UNIVERSITÀ DI TRENTO
Dipartimento di Fisica

CONFINDUSTRIA TRENTO

POLO MECCATRONICA

PROCEEDINGS OF THE EVENT IPSP2021 INDUSTRIAL PROBLEM SOLVING WITH PHYSICS

Trento (Italy)
July 19-24, 2021

Proceedings of the event IPSP2021

Industrial Problem Solving with Physics

Trento, July 19 – 24, 2021

Editors
Mattia Mancinelli
Michele Orlandi
Luca Tubiana

Trento
Università degli Studi di Trento

Proceedings of the event IPSP2018: Industrial Problem Solving with Physics: Trento, July 19 – 24, 2021 / editors Mattia Mancinelli, Michele Orlandi, Luca Tubiana. - Trento: Università degli Studi di Trento, 2021. - 63 p.: ill. - ISBN: 978-88-8443-965-9.

Latex class available at `https://github.com/mrgrass/IPSP_latex_Class`

PREFACE

Confindustria Trento is a partner of IPSP since the very first edition. When the founders of the competition – three PhD students of the Physics Department of the University of Trento – presented us their idea, we decided to join the project mainly for three reasons. First of all, it is a bottom-up initiative managed by proactive students. Second, it contributes to promote the role of graduates and PhDs in Physics within the industrial sector. Third, it can really help to narrow the gap between research and industry.
So far, our experience with IPSP has been extremely positive. Companies who participated in IPSP have obtained innovative solutions to their technical problems, with important effects on their products and productive processes. That is why we will support IPSP in the future as well. Let's innovate together!

Alessandro Santini, Confindustria Trento

Since its founding in 1986, Trentino Sviluppo has been promoting the connection between research, business and education. On these pillars we built Polo Meccatronica, a technology hub where companies, students and researchers work side by side to face the challenges posed by Industry 4.0. Technology transfer, innovation and cross-fertilization are also the basis of ProM Facility, our cutting-edge laboratory where SMEs, industrial groups, startups and researchers develop and patent new business ideas related to the additive manufacturing and prototyping fields.
Recognizing the importance of the circular transfer of knowledge, we are therefore proud to renew our commitment to the IPSP initiative. To us, this competition represents a gainful opportunity for both young talents – who can test their academic skills on the ground – and enterprises – which can experiment innovative solutions to achieve their goals as well as make contact with the most brilliant physicists of the future. Finally, we wish luck and success to the new 2022 edition!

Paolo Gregori, Polo Meccatronica

Industrial Problem Solving with Physics has reached its 6th edition in 2021 and from a Technology Transfer perspective, it has consolidated a new model of industry-academia collaboration which allows industrial problems to be solved quickly and effectively in one week. A growing trend within the initiative consists of a series of meetings among the University and the enterprises involved in IPSP after the event, fostering long-term collaborations encouraged by the contribution of Trentino Sviluppo – Polo Meccatronica and Confindustria Trento. Indeed, one of the main targets of IPSP is the development of a strong link between University laboratories and enterprises. Both applied research and the placement of young scientists can profit from the dialogue promoted by this event.

Vanessa Ravagni, Directorate of Research Services and Valorization,
University of Trento

INTRODUCTION

Industrial Problem Solving with Physics (IPSP) is an annual event organized by the Department of Physics, the Doctoral School of Physics and the Research Support and Valorization Division of the University of Trento, in collaboration with Confindustria Trento and Polo Meccatronica-Trentino Sviluppo. The main objective of the initiative is to promote the collaboration between the Department of Physics and the industrial world.
IPSP reached its sixth edition in 2021, thanks to its huge success since its launch in 2014. The idea of such an event was presented to the Department of Physics and the University of Trento by three physics PhD students and was well received. The three PhD students formed the "Scientific Committee" of IPSP which has been an important element in this initiative until today. To encourage a vast participation and to provide more credit for the initiative, this edition was directed by three physics researchers who had the role of tutor for each team.
IPSP is composed of problems, which are proposed by companies, and participants ("brains"). For one full-time week, the brains (master or PhD students, post-docs, and researchers) try different ideas and various approaches to solve the problems and on the final day of the event present their solutions. The proposed solutions have been helpful and practical in all editions. The strengths of IPSP include the efficient format, the presence of young brains on the one hand and experiences of the companies on the other one and lots of team spirit. In addition, it has successfully promoted the young physicists (scientists) in industries and paved the path for more collaboration with them through industrial projects, doctorate scholarships and traineeship possibilities for undergraduate and graduate students. It has also been an exemplar for some similar initiatives organized by other departments of the University of Trento or by innovation-oriented organizations.
This book contains the proceedings of IPSP2021 and describes in detail the activities carried out by the three teams. It is organized in three chapters, each of which begins with a brief overview of the company and a description of the proposed problem, followed by the undertaken strategies, the technical procedures, and the final solutions. At the end of each chapter suggestions and conclusions are presented.

Mattia Mancinelli, Michele Orlandi, Luca Tubiana

CHAPTER

ONE

PHYSICAL BASED SIMULATION OF A REAL-TIME LIDAR SENSOR WITHIN A RENDERING ENVIRONMENT BASED ON UNREAL ENGINE 4

D. Bazzanella, S. Bontorin, F. Callegari, B. Degli Esposti, S. Rabaglia, L. Wolswijk, and Alessandro Zunino

Tutor: Mattia Mancinelli

1.1 Introduction

1.1.1 The company

AnteMotion [1] is a joint venture of EnginSoft Spa (Italy), LHP Engineering Solutions Inc (USA) and V2R Srl (Italy), three realities of the Automotive Industry R&D, whose skills converge in AnteMotion in order to satisfy the growing simulative needs of the field. Their core business is focused on creating virtual environments through a Real-Time Render Engine for training, validating and assessing the functional safety of both Virtual Drivers and ADAS Systems (Advanced Driver Assistance Systems). The simulation framework is real-time oriented and exploits a Real Time Engine (Unreal) developed for gaming that provides state of the art rendering quality of dynamic environments in virtual reality and dome projection.

1.1.2 The problem

The success of autonomous driving depends on how accurate sensors and software used by the car are. The software is usually based on Artificial Neural Network (ANN) whose aim is to interpret the data coming from a multitude of different kinds of sensors and produces a decision. The sensors can be camera, radar, proximity sensor and LiDAR (Light Detection and Ranging). The accuracy of ANN in general depends on the data set dimension and quality. A good data set should contain 'all' the possible scenarios that the network will manage during

the operation. Once trained the network is tested on different data from the one used for the training. AnteMotion (AM) is capable to provide a platform to test (and eventually train) the accuracy of these ANN by generating virtual data sets. The problem submitted to IPSP 2021 is framed on these concepts. In particular, AM asked to accurately model a LiDAR sensor within the Real-Time Engine framework used to render the virtual reality. A physical model of a LiDAR is required to produce real data from the virtual reality used to test the accuracy of ANN used in the autonomous driving software. A wrong model would produce fake data from the driving session that in turn would give an incorrect judgement of ANN accuracy.In addition, AM requested the model to be lightweight, as their intention is to run it in real time. For this reason, simulation performance and physical accuracy should be balanced in order to satisfy both requests.

1.1.3 Problem solving approach

This section is intended as a guideline for the reading of the whole document: its goal is to show how the problem has been addressed and how the different activities have been carried out by the team. Section 1.2 reports the state of the art of the automotive LiDAR sensors: the different types, the working principles and operating specifications of actual devices on the market have been reported. Then, the document contains a description of the physical phenomena and models implemented in the solution proposed to the company: Section 1.3 explains the equations involved in the different scenarios and the assumptions introduced to perform a realistic simulation. The software and the virtual tools used to implement the proposed solution are shown in Section 1.4: further, it contains also a detailed description of the structure of the algorithms and in the communication between the different layers of the software. Section 1.5 shows the results obtained by the implementation of the physical models in our solution.

1.2 Existing LiDAR devices

1.2.1 Types of LiDAR

LiDAR devices exploit light to identify target positions and map their distances. Typically, a pulsed illumination is used to determine such distances, by measuring the time of flight (ToF) between the emission and the detection of the light pulse. However, also interferometric approaches, relying on the phase detection of a modulated light signal, are widely adopted.
In the automotive sector, LiDARs are fundamental elements for autonomous driving, as they are responsible for scanning the environment around the vehicle. To do so, autonomous driving car manufacturers have adopted four main scanning technologies so far: rotational or spinning, MEMS-based, FLASH and Optical Phase Array (OPA) LiDAR sensors.

Spinning LiDAR Sensors In this kind of device, a set of laser emitters and detectors are assembled in a rotational stage. This configuration relies on

the fast spinning (few tens of Hz) of the sensor to have a 360° scanning of the surrounding area. Thus, a full horizontal field of view (HFOV) can be achieved with this device. The number of emitters/receivers and their orientation determines then the Vertical Field of View (VFOV) and the angular resolution of the illumination.

MEMS LiDAR Sensors By applying the proper voltage signal, MEMS (Micro Electronic Mechanical Systems) can transduce tiny displacements. So, the steering of miniaturised mirrors fabricated on MEMS allows scanning of a certain area, when they are combined with laser beams and detectors. Typically, these sensors do not provide huge FoV but the simultaneous use of many of them provides a good scanning coverage.

FLASH LiDAR Sensors Similarly to a flash camera, FLASH LiDARs use a wide-area laser pulse to illuminate the environment in front of it and an array of photodetectors captures the back-scattered light. Although the FoV is limited also in this class of devices, the data capture rate is much faster.

Optical Phase Arrays (OPA) LiDAR Sensors The OPA principle is based on the control of the optical wave-front shape to control the direction of the illumination beam, so no moving mechanical parts are required. By properly applying different phase shifts to each light emitter, the overall light distribution is effectively steered to point in different directions.

1.2.2 Spinning LiDAR : working principle

Our solution carries out a simulation of a spinning LiDAR sensor. This device allows to determine the distance of an obstacle (also called target) by measuring the delay, usually called Time Of Flight (TOF), between the time at which the light source of the LiDAR emits a light pulse and the time at which the light gets back at the detector, after being reflected or backscattered by the target.
The distance between the LiDAR and the obstacle is usually called range R and is proportional to the time of flight t_{TOF} [2]:

$$R = c\frac{t_{TOF}}{2}, \tag{1.1}$$

where c is the speed of light, which is constant under the assumption that the propagation occurs within the same medium (air). The factor of 2 takes into account the fact that the light travels forth and back.

The maximum measurable range would be theoretically limited only by the maximum time interval that can be measured by the time counter. In practice, being this time interval large enough, the maximum range becomes limited by other factors, which are mainly related to the energy losses during the propagation and reflection of the light pulse. The intensity of the light that reaches the detector depends on the power of the LiDAR laser source, on the losses due to the propagation through the air and on the fraction of light that is reflected back by

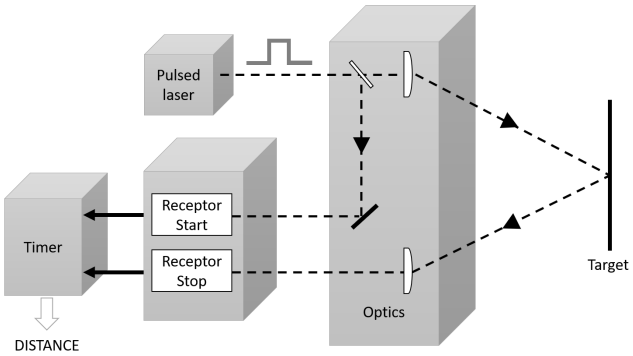

Figure 1.1: TOF measurement principle, image from [2]

a given object reaching the detector's sensitive area. The maximum detectable range results then to be ultimately limited by the signal-to-noise ratio (SNR) of the detector, in terms of the minimum light intensity of the return signal that can be distinguished from the electronic noise of the high-bandwidth detection circuit. To identify the maximum range, another aspect to be considered is the ambiguity distance (the maximum range which can be measured unambiguously) [2], which in the pulsed approach is limited by the presence of more than one simultaneous pulse in flight, and thus related to the repetition rate of the laser. From equation 1.1 the ambiguity distance results of the order of 150 m for a laser with a repetition rate close to the MHz range.
Since the pulsed principle is based on the direct measurement of the round trip time between light pulse emission and the return of the pulse-echo resulting from the backscattering from a target object, the light pulses need to be as short as possible (usually a few ns) and with large optical power in order to obtain a higher SNR.
The peak laser power that can be used is determined by laser safety regulations, which quantify the maximum optical power for a given exposure time [3]. The use of pulsed lasers with short pulse duration allows to remain in the class 1 laser-safety class (where no safety glasses are required) while exploiting high peak powers, up to the order of 500 W for pulse duration of a fraction of ns. Such large irradiance power, together with the use of wavelength-selective optical filters, allows to neglect the solar background, making pulsed LiDARs particularly suitable for outdoor applications as in the automotive field.
An optical power of this order of magnitude might seem large, but one must consider that only a fraction of the optical energy of the emitted laser pulses is received back at the detector, especially if the target surface is, as in most cases, diffusive, which scatters the incoming light in multiple directions. This means that very sensitive detectors are needed in order to detect the faint pulses received from distant diffusive objects. In our simulation we considered a peak power of 100 W, but this input parameter can be easily changed to account for different light sources.
Another important aspect is the range resolution, i.e. the minimum range difference

that can be resolved, which is related to the bandwidth of the detector. The range resolution can change significantly from low cost to more expensive LiDARs , with typical values going from 20 cm [4] to 2-3 cm [5]. This corresponds to minimum time resolutions of the detector $\Delta t = 2\Delta R/c$, respectively of 1.3 ns and 0.2 ns. The bandwidth of the detector must therefore be at least of the order of the GHz. Knowing the required bandwidth, we can estimate the SNR that limits the maximum measurable range. Considering, for instance, the noise equivalent power of a high quality detector [6], $NEP \sim 6.6$ pW$/\sqrt{Hz}$ we can estimate the noise threshold as: $P_{noise} = NEP\sqrt{BW} \sim 0.2\mu W$. In Subsection 1.3.3 we explain more in detail how this estimation of the noise threshold is implemented in our physical model.

1.2.3 Spinning LiDAR : performances

In the next table we report a list of typical parameters and performances available with spinning LiDARs :

	min	max
range	100 m	250 m
range accuracy	/	few cm
Number of emitters/receivers	16	128
HFoV	/	360°
Res. HFoV	0.08°	0.4°
VFoV	20°	40°
Res. VFoV	0,1°	2°
Laser Wavelength	800	1550 nm
Hazard Class	/	Class 1 eye safe

Table 1.1: Typical specifications of rotational LiDARs .

Our simulation is based on the parameters of the AlphaPrime (Velodyne) LiDAR sensor [7]. We used its datasheet to obtain the working parameters and performances required to implement our model. Their values are the following:
Each channel is associated to an emitter-receiver pair: the scattering from a laser beam emitted at a certain angle is coupled to its relative photo-detector through the proper optics. So, 128 laser beams sample the space in the vertical dimension, while the rotation of the sensor provides the 360° HFoV. The angle between the emitted beams determines the Vertical resolution and FoV. As shown in Table 1.2, the VFov is asymmetrical, because the sensor is typically located on the top of a car: a better covering of the negative angles makes the sensing of the regions at human height more accurate.
The light sources used for LiDARs are chosen considering a trade-off between several features, such as peak power, pulse repetition rate, pulse width, wavelength, size, weight, power consumption, shock resistance, operating temperature, cost and availability on the market.

Channels	128
Measurement range	300 m
Range accuracy	3 cm
HFoV	360°
Res. HFoV	0.01° to 0.4°
VFoV	40° (-25°; +15°)
Res. VFoV	0.11°
Frame Rate	5-10 Hz
Laser Wavelength	905 nm
Hazard Class	Class 1 eye safe

Table 1.2: Reference specifications of the Velodyne AlphaPrime Spinning LiDAR .

Typical wavelengths are in the infrared region, in the range from 800 nm to 1550 nm, to take advantage of the high transmission window of the atmosphere, and in particular of water. The most commonly used wavelengths are mainly in three regions: a waveband from 800 nm to 950 nm, dominated by diode lasers combined with silicon-based photodetectors, fiber lasers with a wavelength around 1064 nm and lasers at 1550 nm, available from the telecom industry, which need InGaAs detectors.
Currently the most popular sources for use in automotive LiDAR technology are solid-state lasers and diode-lasers in the near infrared [2], [3].
In the program we developed, we considered a source at 905 nm, which is one of the most common wavelengths, but it is possible to modify this value.

1.3 Physical models

The model that was initially implemented in the software provided by Antemotion considered only the geometric intercepts of the light beams emitted by a rotational LiDAR with the surfaces of the various objects in the surrounding simulated environment. We implemented a physical model that takes the following aspects into account:

- a more accurate model of the interaction between the laser beams and the surfaces, taking into account reflection, diffusion and back-reflection,
- material-dependent interactions,
- atmospheric absorption,
- detection intensity threshold and system noise.

We initially developed a single-ray model, as done in the ray optics approach [8], where each laser beam emitted by the LiDAR is a single geometrical ray. Then we moved on to richer physical models that also take into account the divergence of the laser beams (Subsection 1.3.5) and the occurrence of multiple scattering events (Subsection 1.3.6).

1.3.1 Model of the laser-surface interactions

Urban environments usually contain a variety of surfaces, each of which interacts differently with a laser beam. Our work summarises this large variability to three types of interactions, which cover effectively the vast majority of all possible light-surface interactions.

- The first type is the diffusive material, which is found everywhere in opaque, non-metallic surfaces, e.g. walls, pavement, and wood surfaces.
- The second type is the reflective material, which is associated to either metallic surfaces, e.g. mirrors and metallic paint, or transparent materials, e.g. glass surfaces, in which part of the light is refracted and part is reflected.
- The last type is the back-reflecting material, mostly found on traffic signs and pedestrian crossings, which are often covered with a retro-reflecting surface, to increase their visibility.

The geometrical quantities of a laser beam interacting with a surface are defined as shown in Figure 1.2:

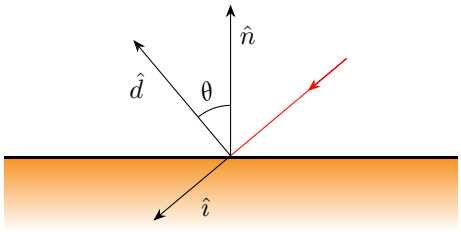

Figure 1.2: Geometrical description of the ray-surface interaction. The incoming laser beam has direction $\hat{\imath}$, the surface is described by its normal $\hat{n}$, and the direction toward the detector is given by the versor $\hat{d}$.

We call r the distance between the laser source and the surface and θ the angle between the direction towards the detector $\hat{d}$ and the unit vector normal to the hit surface $\hat{n}$. In our work we assumed the detector to be in the same position of the emitter. Therefore, $\hat{d} = -\hat{\imath}$, where $\hat{\imath}$ is the direction of the incident ray of light. Thus, the angle is simply $\theta = \arccos(-\hat{\imath} \cdot \hat{n})$.

Diffusive materials

In case of diffusive materials, the incoming optical power is distributed on half the full solid angle, i.e. for θ from $-90°$ to $90°$. Specifically, the fraction of returned power is maximum for surfaces perpendicular to the incoming ray, as shown in Figure 1.3, and decreases quadratically with the distance from the source:

$$I_D(r, \theta) = \frac{\cos\theta}{2r^2}. \tag{1.2}$$

In order to satisfy the energy conservation law, the following equation holds:

$$\int_{-90^\circ}^{90^\circ} I_D(r,\theta)d\theta = \frac{1}{r^2}. \tag{1.3}$$

where the $1/r^2$ term describes the divergence of the back-scattered light.

Reflective materials

In case of reflective materials, the incoming ray is reflected with the same incidence angle, but opposite to the normal to the surface, as shown in Figure 1.3. However, real materials might scatter light also at different angles, mainly because of surface roughness. In order to take this into consideration, we modelled the distribution of reflected rays with a Gaussian curve centred at the reflection angle $-\theta$. In formula:

$$I_R(\theta;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-2\left(\frac{\theta}{\sigma}\right)^2\right), \tag{1.4}$$

where σ is the angular width, specific to each material.
Similarly to the case of diffusive materials, to satisfy the energy conservation law, the distribution is normalized. Indeed

$$\int_{-90^\circ}^{90^\circ} I_R(r,\theta)d\theta \approx \int_{-\infty}^{\infty} I_R(r,\theta)d\theta = 1, \tag{1.5}$$

where the approximation holds because typically $\sigma \ll 90^\circ$.

Retro-reflecting materials

In case of retro-reflecting materials (see Figure 1.3) the ray is reflected backwards with the same intensity of the incoming beam:

$$I_C = 1. \tag{1.6}$$

Realistic materials

The full model of a material is ultimately described by a weighted sum of these three ideal models. The fraction of the initial power that returns back is in total given by:

$$I(r,\theta) = \alpha \cdot I_D(r,\theta) + \beta \cdot I_R(\theta) + \gamma \cdot I_C, \tag{1.7}$$

where α, β, and γ and positive coefficients for the individual models. In order to respect the conservation of energy, the following constraint holds

$$\alpha + \beta + \gamma \leq 1, \tag{1.8}$$

where the equality is observed when no loss is present.

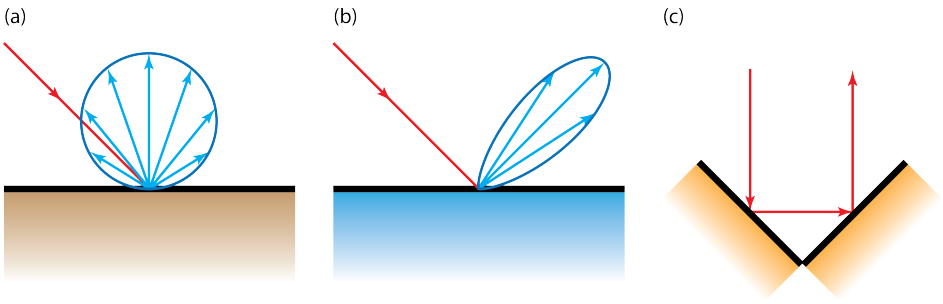

Figure 1.3: Visual description of the behaviour of three different ideal materials. (a) Diffuser, (b) reflector, (c) retroreflector.

1.3.2 Atmospheric absorption

Atmospheric absorption along the optical path can be modelled with the Lambert-Beer law,

$$P(2R) = P(0)e^{-2A_{air}R} \tag{1.9}$$

where $P(0)$ is the optical power in absence of absorption, $2R$ is the total optical path travelled by the laser beam in air and A_{air} is the wavelength-dependent atmospheric absorption coefficient. This coefficient weakly depends also on the atmospheric humidity and increases in presence of fog by a factor of ~ 200 [9], [10]. A more accurate description of the effect of rain and fog may be obtained considering the multiple scattering on the water particles encountered by the light beams [11], as well as the backscattering from precipitation and diffraction effects. In presence of sun, one should also consider the interference of the solar light with the return signal. An accurate modelling of all these effects goes beyond the scope of this work. For the sake of computational efficiency (the simulation should run in real time), we decided to limit the modelling of the atmospheric conditions to their average effect described by the A_{air} coefficient in equation (1.9).

1.3.3 Noise threshold

We set a threshold on the minimum optical power of the return signals that can be detected by the LiDAR . The return power for a given light beam will be a fraction I of the peak optical power P_{peak} of the initial lightpulse emitted by the LiDAR . This fraction is determined by the interaction with the surface of the target, as described in Subsection 1.3.1, and attenuated by atmospheric absorption as explained in Subsection 1.3.2.
Moreover, a real detector has a limited efficiency ε, determined by various factors, for instance the losses along the optical path inside the device. For this reason only a percentage ε of the light that arrives at the LiDAR is actually detected. The actual detected return power will thus be given by

$$P_r = I\ P_{peak}\ \varepsilon. \tag{1.10}$$

In Subsection 1.2.2 we estimated the optical power corresponding to the detector noise as

$$P_{noise} = NEP\sqrt{BW}, \tag{1.11}$$

where NEP is the noise-equivalent-power and BW the detector bandwidth.
To treat the stochastic noise on our detected signals, we add, on top of our return signals P_r, a Gaussian noise distribution P_G, having standard deviation P_{noise}. The actual detected power becomes

$$P_{out} = P_r + P_G. \tag{1.12}$$

Setting a detection threshold P_{min} of three standard deviations, we obtain that a return signal is detectable if:

$$P_{out} > 3P_{noise} = P_{min}. \tag{1.13}$$

1.3.4 Materials database

We built a database containing several materials typical of an urban scenario with their light-surface interaction coefficients: α for the diffusion, β and σ for the reflection, and γ for the retro-reflection, as introduced in Subsection 1.3.1. Specifically, we can divide the materials in two groups: the ideal materials and the more realistic materials.
The first set of materials includes `default`, a material used for surfaces of unspecified material, `diffuser`, `glossy`, and `reflector`, which individually

material	ID_{mat}	α	β	γ	σ
default	ideal0	NaN	NaN	NaN	1
diffuser	ideal1	1	0	0	1
glossy	ideal2	0	1	0	5
reflector	ideal3	0	0	1	1
test	ideal4	0.1	0	0	5
retroreflector	1	0	0	1	1
opaque metal	2	0.01	0.85	0	5
lucid metal	3	0.01	0.98	0	5
glass	4	0	0.01	0	2
rubber	5	0.17	0.10	0	15
asphalt	6	0.09	0.01	0	30
stripes	7	0.30	0.10	0.5	30
concrete	8	0.15	0.10	0	10
wood	9	0.06	0.20	0	30
rock	10	0.13	0.05	0	30
green vegetation	11	0.04	0.20	0	40
non-green vegetation	12	0.05	0.10	0	40

Table 1.3: Materials coefficient table. "NaN" means "Not a Number".

implement the diffusive, reflective, and retro-reflection effects, and a `test` material, used for modelling a flat surface diffusing 10% of the incoming optical power.
The second set of materials contains the main materials that can be found in an urban scenery, among which `asphalt` and `stripes`, which we used to model a road surface, `glass`, `rubber` and `metal`, which we used to model cars, and other materials such as `wood`, `rocks` and `vegetation`.
Since we did not have the possibility to perform reflectivity measurements directly, we looked online for typical reflectivity coefficients of the different materials, in order to obtain a realistic scenery. In [12] the authors describe an experimental procedure to obtain the reflectivity and diffusion coefficients for different materials, using a laser source and measuring the back-scattered light at different angles. They also provide the experimental values for some materials. Other coefficients were obtained using as reference the NASA Ecostress Library [13].
The optical properties of a material depend on the wavelength. We considered a LiDAR operating at a wavelength of 905 nm, for which the coefficients are reported in Table 1.3. Each of these materials is contained in an ordered dictionary and is accessed via a positive integer index ID_{mat}.

1.3.5 Spot divergence model (Roses)

The first change to our single ray model is the inclusion of the spot divergence. Each laser beam is modelled as a 3D cone, so that the width of the beam is a linear function of the distance from the corresponding LiDAR emitter (see Figure 1.4). The divergence Θ of the beam is defined as the angle between the axis of the cone and its surface. Typical divergence of laser diodes is of the order of 1 mrad ($\approx 0.057°$) [7]. Hence, the diameter of the laser spot at a distance L can be evaluated as $d(L) = 2L\sin(\Theta)$. At 50 m and 100 m the spot diameter will be $d|_{50\,\text{m}} \simeq 5\,\text{cm}$ and $d|_{100\,\text{m}} = 10\,\text{cm}$, respectively. This means that, in some cases, the LiDAR will detect a signal reflected from the edge of the beam and not from the centre.
To take this effect into account, we discretised each beam into $N_{roses} = 1 + n$ rays: one in the centre and the remaining n distributed on the beam's conical surface. The number of rays in each *rose* has been fixed to $N_{roses} = 5$, but in principle can

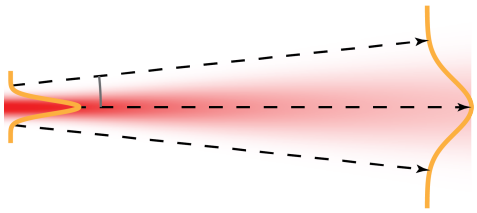

Figure 1.4: Sketch of the modelling of beam divergence. The Gaussian beam is sampled by multiple rays of light with an angular separation defined by wave optics theory.

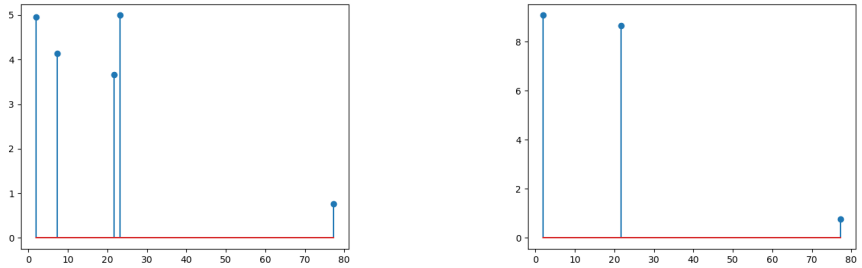

Figure 1.5: Sketch of the modelling of temporal resolution limit of the LiDAR . (a) Ideal response of the detector in presence of multiple returns, (b) Realistic response of the detectors. Returns with a time of flight difference below the temporal resolution are merged and seen as a single return with higher power.

be changed according to the accuracy target of the model. We nicknamed this model *roses* because the distribution of rays vaguely reminds us of a rose's petals (but mostly because we needed a memorable name).
Each ray is then propagated until the first surface interaction just like in the single-ray model. The returned power hence becomes a sequence of pulses with different optical power arriving at different times. These signals have to be distinguished or bundled, depending on the temporal resolution of the detector. In any condition, the model keeps as returning angle that of central ray.

Temporal resolution

After increasing the complexity of our model by discretising each beam with more than one ray, the detector might receive several impulses for a single sample. 1.5 shows two examples of incoming pulse trains.
In a train of pulses, the most important is the first one that exceeds the detection threshold, which should be well above the noise floor. However, LiDAR detection systems have an intrinsic temporal resolution. This means that, if two optical pulses are received in a time interval shorter than the temporal resolution of the instrument, those are seen as a single pulse with higher total power.

1.3.6 Multiple reflections model (Guns)

Another improvement on the basic single-ray model is the introduction of secondary interactions of reflected beams on surfaces which do not cross the direction of the original laser beam from the LiDAR . Ideally a perfectly reflected beam has no components returning directly to the detector (unless the normal to the surface coincides with the direction of the beam) and the surface is therefore not detected. The modification introduced in this section consists in modelling any secondary interactions that the reflected beam may have with the environment. If these

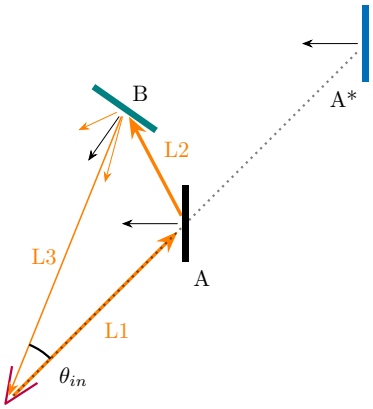

Figure 1.6: Graphical model of the detection of secondary interactions: The laser ray from the LiDAR is directed towards surface A (ideal reflective surface in this example) which is placed at a distance L1. The secondary interaction occurs when the reflected ray, which is not returned to the detector, interacts with a surface B (possibly diffusive), placed at a distance $L2$ from A as described in the graphical model. Part of the rays diffused by B return to the detector covering a distance $L3$ with an input angle θ_{in}. The resulting optical path associated to the original laser ray detected is therefore $L = L1 + L2 + L3$

interactions occur with a diffusive surface, this surface can diffuse components back to the detector.

A simplified scheme of the physical process is presented in Figure 1.6. The LiDAR is pointed to the reflective surface A, and the laser beam is detected after a ray is diffused from surface B after a secondary interaction. Following the notation in the figure, instead of following an optical path equal to $2L1$, the optical path followed has instead a length $L = L1 + L2 + L3$. If the incoming ray has an incidence angle $\theta_{in} <$ Res. HFoV, i.e. which does not exceed the LiDAR angle resolution, and more importantly has a power which exceeds the power threshold ($P_{out} > P_{min}$), the resulting signal can be detected. The LiDAR therefore detects, in the direction it is pointing, a surface (A*) with an associated time of flight $t_{\text{TOF}} \propto L/c$ and an associated range R_{A*} which overestimates the physical distance $L1$ of surface A from the detector ($R_{A*} > L1$).

The take home message is that the beam directed towards the surface A is returned to the detector through secondary interactions with the environment, which ultimately lead to a wrong estimate of the range R of the initial reflective surface. Implementing this process correctly in the physical modelling of the LiDAR detection process is of great importance in order to reproduce real physics features in a virtual environment and to eventually be able to train autonomous driving systems with physics-consistent scenarios. We are aware that the process described in Figure 1.6 is a crude approximation if compared to the large array of secondary interactions that occur. Under the rationale that most secondary interactions from diffusive surfaces do not exceed the power threshold due to surface and atmospheric absorption along the optical path, this addition in the model represents a relatively fine improvement to the single ray detection model.

Algorithm 1 A pseudocode description of `pointcloud_physics_reflections()`

```
set initial parameters
load input data structure
for beam in beams do
    i ← 0
    while i < N_ref do
        get physics information
        if material is a diffuser then
            propagate to detector
            compute surface interaction and absorption
            add noise
            if P_out > P_min and θ_in < resolution then
                store the beam
            else:
                store dummy values and continue to the next beam
            end if
            i ← i + 1
        else material is a reflector:
            compute surface interaction and absorption
            compute relative distance to next N_ref secondary reflection points
            compute the forward power propagated to next point in path
            i ← i + 1
        end if
    end while
end for
return output data structure
```

We nicknamed our multiple reflections model *guns* because the laser reflection phenomenon reminds us of bullet ricochet.

Description of secondary reflections algorithm

In this subsection a brief description of the implementation of the *guns* algorithm is presented. The function `pointcloud_physics_reflections()` takes as input a tensor of size $N_{\text{ppf}} \times N_{\text{ref}} \times 8$, with N_{ppf} being the number of optical paths to be processed, N_{ref} being the maximum allowed number of reflections and 8 being the size of the vector

$$(\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z, \boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z, ID_{\text{mat}}, \theta)$$

that contains physical information about the surface interactions (for more details, see Subsection 1.4.5). A pseudocode description of the function is given in Algorithm 1. The output data structure is a 3D point cloud with a scalar P_{out} (the optical power) associated to each point. We stress that this model returns a single point for each optical path. The difference between this point and the LiDAR 's position has the same direction as the first segment of the optical path, but a possibly greater distance (because the distance depends on the total time of

flight and the interaction with diffusive and retro-reflective materials). The power P_{out} depends on all the surface interactions that occurred in the optical path.
In the current implementation, the algorithm does not evaluate the possibility of a secondary reflection from a pure reflector exactly back to the detector (except for the case of retro-reflectors which is handled separately). All the data is generated from secondary reflections that meet diffusive materials. We remark that the guns model is based on solid physical principles, but still requires additional validation and testing.

1.3.7 Implementation of the physical models

We implement the physical models as a Python module. The main function is `pointcloud_physics()`, whose inputs are NumPy arrays: the rays' position vectors, surface normal's vectors, and material index vectors.
The first operation is to extract from position and normals the distance of the surface from the LiDAR and the angle of incidence on the surface. Secondly, the material parameters corresponding to each material index supplied have to be collected from the material database, which is an ordered dictionary. These two operations are carried out by the auxiliary function `data_slicing()`, so that the correct dimensions of the NumPy arrays are respected, especially when each beam is sampled multiple times (*roses* and *guns*).
The main functions then evaluate the effect of the surface interaction and the atmospheric absorption on the power returned to the detector, employing `surface_interaction()` and `atmospheric_absorption()`, respectively. The returned power is then elaborated by the function `intensity_processing()`, in order to carry out the power thresholding and the noise addition.
The output of `pointcloud_physics()` are the position and norms of the surface detected by the LiDAR along with the intensity of the light pulse. In the case of both *roses* and *guns* models, the position is given by the direction of the central sample and at the distance evaluated by the model. At this stage, our script is able to process around 1×10^6 rays per second in the single-ray model and 200×10^3 rays in the *roses* model, as each beam is sampled $N_{\text{roses}} = 5$ times.

1.4 Implementation of a virtual LiDAR

1.4.1 From a geometrical LiDAR to a physical LiDAR

AnteMotion's simulation platform is based on Unreal Engine 4 (UE4), a game engine developed by the software company Epic Games. Unreal Engine 4 is a highly modular engine, and the simulation logic can be extended either in C++ or in Blueprint, a custom visual scripting language.
As a starting point for our project, we were given by the company a C++ module for Unreal Engine 4 that contained the logic for a basic LiDAR device, which we will refer to as *geometrical* LiDAR . This virtual device performs ray casting in the surrounding virtual environment, but no physical processing after a collision is found. One of the main tasks during IPSP was to extend the geometrical LiDAR to a *physical* LiDAR by implementing the physical model described in Section 1.3. This is a crucial step towards the ultimate goal of reproducing the data collected by an actual LiDAR in a real-life scenario.
Every time that the C++ source code in the LiDAR module is changed, the whole module needs to be recompiled before the simulation can be run again. For this reason, most of the LiDAR properties like the rotation frequency f or the number of channels N_{channels} are not hard-coded, but are instead exposed to the Unreal Engine scene editor GUI so that they can be changed with minimal effort. From a technical point of view, the LiDAR implementation consists of a `ULidarComponent` class. It is in this class that the following essential properties are defined:

- `bEnable`, a Boolean flag that turns the LiDAR on and off
- L_{FoV} and U_{FoV}, the lower/upper field of view of the LiDAR (in degrees)
- N_{channels}, the number of vertical rays that uniformly subdivide the range $[L_{\text{FoV}}, U_{\text{FoV}}]$.
- `range`, the maximum range of the LiDAR in metres (in UE4, rays are approximated with segments)
- f, the frequency of rotation of the laser emitter
- H_{res}, the resolution (in degrees) with which the rotation process is discretized during the simulation
- `bDebug`, a Boolean flag that turns the visualisation of the rays on and off for debugging purposes. Rays are drawn as green lines.

The geometrical LiDAR code performs the intersection between each ray and the world through UE4's built-in function `LineTraceSingleByChannel()`. This function fills a struct of type `FHitResult` that contains essential ray casting information such as a Boolean variable `bBlockingHit` that tells us whether the ray has hit any object at all and a 3D vector `ImpactPoint` that gives us the location of the first hit point (if it exists, i.e. `bBlockingHit` is `true`). In the geometrical LiDAR code, no other information is extracted from the `FHitResult` struct. The position $\boldsymbol{p}$ of the hit point is stored as a triple $(\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z)$ of single-precision floating-point numbers.

For more information on Unreal Engine 4's C++ API, we refer to the official documentation at `https://docs.unrealengine.com` [14].

Surface normal information

The physical model introduced in Section 1.3 requires the surface normal $\hat{\boldsymbol{n}}$ at every hit point. Just like the location of the hit point, this information can be extracted from the `FHitResult` struct that Unreal Engine 4 returns from `LineTraceSingleByChannel()`: it is in the `ImpactNormal` field. Every normal $\hat{\boldsymbol{n}}$ is stored as a triple $(\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z)$ of single-precision floating-point numbers and is, of course, a unit vector.

Physical material information

The physical model that we implement for the LiDAR simulation is based on the study of the interaction between light and materials. Therefore, it is necessary that the program provides also the information on the materials of the hit surface, in addition to the coordinates and surface normal of the impact point. We have solved this problem in the following way.

First, in Unreal Engine 4's editor we have defined 12 custom surface types (see Fig. 1.7) that correspond to the materials considered inside the physical model (see Table 1.3). The number that identifies the surface type is an arbitrary integer identifier ID_{mat}. Second, we have created a custom *physical material* (using UE4's terminology) for each surface type. For example, the `Dark_asphalt` surface type ($ID_{\text{mat}} = 6$) was assigned to a newly created asphalt physical material. Then, the right physical material was assigned to every part of every 3D model in the scene (some models, like cars, have more than one material). Third, we have enabled the detection of the physical material during ray casting by setting both `bReturnPhysicalMaterial` and `bTraceComplex` to `true` in the `FCollisionQueryParams` struct that is passed to every call of `LineTraceSingleByChannel()`. Finally, we have extracted the surface type ID of the material at every hit point by reading the `PhysMaterial.SurfaceType` subfield of the `FHitResult` struct.

In Unreal Engine 4, a static mesh is associated to each object and, based on the collision complexity, it can be either simple or complex. Simple collision is against primitives like cubes, spheres, capsules, and convex hulls, while complex collision is against the complete triangle mesh of a given object. If we use simple collisions, we can associate only one physical material to each object, because the program sees the object as a single homogeneous geometrical shape. With complex collisions, however, every object can be composed by multiple parts, and we can associate a different physical material to each of them. The only drawback of complex collisions is that they have a higher computational cost. Therefore, we have enabled complex collisions only for a small number of objects that we consider essential for our simulation.

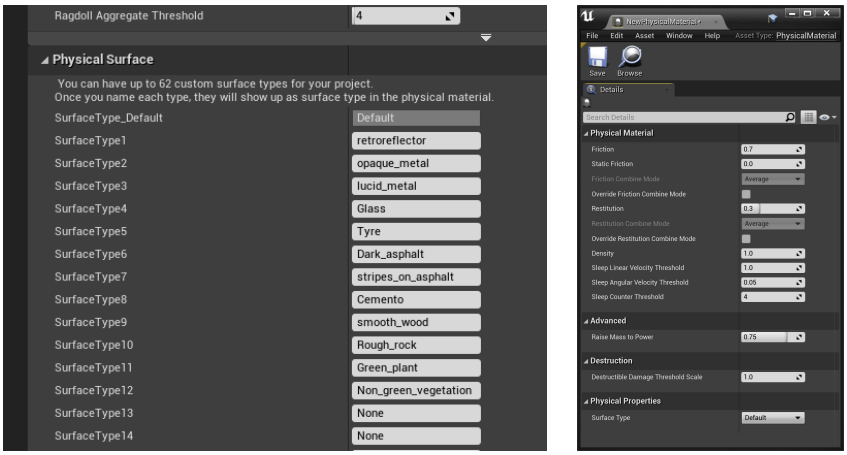

Figure 1.7: UE4's editor settings related to surface IDs and physical materials.

1.4.2 Data flow in and out of the physical model

In computer graphics, the *frame rate* (expressed in frames per second or FPS) is the frequency at which virtual environments are updated and consecutive images (frames) are displayed after rendering. Unreal Engine 4, unless instructed otherwise, tries to compute as many frames as possible every second in order to have smoother animations, more accurate simulations and to react more quickly to external events or user inputs. Some frames may be more expensive to compute than others, so the time required to compute each frame is a variable quantity Δt. The quantity Δt is on average equal to the reciprocal of the FPS, and is used as the time step of the animations and of the game physics.
Let $\omega = 360^\circ f$ be the angular velocity of the laser emitters of a spinning LiDAR device. After a time Δt, the emitters rotate by the angle

$$\Delta\varphi = \omega \Delta t.$$

In our simulation, we want to cast N_{channels} rays every time that the emitters rotate by H_{res} degrees, so the total number N_{rpf} of rays per frame is

$$N_{\text{rpf}} = N_{\text{channels}} \frac{\Delta\varphi}{H_{\text{res}}}.$$

Of course, there is no guarantee that $\Delta\varphi$ is an integer multiple of H_{res}; in practice we solve this problem by rounding down the ratio of two angles. By taking H_{res} to be independent of $\Delta\varphi$, we effectively decouple the number of acquisitions per second of the simulated LiDAR device from the frame rate of the simulation.

Input data structures

We have seen that, for every frame, N_{rpf} rays are cast from the LiDAR emitters. In real life, multiple channels work in parallel, but in a software simulation it makes

sense to serialise every vertical scan by simulating the movement of a single emitter (called `LidarSpanner` in the source code). During a vertical scan, the emitter will move in N_{channels} uniform steps from the angle L_{FoV} to the angle U_{FoV}. The vertical resolution of the device is therefore

$$V_{\text{res}} = \frac{U_{\text{FoV}} - L_{\text{FoV}}}{N_{\text{channels}} - 1}$$

After each vertical scan, the LiDAR emitter rotates horizontally by H_{res} degrees. This process fixes an order in which rays are cast.
As explained in the previous subsection, the physical model of Section 1.3 takes as input three different kinds of data: the location of the hit point, the normal vector to the surface at the hit point and the material ID of the surface. This data is encoded as a vector of 7 single-precision floating-point numbers

$$(\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z, \boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z, \mathit{ID}_{\text{mat}}).$$

The material ID is actually an integer, but we have chosen to encode it as a float in order to keep the data types homogeneous. We remark that integer arithmetic in the $[-2^{24}, 2^{24}]$ range is exact for single-precision floating-point numbers, so we don't face any rounding problems. If the ray has no collisions with any object in the scene (within `range`), the corresponding data is the conventional vector

$$(1, 1, 1, 1, 0, 0, 0).$$

The position and normal information is arbitrary; what really matters is that the material is set to 0. This is how our physical model knows to ignore the collision information and skip the ray.
For efficiency reasons, our physical model doesn't handle one hit point at a time. Instead, all the vectors for each ray in a frame are stacked in a matrix D_{in}, and it is this $N_{\text{rpf}} \times 7$ matrix that is passed to the physical model.

Output data structures

As we have seen in Section 1.3, our physical model takes in a point cloud and outputs two different kinds of data: the reconstructed position $(\boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z)$ of each point (which will always have the same direction as the input position $\boldsymbol{p}$, but a possibly different length) and the optical power P_{out} of the return signal that reaches the LiDAR sensor.
For reasons that will be explained later, we want to keep the number of columns in the output data equal to the one of input data, so the output is obtained by stacking rows of the form

$$(\boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z, 0, 0, 0, P_{\text{out}})$$

to form a matrix D_{out}. The number of rows of D_{out} is at most N_{rpf}. It can be strictly smaller if the physical model discards any input data (either because $\mathit{ID}_{\text{mat}} = 0$, or because P_{out} is below the detection threshold).

1.4.3 Additional LiDAR properties

Frame of reference

A triple of real numbers has no physical meaning by itself: the correspondence between numbers and quantities such as positions, normals, etc, is established by the choice of a frame of reference (FoR).
In the context of LiDAR simulations, there are two natural choices for a frame of reference: one relative to the ground, and one relative to the roof of the car on which the LiDAR is installed. We refer to these two frames of reference as *ground space* and *car space*, respectively. Moreover, a third frame of reference we have to work with is the one defined by UE4's default Cartesian axes. The position, orientation and scaling of all 3D models in any given virtual scene is expressed with respect to this frame of reference. This is why in UE4's documentation this coordinate system is known as *world space*.
Different frames of reference serve different purposes. On the one hand, the training process of the machine learning algorithms behind autonomous driving requires inputs in car space, since this is by definition the frame of reference of a real LiDAR mounted on the roof of a car. On the other hand, the visualisation of the point cloud data generated by the LiDAR is better done in ground space, as it allows us to coherently overlap data captured at different positions of the LiDAR; as the car travels around a neighbourhood, the point cloud can grow and show us how the sensor maps the surrounding environment over time.
During IPSP, we have added a property to the LiDAR component called `refFrame` that allows users to switch the output data between ground space and car space. We remark that this choice has no effect on our physical model, because the simulation is defined in terms of *intrinsic* quantities like angles, distances, etc, and these are invariant up to rigid motions (like those of a car with respect to the ground).
Ground space and world space are at rest with respect to each other, so it is technically possible to choose world space as our ground space. However, there are a few technical reasons to avoid this identification:

- Unreal Engine units correspond to centimetres, while our physical model expects metres (SI base units in general).

- The origin of world space may be very far from the virtual environment of interest. For example, in the 3D scene that we were provided by Antemotion (see Section 1.5), the main street was placed at $(-1902, -10695, 150)$. This offset would make it harder to visualise the results of the simulation.

- World space is a left-handed coordinate system, but right-handed coordinate systems are preferred in physics. The Python visualisation library Open3D (see Subsection 1.4.4) also expects a right-handed coordinate system.

Therefore, we have chosen to define ground space as the frame of reference whose origin is the location of the car at the beginning of the simulation and whose Cartesian axes x, y, z have the same orientation as the axes y, x, z in world space.

We have swapped the x and y axes on purpose, to get a right-handed coordinate system.
The function `LineTraceSingleByChannel()` returns data in world space, so we had to write code to convert this data to either car space or ground space (depending on the setting of the LiDAR component). The code that implements this conversion can be found in the function `ULidarComponent::parseHit()`.

Linear LiDAR movement

In the geometrical LiDAR code, the device is stationary at run time, and it can only be moved manually when the routine is stopped. LiDARs are used as sensors for self-driving cars, so during IPSP we have added a velocity property to the LiDAR component. Every frame, the new function `moveLidar()` performs the movement of the LiDAR by calling the function `AddLocalOffset()` that increments the position on the y axis by $\Delta t \cdot \boldsymbol{v}_y$. We have implemented the movement only along the y axis, but it is possible to introduce a velocity also along the x axis (we recall that the ground is parallel to the xy-plane).

1.4.4 LiDAR data visualiser

The company provides a program that visualises the simulated signals collected by the LiDAR , based on the Python library Open3D. The program opens a white window with drawn two black points, that are used to set the reference scale. The physical model gives as output a point cloud, composed by the coordinates of the impacts points ($\boldsymbol{q}$), for which the returned light intensity is above a fixed threshold, and the corresponding signal intensities (P_{out}).
The program plots these points using an RGB colour map to associate a colour to each intensity, reported with logarithmic scale (Fig. 1.8). The colour map spans from blue, for less intense signals, to red, for more intense ones. The program plots the point cloud *in real time* and periodically clears the windows to avoid excessive overdrawing. The point cloud can be visualised using the *ground space* or the *car space* reference frames (Subsection 1.4.3), selecting the option through the UE4 interface. Moreover, the user can decide to plot the raw point cloud generated by Unreal Engine 4, with a colour map based on the material IDs (ID_{mat}), instead of the intensities. We introduce this feature to check that associations between object and material were right. In this case the colour map spans from yellow to red. Last, the program gives the possibility to overlap the visualised point cloud with the picture of the virtual world (see Fig. 1.9). In this way, we can more easily validate the implemented physical model.

Data flow

We have seen what kind of data is exchanged between Unreal Engine 4, our physical simulation and the visualiser, but we have not yet described *how* data is exchanged. In the geometrical LiDAR code, communication happened in real time by exchanging floating-point numbers encoded as text over UDP sockets. After evaluating possible alternatives with AnteMotion's developer Mattia Affabris, we

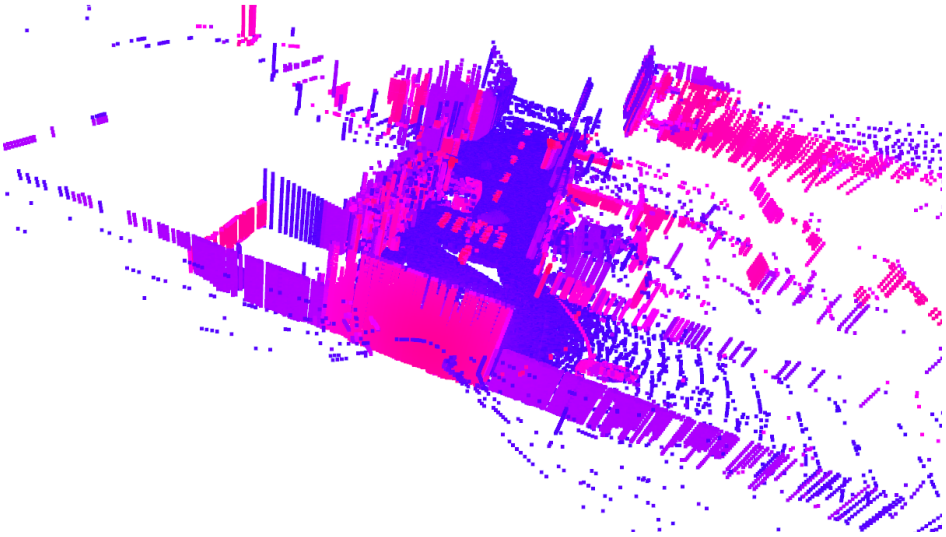

Figure 1.8: Point cloud of the virtual world plotted inside the visualiser (ground space).

decided to keep the UDP sockets infrastructure for its flexibility but switch to binary-encoded floats for additional performance.

As explained in Subsection 1.3.7, our physical model has been implemented as a Python function `pointcloud_physics()` that takes NumPy arrays as inputs and returns other NumPy arrays as outputs. We have therefore written a short Python script `glue.py` that ties everything together. The script does the following:

- Establishes connections over UDP sockets with the Unreal Engine 4 and the visualisation program.
- Reads a batch of binary-encoded rays data and converts it to NumPy arrays using the `struct` library.
- Feeds the arrays to `pointcloud_physics()` and serialises its output into a binary stream.
- Sends the binary stream to the visualiser and prints performance metrics by dividing N_{rpf} by the time required to run the physical simulation.

All of the above happens in real time with an update frequency equal to the frame rate of Unreal Engine 4's simulation. For debugging purposes, the script `glue.py` may also be bypassed: since D_{in} and D_{out} have the same number of columns, the visualiser can easily handle the raw data from Unreal Engine 4. As already mentioned, this feature is very useful to check whether physical materials are assigned correctly to the 3D objects in a given scene.

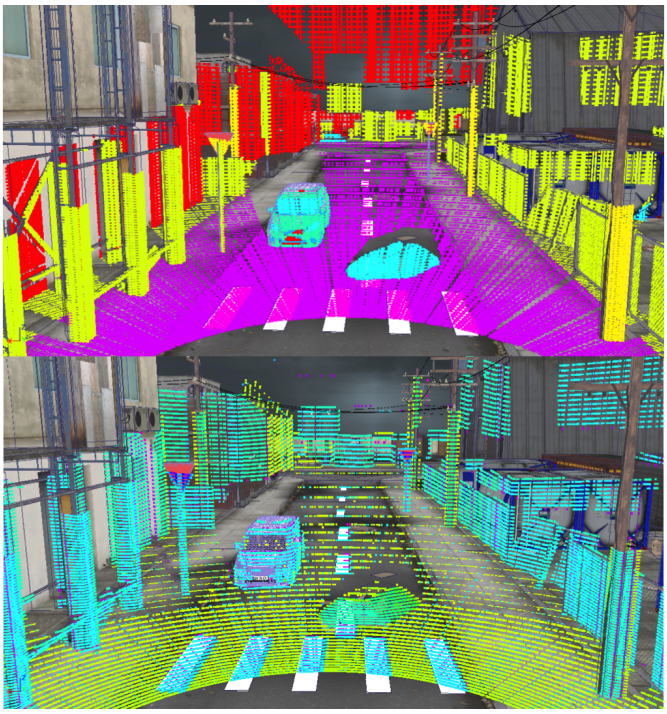

Figure 1.9: Overlay between the UE4 virtual world and the generated point cloud. On top, the point cloud is produced using the geometrical model and the colour map is based on the material IDs. On the bottom, the point cloud is generated using the implemented physical model, and the colour map, that represents low (high) intensities with yellow (blue), is based on the returned intensity of each point.

1.4.5 Data flow for advanced physical models

In this subsection, we explain how the data structures described in Subsection 1.4.2 have to be extended in order to support the richer physical models of Subsections 1.3.5 and 1.3.6.

Data structures for the spot divergence model (roses)

If the basic physical model is extended by taking into account the effects of spot divergence, the simulation of a single laser beam requires multiple rays, following the discretization process outlined in Subsection 1.3.5. As previously stated, we have settled for a 5-rays model, with one ray at the centre of the beam and the other 4 rays arranged in a cross shape around it. The angle between the central ray and any of the surrounding rays is equal to the beam divergence Θ and can be configured by setting the `beamDivergence` parameter of the LiDAR component in UE4's editor. The 5 rays in every beam can be visualised for debugging purposes by turning on the Boolean property `bDebug`.

Once again, the data associated to each ray is encoded as a vector

$$(\boldsymbol{p}_x, \boldsymbol{p}_y, \boldsymbol{p}_z, \boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z, ID_{\text{mat}}),$$

but this time we need to fix an order for the rays that belong to the same beam. We have chosen the following order: the first ray is the central one; the other rays are counted in counter-clockwise order starting from the one on the right of the central one.
Let N_{bpf} be the number of beams per frame. In the basic physical model, data for each ray is stacked into a matrix of size $N_{\text{bpf}} \times 7$. This time around we have an extra dimension (the ray index in every beam), so the input data D_{in} is a tensor of size $N_{\text{bpf}} \times 5 \times 7$. In our code, we have chosen to store D_{in} in row-major order (generalised to tensors, i.e. contiguous elements are accessed by changing the rightmost index). This is because row-major order is the default for both C++ and NumPy.
As explained in Subsection 1.3.5, each beam produces one point of output after processing, so the data structure D_{out} is a matrix with 7 columns and at most N_{bpf} rows. The rows of D_{out} contain the same data as in the base physical model.

Data structures for the multiple reflections model (guns)

If the basic physical model is extended by taking into account the effects of multiple reflections, the description of the optical path of a single laser beam requires multiple segments. Let $\boldsymbol{p}^{i-1}$ and $\boldsymbol{p}^i$ be the endpoints of each segment, with $\boldsymbol{p}^i$ being the position of the i-th hit point after the ray has bounced $i-1$ times. By definition, $\boldsymbol{p}^0$ is the position of the LiDAR detector, and the direction of the segment $\boldsymbol{p}^1 - \boldsymbol{p}^0$ is the same as the LiDAR 's emitter. At every hit point $\boldsymbol{p}^i$, the direction of the outcoming ray is computed by reflecting the incoming ray $\boldsymbol{p}^i - \boldsymbol{p}^{i-1}$ along the surface normal $\hat{\boldsymbol{n}}^i$. The index i goes from 1 to the maximum number of reflections allowed N_{ref}. The variable N_{ref} can be configured by setting the `maxBounces` parameter of the LiDAR component in UE4's editor. The segments of every beam's optical path can be visualised for debugging purposes by turning on the Boolean property `bDebug`.
Let θ^i be the angle (in degrees) between the segment $\boldsymbol{p}^i - \boldsymbol{p}^0$ and the segment $\boldsymbol{p}^1 - \boldsymbol{p}^0$. Then, the data associated to each segment is encoded as a vector

$$(\boldsymbol{p}^i_x, \boldsymbol{p}^i_y, \boldsymbol{p}^i_z, \boldsymbol{n}^i_x, \boldsymbol{n}^i_y, \boldsymbol{n}^i_z, ID^i_{\text{mat}}, \theta^i) \qquad \text{for all } i = 1, \ldots, N_{\text{ref}}.$$

If the ray that travels in the $\boldsymbol{p}^i - \boldsymbol{p}^{i-1}$ direction has no collisions with any object in the scene, the corresponding data is the conventional vector

$$(1, 1, 1, 1, 0, 0, 0, 90).$$

Let N_{ppf} be the number of optical paths per frame. In the basic physical model, data for each ray is stacked into a matrix of size $N_{\text{ppf}} \times 7$. This time around we have an extra dimension (the segment index i in every optical path), so the input data D_{in} is a tensor of size $N_{\text{ppf}} \times N_{\text{ref}} \times 8$. Once again, we have chosen to store D_{in} in row-major order.

If $ID^i_{\text{mat}} = 0$, then our physical model knows that the i-th reflected ray has no intersection with any surface. Therefore, the vectors

$$(\boldsymbol{p}^j_x, \boldsymbol{p}^j_y, \boldsymbol{p}^j_z, \boldsymbol{n}^j_x, \boldsymbol{n}^j_y, \boldsymbol{n}^j_z, ID^j_{\text{mat}}, \theta^j)$$

for all $j = i + 1, \ldots, N_{\text{ref}}$ are technically redundant. However, we store them in D_{in} anyway just to keep the size of every path's data constant.
As explained in Subsection 1.3.6, each optical path produces one point of output after processing, so the data structure D_{out} is a matrix with 7 columns and at most N_{ppf} rows. The rows of D_{out} contain the same data as in the base physical model.

1.5 Description of the data generated by the models

1.5.1 Description of the test scene

The virtual world made available by AnteMotion is an industrial urban scenario structured in blocks with industrial buildings and sheds. Along the roads you can find signs, cars, pedestrian crossings and some obstacles made up of municipal waste. We focused our attention on an area where a car was present to have a precise benchmark for the model.

1.5.2 Remarkable cases description

This section is intended to validate the data produced by the model. We show a simulated point cloud in which physical behaviours are clearly reproduced and make a qualitative comparison with a real LiDAR point cloud taken from the KITTI repository [15]. The outcome of the model mostly depends on the accuracy of the material database, in fact it can be fine-tuned to a specific scene or target by a careful selection of the materials coefficients. The best accuracy would be achieved by including coefficients directly derived from dedicated experiments such as: direct laser reflectivity versus angle measure or real LiDAR data analysis of the collected intensity in a controlled environment.
Figure 1.10 reports an overlay between the rendered scene produced by Unreal and the point cloud produced by the LiDAR model. Major frames highlighted by red circles are described below:

1. pedestrian crossings (partial retro-reflector): they are close to the LiDAR device and have high reflectivity. A very high signal is measured compared to the asphalt.

2. car: it is composed of several materials such as metals, glasses and tires. The angle dependence of the diffusion/reflection from metal is clearly described. The front of the car has a very high signal compared to the other metallic parts. The front glass is also recognised as black because all the light is reflected upwards.

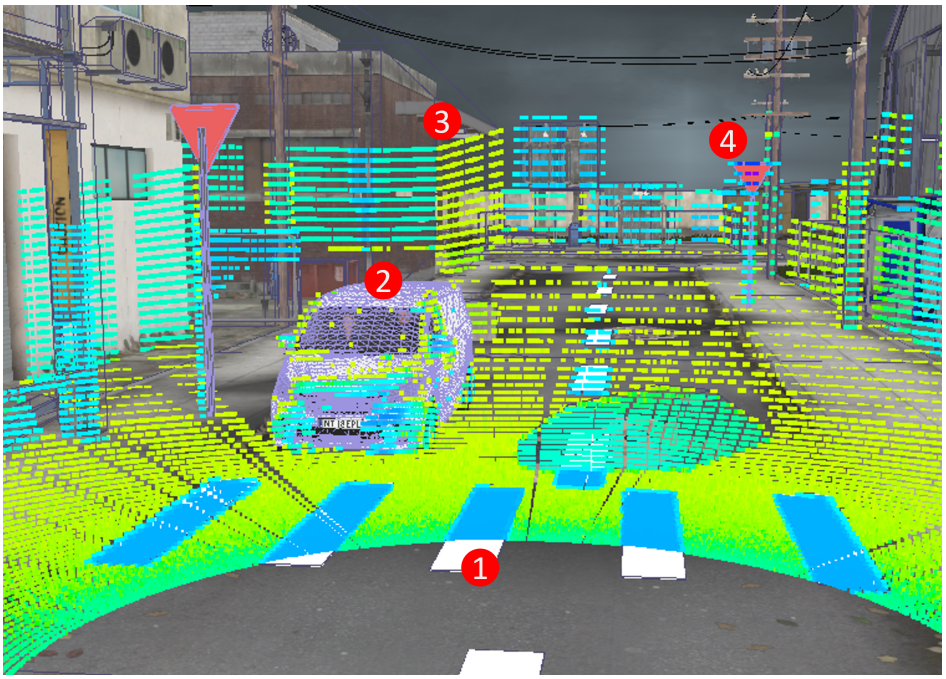

Figure 1.10: Overlay between the rendered scene produced by Unreal and the point cloud related to the LiDAR model. Color scale represents low(high) returned intensity as yellow (blue). More details are in subsection 1.4.4. Red points label relevant details.

3. Street corner: also in this case the returned diffused signal has an intensity related to the LiDAR /wall angle.

4. Road sign (retro-reflector): it appears very bright even if it is far away from the LiDAR .

We can recognize some general behaviours such as the distance-intensity relation linked to the solid angle seen by the detector and the presence of shadows behind tall objects.
Fig.1.11 reports, for comparison, some real data, obtained from LiDAR datasets available at [15]. We made a comparison between a grey-scale image taken by a camera at the LiDAR position, 1.11(A), and the point cloud representing the returned intensity to the LiDAR 's detector, 1.11(B). Yellow lines link some interesting features as detected by the two systems and red boxes highlight details that are also present in the rendered model of Fig. 1.10.
What emerges is a qualitative match between what is shown by our model and what is measured by a real LiDAR , in particular:

- the spatial intensity profile of the point cloud of cars in the real LiDAR data is very close to that of our model, with high signals coming from the normal surfaces (back of the closest car and side of the top left car) and low signal from the car windows and other rounded surfaces.

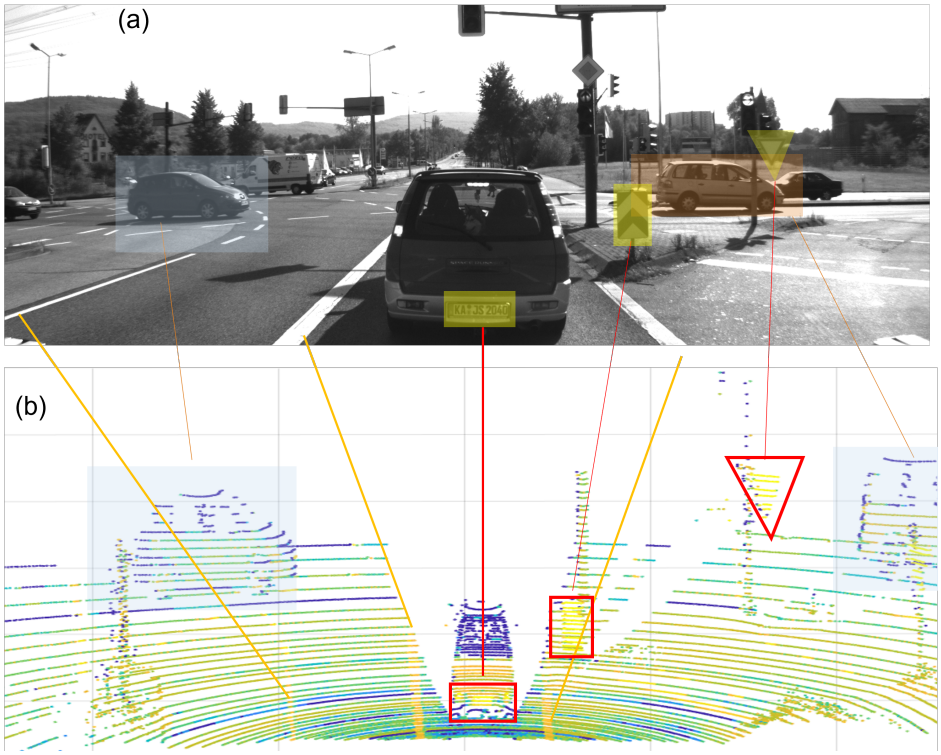

Figure 1.11: (a) Grey-scale photograph of a crossroad seen by the LiDAR . Colored boxes are linked to details in the panel (b). (b) Measured point cloud of a LiDAR : the color-scale represent the received optical signal as yellow (high) and blue (low). Credit to: KITTI Vision Benchmark Suite -A. Geiger, P. Lens, R. Urtasun [15]

- lane separators stand out from the asphalt signal
- road signs are also brighter than surrounding objects even if far away from the LiDAR

1.6 Conclusions

During the IPSP we started by analysing the problem submitted by AnteMotion from several points of view. We adopted the strategy that could guarantee the production of a proof-of-concept model within the event duration. We decided to exploit the already implemented C++ module of UE4 that calculates the geometrical intercepts of 'rays' emitted by the LiDAR . Then, all the physics was implemented in a Python-based script that analyses the intercept coordinates coming from UE4 to calculate the returned power, assuming a certain emitted power. UE4 and the physical model interact through a UDP socket mediated by another Python script called 'glue' that adapts the data formats. All these operations are carried out in real time as asked by the company. In fact, it is

possible to run the physics model while maintaining the typical frame rate at which UE4 runs (120 FPS in our tests).
The physical model accounts for all the principal phenomena that influence the performance of a real LiDAR such as diffusion, reflection, atmospheric absorption, diffraction (roses), multiple reflections (guns), detection noise, and device efficiency. Most of these parameters are material dependent. For this reason, we created a database where materials properties are fixed by three coefficients that can be optimised by the final user. The most accurate way would be to create an experimental database where the coefficients are directly measured through dedicated experiments.
Furthermore, the physics can be improved by adding the refraction effect in dielectric materials. This phenomenon can be crucial in environments where a lot of glass is present as, for instance, in a modern city or along streets with shop windows. Another important effect is the so-called *fata morgana*, a mirage seen along streets during summer. Especially in warm climate cities, the asphalt becomes very hot, determining an abrupt change of the refractive index of the air close to the surface, which causes an upwards deflection of the LiDAR 's rays with large angles of incidence. This reduces the amount of returned rays and so the information related to the horizontal street signs. Even the sun irradiance, that is varying during the day, should be considered because it affects the LiDAR 's detection system that becomes 'blind' under certain circumstances.
In conclusion, within the short time span of the IPSP event, we successfully produced an implementation of a physical model that includes the major physical phenomena involved in the LiDAR working mechanisms. The resulting numerical model is capable of running in real-time together with UE4 simulating the environment and a third software visualising the output.

Bibliography

[1] Antemotion. URL `https://www.antemotion.com/`.

[2] Santiago Royo and Maria Ballesta-Garcia. An overview of lidar imaging systems for autonomous vehicles. *Applied Sciences*, 9(19), 2019. ISSN 2076-3417. doi: 10.3390/app9194093. URL `https://www.mdpi.com/2076-3417/9/19/4093`.

[3] Paul F. McManamon. *LiDAR Technologies and Systems*. 2019. doi: https://doi.org/10.1117/3.2518254.

[4] A. Ortiz Arteaga, D. Scott, and J. Boehm. Initial investigation of a low-cost automotive lidar system. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W17:233–240, 2019. doi: 10.5194/isprs-archives-XLII-2-W17-233-2019. URL `https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W17/233/2019/`.

[5] Bashar Alsadik. Performance assessment of mobile laser scanning systems using velodyne hdl-32e. *Surveying and Geospatial Engineering Journal*, 1(1):

28 – 33, 01 2021. doi: 10.38094/sgej116. URL https://sgej.org/index.php/sgej/article/view/6.

[6] Thorlabs. URL https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=4.

[7] Velodyne. URL https://velodynelidar.com/products/alpha-prime/.

[8] John E Greivenkamp. *Field guide to geometrical optics*, volume 1. SPIE press Bellingham, WA, 2004.

[9] J. Wojtanowski, M. Zygmunt, M. Kaszczuk, Z. Mierczyk, and M. Muzal. Comparison of 905 nm and 1550 nm semiconductor laser rangefinders' performance deterioration due to adverse environmental conditions. *Opto-Electronics Review*, 22(3):183–190, 09 2014. ISSN 1896-3757. doi: 10.2478/s11772-014-0190-2. URL https://doi.org/10.2478/s11772-014-0190-2.

[10] Mario Bijelic, Tobias Gruber, and Werner Ritter. A benchmark for lidar sensors in fog: Is detection breaking down? pages 760–767, 06 2018. doi: 10.1109/IVS.2018.8500543.

[11] Jake Jacobsen. Application in illumination design - analaysing lidar return signal strengths for target optical surfaces and atmospheric conditions. *Synopsys*, white paper, 05 2020. URL shorturl.at/fEOSX.

[12] Stefan Muckenhuber, Hannes Holzer, and Zrinka Bockaj. Automotive lidar modelling approach based on material properties and lidar capabilities. *Sensors*, 20(11), 2020. ISSN 1424-8220. doi: 10.3390/s20113309. URL https://www.mdpi.com/1424-8220/20/11/3309.

[13] Jet Propulsion Laboratory California Institute of Technology NASA. Ecostress spectral library. URL https://speclib.jpl.nasa.gov/library.

[14] URL https://docs.unrealengine.com.

[15] A Geiger, P. Lens, and R. Urtasun. URL http://www.cvlibs.net/datasets/kitti/raw_data.php.

CHAPTER

TWO

COOLING OF A FLUID BY MEANS OF PHASE TRANSITION HEAT SINKS

P. Battocchio, R. Di Falco, R. Iacomino, L. Pace, L. Pagani, S. Quercetti, G. Sacco, and S. Santi

Tutor: Michele Orlandi

2.1 Introduction

2.1.1 The company

Areaderma is a cosmetic company in Trentino that has been producing cosmetics for third parties and class I and IIA medical devices for more than 30 years[1]. The research and development team is constantly looking for new technologies, cutting-edge methodologies, modern raw materials and new formulative solutions to provide to customers. Areaderma aims at the environmental sustainability of its business. In this regard, to reduce more and more the energy consumption, the company would like to change the technology used to cool products during processing. For the current process, the company uses machines where the heating is obtained by heating elements immersed in the reactor tank, while cooling is achieved by circulating cold water in an insulated cavity surrounding the tanks where the product is processed. In addition to being energy-intensive, the cooling phase takes a long time. The goal, therefore, is to be able to optimize times and reduce energy consumption during this phase of processing, switching to a modular setup, capable of processing an emulsion in standardized quantity over a fixed time range. This will allow the possibility to process several different types of products in the same working shift. The idea is to use a removable tank for each different product so that cleaning and maintenance operations can be done offline while the setup is processing the next batch in the sequence.

2.1.2 The problem

The problem proposed by Areaderma during the IPSP 2021 event concerns the study of a cooling system based on Heat Pipes. Since the tank is rotating, removable and made of plastic material, the water-bath configuration cannot be exploited due to the excessive complexity and/or the very low efficiency. A natural question arises: whether a more efficient way of cooling cosmetic creams exists and if such methodology could be in principle implemented in an industrial environment. To find an answer, Areaderma proposed to our group to study if heat pipes can be good candidates to lower the temperature of cream in the shortest time possible and if this device is to be considered among the solutions that can be used in their processes.
The specific goal is to cool a mass of cosmetic cream composed of a mix of 75% of water and 25% of oil, lowering the temperature from 85°C to 35°C in 300 seconds. The surface of the tank available to insert cooling devices is a circular sector since the remaining area is needed for the mechanical parts and to add ingredients.

2.2 Heat Pipe

The heat pipe is a highly effective passive device used to transmit heat. Heat pipes allow high transfer rates over considerable distances, with minimal temperature drops, exceptional flexibility, simple construction, and easy control, all without a need for external pumping power. The heat pipe is essentially a closed, evacuated chamber whose inside walls are covered with a capillary structure that is saturated with a volatile fluid, usually water.
The operation of a heat pipe is based on cyclic phase transitions of the fluid inside it and combines two principles of physics: vapor heat transfer and capillary action. Vapor heat transfer is responsible for transporting the heat energy from the evaporator section at one end of the pipe to the condenser section at the other end. Thus heat is subtracted to the system by exploiting the latent heat of evaporation of water.
Capillary action is responsible for returning the condensed working fluid back to the evaporator section to complete the cycle[2]. The function of the working fluid within the heat pipe is to absorb the heat energy received at the evaporator section and remove it from the system by undergoing a phase transition (that is, evaporation), then transport it through the pipe and release this energy at the condenser end by reverting back to the liquid phase. When a liquid vaporizes, two phenomena happen. First, a large quantity of heat is absorbed from the contact area, because energy is needed to separate molecules that are in contact in the liquid state. The quantity of energy required to evaporate a unit mass of liquid at a given temperature is called the latent heat of vaporization.Second, as the working fluid vaporizes, the pressure at the evaporator end of the pipe increases, due to the thermal excitation of the molecules comprising the newly created vapor. The vapor pressure leads to a pressure gradient between the ends of the pipe that causes the vapor to move toward the condenser section. When the vapor arrives at the condenser section it encounters a temperature lower than that of the

evaporator, therefore it reverts to the liquid state by releasing the thermal energy stored in its heat of vaporization. Since the heat exchange at the two ends of the heat pipe occurs thanks to phase transitions, the temperature along the entire length of the heat pipe tends to remain constant. The movement of the condensed working fluid back to the evaporator side is then provided by capillary action.
Different types of working fluids, such as water, acetone, methanol, ammonia, or sodium can be used in heat pipes based on the required operating temperature. The length of a heat pipe is divided into three parts: the evaporator section, adiabatic (transport) section, and condenser section as depicted in Fig. 2.1 [3]. The most common heat pipes have a diameter between 0.2 cm and 1 cm and a length between 5 cm and 35 cm and can dissipate from 9 W to 220 W.

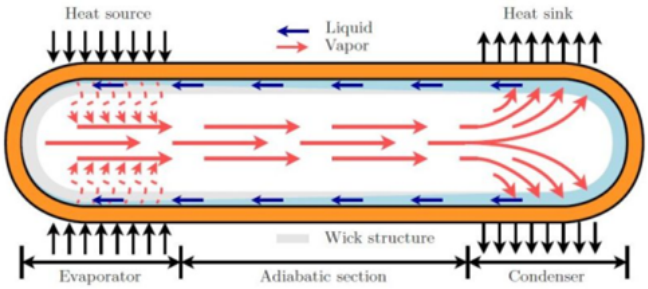

Figure 2.1: Heat Pipe [3]

2.3 Energy Balance

In this section, we describe the energy balance initially considered assuming water as working fluid. We have that: $Q_a = m_(H_2O)C_p\Delta T + m_{H_2O}\lambda$ Where Q_a is the absorbed heat, m_{H_2O} is the water mass, C_p is the specific heat capacity, ΔT is the temperature increment and λ is the latent heat of evaporation. The first term of the right side of the equation corresponds to the sensible heat, which is negligible because it is about three orders of magnitude smaller than the latent heat of vaporization.
Through this equation, we determined that, in order to subtract to the system the desired amount of heat, the total amount of water that should vaporize is $m_{H_2O} = 1.35kg$ and so, corresponding to $\frac{dm}{dt} = 4.48\frac{g}{s}$ in a time interval of 300 s. Considering that the emulsion must reach a temperature of 35°C, we have assumed that the boiling point for the water inside the heat pipe is 30°C, so we determined an inner pressure of 0.03 bar.
We then designed, implemented and tested two different approaches: the first one (Heat Pipe Array Cooler) exploits a series of heat pipes; while the second one (Phase Transition Pressure Cooler) partially exploits the working principles of heat pipes and is developed to overcome their main limitations.

Mean working T [°C]	20	40	60	80	120
Max. exchange power [W] (12 mm diameter heat pipe)	148	178	186	197	218

Table 2.1: Maximum Power Handling Capability from CRS datasheet. The bottom row represents the maximum power for a heat pipe with the external diameter of 12 mm.

2.4 Heat Pipes Array Cooler (HPAC)

The most straightforward solution would be to insert a high number of heat pipes in the hot emulsion in order to reach the desired cooling power of 11 kW. This solution has been called Heat Pipes Array Cooler (HPAC).
By studying datasheets of heat pipes, we found a promising series by "CRS Engineering" [4]. Table 2.1 reports the maximum power transmission for heat pipes operating in a horizontal position. For our application a mean temperature of 60°C should be considered, providing a maximum power of 186 W with a heat pipe of 12 mm diameter.
The function of a heat pipe relies on an effective capillary action system to return the working fluid from the condenser section where cooling is applied to the evaporator section. This function is served by the porous lining wick structure which is built into the heat pipe. If the position of the heat pipe is vertical, with the condenser at the top and the evaporator at the bottom, the gravity improves the transport of water back to the evaporator and could increase the thermal transmission capability of the heat pipe up to twice its horizontal dissipation power, as shown in Fig. 2.2 [4]. In this configuration (that would perfectly fit the need to insert heat pipes in the tank from the top) the previously mentioned heat pipe would reach a maximum dissipation power of 372 W. We need to subtract 11 KW, consequently, about 30 heat pipes of this type would be needed. The surface that we considered to position 30 heat pipes.

2.5 Phase Transition Pressure Cooler (PTPC)

The group also proposed a new possible solution, in alternative to heat pipes. Since the limiting factor for heat pipes is the transport of recondensed water back to the evaporator section, the idea was to develop a new system that exploits the same working principle of the heat pipe, based on heat transfer with phase transitions, but decoupling the condensation stage. H_2O is gradually inserted in a chamber kept at a pressure low enough to ensure water vaporization at the desired temperature; produced vapour is then evacuated by the vacuum pump. In contrast to what happens in the heat pipes, the water follows a unidirectional path, since evaporated water is removed from the system and recondensed externally. Note that this system is active, since vapor is transferred by a pump instead of exploiting vapour expansion. This solution has been called Phase Transition Pressure Cooler (PTPC).

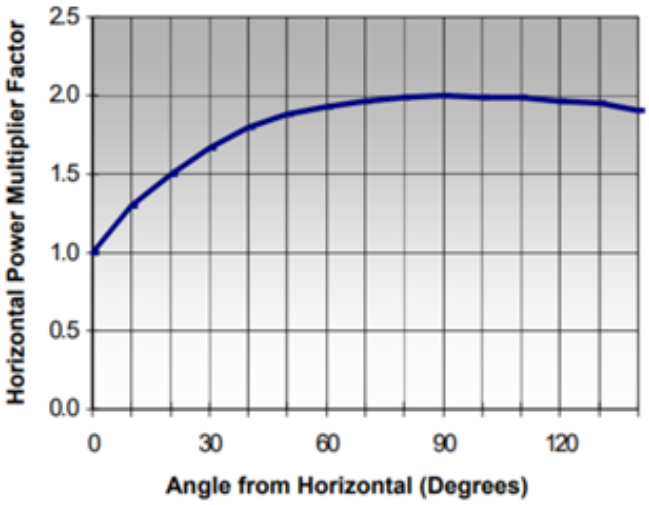

Figure 2.2: Curve Multiplier Factor (of maximum power) vs. Angle (taken from [4])

2.6 Design of experiment

Experiments have been carried out to test the behavior of both HPAC and PTPC in working conditions similar to those proposed by the company. To simulate the actual process of the company described above, distilled water is used instead of emulsion as the substance to be cooled, in order to work with a fluid with a known specific heat capacity. 3 liters of water are placed in a covered plastic container, stirred by a magnetic stirrer at 1000 rpm to simulate the motion of the emulsion in the tank. The cooling device under test is partially dipped in water. The temperature is measured with a resistance thermometer (PT100 probe). The initial temperature of the water is 75°C. The observable of the experiments is then water temperature as a function of time, from which we can calculate the subtracted power.
To be able to estimate the work done by the cooling device itself, excluding the normal cooling of the hot water when left at room temperature, the latter was measured in the same condition (stirred hot water in a covered tank) with no cooling devices inserted in it. The resulting dissipated power was subtracted from the power measured with the operating devices.

2.7 Experiments with HPAC

2.7.1 First prototype of HPAC: Wall – E

To test the heat pipes, we used three identical heat pipes (300 mm length, 10 mm diameter, nominal power 80 W) in parallel.
The first prototype, called “Wall-E”, is shown in Fig.2.3. The condenser section of the heat pipes is inserted in a copper block with three holes that is thermally

connected to a heat sink with two fans. The device should subtract heat from the hot water and transfer it to the cooled copper central body. This system however has not guaranteed good results, since the fans and the heat sink are not able to keep the copper body sufficiently cold and water recondensation inside the heat pipes stops. This experiment confirmed that the bottleneck of a heat pipe system is the cooling of the condenser part.

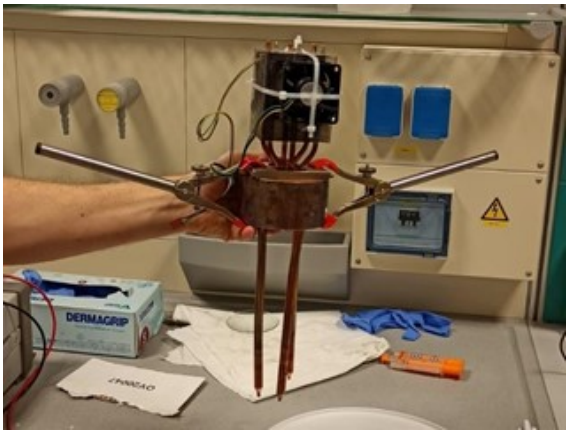

Figure 2.3: First Prototype: Wall-E

2.7.2 Improvements with HPAC

To guarantee water condensation at the heat pipes' condenser end, the cold source was substituted by a small container with running water and/or ice. The container of ice and water has been connected to a valve to secure the continuity of the cold flow, in order to improve the heat exchange. The system is shown in Fig. 2.4. This setup has been tested in three different modalities:

- Stagnant water and ice
- Mixed water and ice
- Running water without ice

Moreover, different experiments were carried out varying the length of heat pipes dipped in water, in order to check variations of cooling power as a function of the effective length L_eff over which the heat transfer occurs, defined as the distance between the middle of the evaporator section and the middle of the condenser section.

2.8 Experiments with PTPC: SVAPO

The first prototype of Phase Transition Pressure Cooler, called "SVAPO", has been designed, produced and tested during the IPSP week. It is shown in Fig. 2.5.

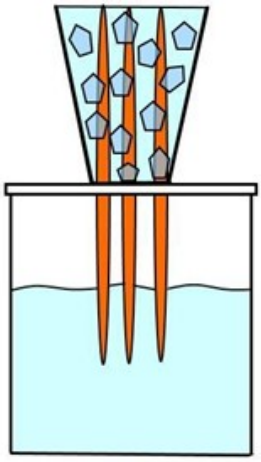

Figure 2.4: Heat Pipe condenser section inserted in water and ice

It consists of a copper cylindrical chamber of 6 cm diameter and 12 cm height, at whose top two tubes of 10 and 6 mm diameter are inserted, meant to connect a pump and insert water, respectively. The pump keeps the pressure inside the cylinder at around 30 mbar, a pressure that allows water evaporation down to 30°C. A flow of 4.48 g/s of distilled water is inserted into the system during cooling operation. The resulting water vapor is then removed by the pump.

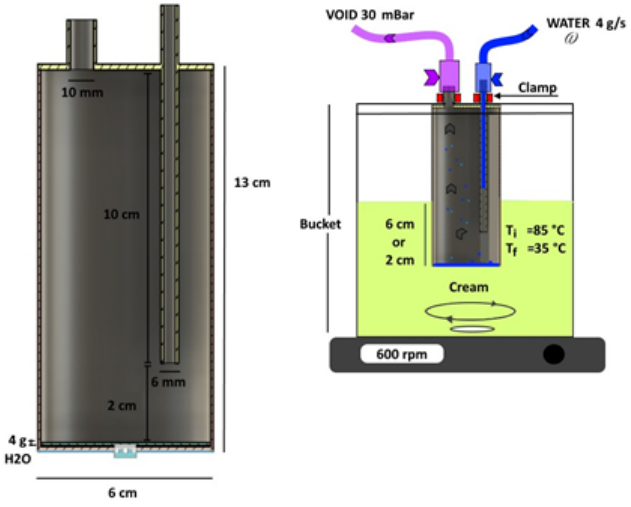

Figure 2.5: New system SVAPO

2.9 Results and discussion

The graph in Fig. 2.6 shows the temperature as a function of time of the two devices designed. The blue curve is obtained with the HPAC system consisting of three heat pipes in mixed water and ice, while the red one shows results obtained with the SVAPO. For comparison, the dashed line represents the trend of temperature as a function of time without the use of any cooling device ("Blank" in the picture).

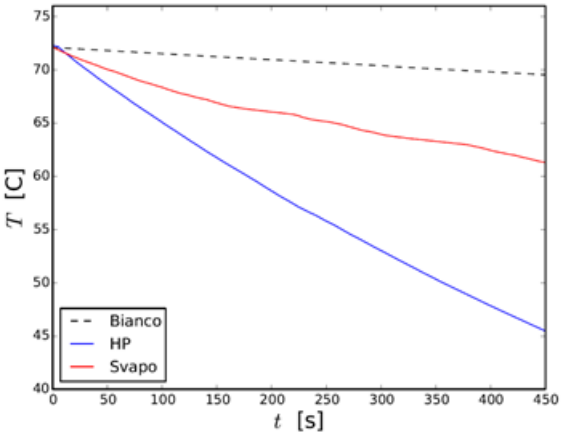

Figure 2.6: Water temperature as a function of time

The HPAC succeeds in lowering the temperature of the water from 73°C to approximately 45°C in 450 seconds, while Svapo, in the same amount of time, succeeds in reaching a temperature of about 63°C. The resulting subtracted power is 680 W for the HPAC and 300 W for SVAPO.
At the same time, simulations of the operation of HPAC were carried out by modifying a COMSOL model [5] and obtaining results in the same order of magnitude of the measured value.
An additional experiment, with a similar procedure to the others, was also carried out with SVAPO to check its correct working and confirm that heat is subtracted from the system by the evaporation of the water inside the copper chamber. During this experiment a fixed quantity of water is injected into the chamber where it evaporates cooling down the hot water in the container; if no more water is injected, once it is completely evaporated the temperature decrease in time slows down since heat is no more absorbed by the system. This can be seen in Fig. 2.7, which shows a step behavior of the temperature with time: each time water is injected into the system, cooling velocity increases, and then it progressively decreases as all the water inside the chamber evaporates, confirming the expected working principle of this apparatus.
The best results in terms of powers subtracted to the hot water mass are obtained using the HPAC system. However, it must be noted that the SVAPO was not working in its optimal conditions and can likely be improved substantially:

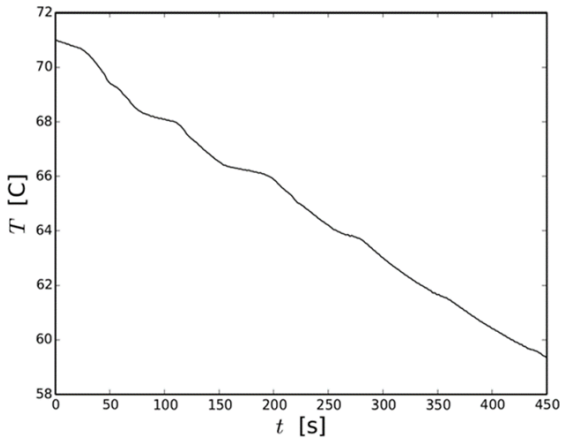

Figure 2.7: Test of the working principle of SVAPO

- The pressure inside the copper chamber was higher than the desired value, reasonably because of leakages in the vacuum system, therefore water evaporation was slower than expected.

- There was no control on the quantity of water injected into the copper chamber, so either there was no water inside and therefore evaporation stopped or there was too much water that then was in poor thermal contact with the chamber walls.

- The shape of the chamber could be changed in order to optimize the contact surface of the chamber walls with the hot water (or emulsion) outside and with the water to evaporate inside.

There are eventually a few more considerations to make regarding the practical actuation of the HPAC and PTPC solutions. The industrial problem to solve is related to the cooling of a given mass in a sufficiently short time, what matters therefore is mainly finding a way to subtract energy efficiently from the cream. The Phase Transition Pressure Cooler is much more interesting than the heat pipes from this point of view, since the heat subtracted from the cream is rapidly brought far from the system as water vapour by a vacuum pump and then it can recondense in longer times somewhere else. On the other hand, using heat pipes requires that the recondensation of vapour on the cold end must be as fast as possible, otherwise the cyclic operation of this system stops and cooling rates drastically decrease. With heat pipes the cooling problem is therefore just shifted from the cream to the condenser part of the pipe, and this may be somewhat more difficult to obtain in practice than simply using a vacuum pump as SVAPO requires.

2.10 Conclusions

The problem proposed by Areaderma consisted in finding a way to rapidly cool down a given mass of cream in the shortest time possible. The method initially proposed by the company was to exploit heat pipes, passive elements with the capacity of fast transferring heat from one side to the other by exploiting vapour expansion inside the pipe.
The work done during the week can be summarized as follows. Initially a first system based on heat pipes called Wall-E was developed, this allowed a better understanding of the working principle of heat pipes and their main limitations. We found that the main problem of the heat pipes is related to the efficient cooling of their condenser end, which otherwise stops the cyclic behavior of these elements. The study of the operation principles of heat pipes was also carried on by performing some simulations in COMSOL.
The work then proceeded with the development of two cooling systems: one based on heat pipes, which is an evolution of Wall-E, and another one that exploits only the evaporation part of the heat pipe cycle, called SVAPO. The evolution of Wall-E consisted in an improvement of the cooling of the heat pipes by using ice, while SVAPO exploited the low temperature evaporation of water obtained reducing pressure in a copper chamber, with the generated vapour removed by means of a vacuum pump.
Some experiments were carried out by cooling a fixed mass of water using these systems in order to measure their efficiency. The system based on heat pipes turned out to be more efficient than SVAPO, however SVAPO was only in an initial stage of development and was not working in its optimal conditions. We then concluded that SVAPO can be a possible interesting solution for Areaderma, both in terms of dissipated power and from the practical actuation point of view.

Bibliography

[1] Areaderma. URL `https://www.areaderma.it/`.

[2] G. Yale Eastman. The heat pipe. *Scientific American*, 218(5):38–47, may 1968. doi: 10.1038/scientificamerican0568-38. URL `https://doi.org/10.1038%2Fscientificamerican0568-38`.

[3] Amir Faghri. Review and advances in heat pipe science and technology. *Journal of Heat Transfer*, 134(12), oct 2012. doi: 10.1115/1.4007407. URL `https://doi.org/10.1115%2F1.4007407`.

[4] URL `https://www.heat-pipes.com/media/a4-heat-pipes-v26-hadston-2010.pdf`.

[5] URL `https://www.comsol.it/model/heat-pipe-with-accurate-liquid-and-gas-properties-90311`.

CHAPTER

THREE

POSITIONING OF A WEATHER STATION FOR THE DETECTION OF ENVIRONMENTAL DATA OF A TURBINE FOR TECHNICAL SNOWMAKING

F. Bellamoli, S. Bini, F. Brugnara, M. Castelluzzo, E. Cozzolino, G. Dalmasso, R. Franchi, and C. Montesano

Tutor: Luca Tubiana

At the request of the company, the technical part of this chapter has been replaced by a generic summary of the work carried out by the team during IPSP2021.

3.1 Introduction

3.1.1 The company

Since 1990 TechnoAlpin has been designing and manufacturing turnkey snowmaking systems for ski resorts all over the world. The company's turnover has reached 230 million euros. The love for snow and the passion for innovation have transformed TechnoAlpin into a leader in the sector, which mainly aims to offer each customer a customized solution.
Each plant is thus subjected to careful evaluation to adapt it to the specific needs of individual customers. The product range is constantly being expanded in order to produce snow of the highest quality with maximum energy efficiency. In addition to outdoor solutions, TechnoAlpin also offers turnkey solutions for indoor snowmaking, in order to make the exclusive snow experience accessible at any latitude. Over 2,400 customers in more than 55 countries rely on TechnoAlpin's know-how, followed by over 700 employees in 35 offices.

3.1.2 The problem

A snowmaking system consists of several components: pipes, pumping stations, wells, basins for water collection and snow guns often called "cannons". The latter can be of different types, mainly we can divide them into turbines or lances. TechnoAlpin produces various models of snow blowers. Each turbine is equipped with its own weather station. In order to produce snow, certain environmental conditions must be met. In particular, the combination of air temperature and humidity, defined as wet bulb temperature for insiders, must be below a certain threshold. It is therefore essential to have precise weather values for each snowmaking point. The turbines are usually located on the edges of the ski slopes at a distance of about 50-100 m from one another. It may therefore happen that one turbine is a few kilometers away from another one and at an altitude of even 1000-1500 m higher. This means that if with the turbine further upstream snow can be produced due to the correct environmental conditions, it does not necessarily mean that there are the same conditions for the turbine further downstream from the plant.
All turbines can operate automatically controlled by the centralized management software. It is therefore essential to correctly measure the weather data, especially in extreme conditions, that is, when a variation of even half a degree means that water and not snow comes out of the turbine. In the same way, an incorrect detection can lead to a shutdown of the turbine when it could instead be productive. The weather station in the old TechnoAlpin turbine models was positioned outside the turbine. In the new models, however, it has been integrated into the turbine cover as can be seen in Fig.3.1.

Figure 3.1: TechnoAlpin TR10 snow generator

Therefore, the challenge proposed by TechnoAlpin is to evaluate an alternative positioning of the turbine weather station such as to be able to detect environmental values with extreme precision without these being minimally influenced by any factors that may lead to an alteration of the values themselves (for example presence of snow, ice near the sensors, etc.)

3.2 The solution

In the study carried out during the IPSP2021, the team of PhD and Master students worked in parallel with experimental measurements on snow generators at the TechnoAlpin headquarters in Bolzano and with theoretical models to characterize the temperature behavior inside the turbine.
The results clearly show that the observed deviation between the temperature measured inside or outside the turbine depends on the operation of the turbine itself. These results derive from different temperature measurements carried out in different conditions: generator inside or outside the laboratory, different climatic conditions outside, etc.
In particular, the measurements confirmed what the company previously observed that the temperature inside the hull is higher than the outside temperature. The team then concentrated on establishing the coefficient of the difference noted.
Through the thermodynamic study of the system, the two main contributions that cause the temperature coefficient have been analyzed: a) the energy transferred from the fan to the air in the form of pressure and kinetic energy and b) the dissipated energy that heats the turbine and the air inside.
It has been seen that the experimental and theoretical approaches have provided consistent results, thus proving that the main physical phenomena involved have been correctly identified. The values obtained from the study give the company the opportunity to evaluate interesting ideas for a new design of the studied device.
The company was advised to carry out a series of experimental tests in typical working conditions, that is, in winter and with low temperatures, to further confirm the proposed model, even if the thermodynamic considerations studied are still robust.

CREDITS

The entire group on the opening day

From left to right, front row on knees: Bruno Degli Esposti, Ersilia Cozzolino, Sofia Santi.

From left to right, back row: Mattia Affabris (*AnteMotion*), Michele Castelluzzo, Luca Pagani, Luca Gasbarro (*AnteMotion*), Sara Rabaglia, Viviana Maria Filì, Riccardo Franchi, Alessandro Zunino, Fabio Callegari, Giuseppe Sacco, Federico Bellamoli, Rossella Di Falco, Pietro Battocchio, Ludovica Pace, Giorgio Dalmasso (*on knees*), Luisa Cunico (*TechnoAlpin*), Romano Iacomino, David Tschager (*TechnoAlpin*), Massimo Stroppari (*Areaderma*), Cesare Montesano, Luigi Miori (*Areaderma*), Davide Bazzanella, Sebastiano Bontorin, Francesca Scandolari (*Areaderma*), Fabio Brugnara, Louise Wolswijk, Flavio Deflorian (*Rector of the University of Trento*), Sophie Bini, Mattia Mancinelli (*Department of Physics, UniTrento*), Giuseppe Caputo (*Head of Research Valorization and Impact Office, UniTrento*), Michele Orlandi (*Department of Physics, UniTrento*), Luca Tubiana (*Department of Physics, UniTrento*).

Foto ©Micaela Paoli per UniTrento

AnteMotion team

From left to right: Mattia Affabris, Alessandro Zunino, Fabio Callegari, Mattia Mancinelli, Sara Rabaglia, Louise Wolswijk, Davide Bazzanella, Sebastiano Bontorin, Bruno Degli Esposti.

Foto ©Mattia Mancinelli per UniTrento

Areaderma team

From left to right: Giuseppe Sacco, Sofia Santi, Ludovica Pace, Rossella Di Falco, Massimo Stroppari, Francesca Scandolari, Michele Orlandi, Pietro Battocchio, Sara Quercetti, Luca Pagani, Romano Iacomino.

Foto ©Mattia Mancinelli per UniTrento

TechnoAlpin team

From left to right Fabio Brugnara, Riccardo Franchi, Luca Tubiana, Giorgio Dalmasso, Cesare Montesano, Luisa Cunico, David Tschager, Sophie Bini, Federico Bellamoli, Michele Castelluzzo, Ersilia Cozzolino.

Foto ©Mattia Mancinelli per UniTrento

ACKNOWLEDGMENTS

Foremost, we would like to express our special thanks to the organizing bodies of Industrial Problem Solving with Physics: Department of Physics, Doctoral School in Physics, Research Support and Valorization Division of the University of Trento, Confindustria Trento and Polo Meccatronica-Trentino Sviluppo. We are particularly grateful to the head of the Department of Physics, Prof. Franco Dalfovo, for his constant interest and support. We thank Vanessa Ravagni and Giuseppe Caputo for their advice and assistance regarding intellectual properties and Alessandro Santini, Daniele Berti and Paolo Gregori for helping us get in touch with some potential participating companies.
We extend our special thanks to all the companies which believed in this project and decided to nominate themselves. Our sincere thanks go to the participating companies representatives: Luca Gasbarro and Mattia Affabris from AnteMotion, Massimo Stroppari and Francesca Scandolari from Areaderma, and Luisa Cunico and David Tschager from TechnoAlpin whose interesting and challenging inputs make this event possible. Besides the participating companies, we are grateful to our technical sponsor, COMSOL, who provided us with the free licence and consultancy for their software.
IPSP would not have certainly been possible without the enthusiasm of the students and researchers who joined us in this adventure. We thank them all for their fundamental contribution.
We would also like to thank the staff of the Laboratori Didattici of the Department of Physics, in particular Mauro Hueller, for their availability, instrumentation, and personal knowledge throughout the week, and the staff of the Mechanical Workshop for their assistance in the realization of prototypes. A huge thank you goes to the Nanolab group who provided us with the workstation used by the AnteMotion team. We also thank the staff of the Secretary of the Department of Physics for their continued support for the organization of IPSP2021 and the Communication Office - Polo Collina of the University of Trento for handling the online contents and paper materials.
Finally, we thank all the people who supported us in the realization of Industrial Problem Solving with Physics, edition 2021.

Mattia Mancinelli, Michele Orlandi, Luca Tubiana
Sientific Committee of IPSP2021

Industrial Problem Solving with Physics (IPSP) is a one-week event organized by the Department of Physics and the Research and Technology Transfer Support Division of the University of Trento, in collaboration with Confindustria Trento and Polo Meccatronica - Trentino Sviluppo.

3 companies and 30 brains (post-graduate students (Master's degree and PhD) and research fellows) are selected and work together to find solutions to practical industrial problems proposed by the participating companies.

Young and motivated researchers have the chance to show off their skills in tackling new practical challenges. The participating companies, on the other hand, obtain solutions to their problems and experience an alternative problem solving strategy.

SCIENTIFIC COMMITTEE

Mattia Mancinelli, Department of Physics
Michele Orlandi, Department of Physics
Luca Tubiana, Department of Physics

ADVISORY BOARD

Franco Dalfovo, Department of Physics
Giovanni Andrea Prodi, Doctoral Programme in Physics
Vanessa Ravagni, Directorate of Research Services and Valorization
Alessandro Santini, Confindustria Trento
Paolo Gregori, Polo Meccatronica

event.unitn.it/ipsp2021

Participating companies

ANTEMOTION INNOVATIVE AUTOMOTIVE ENGINEERING COMPANY

areaderma LA SCIENZA DELLA PELLE

TECHNOSILPIN

Technical sponsor

COMSOL

ISBN 978-88-8443-965-9