generated image. One solution for this would be to superimpose that generated image with a bit of transparency on top of the original video, using various blend modes. This would allow the final result to be a day video while reducing the quantity of artifacts and keeping the features of the night video that the machine learning may remove.

The poor results are due to the lack of a diverse and large dataset. This problem cannot be simply solved. However, further iterations of Pix2pix and CycleGANs are due to use less data than the earlier versions.

The application can also convert a rainy outside scene, with less than good results:

Figure 22: A CycleGANs result example: a generated sunny image.

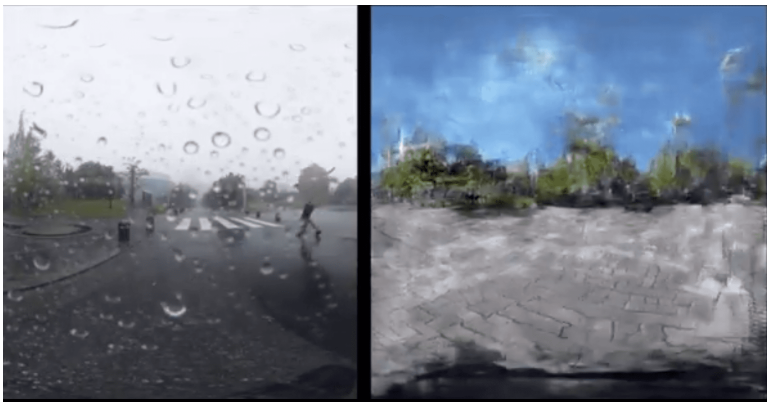


Figure 23: An approximate Pix2pix example: a generated sunny image.

This time, the artifacts are way too invasive for the images to remotely be usable. The scene is barely distinguishable. Some textures are lost as they were not present in the training dataset, such as the road (figure 22). The CycleGANs algorithm even 'invents' parts of the image: it places grass on areas without grass and removes buildings (figure 23). Both methods used for outside dataset capture are not sufficient to yield good results.

Through the results we have seen on the machine learning part, I have understood several facts about datasets. First of all, dataset is said to be the fuel of machine learning. Without enough dataset, no good results can be produced (this point has already been explained in this report).

The second important point is the diversity of a dataset. If a neural network always sees the same places and features, such as in the case of the third dataset (approximative Pix2pix), it will forcefully convert every other unknown features into the known ones (overtraining). That is why balancing training with dataset quality is important.

4.2 CONCLUSION

Despite the numerous difficulties I encountered while making the datasets, we had in the end a sufficient quantity data to work with. The last few weeks of the internship were dedicated to run the machine learning simulations and we were satisfied enough with the application's performances, avoiding another painful dataset making campaign.

Taking care of the dataset has taught me a lot about the data scientist job. Not only a data scientist has to make sure his dataset is of the highest quality, they also have to make sure the dataset fits to the needs of the project, which can wildly vary, depending on the often unpredictable results produced by machine learning. The presence of a data scientist can be very beneficial to a project. During a machine learning training session, or more broadly, a machine learning, they can quickly identify the lacking areas or the malfunctions caused by the dataset, and directly work on those points without having to go through reports and other engineer's interpretations.

One can argue that this internship has been more of a learning one than a research one; this is not wrong. It will take much more time for us to discover anything new, since we had to learn about machine learning from the beginning. However, we believe we have achieved a task that the original Pix2pix project could not: converting night to day images, whereas the Nvidia scientists only attempted to convert day into night (which is considerably easier).

As a conclusion, working with a research dataset has spiked my interest for data science and machine learning. Since this is my first experience working nearly autonomously on a project, I have been able to learn a lot and improve my skills in computer science, time managing, teamworking and, to some extent, Japanese language.

The three months I spent on Nightfall Project were not only a good training for my computer science career, but also a precious human experience. I was able to meet wonderful people at NAIST, including CARE Lab's teachers and students and other people from various labs. Spending my life in Japan with them has truly made me realize how important collaboration and mutual learning is important. I will cherish forever my memories at NAIST as one of the most beautiful time of my life.

Figure 24: A beautiful country.

BIBLIOGRAPHY

Asahi Shimbun

2010 "Evaluation of Achievements Related to the 2nd Medium-term Goals and Plans", http://www.naist.jp/en/about_naist/highlights/ranking.html.

Deep Learning Wizard

2017 *Practical Deep Learning with PyTorch*, https://www.udemy.com/practical-deep-learning-with-pytorch/.

Dev Nag

2017 "Generative Adversarial Networks (GANs) in 50 lines of code", https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros

2016 "Image-to-Image Translation with Conditional Adversarial Networks", *CPVR*, https://arxiv.org/abs/1611.07004.

NAIST

2014 "Next Generation Interdisciplinary Research Project", http://www.naist.jp/kensui/content/en/fusion_nextgen.html.

UNESCO, Institute For Statistics

2017 "How much does your country invest in R and D?", http://uis.unesco.org/apps/visualisations/research-and-development-spending/.

Yunjey

2017 "MNIST-to-SVHN and SVHN-to-MNIST - PyTorch Implementation of CycleGAN and Semi-Supervised GAN for Domain Transfer", https://github.com/yunjey/mnist-svhn-transfer.

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A Efros

2017 "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", *Computer Vision (ICCV), 2017 IEEE International Conference on*, https://arxiv.org/abs/1703.10593.

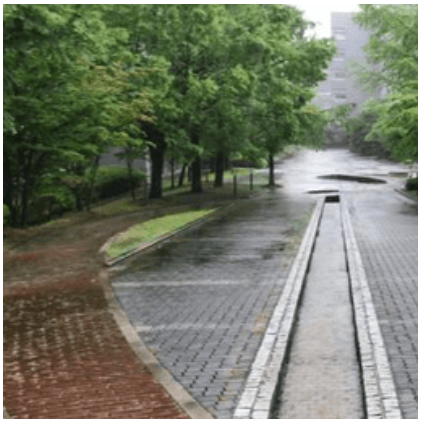
5 APPENDIX

This section contains some additional content related to the projects: some results, the final presentation's video and two readme files. Those documents have been redacted by myself and can be found on the project's Github page.

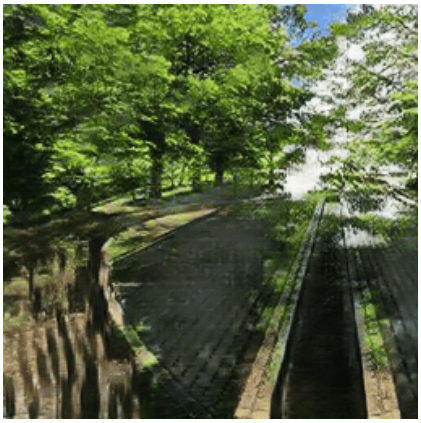
5.1 APPENDIX A: INTERESTING MACHINE LEARNING RESULTS

(a) A real image of rainy weather (135)

(b) The fake generated image. Notice the building's disappearance and the grass overgrowth



(c) Another real image (136)



(d) The fake generated image. Notice the model's attempt to generate foliage shadow on the ground

Figure 25: Some rather messy results of the CycleGAN algorithm.

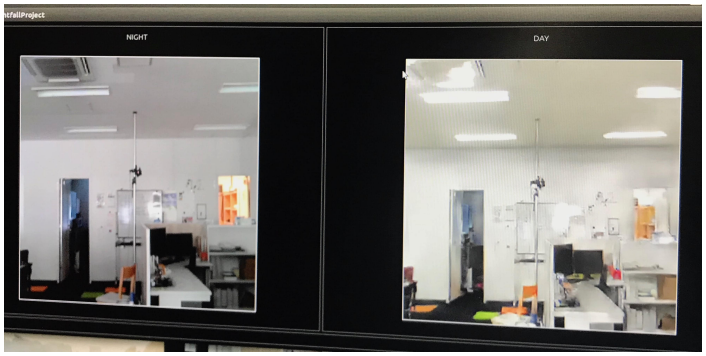


Figure 26: A day image is also converted to a day image, but the neons are on.

5.2 APPENDIX B: A VIDEO DEMONSTRATION OF THE APPLICATION

The video made for the final presentation is available at the following link:

https://www.youtube.com/watch?v=Sq3VLYMJ3ts

5.3 APPENDIX C: THE PROJET'S MAIN README FILE

This document is the main readme of the project's Github page. It is available at this address, properly formatted.

README.md: NIGHTFALL PROJECT

Nightfall Project's objective is to transform a photo (for example night to day or rainy to sunny) using machine learning. This project was developped during a 3-months long internship at CARE Lab (Nara Institute of Science and Technology, Nara, Japan) by a team of two, @Mohzick and @minh-n.

Nightfall Project originally aims to convert a night video into a day video, using an image generating neural network. Transforming night to day has several applications in automated vehicles (to have a clear image of the night, allowing better visibility), surveillance and even tourism or AR attractions. For example, a future tourist will be able to clear rain from a landscape to take the perfect picture.

This project tackles the basics of machine learning and computer vision. It is mostly a learning experience and reuses the work of Nvidia who developped the main ideas behind the Pix2Pix and CycleGAN machine learning algorithms. Both papers can be read here and here.

This project is broadly divided into three parts: in order to train the neural network, I created a dataset composed of various kind of images. Some of them are pairs of day and night images captured by a 360 camera. Some others, like the rain dataset, have been captured by a smartphone and are all unpaired images. The seccond part involves understanding and implementing the Pix2pix and CycleGAN algorithms and creating a model using the dataset. Finally, a real-time application using a PC webcam can be created.

The application

The final product of this project is a real-time Python application displaying a webcam feed and the processed output of the night-to-day neural network. It is only able to convert the laboratory's environment because of the very specific training dataset @minh-n created.

A video demo of the application

Running the application

Prerequisites

The conversion application can be runned as is on Linux and needs the following libraries:

- Python 3.6
- CUDA 9.2
- PyTorch 0.4
- Latest versions of PIL and OpenCV through Python's pip-install

The dataset processing scripts only needs PIL and OpenCV.

Machine learning

The Pix2Pix neural network can be trained using a dataset formatted using the available scripts. The dataset needs to be separated into two folders of paired images (Day and Night or any other transformation). The images' size needs to be 350x350. A good number of pairs would be around 1000 (we could only use the 650 available for the final application).

We ran the training on a desktop PC equipped with a GeForce GTX 1080. Each epoch could take approximately up to 3 minutes, depending on the size of the dataset

and its images. The main training script is train.py and can be run in a Linux shell as is.

Technologies used

With the help of a programmable cleaner robot and an omnidirectional camera, we have captured many pairs of the corresponding night and day-time images from the same viewpoint.

A video of the robot in action

PyTorch is one of the many machine learning libraries available. Developed by Facebook on top of Torch (a Lua library), it is extensively used across the research world and the industry.

PyTorch has two advantages for our project:

- Its syntax is very similar to traditional Python and numpy, making the learning process easier,
- It makes use of CUDA, the GPU acceleration API developed by Nvidia, making the calculation times lower.

That is why we have chosen to work with PyTorch over other libraries like Tensorflow. Furthermore, the 1.0 version of PyTorch will bring a lot of features, including C++ and mobile exports. Those two features are most interesting for our project: a consumer mobile version is the ultimate goal and C++ would only increase its performances.

Future work

A standalone Windows app could be developed in order to use it with a Hololens type of device. Other datasets can be created to make the application useable in various kinds of environments.

Aknowledgments

We would like to thank the team behind Pix2Pix, CycleGAN, PyTorch and CUDA, as well as the teachers and students of CARE Lab. Without them, this whole project would not have been possible.

5.4 APPENDIX D: THE DATASET'S README FILE

This is a dataset of night and day pictures of the same location captured for the project.

1. How to use

The datasets are uploaded as is and are ready to be loaded into a neural network.

Most of the datasets are divised into pairs of Day and Night photos. Those can be used for a Pix2Pix algorithm since both photos have been captured at the same location.

Should you want to check if the pairs have been captured from the exact same location, a python script ('dataset/CombinatedImages/combine.py') is also included in order to combine a pair of images into a single image with a size of 700x350px.

The original files have been placed into ('dataset/Originals'). The Webcam test inputs are in ('dataset/Webcam').

Running the scripts

The dataset processing scripts can be run in a Linux shell. PIL and OpenCV will need to be installed.

python ./crop.py –resize 350 –startpos 0 –divide 2 –inFolder ../path/to/input/large/pictures/.. –outFolder ../path/to/output/cropped/pictures/..

python ./frame.py –inFolder ../path/to/input/videos/.. –outFolder ../path/to/output/frames/..

2. The datasets' content

DN1 is the first attempt of capture (see section 2 for capture method).

DN2 is the highest quality set of pictures and also the most numerous. It was captured with the same method as DN1. However, the robot's path was planned more carefully and all lit screens were shut down to ensure darkness.

DN3 was captured using another method: the camera was tethered to the computer and 4K screenshots were made using MPC-HC. It produced rather a low-quality result and some images are nearly black. However, no exposure adustments have been made: this could be a problem for machine learning, with the numerous artifacts in the night pictures.

DN4 Dark: as its name implies, it contains the pictures that received no or insufficient illumination during night. We will see if those are exploitable.

DN5 all is a combination of DN1 and DN2 which are relatively similar datasets. It yields pretty good results when feeded to the neural network. Other dataset of this kind will be made in the near future. After a 1000 epochs training, it gives better results when applied to DN3.

DN6 is a combination of DN1 and DN2, cut into four sub-photos instead of being resized. The idea behind DN6 is that the other datasets required to images to be resized from 1440px to 350px, in order to help the neural network by reducing the amount of calculations needed. A lot of data is lost and wasted this way. Instead, what can be done is to crop the full images into smaller ones, eliminating the need for resizing. DN6 especially focuses on details. Each image size is 450x450 px (4 images from a 900x900 image). To be tested.

DN7 is a combination of DN5 and resized DN6 (both sizes are 350x350 px). It gives details (4/5 of the dataset) and a general view (1/5) of the room at the same time. It is the most extensive dataset as for now, containing 1084 pairs (865 from DN6 and 219 from DN5). The results are mixed: for 250 epochs which took more than 15 hours, the neural network did NOT produce a more accurate model than before.

KG1 is a dataset from another student of CARE Lab, Koga. He graciously offered us his 106-pictures dataset from last year. This dataset was captured during the day, so no image is completely black as sunlight could get through the lab windows. The result is 424 pairs with the lights on and off.

CGN1 is a conbination of DN1, 2, 3 and 4, totalling 312 pairs. Although CycleGAN doesn't need pairs, we don't have anything else at the moment.

Weather/RN1 is a new dataset used in the CycleGAN training. It is composed of more than 30 000 unpaired images, since CycleGAN does not need pairs. It is divided into four categories: Typhoon 1 and 2, Day and Night. Packs of 1000 randomly selected images are also available for lesser training time.

Webcam is a very dark dataset captured with the Logitech 720 webcam. It is intended to be used as a test dataset; to see how the model reacts with a common webcam and a dark setting.

Further dataset types will come, depending on the needs of the project.

3. Methodology

For image capture, I used a Ricoh Theta S camera producing 360 degrees, 4K pictures. The Ricoh was mounted on a Roomba iRobot driven by a tethered laptop. The Roomba was guided by masking tape on the lab floor, using Python/OpenCV for its line recognition algorithm. The 360 pictures were then post-processed in Adobe Photoshop with various scripts, then renamed with the help of Advanced Renamer.

As the lab isn't able to be fully in the dark, the photos nearly all have illumination from outside. This should be close to realistic conditions. Furthermore, the webcam is not able to capture any image under a certain luminosity threshold. Thus, the exterior illumination should be seen as an advantage of this dataset. Some fully dark images are also randomly included. They are to be removed if they do not prove to be useful for the training.

Some further research may be needed to improve the process' automation.