Mozart: A Mobile ToF System for Sensing in the Dark through Phase Manipulation

ABSTRACT

Sensing in low-light and dark environments has a wide range of applications. However, existing sensing technologies suffer several major challenges such as excessive noise and low resolution. This paper proposes Mozart - a new mobile sensing system that leverages off-the-shelf Time-of-Flight (ToF) depth cameras to generate high-resolution and rich-in-texture maps for applications in dark scenarios. The design of Mozart is based on our key observation that the phase components of ToF measurements can be manipulated to expose texture information. Through in-depth analysis of the physical reflection model, we show that the textures can be exposed and enhanced using highly compute-efficient phase manipulation functions. By exploiting the physics texture models, we propose an autoencoder-based unsupervised learning approach that can automatically learn efficient representations from phase components to generate high-resolution maps. We implemented Mozart on several Android smartphone models[1]. and an edge testbed with standalone ToF camera platforms for various applications in the dark. The results show that Mozart can work in real time and delivers significant improvement over existing sensing technologies. Therefore, Mozart offers a low-cost, high-performance sensing technology for next-generation applications in the dark.

1. INTRODUCTION

Sensing in low-light and dark environments has a wide range of applications, such as smart building, smart health, and robot navigation. For example, a highly desirable feature of smart door locks is automatic unlock via face recognition or secret hand gestures in dark environments [1, 2]. Moreover, many health monitoring systems require 7/24 sensing capabilities, e.g., detecting Sudden Infant Death Syndrome (SIDS) during sleep using a smart baby monitor [3].

As summarized in Table 1, although there already exist many sensing technologies that function to some extent in dark conditions, they cannot meet the requirements of high-resolution sensing applications. RF-based systems such as mmWave radar and Wi-Fi are not interfered by visible light. Unfortunately, their sensing data are highly sparse [4, 5, 6], making them poorly suited for applications that require high-resolution results such as human faces and hand gestures. Thermal, IR, and depth cameras can work in the dark. However, thermal cameras have limited resolution [7]. Although IR cameras can provide more detailed information, they rely on strong IR emissions, which incur high power consumption ranging from 5 to 20W [8, 9]. Moreover, off-the-shelf commercial IR cameras suffer from the over-exposure effect when objects are too close and can only capture image details within a short distance (typically up to 5m [10]) due to the fast decay of light intensity [11]. ToF depth cameras have a longer range and lower power consumption, and are increasingly embedded in smartphones or used as standalone sensors for 3D applications. However, by design, ToF depth cameras cannot capture most texture information of the scene [12].

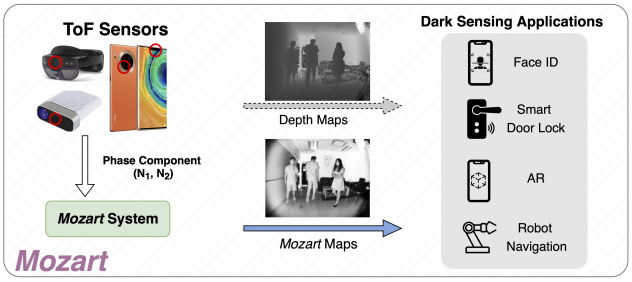


Figure 1: Typical applications of *Mozart* in the dark sensing scenario. In addition to the depth maps from ToF cameras, *Mozart* also generates high-resolution and rich-in-texture maps, which can significantly augment the performance of sensing tasks in the dark.

In this paper, we proposed *Mozart*, a novel sensing system that leverages off-the-shelf ToF cameras to generate high-resolution and rich-in-texture maps for dark scenarios. As shown in Figure 1, *Mozart* maps can significantly augment the performance of various sensing tasks in the dark. The design of *Mozart* is based on our key observation that the phase components of ToF measurements can be carefully controlled (which we refer to as "*phase manipulation*") to generate high-resolution maps with detailed textures. To design *Mozart*, we first present an in-depth analysis on the physical reflection model for exposing texture information through phase manipulation. Our key finding is that the textures can be exposed and enhanced using highly compute-efficient phase manipulation functions. Lastly, we propose an end-to-end autoencoder-

[1]A demo video of Mozart smartphone App working in the dark is available at `https://youtu.be/qBEffXVft_8`.

Sensing Technologies	Modality	Work in the Dark?	Cost	Power	Noise	Typical Range (with texture details)	Texture Resolution
RGB Camera	Visible Light	No	**Low**	Low	High	5m	Low
mmWave Radar	Radio Waves	Yes	Medium	low	Very High	0m	Very Low
Thermal Camera	Longwave Infrared	Yes	High	High	Medium	5m	Low
IR Camera	Near Infrared	Yes	Medium	High	Medium	5m	Medium
ToF Camera	Near Infrared	Yes	Medium	Low	High	5m	Medium
Mozart (Ours)	Near Infrared	**Yes**	Medium	**Low**	**Low**	**10m**	**High**

Table 1: Comparison of various technologies for sensing in the dark.

based unsupervised learning approach to automatically learn efficient representations from the phase component maps to generate *Mozart* maps. To train the deep autoencoder, we design three novel loss functions by exploiting the physics models we proposed, including the *albedo similarity loss*, the *illumination attenuation loss*, and the *uniform distribution loss*. Our approach offers several key advantages, including requiring no labeled training data and being highly scalable in different applications without manual system tuning.

We implement *Mozart* on several Android smartphone models and mainstream standalone ToF cameras. The results show that *Mozart* maps can be generated in real-time on smartphones due to the extremely low overhead. Moreover, *Mozart* can generate high-resolution maps at about 23 frames per second on edge computing platforms, and is compatible with mainstream ToF cameras. We evaluate the performance of *Mozart* using three new datasets we collected in low-light and dark conditions, including human tracking, face recognition, and gesture recognition, which involves a total of 33 subjects and contains over 1,000,000 data frames. The results show that *Mozart* maps outperform all baseline modalities (including RGB, IR, depth, and mmWave Radar) in dark environments. For example, in gesture recognition, *Mozart* outperforms RGB images, mmWave Radar, depth images by 93.4%, 88.46%, 45.76%, respectively. Moreover, in comparison with IR maps, Mozart delivers a performance improvement up to 29.1% with a substantially smaller variance.

Our key contributions are summarized as follows:

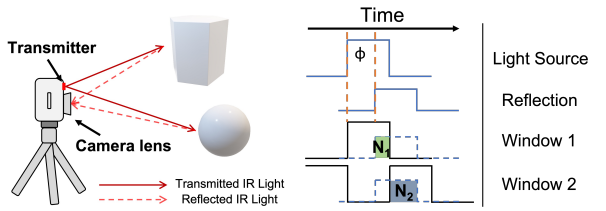
- To the best of our knowledge, *Mozart* is the first low-cost, high-performance sensing system that can generate high-resolution maps using a single off-the-shelf ToF camera in low-light and dark environments.
- We provide an in-depth analysis on the physics models for exposing and enhancing textures. Our key finding is that the textures can be generated using highly lightweight phase manipulation functions.
- By exploiting the physics texture models, we propose a deep autoencoder-based texture generation approach that can automatically learn efficient representations from phase maps to generate *Mozart* maps.
- We implement *Mozart* on several smartphone models as well as edge platforms with mainstream ToF modules. Our evaluation using three self-collected datasets in the dark shows that *Mozart* outperforms existing baselines and can work in real time.

2. RELATED WORK

Sensing Technologies for Applications in the Dark. Sensing in the dark has a wide range of applications such as robot navigation, face authentication, gesture recognition and surveillance [6, 13, 2, 14]. Most of the current approaches in this area are based on vision or RF sensors. RGB camera is a ubiquitous vision system, which cannot work in dark conditions [15]. Other vision sensors such as thermal, IR and depth cameras have shortcomings such as low resolution [7], high power consumption [9, 8] and limited texture exposure [12]. In particular, the intensity maps [16, 17] collected by ToF cameras are essentially the IR maps collected by IR cameras. RF-based sensing technologies such as mmWave radar and Wi-Fi are not interfered by visible light. Unfortunately, their sensing results are highly sparse [4, 5, 6]. Compared with these existing solutions, our system can produce high-resolution images in the dark by extracting detailed textures from off-the-shelf ToF cameras.

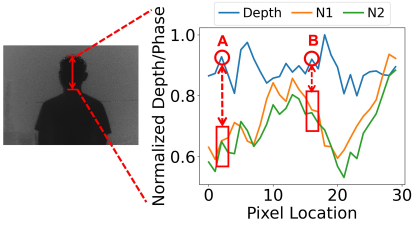
Image Enhancement in Low-light Conditions. There have been extensive efforts on improving the RGB image quality in low-light conditions. A multi-task learning framework is proposed in [18] to explore the intrinsic pattern behind illumination translation for object detection in poor light environments. Moreover, in [19], thermal images are synthesized from RGB images with a Generative Adversarial Network to enable monitoring in low-light conditions. Recently, several studies formulate the light enhancement of RGB images as a deep curve estimation problem, which requires the reference images as the input [20]. These approaches can not work in fully dark conditions without any illumination. Moreover, designed for RGB cameras, they can not be directly applied to ToF cameras due to the fundamental differences between the two sensor modalities.

ToF Augmentation. A family of techniques has been proposed for depth image enhancement, most of which are focused on improving the noise models of ToF cameras for accurate distance measurement [21, 22, 23]. An energy-efficient epipolar imaging approach is proposed in [24] to improve the robustness of depth measurement in extreme scenarios, and the centimeter-wave and interferometric imaging are utilized in [25] to improve the precision of iToF cameras. A recent work [12] illustrates



(a) Capturing depth maps of the scene with a ToF camera (b) Principles of iToF depth cameras

Figure 2: ToF depth cameras obtain depth maps by emitting and receiving the IR light to calculate the time of flight, which is unaffected by the ambient light.



(a) The original depth map and values of phase components

(b) The new depth map obtained by shifting N_1.

Figure 3: Compared to the depth map, the values of phase components have larger fluctuations across the face. Adding a shift to N_1 when calculating depth using Eqn. (1) leads to an image with fine-grained textures.

the feasibility of extracting rich textures from depth maps. However, it requires additional hardware such as an external IR emitter and distorts depth measurements during texture exposure, making it incompatible with current depth-based applications. In contrast, Mozart generates high-resolution texture maps entirely based on on-device sensing data processing. As a result, it not only can be implemented on mainstream off-the-shelf ToF cameras and ToF-enabled smartphones, but also can obtain high-resolution texture maps and depth maps simultaneously.

3. BACKGROUND AND MOTIVATION

In this section, we introduce the basic principles of ToF sensing. Then we study the impact of phase component manipulation and compare the manipulated maps with IR maps to motivate our design.

3.1 Principles of ToF Depth Sensing

Depth measurement from time-of-flight. As shown in Figure 2a, a ToF depth camera emits IR light, illuminates the scene to be captured, and receives the IR light reflected by the objects in the scene. The distance is measured based on the fact that the round trip time-of-flight (t) of the IR signal between the scene and the camera is strictly proportional to the distance. Specifically, we have $t = 2d/c$, where d is the distance of the scene and c is the speed of light. Off-the-shelf ToF cameras fall into two categories based on how the time-of-flight is measured: direct Time-of-Flight (dToF) and indirect Time-of-Flight (iToF). Compared with dToF, iToF is more suitable for 3D imaging applications due to its low cost and high-resolution [12]. Currently, most of the ToF modules on mobile devices (especially Android smartphones) on the market adopt the iToF technology [26, 27]. *Mozart* is designed to work with iToF cameras, and all ToF cameras in this paper refer to iToF cameras unless otherwise indicated.

Measuring ToF based on received signal phase. The iToF camera has two successive windows (in-phase and quadrature) to receive the reflected light and uses the phase shift of the returned light to calculate the time of flight. Figure 2b illustrates the general principle of calculating the phase shift in a ToF camera. The time of flight t can be calculated by:

$$t = \frac{N_2}{N_1 + N_2} \cdot T_p, \tag{1}$$

where T_p is the width of pulse, N_1, N_2 are the amount of received light in successive in-phase and quadrature windows, which we refer to as *phase components*.

Besides the basic designs, mainstream off-the-shelf iToF cameras also adopt several advanced techniques to mitigate the influence of ambient light on distance measurement. For example, the continuous-wave iToF cameras take multiple samples per measurement (i.e., using more than two windows) to calculate the phase shift, which can reduce the energy offset caused by ambient light during the process of each distance measurement [28]. For different types of iToF cameras, we can always obtain the phase components N_1, N_2 through a direct transformation of Eqn. (1).

3.2 A Motivation Study

Impact of Phase Components. As shown in Eqn. (1), the depth measurements are calculated from the phase components (N_1, N_2). Moreover, the process of calculating the depth of a point in the scene is equivalent to the dimension reduction from 2-D (N_1, N_2) to 1-D distance d, which will inevitably lose some information. In other words, *the phase components contain more information about the captured scene than the depth measurement.* This key observation provides opportunities for exposing detailed texture in the calculated map.

To validate this observation, we compare the depth measurements and phase components for the points across a vertical line of a human face in Figure 3a, where the blue curve denotes the normalized depth values, and the orange and green curve denote N_1 and N_2 components, respectively. We observe that the depth values are less volatile, while the phase components (N_1, N_2) fluctuate drastically. Moreover, for two points **A** and **B** with the same distance from the ToF camera, they have totally

(a) Collected by IR cameras (b) Collected by ToF cameras

Figure 4: The IR maps collected by IR cameras and ToF cameras both suffer over-exposure for near objects and under-exposure for distant objects.

different phase components (N_1, N_2). Therefore, by exploiting such information encoded in the phase components, it is possible to show detailed texture information of different points in the scene.

We then conduct a simple manipulation operation on the phase components to show the feasibility of exposing more texture information of the scene. Specifically, we add a small shift to N_1 when calculating depth using Eqn (1). Figure 3b shows the resulting depth map, which exhibits significantly finer grained textures. Next, we apply a commonly used object detection model [29] to the original depth maps and the new maps with simple phase component manipulation. The results show that the human detection rate increases from less than 20% to more than 90%. This result clearly shows that the generated maps can provide significant performance improvement in perception tasks, especially in the dark.

Limitations of Existing IR-based Methods. IR-based techniques are mainstream solutions for providing detailed textures and sensing in the dark. Most iToF cameras provide intensity maps [16, 17], which are essentially the IR maps collected by IR cameras. However, IR/intensity maps represent the amplitude of received IR signals and cannot fully expose texture information due to the fact that they fail to account for the relationship between received signals and object textures. As Figure 4 shows, IR maps collected by IR and ToF cameras have the same key drawbacks. When an object is close to/far from the IR/ToF camera, its texture details will be overwhelmed by saturation/lost due to the extremely weak signal strength. In particular, adding IR power to sense distant objects is infeasible on battery-sensitive mobile platforms, such as smartphones.

We then examine the face detection rate on the IR images collected by ToF cameras and compare them with the maps generated by phase component manipulation. It turns out the detection rate of IR images is merely 2% while the rate of manipulated maps is more than 80%, which indicates that the performance of IR images is significantly limited in distance, while the manipulated maps suffer less from distance. Moreover, the IR maps collected by the IR and ToF cameras show the same properties. Therefore, we will not differentiate IR maps collected by IR camera and ToF camera in the rest of this paper unless otherwise indicated.

Summary. We now summarize the key observations on phase component manipulation during ToF measurement. First, the original depth maps are calculated using the phase components of the received IR signal. However, the transformation from phase components to depth suffers dimension reduction. In other words, the phase components contain more information than the depth map. Second, phase component manipulation can exploit such information and therefore expose more textures, which can be used to augment the performance of various depth applications in the dark. Moreover, phase component manipulation can overcome the key shortcomings of traditional IR images, including short sensing range due to rapid signal decay, significant noises, and the over-exposure effect.

4. APPLICATION SCENARIOS

Mozart exploits the phase components of infrared light during ToF measurements to expose high-resolution textures of the scene. As the infrared light is not interfered by visible light, *Mozart* can work in all light conditions. However, we focus on sensing in the low-light and dark environments in this work, since there currently does not exist a ubiquitous high-resolution vision technology in these challenging conditions, i.e., the counterpart of RGB camera in good lighting environments. The robust high-resolution sensing in the dark has many applications, e.g., longitudinal assessment of physical and mental health [30, 31, 32] of elders or babies. In general, *Mozart* can enable applications mainly in the following two manners.

Mozart-only applications. Thanks to its high-quality textures, the *Mozart* map alone can enable or augment various applications in the dark. For example, ToF-based face recognition would typically fail when the user's face is away from the ToF camera more than 0.8m, due to the excessive noise of depth measurement. In such cases, *Mozart* maps can be applied to augment face recognition, which is an essential function for smartphones, smart door locks and smart surveillance systems [1, 33]. Besides, *Mozart* on mobile phones can enable accurate facial expression recognition under all lighting conditions to monitor the user's emotional state, which enables a more natural user interface adaptive to the user's emotions [34]. Other representative applications in the dark include complex gesture recognition [2], security surveillance [14], robot navigation [6], etc. For instance, in a smart building embedded with depth ToF cameras on the wall, users can use gestures to control lights and other appliances, even in low-light and dark conditions.

Integration with depth map. The sensing outcome of *Mozart* can not only provide high-quality input for downstream applications, but also be integrated with depth

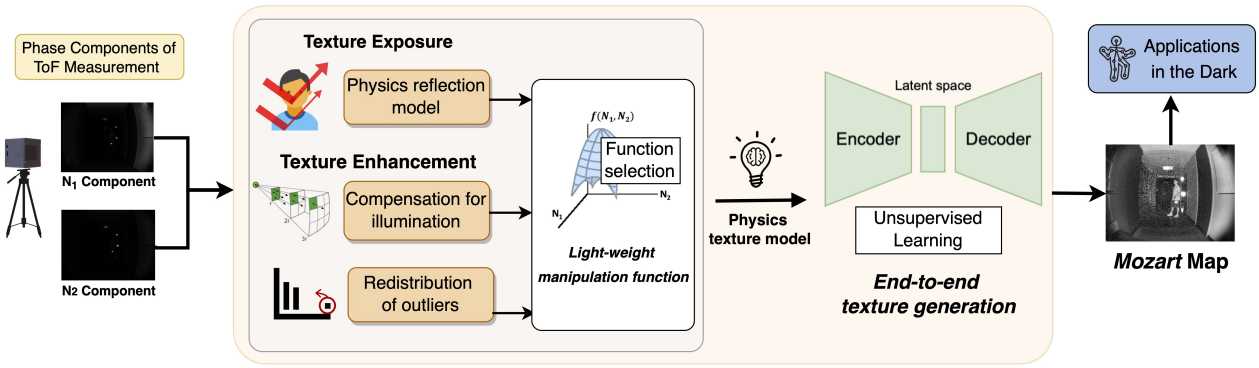


Figure 5: *Mozart* is designed based on the physics models for exposing and enhancing textures through phase manipulation. The high-resolution textures can be generated by both highly compute-efficient phase manipulation functions and an autoencoder-based approach.

maps to augment ToF-only 3D applications. First, *Mozart* maps provide a new mechanism of training machine learning models for depth data. Specifically, the high-quality texture details can generate accurate labels by utilizing CV algorithms, which can be used for quickly training models on fully aligned depth maps. Second, the accuracy of tasks can be improved by fusing the features of Mozart maps and depth maps. Finally, better 3D models [35] can be built with detailed textures of Mozart maps and the corresponding depth maps for ToF-only modules.

5. SYSTEM ARCHITECTURE

Mozart features a novel approach called *phase component manipulation*, which exploits effective mapping of the phase components during ToF measurements (i.e. N_1, N_2 in Eqn. (1)) to generate high-resolution texture of the scene. Figure 5 shows the system architecture of *Mozart*. Unlike other depth camera systems that obtain depth maps directly, *Mozart* takes advantage of the phase components (N_1, N_2) during ToF measurements. As the theoretical foundation of *Mozart* design, we first present an in-depth analysis on the relationship between phase components and exposed textures of the scene, based on the physical reflection model for the received IR light. Then we further introduce two techniques for enhancing the exposed texture map, including *redistribution of total reflection outliers* and *compensation for illumination attenuation*. Based on these analysis, our key finding is that the textures can be exposed and enhanced using highly compute-efficient phase manipulation functions. Lastly, we propose an end-to-end unsupervised learning approach, which employs autoencoder to automatically learn efficient representations from the phase component maps to generate *Mozart* maps. Specifically, the autoencoder neural network first converts the phase components into deep latent space and then reconstructs high-dimension *Mozart* maps from the deep embeddings. To train the deep autoencoder, we design three novel loss functions by exploiting the physics models for texture exposure and enhancement we proposed, including the *albedo similarity loss*, the *uniform distribution loss*, and the *illumination attenuation loss*. By combining these learning objectives, the autoencoder-based *Mozart* can effectively generate high-resolution texture maps for various scenes and applications. Our approach has several key advantages. First, the autoencoder is trained in an unsupervised manner, which does not require any manual labeling or reference images. Second, the autoencoder is scalable in generating various high-resolution *Mozart* maps, as it can be directly applied to different applications without manual system tuning.

6. METHODOLOGY

As introduced in Section 3.1, the phase components of ToF measurements vary for the points at the same distance, which essentially encode inherent information besides the depth maps. Therefore, our key idea for revealing texture information is to manipulate the phase components of ToF measurements for augmenting ToF sensing in the dark.

Next, we propose a first-principle physics model in Section 6.1 for exposing textures from ToF phase components and introduce two techniques for enhancing the revealed texture map during the manipulation, including *compensation for illumination attenuation* and *redistribution of total reflection outliers*. Based on the physics texture models, our key finding is that the textures can be exposed and enhanced using highly compute-efficient phase manipulation functions. Then we provide some examples of lightweight mapping functions and discuss the guidance in selecting effective functions for different applications. Lastly, we propose an end-to-end autoencoder-based texture generation approach by designing highly effective learning objectives of the autoencoder according to the physics models for texture exposure and enhancement, which automatically learns

efficient representations from N_1, N_2 maps.

6.1 Physics Model of Exposing Textures

In this section, we show how to expose and enhance textures from ToF phase components based on the physical reflection model of IR light.

6.1.1 Modeling Textures Using Received IR Light

As shown in Section 3.1, the IR light is emitted by the ToF camera, reflected by objects in the scene, and finally received by the lens of the ToF camera. Therefore, the texture information of the scene is encoded in the intensity of the received IR signal. Moreover, in most cases, the IR light emitted by the ToF camera will be *diffusely reflected* on the surface of the object. According to the Lambertian reflection model (Figure 6a) [36], the amount of received IR light reflected by the object at the distance d can be calculated by:

$$E_d = \frac{E_0 \alpha \cos\theta}{8d^2} = \frac{E_0 \beta}{8d^2}, \tag{2}$$

where E_0 is a constant determined by the emission power of the ToF camera, α is the reflectivity of the object and θ is the angle of incidence. Therefore, we can see that the intensity of received light is determined by objects' reflectivity α, the incidence angle θ, and the distance d. The former two factors together form the textures of objects [37]. Therefore, in this paper, we define a new variable "*albedo*" $\beta = \alpha \cos\theta$, to quantify the two factors from the object itself that have an impact on the intensity of the received light.

Based on Eqn. (2) and the physical meaning of N_1, N_2 (see Section 3.1), we can establish the relationship between the phase components and the texture information β:

$$N_1 = \frac{E_0}{8D} \cdot \frac{\beta(D-d)}{d^2}, \quad N_2 = \frac{E_0}{8D} \cdot \frac{\beta}{d}, \tag{3}$$

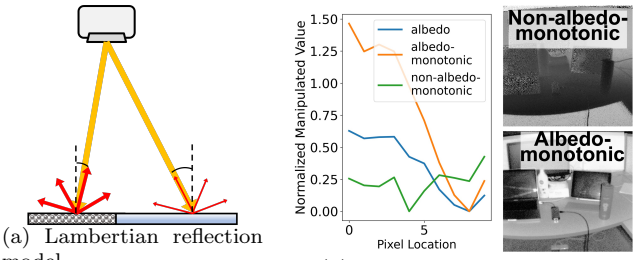
where D is the ToF camera's range of measurement. Note that E_0, D are all constants, thus N_1, N_2 are functions of albedo β and distance d, namely $N_1 = n_1(\beta, d), N_2 = n_2(\beta, d)$. Therefore, we can manipulate N_1, N_2 to expose texture information of the scene β for augmenting ToF sensing in the dark.

6.1.2 Manipulating Phase Components to Expose Textures

Next, we show how to manipulate the phase components N_1, N_2 to effectively expose texture β. We formulate the phase component manipulation problem as a mapping from a 2-D vector to a scalar as:

$$f(N_1^i, N_2^i) \to S_i, \tag{4}$$

where i is the index of the pixel in the map, and S_i is the mapped scalar in the generated map. The function $f(\cdot)$ will be applied to every pixel i in the whole map.



(a) Lambertian reflection model.

(b) Pixel value over the map.

Figure 6: (a) The reflected IR intensity is determined by both the reflectivity and the incidence angle of light. These two factors together form the textures of objects, which we refer to as albedo β. (b) The mapping functions that are albedo-monotonic can well expose textures of the scene and vice versa.

The original depth maps do not contain detailed texture information because points with the same distance d but different albedos β can not be differentiated. Therefore, to expose detailed texture information, the phase components mapping $f(\cdot)$ should: (1) make the points at the same distance have different values; (2) keep the same order as albedos β. Therefore, for any two points **A** and **B** with the same distance d in the scene, if $\beta_A < \beta_B$, the mapping should have $f(N_1^A, N_2^A) < f(N_1^B, N_2^B)$. That is $\frac{\partial f(N_1, N_2)}{\partial \beta} > 0$. Combining this with Eqn. 3, we have the following constraints of the phase manipulation mapping:

$$\frac{\partial f(N_1, N_2)}{\partial N_1}(D-d) + \frac{\partial f(N_1, N_2)}{\partial N_2} d > 0 \tag{5}$$

Eqn. (5) ensure that the transformed result $f(N_1, N_2)$ is monotonically increasing in terms of albedo β at a given distance d, which we refer to as *albedo-monotonic*. Such monotonicity keeps the same structure in the transformed map as the albedo map, thus exposing detailed textures without introducing any artifacts (see Figure 6b). It is worth mentioning that if the inequalities in Eqn. (5) are completely opposite to the current ones, the monotonicity will still hold. However, the texture structure in the transformed map will be reversed to albedo, resulting in "negative images" [38]. For example, negative images are useful for enhancing white or grey detail embedded in dark regions of an image. Moreover, note that Eqn. (5) is a sufficient but not necessary condition, which means all transformations that meet this constraint can effectively expose textures.

6.1.3 Compensation for Illumination Attenuation

According to Eqn. (3), the phase components $N_1 = n_1(\beta, d)$ and $N_2 = n_2(\beta, d)$ decrease with the distance. Therefore, as shown in Figure 7, in a typical map generated by phase manipulation, the near objects will be much brighter than distant objects, reducing the utility of distant points of the image. More specifically,

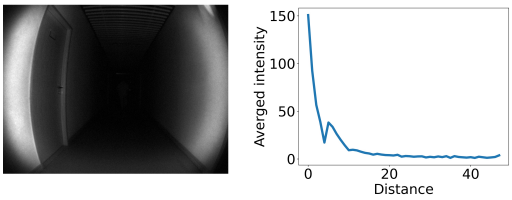


Figure 7: Left: In a typical texture map, the near region is brighter while the far region is dark. Right: The averaged intensity of received light decreases drastically with the distance.

the averaged intensity of received light decreases drastically with the distance. Therefore, in this section, we propose to compensate for the illumination attenuation introduced by longer distances to further enhance the exposed textures during phase manipulation. To achieve this, we can remove d from Eqn. (3) and obtain the following function:

$$g(N_1, N_2) = \frac{N_2^2}{N_1 + N_2} = \frac{E_0}{8D^2} \cdot \beta, \tag{6}$$

where E_0, D are all constants, which means that the transformed result by Eqn. (5) will not be affected by the distance while only related to the texture information β. Moreover, $g(N_1, N_2)$ strictly satisfies the constraints of exposing textures in Eqn. (5). Therefore, the function $g(N_1, N_2)$ can correct illumination attenuation introduced by the distance of objects d while enhancing the detailed textures of the scene.

6.1.4 *Redistribution of Total Reflection Outliers*

Based on the physics model proposed in Section 6.1.2, we are able to expose texture information through ToF phase manipulation. In practice, there will always exist *total reflection* on the surface of certain objects, such as metals and glass. Therefore, the points with total reflection will not satisfy the Lambertian reflection model in Eqn. (2), and the corresponding N_1, N_2 received by the ToF camera will far exceed the normal values. For example, a typical value of N_2 is less than 100, while N_2 value of total reflection can be more than 1,000. Then the mapping results of these outliers through the non-decreasing functions defined in Eqn. (5) will also exceed the normal range. As shown in Figure 8, if we normalize the texture map to grayscale, most of the pixels will be squeezed into a small range near 0, while the points with total reflection exhibit isolated bright spots. Therefore, in this section, we introduce how to enhance the exposed textures during phase manipulation by redistributing the total reflection outliers.

To alleviate the impact of outliers introduced by total reflections, we should expand the dense value around 0 and compress the sparse outliers with large values. To achieve this, we add a new mapping $r(S)$ outside the function $f(\cdot)$ defined in Eqn. (5), where S denotes the

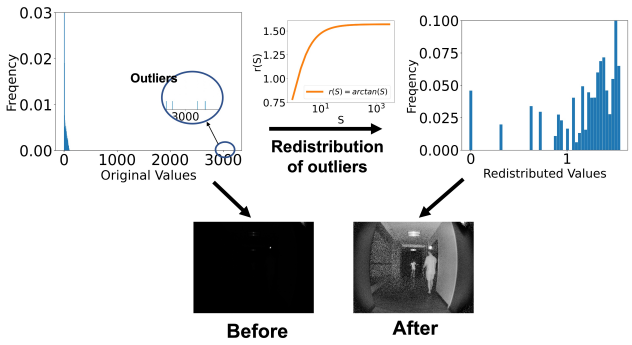


Figure 8: Redistribution of outliers reduces the influence of total reflection and reveals more textures.

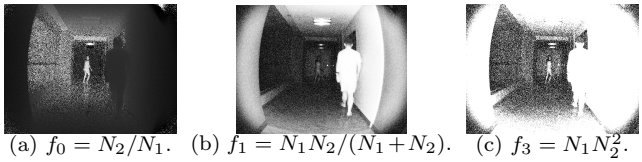
transformed result of $f(N_1, N_2)$. Here the continuous function $r(S)$ must be monotonically increasing, and the rate of increase is getting smaller with the pixel value S. Figure 8 shows an example of the redistribution function r(S), where the dense distribution at 0 can be expanded, and the sparse distribution at larger values can be squeezed. Therefore, $r(S)$ can limit the bound of higher outlier values. The lower part of Figure 8 shows the texture map before and after the redistribution of phase manipulation, where the manipulated map after redistribution has a substantially higher contrast and more uniform distribution of pixel values.

6.2 Light-weight Manipulation Functions

Now we introduce how to design functions $f(N_1, N_2)$ based on the proposed physics model in Section 6.1 in practice to expose textures.

Selecting functions for texture exposure. We first compare the generated maps using the following functions with different polynomial degrees with respect to N_1, N_2: $f_0(N_1, N_2) = \frac{N_2}{N_1}, f_1(N_1, N_2) = \frac{N_1 \cdot N_2}{(N_1+N_2)}, f_3(N_1, N_2) = N_1 N_2^2$, where the polynomial degrees of f_0, f_1, f_3 are 0, 1, 3, respectively, and f_1, f_3 satisfy Eqn. (5) while f_0 does not. Figure 9 shows the generated maps by applying f_0, f_1, f_3 to the phase components, respectively. We observe that f_0 cannot expose textures of the scene while both f_1, f_3 can, which is consistent with the principle we proposed in Section 6.1. Moreover, the generated map f_3 exhibits an over-exposure effect, thus reducing the quality of exposed textures, which indicates that the functions with larger polynomial degrees amplify many normal values to larger values. Therefore, besides satisfying Eqn. (5), an effective function $f(\cdot)$ to expose textures should not have a large polynomial degree with respect to N_1, N_2.

Selecting functions for illumination compensation. To compensate the illumination attenuation of the texture maps, the functions $f(N_1, N_2)$ should be the variant of $g(N_1, N_2) = N_2^2/(N_1 + N_2)$ or itself. Moreover, based on the observations in Section 6.2, the degree of polyno-



(a) $f_0 = N_2/N_1$. (b) $f_1 = N_1N_2/(N_1+N_2)$. (c) $f_3 = N_1N_2^2$.

Figure 9: The functions with large polynomial degrees will turn normal values into outliers, resulting in over-exposure.

mials for the manipulation function should not be too large. Therefore, to compensate the illumination attenuation during phase manipulation, we can choose to use $f(N_1, N_2) = g(N_1, N_2)$ directly or a linear transformation of $g(N_1, N_2)$ in most common scenarios.

Selecting Redistribution Functions. Now we show how different redistribution functions $r(S)$ affect the generated maps. Here we give three examples of redistribution functions that satisfy the requirements defined in Section 6.1.4, including $r_1(S) = \sqrt{S}$, $r_2(S) = ln(S)$ and $r_3(S) = arctan(S)$. For the same outlier $S_0 = 3,000$, $r_1(S_0) = 54.7, r_2(S_0) = 8.00, r_3(S_0) = 1.57$. We can see that different functions $r(S)$ have different redistribution performances for larger pixel values introduced by total reflection outliers. For those scenarios where the total reflection is strong, we can select functions like $r_3(S)$ for better redistribution of the total reflection outliers. On the contrary, we can choose a more mild function like $r_1(S)$ so that the distribution will not affect the pixels with normal values.

6.3 Autoencoder Network for Texture Generation

The function-based texture generation in Section 6.2 requires careful design and manual tuning for different applications, which is labor-intensive and requires substantial domain expertise. Therefore, we propose an end-to-end autoencoder-based texture generation approach, which automatically learns efficient representations from N_1, N_2 maps. Our key idea is to utilize the physics models for texture exposure and enhancement proposed in Section 6.1 to design highly effective learning objectives of the autoencoder.

Our approach has several key advantages. First, the autoencoder neural network is trained in an unsupervised manner, which does not require manual labeling or reference images, in contrast to previous supervised image enhancement solutions [39]. Second, by exploiting the physics models for texture exposure and enhancement proposed in Section 6.1, we design several effective loss functions to train the deep autoencoder neural network. As a result, the autoencoder can output high-resolution texture maps without introducing artifacts or large noises. Third, compared with the manually crafted manipulation functions, the autoencoder network is more scalable in generating high-resolution

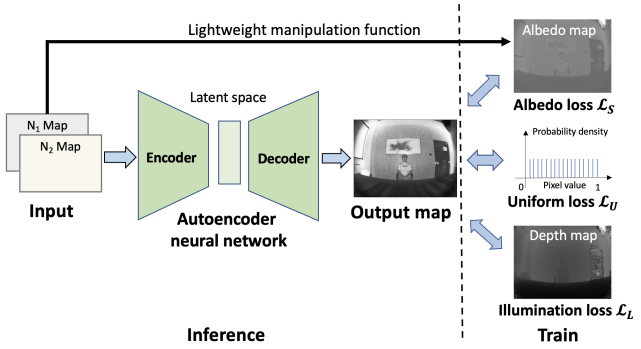


Figure 10: The autoencoder can learn efficient representations from N_1, N_2 maps to generate robust *Mozart* maps for different applications. The loss functions are designed based on the physics models in Section 6.1.

Mozart maps for different applications. For example, for the two applications (e.g., human tracking and gesture recognition) with different outlier distributions or IR illumination attenuation effects, the autoencoder-based texture generation can be directly applied without any modification or manual system tuning.

6.3.1 Autoencoder Design

Autoencoder is a widely-used unsupervised learning approach in computer vision tasks that can learn efficient features from unlabeled data [40, 41]. We use a deep autoencoder neural network as a generative model to adaptively generate high-resolution *Mozart* maps from N_1, N_2 maps. As shown in Figure 10, there are two main components in the autoencoder neural network, including the encoder and the decoder network. During the training of the deep autoencoder, the phase component N_1, N_2 maps will be input to the neural network. Then the encoder network maps the input N_1, N_2 maps into deep latent space, and the decoder network reconstructs high-dimension *Mozart* maps from the deep embeddings. In this way, the autoencoder neural network can learn invariant features underlying the N_1, N_2 maps collected from different scenarios [42]. We use a 3D-CNN for the encoder and decoder to explore the inter-channel relationships between N_1, N_2 maps. Finally, the output maps will be used to calculate the unsupervised training loss, where the loss functions are designed based on the physics models for texture exposure and enhancement proposed in Section 6.1.

The goal of training the autoencoder neural network is to automatically learn efficient representations from N_1, N_2 maps to generate high-resolution texture maps. Therefore, similar to the lightweight mapping function in Section 6.2, the autoencoder is trained to exploit efficient manipulations to the phase components for exposing texture information.

6.3.2 Design of Loss Functions

To train an autoencoder neural network that can gen-

erate high-resolution texture maps, we carefully design three loss functions according to the physics models proposed in Section 6.1, including the *albedo similarity loss*, the *illumination attenuation loss*, and the *uniform distribution loss*. Suppose $\mathbf{P}$ denotes the *Mozart* output of the autoencoder neural network. We design the three loss functions as follows.

Albedo similarity loss. Unlike the lightweight phase manipulation functions that can maintain the pixel topology of N_1, N_2 maps, the neural network-based method is more like a black box, which may generate artificial textures that do not exist in the actual scene. As shown in Section 6.1, the albedo map calculated by the manipulation function $f(\cdot)$ contains textures of the scene. Therefore, we use the structural similarity [43] between the *Mozart* output and the corresponding albedo map to guide the training of the *Mozart* Autoencoder model. Suppose $\mathbf{B}$ denotes the albedo map, then the albedo similarity loss is calculated as follows:

$$\mathcal{L}_S = 1 - S(\mathbf{P}, \mathbf{B}), \tag{7}$$

where $S(\mathbf{X}, \mathbf{Y}) \in [0, 1]$ denote the structural similarity index of two images $\mathbf{X}$ and $\mathbf{Y}$. A larger $S(\mathbf{X}, \mathbf{Y})$ means more similarity between the map $\mathbf{X}$ and $\mathbf{Y}$.

Illumination compensation loss. As shown in Section 6.1.3, the phase components N_1 and N_2 decrease with the distance, making the near objects much brighter than distant objects. Therefore, we propose an illumination compensation loss to penalize the high-intensity pixels near the ToF camera.

By design, the depth maps have larger distance values for distant objects and smaller distance values for close objects. Therefore, the depth maps (or maps that are positively correlated to distance) can serve as a reference kernel to correct the uneven light field of *Mozart* output. Moreover, as the depth maps usually have lots of noises that can affect the quality of generated *Mozart* maps, we use the denoised depth maps $\mathbf{D}$ after median filter [44] to calculate the light compensation loss:

$$\mathcal{L}_L = \mathbf{P} \bigotimes \mathbf{D}, \tag{8}$$

where $\bigotimes$ means element-wise multiplication.

Uniform distribution loss. As shown in Section 6.1, the outlier values introduced by total reflection will distort the distribution uniformity. However, when the histogram of a map is uniform across the entire value range, the map will have a higher contrast [45]. This feature is also friendly to many existing object detection and tracking algorithms [46, 47]. Therefore, consistent with the redistribution function in Section 6.1.4, we design a uniform distribution loss by minimizing the negative histogram entropy for the output map:

$$\mathcal{L}_U = \sum_{c=0}^{255} p_c(\mathbf{P}) \log p_c(\mathbf{P}), \tag{9}$$

Phone Model	ToF Camera(s)	ToF API	Resolution
Huawei P30 Pro	rear	AREngine	240 × 180
Huawei Mate30 Pro	rear & front	AREngine	240 × 180
Samsung S20 Ultra	rear	ARCore	640 × 480

Table 2: Summary of the mobile phones with *Mozart* implemented.

where we change the generated map to grayscale with the value range [0, 255] and $p_c(\cdot)$ is the normalized histogram counts of value c for a map.

Overall Training Loss Function. Putting the above three loss functions together, the overall loss function for training the autoencoder neural network is:

$$\mathcal{L} = \lambda_s \mathcal{L}_S + \lambda_l \mathcal{L}_L + \lambda_u \mathcal{L}_U \tag{10}$$

Here λ_s, λ_l, λ_u are the coefficients that weight the contribution of each loss function and can be adjusted easily in different applications. For example, when the depth maps are very noisy, we can set a smaller λ_l to reduce the impact of depth on the light compensation, leading to smaller noise in the generated *Mozart* maps.

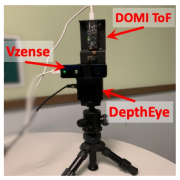
7. SYSTEM IMPLEMENTATION AND OVERHEAD

7.1 Implementation on Various Platforms

Smartphones Platforms. A number of smartphones (e.g., Huawei P/Mate series, Samsung S/Note series) are equipped with ToF cameras for various applications such as FaceID, In-Air Gesturing, and AR/VR. We first implement *Mozart* on three off-the-shelf smartphones with embedded ToF cameras, whose specifications are shown in Table 2. To demonstrate the real-time performance of Mozart, we built an Android App that can help users identify objects under dark environments. The demo video (`https://youtu.be/qBEffXVft_8`) shows that, compared with depth maps, the Mozart App can provide substantially more texture details in the dark environment in real-time.

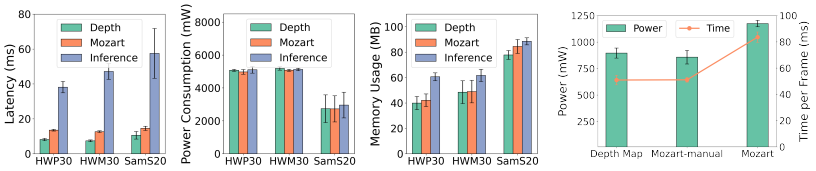
The raw phase components from ToF cameras are not available on Android smartphones. To address this challenge, we calculate the phase components indirectly from depth and intensity maps (i.e., the confidence maps in Android documentation), which can be obtained from all Android smartphones using *Camera2 API* [48]. Moreover, smartphones from a few manufacturers provide dedicated ToF APIs to access the ToF data. For example, *AREngine* [49] available on Huawei devices can provide 3-bit confidence maps and 13-bit depth maps. *ARCore* [50] available on Google-certified models such as Samsung S20 Ultra can provide 8-bit confidence maps and 16-bit depth maps. After obtaining the phase components, we calculate the *Mozart* maps using manually designed functions on the phones for downstream applications. *Mozart* on smartphones is implemented using

(a) Face emotion App.



(b) Three ToF modules.

Figure 11: Implementation of *Mozart* with smartphones and standalone ToF cameras.



(a) Generating depth, *Mozart* maps, and inference with *Mozart* maps on smartphones.

(b) Generating maps on Jetson Xavier.

Figure 12: System overhead on smartphone and edge platforms.

Java on Android Studio. Note that the latency of generating a single *Mozart* map on all the three smartphones is merely around 15 ms (see Figure 12a), thus enabling real-time applications.

Edge Platforms. We also implement *Mozart* with three mainstream standalone ToF depth cameras, including DMOM2508 [51], Vzense DCAM710 [52], and DepthEye Wide ToF [53], on Nvidia Jetson Xavier [54]. The DepthEye ToF camera adopts an IMX556PLR CMOS from Sony, from which we can easily obtain the phase components of ToF measurements directly via APIs. The phase components of the other two ToF cameras need to be calculated indirectly from depth maps and intensity maps. *Mozart* on Nvidia Xavier (running Ubuntu 18.04) is implemented using C++ and Python.

On edge platforms, we implement two different texture generation approaches, including *Mozart-manual* where we manually select the best functions for different applications according to the principles proposed in Section 6.2, and *Mozart* where we train the autoencoder neural network using the loss functions defined in Section 6.3. We note that *Mozart-manual* requires substantial efforts and domain expertise to choose a proper function in a trial and error manner. Moreover, such a labor-intensive process must be repeated for different applications. On the contrary, the proposed autoencoder can automatically learn efficient representations from phase components to generate texture maps while achieving similar performance with *Mozart-manual*.

7.2 System Overhead

In this section, we evaluate the system overhead of *Mozart* on three smartphones and on Nvidia Jetson Xavier with three mainstream standalone ToF modules. First, we developed a smartphone App to classify facial expressions from a typical expression set [55] (i.e., angry, disgust, fear, happy, sad, surprise, and neutral) under different illumination conditions[2]. Results show that *Mozart* on smartphones outperforms depth maps by 20% and IR maps by 40% in mean recognition accuracy. We further implemented an object detection task on Nvidia Xavier with the three ToF modules. The results indicate that the performance improvement of *Mozart* maps is significant for all ToF cameras. Specifically, on the Vsenze ToF camera, *Mozart* outperforms depth and IR maps by 63.76% and 45.28%, respectively.

System Overhead on Smartphones. We compare the system overhead of *Mozart-manual* with the native ToF system that generates depth maps from DEPTH16 data [56] in terms of latency, power consumption, and memory usage. We use PerfDog [57] developed by Tencent to collect the power consumption and memory usage of each smartphones. The results are shown in Figure 12a. First, the latency of calculating a single Mozart map on mobile phones is smaller than 14.5 ms, which can easily support real-time applications with a frame rate of 30 fps. Moreover, calculating *Mozart* maps does not significantly increase any resource consumption, including power consumption and memory usage.

System Overhead on Edge Devices. We compare *Mozart* against *Mozart-manual* and the native depth system that generates depth maps from ToF phase components. As discussed earlier, *Mozart-manual* is designed by an extensive search of compute-efficient texture generation functions, which requires substantial efforts and domain expertise. As a result, the compute overhead of *Mozart-manual* at runtime is expected to be smaller than autoencoder-based *Mozart*. During the end-to-end experiments, we measure the averaged computation time and power consumption for obtaining one map frame. The power consumption is obtained using tegrastats [58] provided by Nvidia.

The results are shown in Figure 12b. First, *Mozart-manual* outperforms the native depth system in both computation time and power consumption. This shows that the phase manipulation functions we designed for *Mozart-manual* are more efficient than the built-in transformation of phase components to depth maps. Specifically, *Mozart-manual* produces each frame in merely 43.5ms, i.e., at a rate about 23 fps, which allows it to be executed in real-time on embedded platforms. The autoencoder-based *Mozart* takes more time and power consumption, while it can still achieve about 12 fps, which is acceptable in most depth applications [59, 12]. We note that the system overhead can be further reduced by various existing techniques [60], including optimizing the computation pipeline in a hardware-

[2]All the data collection involving human subjects was approved by IRB of the authors' institution.

software co-optimization and adopting sparse autoencoder, which is left for future work.

8. COMPARISON WITH OTHER SENSOR MODALITIES

In this section, we compare *Mozart* with the baseline systems that employ the following sensor modalities: ***RGB***, ***RGB-enhanced***, ***Depth***, ***IR*** and ***mmWave Radar***, using three new real-world datasets collected in dark environments. The RGB-enhanced maps are generated by Zero-DCE [20], a state-of-the-art image enhancement approach from computer vision literature.

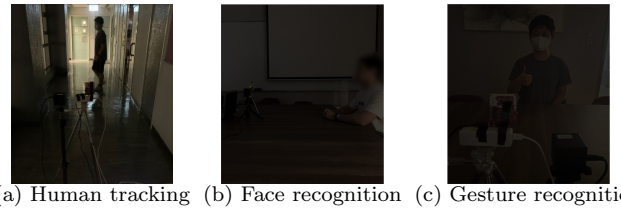
8.1 Data Collection in the Dark

The reasons for collecting the new datasets are as follows. First, existing depth camera-based datasets do not contain corresponding samples of other sensor modalities. Besides, there are few multimodal datasets (including RGB/Depth/IR/Radar) collected under dark environments, which is the main application scenario of *Mozart*. The three datasets consist of over 1,000,000 frames with a total of 33 subjects.

For a fair comparison, we collect the data of all these sensors and the ToF phase components simultaneously at 10 Hz. The RGB images are collected using the Vzense camera [52] and the IR/depth images are collected using the DepthEye ToF camera [53]. The radar point clouds are collected using a 60-64GHz mmWave radar TI IWR6843 [61]. We convert the IR, RGB, RGB-enhanced, Depth and *Mozart* maps to the same dimension (640, 480) and apply the same models to them for each task. We convert each frame of radar points into voxels of a fixed dimension (2, 16, 32, 16) and apply existing detection or classification models in each task.

Human tracking dataset. Continuous human tracking in the dark is a typical task for applications like security surveillance, which need to capture high-resolution textures of the scene. As shown in Figure 13a, we collect a real-world human tracking dataset under low-light conditions in three different environments (i.e., square, room, and corridor). The volunteers are asked to walk freely within 10m from the sensors. For each environment, we collect data under both single-person settings and multi-person settings (where 2, 3 and 4 persons appear simultaneously). In total, we collect over 8,000 frames from nine subjects for each modality.

Face recognition dataset. Face recognition in the dark is important for user authentication, e.g., for smartphones or smart doors. However, existing technologies such as Apple's FaceID can only work in close range (e.g., 0.8m) [62]. In this case, *Mozart* can boost the performance at a significantly longer distance. As shown in Figure 13b, we collect a face recognition dataset in a dark room. We recruit 12 volunteers and ask them to sit in front of the sensors at a distance of 1m (the near setting) and 2m (the far setting), respectively. We also collect data under four different illumination conditions by adjusting the lights in the room. We totally collect over 15,000 data frames of data for each modality.



(a) Human tracking (b) Face recognition (c) Gesture recognition

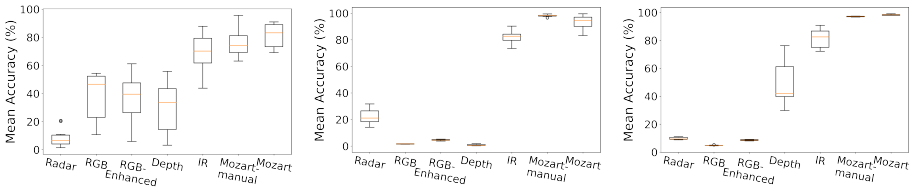
Figure 13: Experiment settings of different datasets. The photos are taken using iPhone XR camera with default mode.

Hand gesture recognition dataset. Hand gesture recognition is important in human-computer interaction applications (e.g., controlling smart home appliances). However, due to the small dimensions of hand gestures, it is extremely difficult to achieve a robust performance in the dark. As shown in Figure 13c, we collect a hand gesture dataset in a dark room. We recruit 12 volunteers and ask them to perform 20 different gestures, including calling, dislike, like, victory, fist, ok, one, three, four, palm, rock, stop, mute, crossed fingers, no, pause, grabbing, gun, pointing, and holding up. The gestures are collected at a distance of 1.5m from all sensors.

8.2 Accuracy in Different Applications

We first compare *Mozart* against different sensing approaches for the three datasets in the dark. For the human tracking task, we use a widely-used object detector YOLOv5 [29] to detect the persons in each frame, which will output the predicted objects and the confidence of prediction for each object. However, this evaluation is not applicable for radar point clouds as they do not support semantic-based tasks due to the data sparsity. Therefore, the radar baseline in this experiment only tracks moving objects in the scene without detecting the type of objects (i.e., the person). For the face recognition task, we first detect the faces in the image maps using a pre-trained RetinaFace detector [63]. Then a pre-trained ArcFace [64] model is used to transform the detected faces areas to 512-dimension feature vectors. For voxels of radar data, we train a 3D-CNN model to classify the faces of 12 different people. To recognize the hand gestures, we first detect and localize the hands by applying the MediaPipe [65] to the RGB, depth, IR and *Mozart* maps. Then we input the cropped hands area into a lightweight 2D-CNN model to classify the gestures. The radar voxels are trained using a 3D-CNN model directly to classify 20 different gestures.

The results of different datasets are shown in Figure 14. First, the baselines perform very poorly for applications in the dark. For example, the RGB and depth



(a) Human tracking (b) Face recognition (c) Gesture recognition

Figure 14: Overall accuracy on different datasets. Both *Mozart* and *Mozart-manual* consistently outperform the baselines.

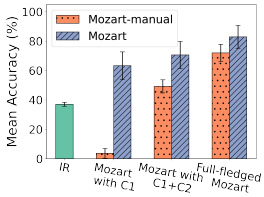


Figure 15: Ablation Study.

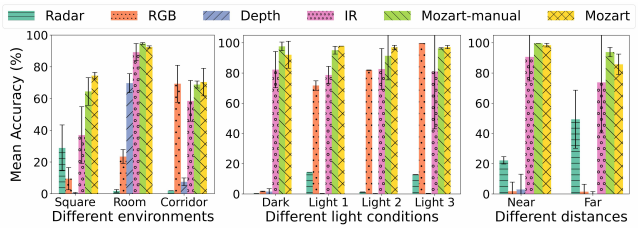


Figure 16: Performance comparison in different environments, light conditions and distances, respectively.

images only achieve 1.66% and 1.54% mean accuracy in the face recognition task. Second, both *Mozart* and *Mozart-manual* consistently outperform other baselines in different datasets with various settings. Moreover, *Mozart* can approach or even surpass the performance of the manually designed *Mozart-manual* in different datasets, which shows that the autoencoder texture generation is robust and scalable for different applications.

8.3 Performance in Dynamic Conditions

In this section, we evaluate the effectiveness of *Mozart* under dynamic conditions, including different environments, illumination conditions and distances of objects. The results are shown in Figure 16.

Different Environments. We first evaluate the impact of different environments in human tracking, including the square, room and corridor. First, both *Mozart-manual* and *Mozart* have a robust performance across different environments, consistently outperforming all baselines. Radar performs extremely poorly in rooms/corridors due to significant multi-path effects. RGB images have a extremely low accuracy for all environments in the dark. Depth and IR maps perform poorly in the squares where people walks in a large range (e.g., 5-10m).

Different Light Conditions. We then evaluate the impact of different light conditions in face recognition, where Light 1, 2, 3 have sequentially increasing ambient light intensity. First, the performance of *Mozart* and *Mozart-manual* are very stable under different light conditions and outperforms all baselines except for RGB images under Light 3 (full illumination). Besides, depth, IR and radar also yield a stable performance under different light conditions as they do not rely on ambient light. However, the performance of RGB drops drastically with lower light levels since it is highly susceptible to non-ideal environmental light conditions.

Different Distances of Objects. Lastly, we evaluate the impact of different distances of objects in face recognition. Almost all the approaches perform worse in the far setting. This is expected as the face area will be smaller in the maps. However, both *Mozart-manual* and *Mozart* only suffer subtle performance degradation and always outperform the baselines.

8.4 Ablation Study

In this section, we evaluate the effectiveness of different design components for texture generation based on the human tracking task. Both *Mozart-manual* and *Mozart* are implemented using different components combinations. Specifically, *exposing textures based on the physics model* is denoted as C1, *redistribution of total reflection outliers* is denoted as C2 and *compensation for illumination attenuation* is denoted as C3.

The results are shown in Figure 15. First, the *Mozart-manual* with C1+C2 already shows significant accuracy improvement. Second, the results with different loss combinations of *Mozart* all show significant accuracy improvement (i.e., at least 20%) over IR maps. Lastly, the performance of *Mozart* is more stable than *Mozart-manual* under different configurations, which shows the robustness of autoencoder-based texture generation that exploits the physics texture models.

9. CONCLUSION

In this paper, we present *Mozart*, a new sensing system that leverages off-the-shelf ToF depth camera to generate high-resolution and rich-in-texture maps for low-light and dark scenes. Extensive experiments show that *Mozart* significantly outperforms existing sensing technologies and can work on smartphones and edge platforms in real-time. In the future, we will study how to combine the native depth maps and *Mozart* maps for advanced 3D sensing applications such as 3D reconstruction of dark environments.

References

[1] Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Earlence Fernandes,

and Blase Ur. Rethinking access control and authentication for the home internet of things. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 255–272, 2018.

[2] Xiulong Liu, Dongdong Liu, Jiuwu Zhang, Tao Gu, and Keqiu Li. Rfid and camera fusion for recognition of human-object interactions. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 296–308, 2021.

[3] Anran Wang, Jacob E Sunshine, and Shyamnath Gollakota. Contactless infant monitoring using white noise. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[4] Yanzi Zhu, Yuanshun Yao, Ben Y Zhao, and Haitao Zheng. Object recognition and navigation using a single networking device. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 265–277, 2017.

[5] Kun Qian, Zhaoyuan He, and Xinyu Zhang. 3d point cloud generation with millimeter-wave radar. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–23, 2020.

[6] Chris Xiaoxuan Lu, Muhamad Risqi U Saputra, Peijun Zhao, Yasin Almalioglu, Pedro PB De Gusmao, Changhao Chen, Ke Sun, Niki Trigoni, and Andrew Markham. milliego: single-chip mmwave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 109–122, 2020.

[7] Linjie Gu, Zhe Yang, Mithun Mukherjee, Zhigeng Pan, Mian Guo, Xiushan Liu, Rakesh Matam, and Jaime Lloret. Hawk-i: a remote and lightweight thermal imaging-based crowd screening framework. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 885–887, 2021.

[8] Ke Sun, Wei Wang, Alex X Liu, and Haipeng Dai. Depth aware finger tapping on virtual displays. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 283–295, 2018.

[9] H Andrei, V Ion, E Diaconu, A Enescu, and I Udroiu. Energy consumption analysis of security systems for a residential consumer. In *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–4. IEEE, 2019.

[10] Jufeng Zhao, Yueting Chen, Huajun Feng, Zhihai Xu, and Qi Li. Infrared image enhancement through saliency feature analysis based on multi-scale decomposition. *Infrared Physics & Technology*, 62:86–93, 2014.

[11] Wikipedia. Inverse-square law, 2022. `https://en.wikipedia.org/wiki/Inverse-square_law`.

[12] Zhiyuan Xie, Xiaomin Ouyang, Xiaoming Liu, and Guoliang Xing. Ultradepth: Exposing high-resolution texture from depth cameras. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 302–315, 2021.

[13] Li Lu, Jiadi Yu, Yingying Chen, Hongbo Liu, Yanmin Zhu, Yunfei Liu, and Minglu Li. Lippass: Lip reading-based user authentication on smartphones leveraging acoustic signals. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1466–1474. IEEE, 2018.

[14] Lei Yang, Qiongzheng Lin, Xiangyang Li, Tianci Liu, and Yunhao Liu. See through walls with cots rfid system! In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 487–499, 2015.

[15] Xian Shuai, Yulin Shen, Yi Tang, Shuyao Shi, Luping Ji, and Guoliang Xing. millieye: A lightweight mmwave radar and camera fusion system for robust object detection. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pages 145–157, 2021.

[16] Robert Lange and Peter Seitz. Solid-state time-of-flight range camera. *IEEE Journal of quantum electronics*, 37(3):390–397, 2001.

[17] Marvin Lindner and Andreas Kolb. Compensation of motion artifacts for time-of-flight cameras. In *Workshop on Dynamic 3D Imaging*, pages 16–27. Springer, 2009.

[18] Ziteng Cui, Guo-Jun Qi, Lin Gu, Shaodi You, Zenghui Zhang, and Tatsuya Harada. Multitask aet with orthogonal tangent regularity for dark object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2553–2562, 2021.

[19] Ishani Janveja, Akshay Nambi, Shruthi Bannur, Sanchit Gupta, and Venkat Padmanabhan. Insight: monitoring the state of the driver in low-light using smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–29, 2020.

[20] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1780–1789, 2020.

[21] HyeonJung Park, Youngki Lee, and JeongGil Ko. Enabling real-time sign language translation on mobile platforms with on-board depth cameras. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–30, 2021.

[22] Yunfan Zhang, Tim Scargill, Ashutosh Vaishnav, Gopika Premsankar, Mario Di Francesco, and Maria Gorlatova. Indepth: Real-time depth inpainting for mobile augmented reality. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–25, 2022.

[23] Benjamin Langmann, Klaus Hartmann, and Otmar Loffeld. Increasing the accuracy of time-of-flight cameras for machine vision applications. *Computers in industry*, 64(9):1090–1098, 2013.

[24] Supreeth Achar, Joseph R Bartels, William L'Red' Whittaker, Kiriakos N Kutulakos, and Srinivasa G Narasimhan. Epipolar time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(4):1–8, 2017.

[25] Seung-Hwan Baek, Noah Walsh, Ilya Chugunov, Zheng Shi, and Felix Heide. Centimeter-wave free-space neural time-of-flight imaging. *ACM Transactions on Graphics (TOG)*, 2022.

[26] DayDayNews. The civil war for tof technology is far from over, 2020. `https://daydaynews.cc/en/technology/683608.html`.

[27] Richard LIU Chenmeijing LIANG, Pierre CAMBOU. Status of the cmos image sensor industry 2020, 2020. `https://s3.i-micronews.com/uploads/2020/11/YDR20106-Status-of-the-CMOS-Image-Sensor-Industry-2020_sample.pdf`.

[28] S Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor-system description, issues and solutions. In *2004 conference on computer vision and pattern recognition workshop*, pages 35–35. IEEE, 2004.

[29] Yolov5, 2022. `https://pytorch.org/hub/ultralytics_yolov5/`.

[30] Tian Hao, Guoliang Xing, and Gang Zhou. isleep: Unobtrusive sleep quality monitoring using smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, 2013.

[31] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 54–66, 2021.

[32] Xianda Chen, Yifei Xiao, Yeming Tang, Julio Fernandez-Mendoza, and Guohong Cao. Apneadetector: Detecting sleep apnea with smartwatches. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(2):1–22, 2021.

[33] Weijia He, Valerie Zhao, Olivia Morkved, Sabeeka Siddiqui, Earlence Fernandes, Josiah Hester, and Blase Ur. Sok: Context sensing for access control in the adversarial home iot. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 37–53. IEEE, 2021.

[34] Daniel McDuff, Abdelrahman Mahmoud, Mohammad Mavadati, May Amr, Jay Turcot, and Rana el Kaliouby. Affdex sdk: a cross-platform real-time multi-face expression recognition toolkit. In *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*, pages 3723–3726, 2016.

[35] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, volume 37, pages 625–652. Wiley Online Library, 2018.

[36] Frank L Pedrotti, Leno M Pedrotti, and Leno S Pedrotti. *Introduction to optics*. Cambridge University Press, 2017.

[37] Wei Zhai, Yang Cao, Zheng-Jun Zha, HaiYong Xie, and Feng Wu. Deep structure-revealed network for texture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11010–11019, 2020.

[38] Anil K Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[39] Jun Luo, Wenqi Ren, Tao Wang, Chongyi Li, and Xiaochun Cao. Under-display camera image enhancement via cascaded curve estimation. *IEEE Transactions on Image Processing*, 31:4856–4868, 2022.

[40] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[41] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[42] Kin Gwn Lore, Adedotun Akintayo, and Soumik Sarkar. Llnet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition*, 61:650–662, 2017.

[43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[44] Tao Chen, Kai-Kuang Ma, and Li-Hui Chen. Tri-state median filter for image denoising. *IEEE Transactions on Image processing*, 8(12):1834–1838, 1999.

[45] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[46] Chengxi Li, Xiangyu Qu, Abhiram Gnanasambandam, Omar A Elgendy, Jiaju Ma, and Stanley H Chan. Photon-limited object detection using non-local feature matching and knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3976–3987, 2021.

[47] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[48] Camera2 overview, 2022. `https://developer.android.com/training/camera2`.

[49] Arengine documents, 2022. `https://developer.huawei.com/consumer/en/doc/development/graphics-Guides/introduction-0000001050130900`.

[50] Arcore documentation, 2022. `https://developers.google.com/ar/develop`.

[51] DOMI. 3d tof camera-dmom2508cl, 2022. `https://www.domisensor.com/products/17-dmom2508c`.

[52] Vzense dcam710, 2022. `https://www.vzense.com/RGBDToFProducts.html`.

[53] Deptheye wide, 2022. `https://www.seeedstudio.com/DepthEye-Wide-ToF-Camera-with-Sony-IMX556PLR-DepthSense-p-4809.html`.

[54] Nvidia jetson xavier, 2022. `https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit`.

[55] Myunghoon Suk and Balakrishnan Prabhakaran. Real-time mobile facial expression recognition system-a case study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 132–137, 2014.

[56] Imageformat, depth16, 2022. `https://developer.android.com/reference/android/graphics/ImageFormat#DEPTH16`.

[57] Perfdog., 2022. `https://perfdog.qq.com/`.

[58] tegrastats utility., 2022. `https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html`.

[59] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[60] Wei Luo, Jun Li, Jian Yang, Wei Xu, and Jian Zhang. Convolutional sparse autoencoders for image classification. *IEEE transactions on neural networks and learning systems*, 29(7):3289–3294, 2017.

[61] Ti iwr6843, single-chip 60-ghz to 64-ghz mmwave radar, 2022. `https://www.ti.com/product/IWR6843`.

[62] About face id advanced technology, 2022. `https://support.apple.com/en-us/HT208108`.

[63] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5203–5212, 2020.

[64] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[65] Mediapipe hands, 2022. `https://google.github.io/mediapipe/solutions/hands.html`.

APPENDIX

A. ARTIFACT APPENDIX

A.1 Abstract

Our artifact provides a full-stack implementation of Mozart proposed in the paper, including C++ and Python codes for the raw data collection on different types of ToF depth cameras, Python codes of light-weight phase manipulation and autoencoder for texture generation, and Java code for running Mozart on smartphones in real-time. With the hardware devices (ToF cameras and smartphones) or included dataset, it can be used to evaluate the texture generation performance of Mozart.

A.2 Artifact check-list (meta-information)

- **Algorithm: Two approaches of ToF phase manipulation for texture generation**
- **Program: OpenCV, PyTorch**
- **Compilation: Only for collecting raw depth data**
- **Model: Customized models included in the codes.**
- **Data set: Self-collected dataset using different ToF depth cameras**
- **Run-time environment: MacOS, Linux, Android**
- **Hardware: ToF Sensors (Standalone ToF cameras or ToF cameras on smartphones), Computing platform (Nvidia Jetson Xavier, smartphone)**
- **Metrics: The accuracy performance in downstream applications**
- **Output: Maps with more detailed textures**
- **Experiments: README**
- **How much disk space required (approximately)?: 20 GB**
- **How much time is needed to prepare workflow (approximately)?: 10 minutes**
- **How much time is needed to complete experiments (approximately)?: 2 hours**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: MIT License**
- **Data licenses (if publicly available)?: MIT License**

A.3 Description

A.3.1 How to access

The codes and datasets can be downloaded or clone repository from the GitHub link: `https://github.com/zhiyuancu/Mozart`.

A.3.2 Hardware dependencies

- ToF data collection: Various ToF depth cameras, including DMOM2508, Vzense DCAM710, DepthEye Wide ToF, and ToF cameras on smartphones (e.g. Huawei P30 Pro, Huawei Mate30 Pro, Samsung S20 Ultra).
- Phase pre-processing and manipulation: Linux machine with at least 4GB memory.
- Real-time Android App: Android smartphones with ToF cameras.

A.3.3 Software dependencies

- ToF data collection:
  - DepthEye Wide ToF: MacOS / Linux, libPointCloud SDK, OpenCV, cmake, python packages (struct, pickle, numpy).
  - Others: MacOS / Linux, python, sys, numpy, OpenCV
- Phase pre-processing and manipulation:
  - OS: Ubuntu 18.04
  - Packages: Python 3.6.9, PyTorch 1.8.0, CUDA Version 10.2, os, sys, argparse, numpy, sklearn.
- Real-time Android App: Android 8.0 (API 26) or above.

A.3.4 Data sets

The three datasets are all collected by ourselves in the dark for different applications, namely human tracking, face recognition, and hand gesture recognition. The datasets contain strictly aligned multimodal data (mmWave Radar, IR, RGB, RGB-enhanced, Depth, Mozart). The download link of these datasets can be found in `https://github.com/zhiyuancu/Mozart`.

- Human tracking dataset: over 8,000 frames from 9 subjects.
- Face recognition dataset: over 15,000 frames from 12 volunteers.
- Hand gesture recognition dataset: 20 hand gestures from 12 volunteers.

A.3.5 Models

We use customized models for autoencoder-based texture generation. The model configurations are included in the codes. Several third-party models are required in downstream applications, including YOLOv5 for human tracking, RetinaFace and ArcFace for face recognition, and MediaPipe for hand gesture recognition.

A.4 Installation

The codes can be downloaded from the GitHub link: `https://github.com/zhiyuancu/Mozart`.

A.5 Experiment workflow

- Download the codes in the repo. There are four folders containing different modules for different stages of Mozart. In each folder, small-scale dataset is provided for validation of each module.

- If you have one DepthEye ToF camera, go to the "libPointCloud" folder under "data collection" folder, follow the README.md for collecting the ToF phase components data using the DepthEye camera.
- If you have other ToF cameras that only provide depth and IR maps, go to the "CalFromDepthAndIR" folder, run ConvertPhase.py to generate the ToF phase components data.
- In the "light-weight-manipulation" folder, you can follow the README.md to generate Mozart maps with light-weight functions.
- In the "autoencoder" folder, you can follow the README.md to train an autoencoder for generating texture-rich Mozart maps.
- In the "Android App" folder, you can simple download the whole Android project, compile the project in Android Studio, and run the project on Android smartphones with a ToF camera.

A.6 Evaluation and expected results

Run the wrapping codes in the workflow will result in two outputs.

- The performance of generated Mozart maps in different downstream application scenarios.
- The training overhead of the autoencoder-based Mozart.

A.7 Notes

The above introductions are only for quick artifact evaluation. The complete workflow and dataset will be updated on GitHub (`https://github.com/zhiyuancu/Mozart`) in the future.

A.8 Methodology

Submission, reviewing and badging methodology:

- `https://www.acm.org/publications/policies/artifact-review-badging`
- `http://cTuning.org/ae/submission-20201122.html`
- `http://cTuning.org/ae/reviewing-20201122.html`