Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

An Investigation into the Simulation of Capacitive Fiducial Markers using Deep Learning

Benedict Steuerlein

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Michael Sedlmair
Supervisor:	Prof. Dr. Sven Mayer
Commenced:	February 28, 2021
Completed:	August 27, 2021

Abstract

Tangibles have shown to enrich the interaction space on touchscreens already in the early 2000s. On early tabletop installations, camera-based systems were used for tracking tangibles. On mainstream projected-capacitive screens, the ability to recognize objects other than fingers is limited. Lately, researchers have utilized deep learning to bring back the capabilities of recognizing conductive tangibles on capacitive screens. The main drawback of sufficiently working neural networks is that they require huge amounts of data for training, domain-specific knowledge for hyper-parameter tuning, and are often single-purpose networks. With this thesis, we propose a toolkit that allows designers and developers to train a deep learning recognizer that is purely trained on simulated data. Our toolkit makes use of a pre-trained Conditional Generative Adversarial Network that, based on sketches of the footprint of conductive tangibles, simulates the corresponding capacitive representation. Furthermore, we use this simulated data to train a deployable recognizer network. Therefore, using our toolkit, designers require no domain knowledge or need to collect data. Our evaluation shows that our approach can reliably recognize conductive fiducials with an average accuracy of 99.3 % with a recognizer network solely trained on simulated data. Additionally, our recognizer architecture can predict the tangible's orientation with an average absolute error of 4.8°.

Kurzfassung

Bereits in den frühen 2000er Jahren haben sich Tangibles als Bereicherung des Interaktionsraums mit digitalen Medien erwiesen. Auf den ersten Tabletopinstallationen wurden kamerabasierte Systeme zur Verfolgung von Tangibles eingesetzt. Auf alltäglich verwendeten, projiziert-kapazitiven Touchscreens sind die Möglichkeiten, andere Objekte als Finger zu erkennen, begrenzt. Zuletzt wurde von Forschern Technik aus dem Bereich des Deep Learnings angewandt, um die Fähigkeit zur Erkennung von leitfähigen Tangibles auf kapazitiven Bildschirmen wiederzuerlangen. Der größte Nachteil gut funktionierender neuronaler Netze besteht darin, dass große Datenmengen für das Trainieren vonnöten sind. Zusätzlich wird domänenspezifisches Wissen zur Abstimmung der Hyperparameter benötigt. Die resultierenden Netze sind trotz hohem Aufwand oft nur für einen Zweck geeignet. In dieser Arbeit schlagen wir ein Toolkit vor, mit welchem Designer und Entwickler ein Deep Learning Erkennernetzwerk auf ausschließlich simulierten Daten trainieren können. Unser Toolkit nutzt ein vortrainiertes Conditional Generative Adversarial Network, welches auf der Grundlage von Skizzen des Fußabdrucks von leitfähigen Tangibles die entsprechende kapazitive Repräsentation simuliert. Ferner verwenden wir diese simulierten Daten, um ein einsatzfähiges Erkennernetzwerk zu trainieren. Mit unserem Toolkit benötigen Designer weder großes Fachwissen, noch müssen Daten für das Trainieren aufgenommen werden. Unsere Ergebnisse zeigen, dass der von uns entwickelte Ansatz es ermöglicht, aufgenommene Bilder leitender AprilTag Marker mit einer durchschnittlichen Genauigkeit von 99,3 % zu klassifizieren. Jenes Erkennernetzwerk wurde ausschließlich auf simulierten Daten trainiert. Darüber hinaus kann unsere Erkennungsarchitektur die Rotation der Tangibles mit einem durchschnittlichen absoluten Fehler von 4,8° vorhersagen.

Contents

List of Figures

1 Introduction

Interaction with digital environments has become ubiquitous to the point where the functioning of our society depends on it. Especially, interaction with touch screen devices such as mobile phones has increased over the recent years [20]. Our primary way of interacting with the digital world is through a graphical user interface (GUI). Here touchscreens fulfill two functions; first, they visualize the content of the digital world, and second they resemble the manipulable barrier between the physical and the digital world. For GUIs, one might think that simple touch input is sufficient, but the capacities of touch-based techniques are not even close to being fully utilized with such input. Research tried to introduce additional input modalities to enrich present interaction possibilities with GUIs, e.g., through the use of the palm [40], the knuckle [73], or touch pressure as an additional modality for input [64]. While GUIs are currently our primary way to interact with digital environments physically, earlier research envisioned and implemented so-called *tangibles* as a means to embed and control digital content through physical objects [11]. Tangibles immerse digital information in graspable, manipulable objects that reflect the coupled digital information within so-called tangible user interfaces (TUI) [25]. The first TUIs were realized on large and bulky tabletop installations that used camera-based tracking systems to differentiate between fingers and tangibles, cf. [30, 59]. With the introduction of capacitive devices operated by touch input, the manipulation of digital content was easy to implement, and thus, tabletop systems have been mostly displaced [51]. Integrating TUIs into touch-based capacitive devices has therefore become a new challenge for research and is, to this day, only partially solved [51, 71].

Current touchscreens and related touch-sensing technologies in consumer devices are optimized to detect single or multiple fingers touches to initiate the actions desired by the user. Ordinary devices are not able to distinguish which finger has triggered a touch event [41]. In fact, users can trigger touch events using any conductive object, as long as the user grounds the object. Without any additional software running in the background, devices simply register touch events. Additionally, the resolution with which capacitive screens sense input is many times lower than the resolution with which content is displayed to the user [76]. This poses the most prominent challenge for enabling TUIs on capacitive devices; how to enable a stable detection of tangibles. Research has shown that additional information about objects triggering touch events can be obtained if the entire contents of the capacitive images are used [40, 50, 73]. By treating the capacitive matrix as an image containing rich information, researchers utilized Convolutional Neural Networks (CNN), popular for extracting information from images to detect diverse objects touching the screen otherwise classified as simple touch events. CNNs are promising in the image domain, but a sufficiently performing network requires a large-scale and well-annotated dataset [37, 38, 74]. Concerning detecting tangibles on capacitive screens, this would require a comprehensive dataset containing the capacitive footprints of all tangibles later in use. Additionally, it would be required to record new capacitive images and repeat the training of a classifier for adding new tangibles [71].

To avoid time-consuming data collection studies, we suggest using a *simulator network* as a data source for generating capacitive images suited for domain-specific use cases. Research has shown that Generative Adversarial Networks (GAN) can be utilized as means to circumvent the data collection study process [6, 28]. As a simulator network, we propose using a Conditional Generative Adversarial Network (cGAN) [53] that can be conditioned on sketches of the tangibles that are later in use, thereby allowing us to simulate the respective capacitive images of tangibles on capacitive touchscreens. Using a simulator network as a data source, we can obtain capacitive images and immediately train a recognizer network to identify conductive tangibles on capacitive screens [71].

In this thesis, we develop a pipeline that utilizes a simulator network to immediately train a recognizer network only on synthesized capacitive footprints of fiducial tangibles. We show that our simulator, which we trained on 10 different marker templates of 3 different sizes, can reliably synthesize diverse unseen capacitive footprints of fiducial tangibles (e.g., AprilTags [85]) in our test set with an average absolute pixel error of 7.9 (SD = 18.4). Additionally, we show that by using our simulator as a data source, our recognizer architecture can classify capacitive images in our test dataset with an average accuracy of 99.3%. With our toolkit, consisting of the pre-trained simulator network and the recognition architecture, designers of TUIs can focus on developing new interaction modalities on capacitive screens without worrying about hyper-parameter tuning and data collection.

2 Related Work

This thesis is situated in the field of tangible interaction, capacitance in HCI, and data simulation tasks. We, therefore, structured this chapter the following: first, we give an overview of previous research on tangibles followed by an investigation on how tangibles can enrich existing or create new interaction taxonomies. Since our work builds on accessing the capacitive matrix of mobile devices, we look into related work that used this data to solve diverse sets of problems. However, the main challenge of this work is the simulation of capacitive data, which is why the last part of this chapter deals with data generation through simulation.

2.1 Tangibles

Today, the interaction between users and digital devices is ubiquitous. With our hands as the primary actuator, we operate user interfaces, e.g., through computer mice, keyboards, or on mobile devices by touch. Embedding inputs into various physical objects, augmenting items with digital information, and thereby pushing computers into the background was explored early on [67, 77, 87, 88, 89]. As an additional input modality for digital devices Fitzmaurice et al. [11] proposed "Bricks", a physical mechanism to enable graspable user interfaces. "Bricks" lay the foundation for having physical objects able to manipulate virtual, graspable user interfaces. Acting as an input device, a *brick*, being tightly coupled to the computer, passes on information about its position, rotation, and selection information. They argue that using *bricks* can be superior to using our hands, as the tactile feedback guards and guides users' intention. Ishii and Ullmer [25] philosophically called this coupling between the digital and physical world "Tangible Bits" and introduced a new terminology for this interaction taxonomy called *Tangible User Interfaces* (TUI). TUIs extend the interaction with digital devices by abstracting controls in physical objects, so-called *tangibles*. Tangibles embody digital information in physical objects and enable the user to guide actions within digital media through the physical manifestation. Not only do they sense their location, but they were envisioned to sense their surroundings, including other tangibles as well.

Early TUI research used different approaches to track physical objects ranging from electronic ID tags [80], over triangulation laser scanners [62] and Radio Frequency Identification (RFID) tags [27], to electromagnetic sensing through sensing tablets [58]. The development of technology encouraged camera-based systems as the preferred way for tracking physical objects, allowing to recognize different shapes [69, 91], or visual fiducial markers [19, 30, 31].

2.2 Tangible User Interfaces

Classical graphical user interfaces (GUI) are operated using a mouse and a keyboard to access data or change visual aspects of the data representation. One might think that mice and keyboards are tangibles since they have a physical form and manipulate digital bits. While this is partially true, the physical representation of mice and keyboards has little to do with manipulating the digital bits; quite unlike actual tangibles, whose physical appearance directly impacts the expected result of the manipulation. An extensive implementation of this property was done by Underkoffler and Ishii [81]. They used tangibles in the form of buildings on a workbench to simulate and visualize different variables of urban planning (Urp). Placed tangibles, resembling buildings, allowed shadows to be cast depending on the daytime, mapped to a clock tangible. Different materials could be assigned to buildings using a material-changing tangible, or airflow around the buildings could be simulated by placing a physical wind tool on the workbench. Since the Model-View-Controller pattern (MVC) could not represent these interactions, Ullmer and Ishii [79] introduced a new interaction model for TUIs called “MCRpd”, short for model-control-representation (physical and digital). This revised model splits the view component of MVC into a graspable, physical representation of digital information (realized through tangibles) and a non-graspable, digital representation (realized through video projection or sound). The control component is coupled tightly with the physical representation enabling mechanisms for interactive control. In turn, the physical representation is “perceptually coupled to actively mediate with the digital representation” [79]. As an example, the physical representation of the time-tangible by Underkoffler and Ishii [81] gives the user control over changing the digital representation of, for example, the cast shadows of buildings. Other basic design principles were explored by Ishii [24].

Many applications implementing TUIs made use of tabletops as a ground plane for placing tangibles. Some research used the Microsoft Surface (later rebranded PixelSense [52]) to build TUIs [2, 5]. Jordà et al. [30] built the *reacTable*, a tabletop installation using multiple different tangibles to compose music. By pairing or placing building blocks on the tabletop, each of which modulates or generates a sound, users can assemble live music pieces. Using an infrared camera, they were able to track fingers and tangibles from beneath the tabletop surface. The tangible tracking was performed using reacTIVision fiducials at the bottom of tangible objects [31]. Jordà [29] later argues that by arranging tangibles on the reacTable, a horizontal, circular tabletop was envisioned to support collaboration because data can be controlled in real-time by multiple users. Zufferey et al. [101] used cameras and ARTag fiducials for a paper-based tangible tabletop simulation. Olwal and Wilson [56] combined computer vision techniques with RFID technology to take advantage of both approaches. Using RFID allowed them to recognize which tangibles are placed on the tabletop while computer vision allowed locating objects. Dalsgaard and Halskov [9] presented a 3D tangible tabletop that uses three projectors that allow augmenting the tangibles and the tabletop with additional information, e.g., augmenting a physical object to act as a lens on a map that is projected onto the tabletop. The tangible tracking was performed with a camera under the translucent table using reacTIVision fiducials [31]. Pedersen and Hornbæk [59] also made use of reacTIVision fiducials to realize tangible bots. Their setup is similar to that of Jordà et al. [30]. A projector and two cameras underneath the tabletop surface were used to augment the surface and finger and fiducial tracking, respectively. The motorized tangible bots and the tabletop implemented multiple interaction techniques, including, for example, *haptic feedback* through motor activation, and *interaction assistance* through visual guidance. Additionally, *group interactions* allowed to group multiple active tangibles and control them by interacting with only one group member. [44]

present *Geckos*, tangibles for pressure-based interaction. Using pressure-sensitive foil and unique tangible footprints, they could track and identify different physical objects. One tangible object could be used for multiple functions because its footprint can be dynamically changed by magnets retracting certain metallic parts of the footprint used for detection.

TUIs are considered superior to GUIs by some researchers [10, 11] because they can create environments used for collaboration, learning, and planning. To this day, TUIs are hardly used in everyday life, if at all. Although using a GUI may be objectively more practical, a well-implemented tangible interface allows users to experience a physical interaction with a digital environment that provides haptic feedback, creates a high level of realism, and thus generates a more positive subjective perception [100]. However, it is essential to analyze when a TUI is preferable to a GUI since expensive hardware is often required, and task efficiency might suffer.

2.3 Application Examples of Tangibles on Capacitive Screens

With thinner form factors, lower cost, and greater accessibility of the wide crowd, devices with capacitive sensing technologies displaced camera-based tabletops. Capacitive sensing on mobile devices can be classified as surface-capacitive and projected-capacitive techniques [55]. For surface-capacitive techniques, a capacitive layer with four synchronized electrodes at each corner senses contact with a conductive object. This technique does not recognize multiple touches at the same time. Projected-capacitive techniques use two separate conductive layers, one of which senses contact with conductive objects horizontally and one vertically, thus allowing to capture multiple touches at different x and y positions. Modern smartphones and tablets utilize the technique of projected-capacitive [4]. However, the resolution with which current commercial capacitive screens sample input is low, making the detection of small structured elements like tangibles hard.

To overcome these limitations, researchers introduced various techniques to realize the usage of physical objects on capacitive platforms [36, 46, 66, 96]. Yu et al. [95] built *TUIC*, a technique that can simulate finger touches. *TUIC* utilizes passive materials and a frequency modulator to encode information in a low resolution spatial tangible. By modulating, for example, IDs through touch activation at specific intervals, they could decode this information to trigger certain actions, e.g., opening a keyboard or unlocking a device. However, their approach requires the user to touch the tangibles to be recognized by the screen. Additionally, the tangibles must be equipped with batteries; otherwise, the frequency modulation can not be achieved. To overcome the problem of having to touch and thereby ground the tangibles, passive tracking tangibles can be used [83, 84]. However, these types of tangibles are fabricated in specific ways for being recognized passively. Chan et al. [8] enriched touchscreen interaction with *CapStones and ZebraWidgets*. Their tangibles consist of multiple building blocks stacked on top of each other. Combining these building blocks, they can sense the number of elements incorporated and thus trigger different functionalities based on the activated conductive elements of the tangible's footprint. Xiao et al. [93] paired devices using the capacitive screen of one device and the camera of the other. Pairing works through selectively activating specific LCD pixels. By estimating the position of the rear camera using a contour-based approximation of the device dimensions through capacitive images, said pixels under the rear camera are activated. By illuminating the pixels in certain colors in specific sequences, they could transfer data with a throughput of 150 bits per second. *Project Zanzibar* by Villar et al. [82]

presents a portable TUI. The flexible mat utilizes capacitive sensing to detect hovering gestures, touch, rotation of placed conductive objects, and up to 16 tangibles placed on it. Through NFC, the mat can identify and communicate with tagged objects up to 30mm above it.

A large body of research looked at how to 3D-print tangibles for capacitive surfaces [13, 14, 15, 22, 34, 35, 70]. 3D printers allow designers to quickly fabricate geometric shapes previously modeled using digital tools. Typical 3D printers can switch between the materials they use for printing, allowing them to print both conductive and normal polylactide in one printing session. Schmitz et al. [72] built *Flexibles*, flexible tangibles that can sense spatial and intensity deformation. The key contribution of this work is informing designers how to build deformation-sensitive tangibles for capacitive screens using standard 3D printers. *3D-Auth* by Marky et al. [48] tries to extend authorization processes on capacitive touch devices by using different 3D-printed tangibles as the secret for authorization. A problem most tangibles on capacitive screens have is that they occupy large areas of valuable screen space. Unlike tabletop displays that span 40", commercial smartphones only span around 6" and tablets around 10". Schmitz et al. [71] built *Itsy-Bits*, tangibles for capacitive screens with small footprints to cover as little display as possible. They fabricated 40 distinct tangibles (10 different shapes in 4 sizes) and recognized them on screen using a Convolutional Neural Network (CNN). Their CNN approach allowed them to recognize the different markers and sizes and approximate the tangibles' orientation. This work eases the process of detecting small tangibles on capacitive screens.

On tabletop TUIs, the tangibles are mostly tracked by camera-based systems and identified through, for example, RFID. A small portion of researchers tried to identify and track tangibles on capacitive devices by using conductive fiducials and their resulting capacitive footprint. Ikeda and Tsukada [21] built markers that are conductive on one side and have a visual fiducial on the other side, allowing the markers to be detected by both a camera (using the visual fiducial side) and the capacitive screen. With *ForceStamps*, Han et al. [16] present a pipeline for rapid prototyping of 3D-printed fiducial conductive markers to enable physical controls for pressure-sensitive touch surfaces. Their work guides designers in how to build and utilize them in different applications. However, they still discovered problems in correctly identifying fiducials the moment they touched the screen. In addition, they note that currently, the design of physical markers is dependent on two things: the physical size of the devices on which the tangibles will be used and the resolution of the capacitive matrix. Although the trend went towards ever-larger devices, the capacitive resolution remained low. This problem was addressed by Mayer et al. [51] with *Super-Resolution Capacitive Touchscreens*. By borrowing super-resolution techniques from biology and astronomy, they could reliably upscale low-resolution capacitive images. Their technique scales up conductive objects on capacitive screens, so the details that are otherwise hidden become visible. They show standard and up-scaled capacitive images of different conductive objects, e.g., coins, keys, or AprilTag [85] fiducials. They show the feasibility by using an out-of-the-box AprilTag detector and show a significant increase of detection rate between standard and super-resolved images.

Previous work shows that TUIs on capacitive devices can enrich the current interaction space. However, ordinary capacitive screens are limited by their resolution making the detection of tangibles hard. To identify tangibles on capacitive screens, researchers used their unique footprints that imitate multiple finger touches [95]. Schmitz et al. [71] envisioned different shaped physical objects with small footprints and utilized a DCNN to identify them based on the capacitive matrices. Neural networks require vast amounts of data to train sufficient recognition models. One way to gather them is through simulation.

2.4 Generating Data using GANs

With the introduction of GANs by Goodfellow et al. [12] in 2014, the training of neural networks that generate data and match a data distribution took a huge step. GANs are composed of two separate adversarial networks, namely a generator G and a discriminator D, that try to minimize their own loss while maximizing the others' in a minimax-game. G tries to generate images indistinguishable from real images, while D tries to determine if images are from the data distribution or from G. In the classical GAN formulation, G learns a mapping from a latent noise space p_z to the data space p_{data}. To prove their concept, they generated new samples of different datasets, including handwritten digits from MNIST [42] and low-resolution images from CIFAR-10 [37]. Where in the original GAN formulation, G and D could be any differentiable function, i.e., a multilayer perceptron, Convolutional Neural Networks (CNN) were found to be appropriate for image modeling tasks [63]. Over the course of 7 years, GANs have evolved rapidly and have found use in e.g., producing high-resolution photorealistic images [7, 32, 33], generating 3D shapes [92], resolving blurry images [43, 94, 97], or enhancing audio signals by denoising [57]. Streli and Holz [76] implemented a Wasserstein GAN [3] to upsample low-resolution capacitive images of finger touches to make adjacent touches more distinguishable. Goodfellow et al. [12] state that instead of generating random images by sampling from the latent noise space, both G and D can be conditioned on certain information y by passing it as additional input to G and D. Mirza and Osindero [53] introduced Conditional Generative Adversarial Networks (cGAN). CGANs give specific control over what the output will look like. CGANs condition both generator and discriminator on the multimodality of the data. Therefore bot can learn the data distribution of different classes rather than that of the entire dataset. Say the distribution to be learned is of pictures of garments. With classical GANs, a noise vector z from the latent space produces samples that can be anything from, for example, a shoe, a sweater, or a hat. With cGANs, one can exclusively sample shoes or exclusively sample hats by passing a class information representation. Instead of conditioning on class labels, Reed et al. [65] conditioned their GAN on text, allowing them to synthesize images based on text descriptions.

Image-to-image translation is a challenge in computer vision. These problems define a source domain X, a target domain Y, and the objective is to learn a function $f : X \longrightarrow Y$ that maps images $x \in X$ from the source domain to images $y \in Y$ of the target domain. It has been found that cGANs can cope with these kinds of problems. Isola et al. [26] propose *pix2pix*, an cGAN for image-to-image translations. Their generator is a U-Net [68], a unique form of an encoder-decoder network [18], that uses skip connections from encoder layers directed to decoder layers in order to pass over information that might otherwise get lost during the encoding step. Their discriminator, a Markovian discriminator [45] which they call PatchGAN, is a deep convolutional neural network (DCNN) that, instead of classifying if a whole picture is real or fake, classifies if $N \times N$ patches are real or fake. Isola et al. [26] prove their architecture by conditioning their cGAN on several problems, among which fall generating photo-realistic images from segmentation maps, generating segmentation maps from photos, coloring images, or generating satellite images by conditioning on maps. Zhu et al. [99] extend the image-to-image translation to a cyclic translation, simultaneously learning two mappings, one from domain X to domain Y and one from Y from X. Recently, Hao et al. [17] translated pixelated Minecraft worlds into photo-realistic scenes utilizing GANs. Their approach renders high-quality scenes from low-resolution information.

One promising application domain for image-to-image translation GANs is synthesizing new, unseen data. This simulated data can then, in turn, be used to train classifiers without the need for recorded datasets, as the generated data may already have labels attached to it [6, 98]. Jahanian et al. [28] trained classifiers by utilizing implicit generative models [54] as dataset synthesizers, while having no control over the training data of the generative models. Based on their results, they propose to include GANs as a viable data source for training tasks of image-based neural networks.

2.5 Summary

In this thesis, we synthesize capacitive images of fiducial markers using an cGAN to ease the process of data collection and train recognizer networks solely on simulated data. These recognizers can be used to identify active tangibles equipped with conductive fiducials on capacitive touchscreens. Thus, we presented an overview of tangibles and their application within TUIs. Such TUIs give a physical meaning to digital information and enable digital manipulation through physical interaction [25, 79]. While early research about TUIs utilized large tabletop installations that mainly used cameras for tangible tracking, recently, researchers tried to enable tangibles on capacitive sensing surfaces [36, 46, 51, 66, 71, 96]. One way for identification of tangibles on capacitive screens is to use CNNs, which perform well in image recognition tasks [71]. While CNNs and other types of neural networks require vast amounts of training data, GANs [12] and other types of generative networks can function as a form of a dataset by synthesizing new training data [28]. Next, we will discuss how we collected our ground truth capacitive images used for training, validating, and testing our cGAN.

3 Data Collection for Simulator

Our objective is to simulate capacitive images. We define our simulation problem as a mapping from conductive marker image space to capacitive image space. We fabricated and recorded a wide range of conductive fiducial markers on a capacitive touchscreen to train and test our simulator. Our collected data comprises a total of 448,261 capacitive frames of 120 recorded fiducial markers. The following section describes how we recorded the ground truth capacitive images and the pre-processing steps required for training our cGAN.

3.1 Apparatus

To record our ground truth capacitive data, we used a Samsung Galaxy Tab S2 SM-T813. The display spans 9.7", having a resolution of 2048 × 1536px. Since commercial smartphones and tablets rarely grant permissions for accessing the capacitive screens' matrix, we flashed a custom kernel onto the device to access the 37 × 49 capacitive images (6.33PPI, 4.0127mm per pixel). We developed an application that logs the capacitive matrices at a sample rate of 9FPS.

In line with prior work, e.g., [50, 71], we tracked the rotation of the capacitive markers using an *OptiTrack-V120:Trio*, an optical motion capture system, which records motion data at 120FPS. We calibrated the upper left corner of the tablet as the origin of the tracking area. Using three reflective markers attached to a custom apparatus, we defined a *Motive* rigid body to track the orientation of the tangibles relative to the tablet.

Since we collect capacitive data on the tablet and *OptiTrack* data in *Motive* simultaneously, we captured each data point with a continuous timestamp. For later timestamp synchronization, we connected the computer running Motive and the tablet sampling the capacitive images (using an application called *ClockSync*) to the same NTP server.

As capacitive markers we used 2 TYPES of fiducials, AprilTags of *tag family 16h5* and of *tag family 36h11*. Each fiducial was built with 3 different SIZES per pixel (4, 6, 8mm). For each TYPE we built 15 markers which resulted in a total of 2 × 15 × 3 = 90 fiducial markers. Additionally, to ensure the generalization ability of our simulation we built 10 custom *shapes* with three different WIDTHS (8, 12, 16mm). Combining the custom *shapes* with the AprilTag fiducials, we built a total of 120 capacitive markers. Figure 3.1 shows two AprilTag markers of TYPE 36h11 and 16h5 in all SIZES.

The construction process for each physical marker was the same. We scaled the graphics of the markers to the intended SIZES and cut them out on aluminum foil which was attached to thin cardboard. Additionally, we added an extra strap to each physical marker so the current can flow into the recording person's finger without touching the capacitive display. The cut-out markers were then attached to a 5mm thin foam plate.

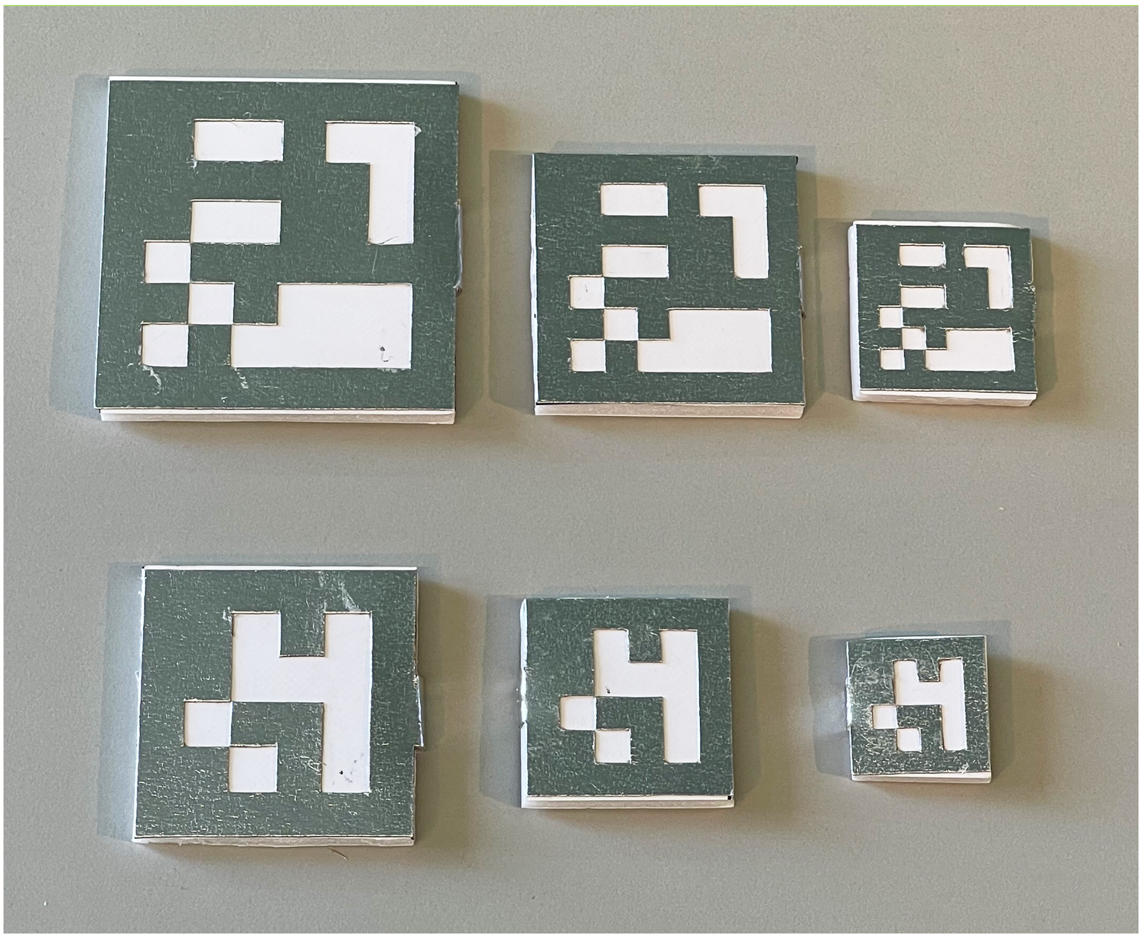


Figure 3.1: Comparison of the physical fiducial markers 48_{36h11}, 21_{16h5} in all SIZES.

3.2 Procedure

Using double-sided adhesive tape, we attached the tablet to our recording table. With the three reflective markers on the corners of the tablets' display, we calibrated the OptiTrack system, setting a ground plane and the direction of the tracking axes.

Before each recording, we synchronized the tablet's and computer's local time to the same NTP server. We attached our rotation tracker to the tangibles using velcro adhesives. Subsequently, we started the recording application on the tablet and the *Motive* recording on the computer. Using circular motion on the tablet while simultaneously manipulating the markers' yaw axis, we captured multiple frames of every marker in all rotations. To record clear capacitive images, we were grounded by touching the tablet frame.

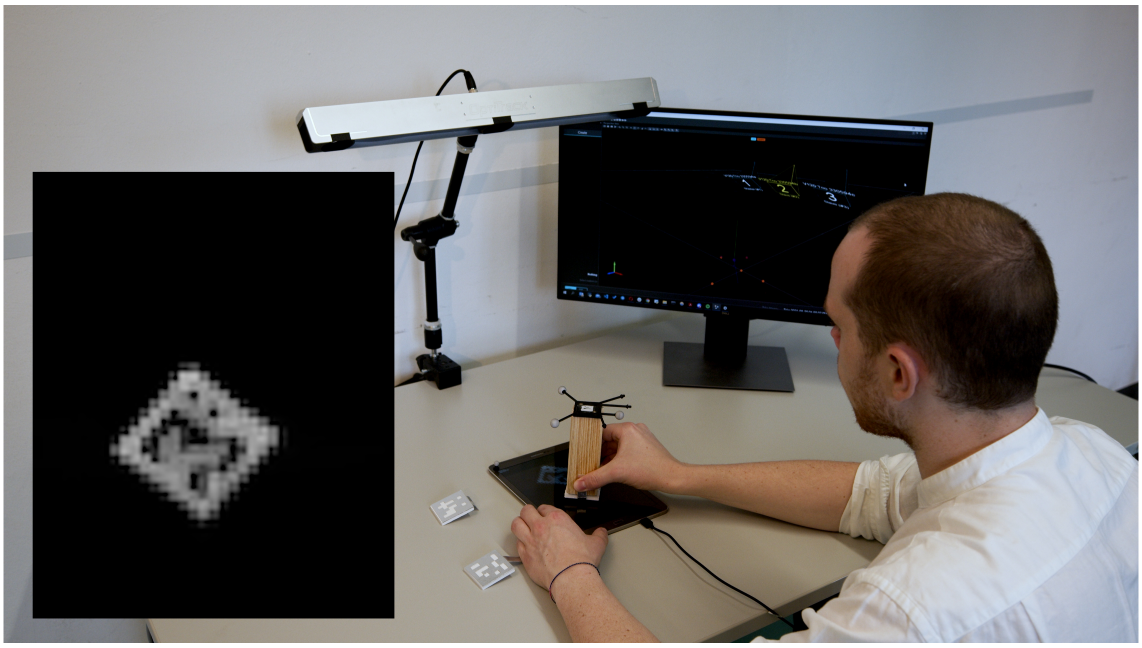
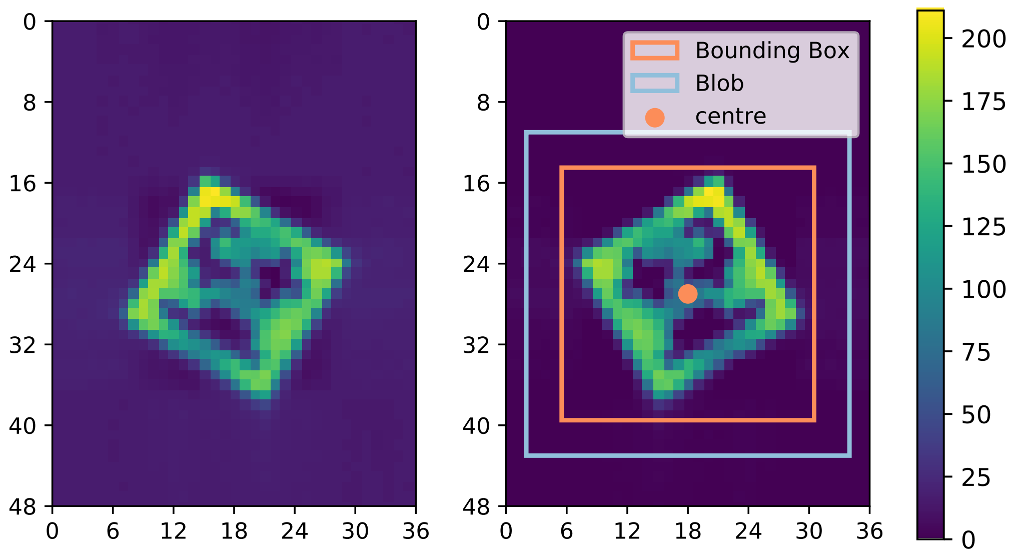


Figure 3.2: Our data capturing setup showing the *OptiTrack-V120:Trio* that streams motion data of the rigid tracking body attached on the physical markers to our Motive PC. Simultaneously we record capacitive images on a Samsung Galaxy Tab S2 of conductive fiducials.

3.3 Pre-processing & Data Augmentation

To generate our dataset, we mapped the recordings of the capacitive touch screen to the rotational data captured via *OptiTrack* using time stamps. Given *OptiTrack's* documented system latency of 8.33ms, we manually matched the sub-latency shifts by visually comparing the orientation and change of the capacitive image to the corresponding recorded angles. In total, we recorded 448,261 capacitive frames over the course of 13h and 41 min. We recorded each marker for an average duration of 6min 51sec (SD = 33sec). We identified all capacitive blobs (an imprint of the fiducial maker) during pre-processing by finding the contours [78] on a thresholded and flipped version of the capacitive image. We flipped the capacitive images to match the imprints to the footprints of the physical fiducial markers. Here, we removed all images without any fiducial markers present. Next, we cropped a 32×32 patch around the center of the blob. This allows us to recognize fiducial markers with a diameter of up to 128mm (32dots * 4.0127mm dot pitch). For data augmentation, we rotated each 32×32 sample 3 times by 90°, adding three times the amount to the initial data. Thus, we are left with 1,699,044 samples (655,392 36h11 AprilTags, 640,684 16h5 AprilTags, and 402,968 shapes). Finally, we clipped the capacitive image values between 0 and 255 and normalized the resulting images between -1 and 1 to help the training process. Figure 3.3 highlights the steps performed for each capacitive matrix without the normalization step.



(a) (b)

Figure 3.3: Pre-processing pipeline of capacitive images collected during our data collection. Image a) shows an example 37×49 capacitive image of a capacitive marker (70_{36h11}, 8mm) gathered during our data collection. Image b) visualizes the steps required to extract capacitive blobs from our recordings. First, we clip the image between 0 and 255 and find the center (depicted by the orange point) and the fiducial marker's bounding box (orange square). Next, we crop a 32×32 patch around the center (light blue square) to extract the capacitive blob. If the borders of the patch exceed the screen's dimensions, we fill the exceeding pixels with zero values.

4 Recognizer Pipeline

To help designers create applications that work with capacitive images and conductive tangibles, we envision a pipeline that outputs a trained neural network that can differentiate between multiple capacitive markers and predict their current rotation. The key feature of our pipeline is that no recorded capacitive data is required to train our recognizer. Our pre-trained simulator network provides all training data. Our simulator network only needs to be trained once in order to simulate capacitive footprints of conductive fiducials. Additionally, there is no need to collect capacitive data when adding new markers to a marker set of an envisioned application. Designers and developers need to restart the pipeline and add the new marker to the marker set to be simulated. The resulting recognizer network is trained on the additional marker and can distinguish it from others. This section covers the pre-processing steps that are required to prepare the data for training the simulator network. We explored multiple network architectures for our simulator network components and our recognizer model. We explain the best performing network structures of the components of our simulator network, our recognizer network, and how they are interconnected to form our pipeline.

4.1 Pipeline Pre-processing

We call the final set of data obtained after our pre-processing and data augmentation step DS (see Section 3.3). DS contains paired data $\{x_t, x_c, x_a\}$ of templates x_t, capacitive recordings x_c, and recorded angles x_a. We excluded the shape recordings from DS and put them into a separate dataset DS_{Shape}. The sketches of these shapes are displayed in Figure 5.5 (a). To train, validate, and test the simulator network, we used all recorded AprilTags of TYPES 16h5 and 36h11. More specific, we split DS into two sets: one to train and validate the *simulator network* DS_{GAN}, and one to test the capabilities on simulating unseen data DS_R monitored through a *recognizer network* R. We split the data, so that one AprilTag in all SIZES is either in DS_{GAN} or DS_R. This guarantees no occurrence of overfitting between the *simulator network* and the *recognizer network*. DS_{GAN} contained 5 different AprilTag ids of TYPE 16h5 and 5 different AprilTag ids of TYPE 36h11, see Figure 4.3. DS_R therefore contains the remaining 20 ids (10 of TYPE 16h5 and 10 of TYPE 36h11), see Figure 4.4. In total DS_{GAN} contained 433,664 samples and DS_R contained 862,412 samples.

Subsequently, to train and validate our *simulator network* we split the marker set DS_{GAN} into the subsets DS_{GAN}^{train} and DS_{GAN}^{val} used for training and validation, respectively. We, therefore, used a $70\%/30\%$ split, resulting in $303,565$ samples in DS_{GAN}^{train} and $130,099$ samples in DS_{GAN}^{val}. While this could potentially lead to overfitting, the simulator network has to generalize beyond the markers used for training. This can be seen when training R, where the *simulator network* has to generate markers contained in DS_R that are not used during the training of our simulator. Therefore there is no reason for a test set for the *simulator network* as the *recognizer models* will highlight any

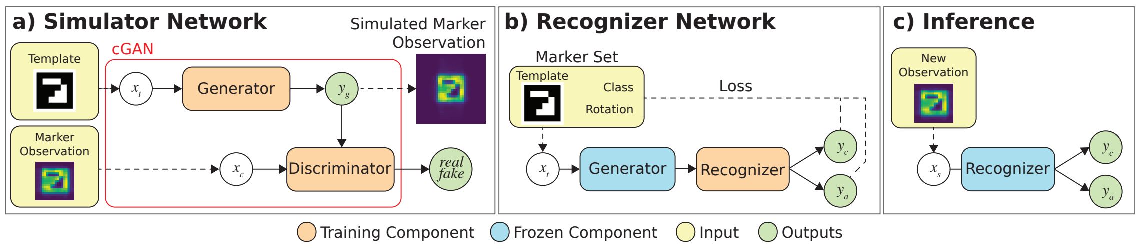


Figure 4.1: This image shows the training pipelines and the usages of our different models. Image a) shows the training process of our cGAN. Image b) shows the training of our recognizer network on a specific marker set where we use the frozen generator component acquired from step a) to generate our training data. Image c) depicts the usage of a fully trained recognizer network deployed on an end-device used to infer classes and rotations of capacitive fiducial marker observations.

weakness of the *simulator network*. Because we train our *recognizer models* only on synthesized data, we do not need to split DS_R, allowing us to use 100% of our test set (862,412 samples) to evaluate our *recognizer models*.

4.2 Simulator Network

Generative models have been shown to accomplish similar tasks to ours, e.g., style transfer [26, 99]. A generative model G tries to replicate a data distribution $p_{Data}(y)$ with y being the ground truth images by generating images $\tilde{y} \in p_{Noise}(z)$. However, instead of a classical GAN [12], that tries to map a noise distribution $p_{Noise}(z)$ to a data distribution Y, $G(z) \longrightarrow y$ with $y \in Y$, we use an cGAN, a refined GAN that can be conditioned on a given input. In our case, we condition our cGAN on templates of fiducial markers (x_t). We train our simulator network using the dataset DS^{train}_{GAN} introduced in Section 4.1, to learn a mapping from structural information of templates x_t to capacitive images x_c. Thus, G is a mapping $G(x_t) \longrightarrow y_g$.

4.2.1 Simulator Network Objective

During our training process of the cGAN, the discriminator D tries to detect if an image is a fake image (simulated image) or a real image. On the other hand, generator G wants to produce better quality images to fool the discriminator (see Figure 4.1a)). This behavior can be expressed as:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) \tag{4.1}$$

Traditional GANs and various cGANs use a noise vector z to produce output that is not deterministic [86]. However, often the model learns to ignore the noise [26, 49]. One common approach to overcome this issue is to embed Dropout layers [75] into the model structure during training to generate non-deterministic output, which will allow the generation of a wider range of outputs [26], in our case, simulated capacitive markers x_g. As we do replaced the typical noise vector z with Dropout layer, our adversarial loss $L_{cGAN}(G, D)$ can be described as the following:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x_t, x_c}[\log D(x_t, x_c)] + \mathbb{E}_{x_t}[\log(1 - D(x_t, G(x_t)))]. \tag{4.2}$$

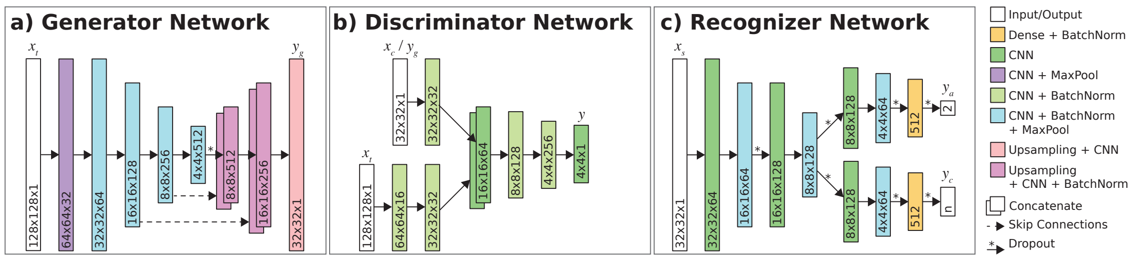


Figure 4.2: The structures of the different networks used in this work. a) is an encoder-decoder network with skip connections that uses templates as input and produces the respective capacitive images. b) shows the discriminator structure that is a CNN that distinguishes 4×4 patches of the input image as real or fake. c) shows the structure of our recognizer used to classify capacitive images and predict the orientation of conductive tangibles.

Moreover, prior work has shown that it is beneficial to only relay on the discriminator loss but also on traditional losses [26], e.g., L1 loss. Thus, we add the L1 loss (pixel-wise loss) of G defined as

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x_t,x_c}[\|x_c - G(x_t)\|_1] \tag{4.3}$$

to the initial objective function with a weighting parameter λ. This results in the final objective function:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda * \mathcal{L}_{L1}(G) \tag{4.4}$$

We weigh the pixel-wise loss 100 times more than the adversarial loss.

4.2.2 Generator

We define the objective of our generator G as a style transfer objective. Here, G learns to apply a style from images y of a target domain Y to images x of a source domain X, thus $G : X \rightarrow Y$. In our case, G learns the mapping from templates x_t to respective capacitive images x_c, thus the objective of our generator can be described as $G : T \rightarrow C$, with $G(x_t) = y_g$. G is a special type of encoder-decoder network [18], called U-Net [68], which was also utilized by Isola et al. [26] with Pix2Pix. G encodes its input in downsampling layers and applies the style transfer in its upsampling layers. Additionally, skip connections pass partially encoded information from downsampling layers to upsampling layers. Therefore, information that otherwise might get lost during later encoding steps can enrich the information space while decoding.

Model Structure

As shown in Figure 4.2 a), our generator is an encoder-decoder network. It has a total of 5,942,369 parameters. The structure of our generator is a modified version of the Pix2Pix model by Isola et al. [26]. We chose to modify the Pix2Pix model as it comprises simplicity but also produces strong results. Our encoder uses modules of structure 2D Convolution - LeakyReLU [47] - BatchNorm [23] - MaxPooling, which we call *Conv-Block*. Our decoder uses modules of structure Upsample2D - 2D

Convolution - ReLU - BatchNorm, which we call *Deconv-Block*. A Conv-Block downsamples the input by a factor 2 using MaxPooling with a pool size of 2×2. Deconv-Blocks upsample their input by a factor 2.

The input layer of our encoder is Conv-Block^1 without BatchNorm. The 2D convolutions in this layer are performed with 32 filters. After the input layer, 4 consecutive Conv-Blocks^{2,3,4,5} follow with 64, 128, 256, 512 filters, respectively. To avoid deterministic outputs we employ a Dropout layer after the last Conv-Block with a dropout rate of .5. The decoder receives the output of the Dropout layer and passes it through two Deconv-Blocks^{1,2} with 256 and 128 filters, respectively. We add skip connections from Conv-Block^4 to Deconv-Block^1 and from Conv-Block^3 to Deconv-Block^2. The last Deconv-Block is followed by an Upsample2D layer and a Conv2D layer to map to the number of output channels of the image (in our case with filter size 1).

All convolutions have a kernel size of 4×4 with a stride of 1. We use LeakyReLU units in the encoder with a slope of .2 and standard ReLu units are used in the decoder. Each BatchNorm unit has a momentum of .8. We initialized the weights of the generator from a normal distribution with $\mu = 0$ and $\sigma = .02$. Our generator G was trained with an Adam optimizer with a learning rate of .0002 and the momentum parameter $\beta_1 = .5, \beta_2 = .999$.

4.2.3 Discriminator

The discriminator D is the adversary of our generator G described in Section 4.2.2. We adopt the PatchGAN discriminator by Isola et al. [26], which is a Markovian discriminator [45] that distinguishes if $N \times N$ patches of the input are either real or fake. This has shown to enforce G to produce less blurry images. Additionally, PatchGAN discriminators have fewer parameters than classical discriminators, making them faster to train and run. D and G are conditioned on the templates x_t used to produce the capacitive recordings x_c or simulating capacitive images y_g.

Model Structure

As shown in Figure 4.2b), our discriminator D is a Convolutional Neural Network (CNN). It has a total of 736,337 parameters. It has two input branches, one branch for 32×32 capacitive images and one for 128×128 templates for conditioning D. The first branch adjusts capacitive images to the same dimensions as the result of the second branch. The second branch is the input for conditioning D on geometric templates x_t. Here, the templates are passed through two blocks consisting of 2D convolutions (4×4 kernel with a stride of 2) activated by a LeakyReLU unit and followed by a BatchNorm unit. This results in the convolved templates having the same shape as our capacitive images from the first input branch. The output of both branches is concatenated and fed through three blocks consisting of 2D convolutions (4×4 kernel with a stride of 2) activated by a LeakyReLU unit and followed by a BatchNorm unit (except for the first block). Finally, the output of D is a 2D convolution with 1 filter to match the input channels of our capacitive images and a kernel size of 4×4 with a linear activation function. Thus, resulting in 4×4 patches that are classified as either real or fake.

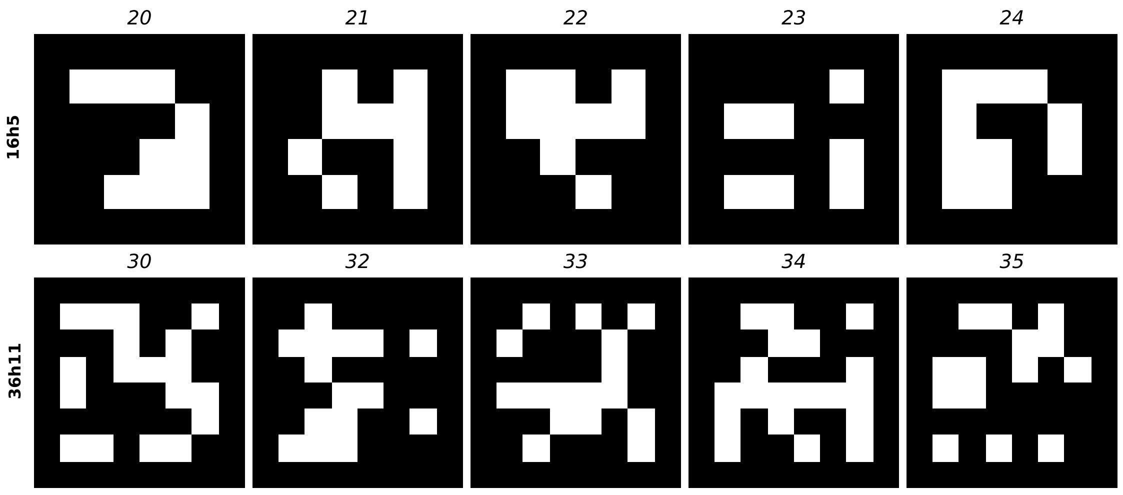


Figure 4.3: Images of the AprilTag fiducials used for recording the training data for our simulator network.

See Figure 4.2b) for the number of filters used in each 2D convolution layer. Our discriminator D was trained with an Adam optimizer with a learning rate of .0002 and the momentum parameter $\beta_1 = .5$, $\beta_2 = .999$. We initialized the weights of the discriminators from a normal distribution with $\mu = 0$ and $\sigma = .02$. All LeakyReLU units have a slope of .2. Each BatchNorm unit has a momentum of .8.

4.2.4 Simulator Network Training

As proposed by Isola et al. [26], we train our simulator network consisting of G and D by alternating between a gradient descent step on D, followed by a gradient descent step for G for each batch. We trained the generator and discriminator for 500 epochs with a batch size of 128. Every 20 epochs, we saved a checkpoint containing weights and biases of both networks. The checkpoints allowed us to reload the state of the training phase in which both the generator and the discriminator performed well in terms of the desired recognition rate of generated and real images by D and the performance of G in generating realistic images (in terms of L1 loss between real and generated images). The training time for the generator model was 66 hours on an Nvidia Tesla V100.

The task of G is to generate 32×32 capacitive images y_g based on templates x_t representing the footprint of fiducial markers. Initially, we built G to generate y_g based on 32×32 representations of the templates. However, we found it beneficial to reduce the density of information stored in one pixel by increasing the overall size of the template passed to G. Thus, we scaled x_t by a factor of 4. Therefore, G expects input templates of size 128×128px. See Figure 4.3 for the unscaled marker templates used for training the simulator network.

To increase the variance of our training set M_{train}, we shifted templates x_t by ± 4 and ground truth capacitive images x_c by ± 1 in random x and y directions. The training time for the generator model was 66 hours on an Nvidia Tesla V100.

4.3 Recognizer Network

Our recognizer network R is a CNN that determines the class y_c and the rotation y_a given the capacitive image x_s of a conductive fiducial. Thus, $R(x_s) \rightarrow y_c, y_a$. These models are common for image recognition tasks, even in the domain of capacitive images [39, 71, 73].

4.3.1 Model Structure

Figure 4.2 c) shows the structure of our recognizer R. The input consists of a normalized 32×32 capacitive image between -1 and 1. It is passed through two blocks, each applying two layers of 2D convolution (3×3 kernel, stride of 1) followed by a BatchNorm layer, a MaxPooling layer, and a Dropout layer with a dropout rate of 0.4. The output is split into two separate branches, a rotational regression branch, and a classification branch. Each branch passes its input through one of the aforementioned blocks whose output is flattened and passed through a fully connected hidden layer, followed by a BatchNorm layer and a Dropout layer with a dropout rate of 0.4. Finally, the output layers for the classification branch, which uses a softmax activation function, and the rotational regression branch, which uses a linear activation function, follow. The number of output neurons for the classification branch is dependent on the number of classes n on which we train our recognizer R. The number of classes n also impacts our recognizer networks parameter count (n=2: 1,758,788, n=10: 1,762,892, n=30: 1,773,152, and n=60: 1,788,542).

4.3.2 Loss Function

Contrary to related work by, e.g., Mayer et al. [50], our recognizer predicts the sine and cosine components of the angle y_a of marker observations x_s, as suggested by White [90]. Schmitz et al. [71] also used the sine and cosine components of the predicted angle but calculated them post-prediction. We calculate the predicted angle a of our recognizer R as:

$$a = arctan2(pred_{sin}, pred_{cos}), \tag{4.5}$$

where $pred$ is the rotation branch output y_a of R. For our rotational regression branch, the network tries to minimize the error

$$\mathcal{L}_{Angle}(R) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(0.5 * ((true_{sin,i} - pred_{sin,i})^2 + (true_{cos,i} - pred_{cos,i})^2))} \tag{4.6}$$

The loss for our classification branch is a categorical crossentropy with

$$\mathcal{L}_{Class}(R) = -\frac{1}{N}\sum_{i=1}^{N} log p_R[y_i \in C_{y_i}]. \tag{4.7}$$

The final objective of our recognizer is described by

$$R^* = arg \min_R \mathcal{L}_{Angle}(R) + \mathcal{L}_{Class}(R). \tag{4.8}$$

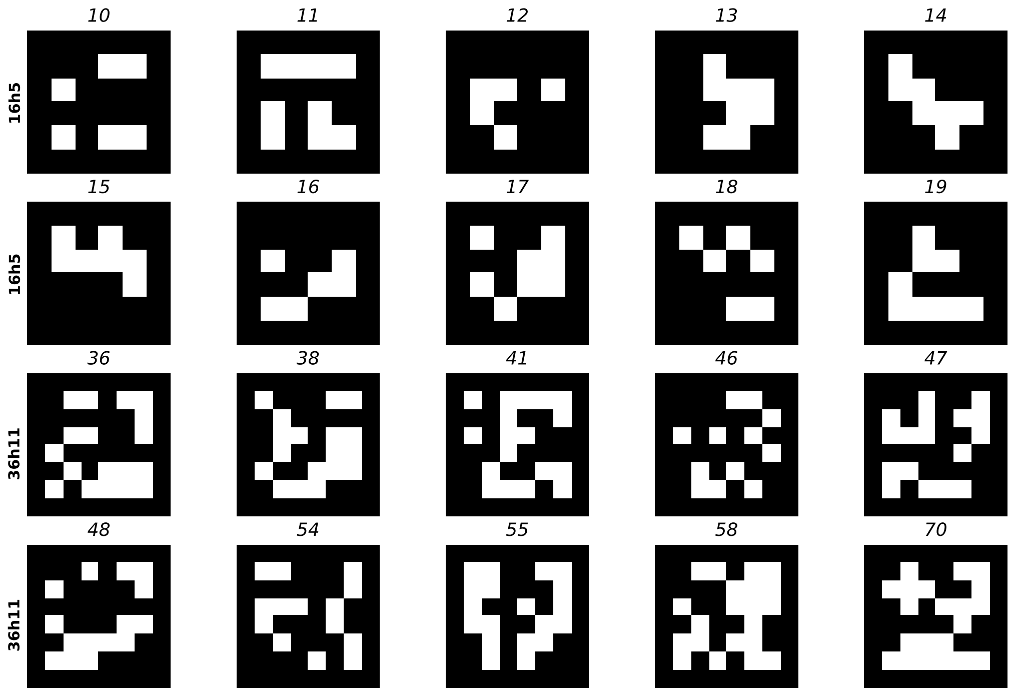


Figure 4.4: Image showing the different AprilTag fiducials contained in our test set. We recorded each marker with pixel sizes of 4, 6, and 8mm.

4.3.3 Recognizer Network Training

The unique feature of our recognizer R is that it is trained exclusively on simulated data. We trained R using an Adam optimizer with a learning rate of .001. We used a batch size of 64 and set the number of training epochs of R to 400. However, we applied early stopping with patience of 20, monitoring the combined loss of the classification and rotation branch. The training time for a single recognizer is between 0.07h and 7.38h on an Nvidia Tesla V100; however, the time is impacted by the class count and the monitored loss during training (n=2: .07h, n=10: 1.17h, n=30: 3.42h, and n=60: 7.38h).

The goal of R is to correctly classify capacitive fiducials and predict their rotation, while being trained solely on simulated data (see Figure 4.1 b)). The simulated data depends on the fiducials that will be in use during inference on an end-device. We, therefore, generate a marker set M containing all permutations of templates x_t in all rotations ($1°$ steps in $[0, 360°)$). Additionally, we shift each template permutation in M by ± 4 as performed during the simulator training (see Section 4.2.4). This increases the variance and counteracts inaccurate blob detections on the end-device, where the detected blob x_s may not be exactly centered for R's input. As an example, a designer wants to use 10 fiducial markers during inference. The number of simulations used for training can therefore be calculated as: $\#markers \times \#rotations \times \#shifts = 10 \times 360 \times 9 = 32,400$. In turn, we simulate one fiducial marker in 3240 different states.

To counteract overfitting and to add more variance to our training set, we add Perlin noise [61] to the activated areas of simulated capacitive markers y_g.

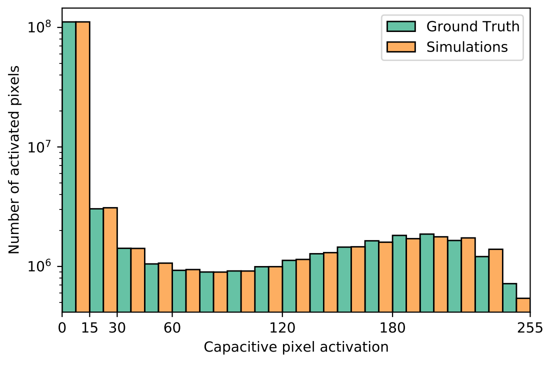
5 Evaluation

The goal of this thesis was to develop a pipeline that utilizes a pre-trained simulator as data generator for training recognizer networks. Our simulator network synthesizes training data in the form of capacitive footprints of conductive fiducial markers. We trained each recognizer network with synthetic capacitive data of markers which our simulator network has not seen before. We tested our trained recognizer network only on recorded capacitive images. When evaluating our trained recognizer networks, we considered both the classification accuracy, and the absolute errors between the predicted rotation y_a and actual rotation x_a of the recorded capacitive images x_c. By using recordings of markers unknown to our simulator network, we infer insights into the generalization ability of the simulator. Thus, instead of deploying each trained recognizer model on an end-device for evaluation, we treated our recorded capacitive data as new, previously unseen marker observations (see Figure 4.1c). When inferring baseline metrics, we could not pass two-dimensional image data to the machine learning algorithms. We therefore flattened each 32×32 capacitive simulation for training and each 32×32 recorded capacitive blob into a one-dimensional feature representation of length 1024. We performed a grid search on the number of estimators for each baseline evaluation to find the best parameters. The overall results, including Random Forest baselines for all evaluations, are listed in Table 5.1.

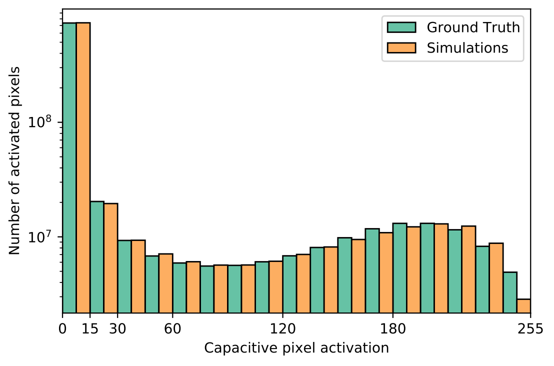
We built our complete software stack for evaluation and visualization in *Python 3.8.10* to enable rapid prototyping. For image processing and manipulation, including value clipping, normalization, and blob detection, we used *OpenCV 4.5.3*. For our baseline evaluation, we used *scikit-learn 0.24.1* [60]. To train, validate, and test our different network architectures and our pipeline, we used *TensorFlow 2.5.0*. We trained our models on an *Nvidia Tesla V100-SXM2* graphics card with 32GB memory.

5.1 Simulator Network

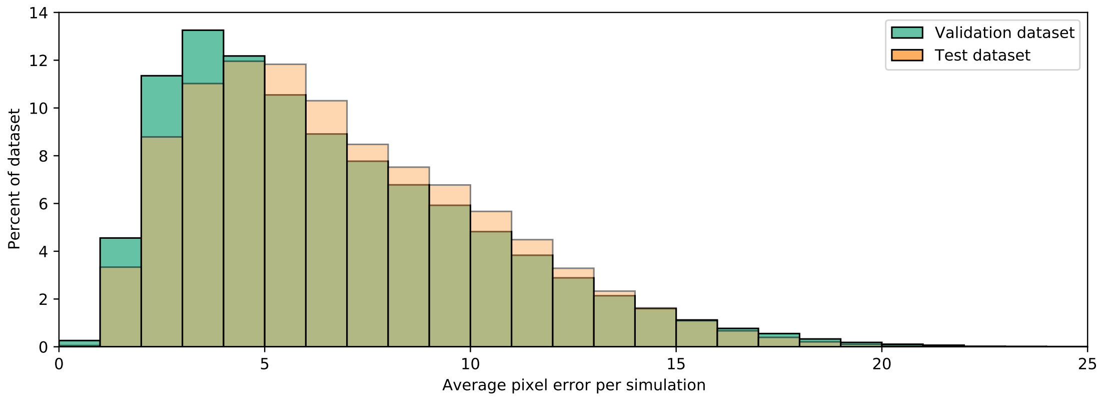
Our first step was to analyze the quality of our simulator. The quality of our simulator network is largely determined by the similarity of synthesized images to real ones. During training, simulated images that deviated from ground truth images were penalized by the L1 loss and the adversarial discriminator. Therefore, our generator learned to map the data distribution in our training data set. To gather the first insights into the generalization ability of our generator network, we investigated the quality of synthesized images. Therefore we simulated each template in our datasets DS_{GAN}^{val} and DS_R. We denormalized the simulations and capacitive ground truth blobs from pixel values between -1 and 1 to range from 0 to 255. Figure 5.1 (a) and (b) show the capacitive pixel value distribution of our validation and test set and the respective simulations. Each bar represents a pixel value range of 15. We see that our simulator reliably maps the target data distributions of both the validation and test set in the lower range of pixel values. However, our simulator produces more pixel values in the range of 210 to 240 and fewer pixel values in the range from 240 to 255



(a) Pixel value distribution of DS^{val}_{GAN}



(b) Pixel value distribution of DS_R

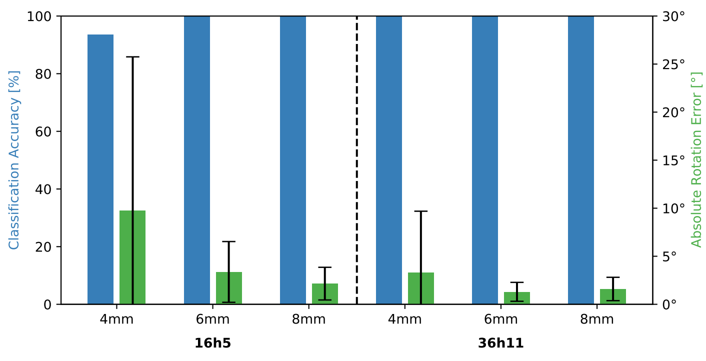


(c) Average pixel error on our validation and test set

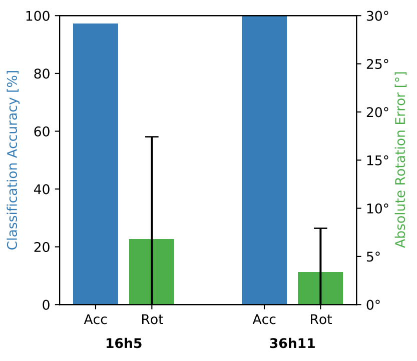
Figure 5.1: (a) and (b): pixel value distributions of simulations and ground truth capacitive recordings. (c) Histogram depicting the average pixel errors between capacitive recordings and the respective simulations. The height of a bar represents the percentage of the dataset with pixel errors falling in the range shown on the x-axis of the graph.

than the ground truth data. Figure 5.1 (c) shows the average absolute pixel differences between simulated and ground truth capacitive images for both DS^{val}_{GAN} and DS_R, denoted as validation and test dataset, respectively. We display which percentages of each dataset fall within a specific average pixel error range. For our validation dataset DS^{val}_{GAN}, our simulator achieves an average absolute pixel error of 7.6 (SD = 18.1) and an average absolute pixel error of 7.9 (SD = 18.4) for DS_R. This puts the concerns about different pixel activations from Figure 5.1 (a) and (b) into perspective. Even though there are significant differences in the number of pixel values in the range from 210 to 255, the average differences are 7.6 and 7.9 for DS^{val}_{GAN} and DS_R.

To further investigate the feasibility of our simulator network, we look at actual simulated capacitive fiducials. For this visual inspection, we simulated a total of 12 different classes from DS_R. Figure 5.3 contains selected templates used for simulating capacitive footprints, respective ground truth capacitive images recorded during our data collection, and simulated capacitive images. The 12 classes originate from 4 different markers, each in Size 4, 6, and 8mm. In addition to varying Sizes of each marker, we simulate rotation changes with increasing Size by rotating the template by 45° and 90° for sizes 6mm and 8mm. Visually, our simulations reflect the ground truth capacitive images in most details. Due to the limited touchscreen sensor resolution, our simulations and our



(a) Model results by AprilTag Type and Size



(b) Model results by AprilTag Type

Figure 5.2: This image shows the classification accuracies and absolute rotation errors for our first and second experiments concerning our recognizer pipeline. (a) shows the results for the 6 recognizers trained on markers of the same Type and have the same pixel Size. (b) shows the results of two recognizers trained on all markers of one Type.

ground truth capacitive images of fiducials with a pixel size of 4mm lose most of their fine-grained inner structure. With increasing pixel size, the visibility of the inner structure becomes clearer for both simulations and ground truth images.

5.2 Recognizer Network on AprilTags

Since we trained our simulator on AprilTag fiducials, the second step of our evaluation involves analyzing our detection pipeline on such fiducials. Therefore, we explored several combinations for training our detection pipeline to gain further insight into its performance. We trained every recognizer listed in this chapter as described in Section 4.3.3. Additionally, we only make use of templates contained in DS_R to train our recognizer on AprilTag fiducials which our simulator was not trained on (see Figure 4.4). Depending on the experiment, we split DS_R, and only access desired parts of it.

5.2.1 Recognition Within Type and Size

Our first experiment investigated the performance of our recognizer pipeline concerning AprilTags of the same Type and Size. There are 10 markers for each Type (16h5, 36h11) of each Size (4, 6, 8mm) contained in DS_R. Thus, we trained one recognizer for each combination of Type $\times$ Size, resulting in 6 independent recognizer networks that classify capacitive markers and predict their rotation. We evaluated each recognizer on the respective ground truth capacitive images in DS_R of specified Type and Size. Figure 5.2a displays the test accuracies and the average absolute rotation errors for this experiment. For the capacitive images of Type 16h5 contained in DS_R, we achieved classification accuracies of 93.5%, 100%, 100% for Sizes 4, 6, and 8mm, respectively. For the capacitive images of Type 36h11 contained in DS_R, we achieved classification accuracies of 99.9%, 100%, 100% for Sizes 4, 6, and 8mm, respectively. Concerning the absolute rotation errors, our recognizers predict the rotation of 16h5 capacitive images with average errors of 9.8° (SD = 16°),

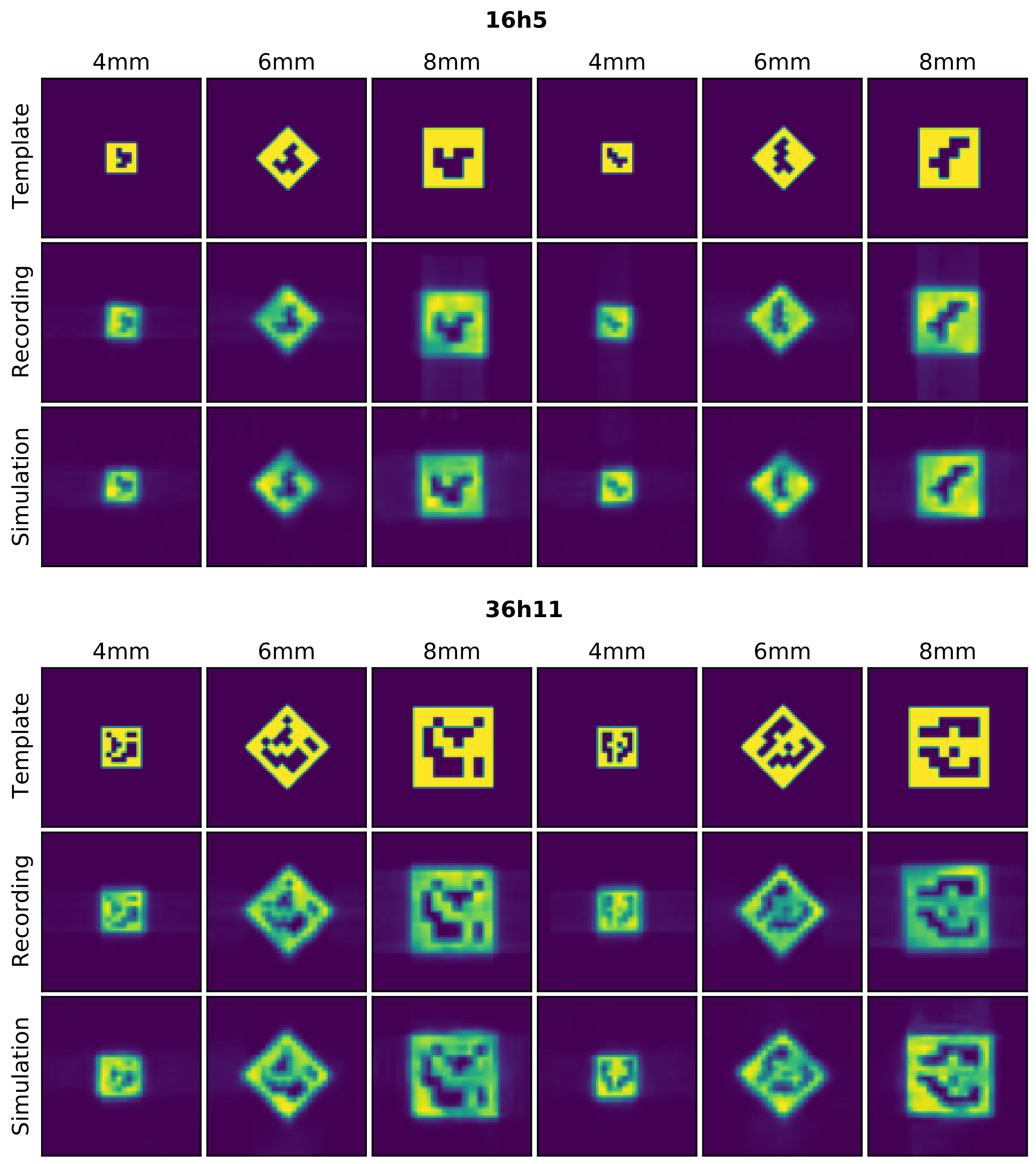


Figure 5.3: Each column shows a geometrical template, a recorded capacitive blob, and a simulation result of the respective geometric template. The simulated templates in the top image are $13, 14_{16h5} \times 4, 6, 8$mm. The simulated templates in the bottom image are $38, 55_{36h11} \times 4, 6, 8$mm. We illustrate the capabilities of our simulator network by showing simulations in different orientations. All templates and capacitive recordings shown in this image were not used for training the simulator network. For both capacitive ground truth images and simulated capacitive images, the inner structure of the AprilTag fiducials gets clearer with increasing Size.

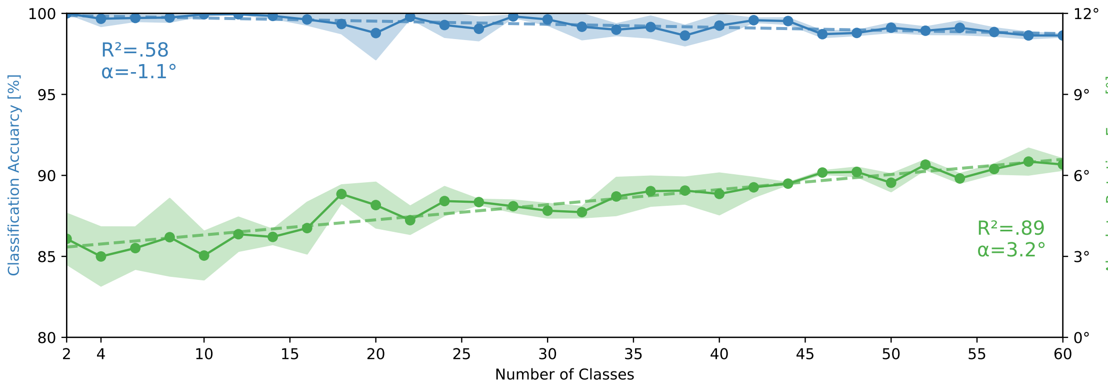


Figure 5.4: Test results for increasing the number of randomly sampled classes from DS_R. The x-axis denotes the number of classes a recognizer was trained on. The left y-axis shows the average classification accuracy, and the right y-axis depicts the average absolute rotation error. We trained each step independently a total of three times. The light areas enclosing data points correspond to the standard deviation over all runs.

3.4° (SD = 3.2°), 2.2° (SD = 1.7°) for SIZES 4, 6, and 8mm, respectively. For markers of TYPE 36h11 the average rotation errors are 3.3° (SD = 6.4°), 1.3° (SD = 1°), 1.6° (SD = 1.2°) for SIZES 4, 6, and 8mm, respectively.

5.2.2 Recognition Within Type

In our second experiment, we looked at all markers of one TYPE, omitting the variable SIZE. Additionally, we wanted to investigate how our model performs on more than 10 classes. Thus, we trained two independent recognizer networks. Each network is responsible for distinguishing between 30 different markers and predicting the rotation of the markers, all of the respective TYPE. Figure 5.2b shows the results of the recognizers trained in this experiment. On the test set DS_R our recognizer for markers of TYPE 16h5 achieves a classification accuracy of 97.3 % and an average absolute rotation error of 6.8° (SD = 10.6°). Our recognizer for markers of TYPE 36h11 has a classification accuracy of 97.3 % and an average absolute rotation error of 3.4° (SD = 4.5°).

5.2.3 Stepwise Increase of Classes

Our third experiment aimed to determine the applicability and robustness of our recognizer pipeline for a variable but steadily increasing number of classes. Here, we started with two random classes and increased the number of classes step-wise by 2. The final class count was 60 (10 markers × 2 TYPES × 3 SIZES). To get a more informative result, we performed each training step three times and thereby trained 90 recognizers. In summary, taking all trained recognizers into account, a total of 40,107,376 capacitive images were used for testing. We present the final results of all 90 trained models in Figure 5.4. With an increasing number of classes, our recognizer still accurately distinguishes between classes and predicts the rotation of markers. The average classification error over all models is 99.31% (SD = .66). Classification accuracies range from 96.47% (20 classes) to 100%. The average rotation errors of all 90 models range from 1.71° to 7.01°, and average 4.97°

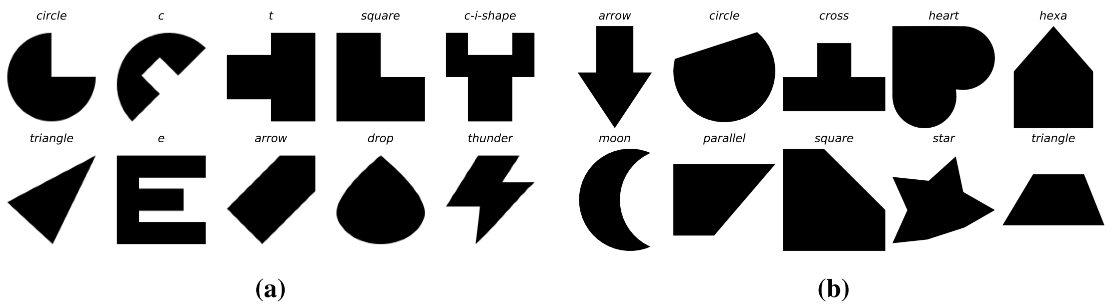


Figure 5.5: This image depicts the shapes we used for testing our simulator network on generating other shapes than AprilTag fiducials. (a) shows the shapes we used to generate DS_{Shape}. (b) shows the shapes by Schmitz et al. [71].

(SD = 1.19°) for all models. For increasing classes, the models have an overall slight downward trend for accuracies, as a linear fit reveals with $R^2 = .58$ (*slope* = −0.020). For 60, the maximum number of distinct AprilTags in DS_R, the 3 trained models achieve an average accuracy of 98.6%, meaning that on average we can correctly classify 850,338 samples in DS_R, with an average rotation error of 6.4° (SD = 9.2°). However, for increasing classes, we also observe an increasing rotation error. Using a linear fit, we show $R^2 = .89$ (*slope* = .056). Table 5.1 highlights the average results for the recognizers trained in the setting of 10, 30, and 60 classes.

5.3 Recognizer Network on Alternative Markers

To test the limits of our pipeline, we investigated the simulation of other markers than AprilTag fiducials. For external validity and showcasing the quality of our simulator network, we investigated the performance on other markers than AprilTag fiducials. To test our recognizer network only trained on simulated capacitive footprints, we used the capacitive images contained within DS_{Shape}. Additionally, we used the recorded data provided by Schmitz et al. [71] to test our pipeline. This allowed us to perform a cross-device validation as the authors used an LG Nexus 5 smartphone (15×27 8-bit raw capacitive matrix, 4.1mm dot-pitch size) to capture capacitive images. We trained our recognizer pipeline as described in Section 4.3.3. However, as templates x_t, we used the shapes used to generate our DS_{Shape} and the dataset by Schmitz et al. Both our shapes and the shapes by Schmitz et al. can be seen in Figure 5.5 (a) and (b), respectively. The results gathered for experiments in this section can be seen in Figure 5.7, and exact numbers can be found in Table 5.1.

We performed a visual inspection to gather insights into the quality of simulations for DS_{Shape} and small markers in general. Figure 5.6 depicts selected templates x_t, capacitive ground truth blobs x_c, and simulations y_g of all Widths. The Widths (8, 12, 16mm) of our shapes are multiples of the size of the capacitive pixels (4mm) with which the images were captured. However, for our 8, 12, and 16mm shapes the footprint is 89%, 75%, and 56% smaller than our smallest AprilTag (16h5 4mm $\hat{=}$ 576mm^2). Thus, capacitive images are blurry, and their structure is hard to recognize. With increasing size, the capacitive images gain more structural information, and our simulator can pick up details.

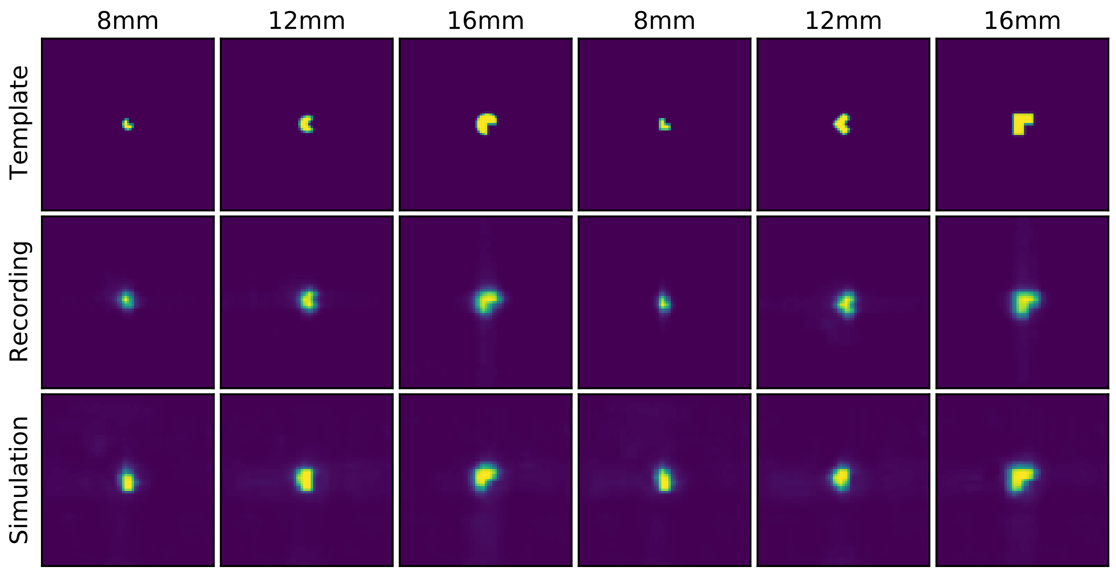
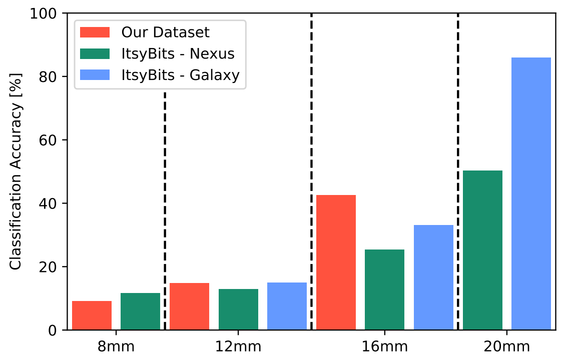


Figure 5.6: Each column shows a geometrical template, a recorded capacitive blob and a simulation result of the respective geometric template. The shape markers simulated are $circle, square \times 8, 12, 16$mm.

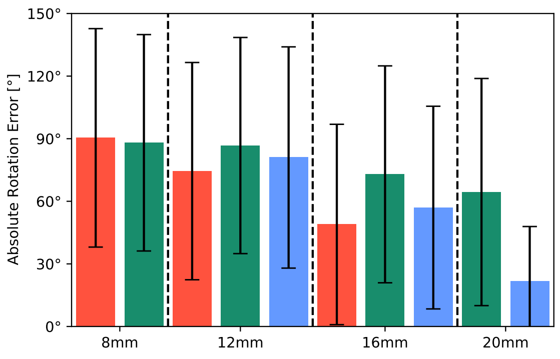
We trained 3 different recognizers on all 10 shapes within the three shape WIDTHS. Compared to our AprilTag fiducials, the accuracies strongly dropped with a value of 9.2% for WIDTH 8mm, being worse than picking a random class. However, for increasing WIDTH, accuracies grow to 14.7% and 42.9% for 12 and 16mm, respectively. The same pattern applies to rotation errors, where the recognizer trained on 8mm shapes predicts the angle on average 90.4° (SD = 52.3°) different from the original orientation. With increasing WIDTH errors decrease to 74.5° (SD = 52.1°) and 48.9° (SD = 48°) for 12 and 16mm, respectively.

Next, we used the shapes provided by Schmitz et al. [71], which were also recorded on a Samsung Galaxy Tab S2 tablet. The training procedure was the same as for our shapes in DS_{Shape}, where we trained 3 different recognizers; however, now for the recording WIDTHS 12, 16, and 20mm. Especially for the 20mm markers, there is a drastic improvement allowing to correctly classify 85.9% of these markers. With decreasing WIDTHS, we can see a trend towards the results gathered for our 12mm and 16mm markers, but with lower classification accuracies and higher rotation errors for the data by Schmitz et al. [71]. Therefore, we assume that this is due to the shape design.

When comparing our results of recognizer networks trained only on simulated data to the domain-specific model results by Schmitz et al. [71], we see an overall reduction in performance. On average, we are 48.7% worse in classification and 35.4° less accurate in predicting the correct orientation. It is not surprising that our results are not as good as a domain-specific model. However, the testing results on externally recorded data or with our recorded data are comparable and even reach > 80% for large markers. This suggests that the recognizer network is not causing the bad performance; but instead, our simulator is not yet capable of generating capacitive footprints of small markers. This can be supported by our visual investigation that shows simulations of small markers are blurry and not distinguishable from each other. However, this is unsurprising since our simulator network has never seen such small markers during training.



(a) Shapes classification accuracies



(b) Shapes rotation errors

Figure 5.7: Classification accacuracies (a) and absolute rotation errors (b) for our shape dataset and the datasets by Schmitz et al. [71].

5.3.1 Recognition of Nexus 5 Data

We carried out all experiments with data recorded on one device with the same dot-pitch size of 4.022mm. Thus, we next investigated if our simulator network can generate sufficient data for devices with a different dot-pitch by appropriately scaling the marker templates. We used the second half of the dataset by Schmitz et al. [71], which contains capacitive recordings of fiducials from a Nexus 5 device. Since we did not have a detailed description for the data extraction, we performed our pre-processing steps based on a data investigation. Of the 269,867 frames from the Nexus 5 recordings, we filtered out 576 frames due to corrupted or missing data. Next, we used a 3×3 inverse identity kernel to remove noise artifacts that are no larger than one pixel. Afterward, we filtered 7580 frames that did not contain a detectable blob. We used the remaining 261,711 blobs for cross-device validation.

We trained four different recognizers for the four different WIDTHS in this dataset. The trend for the results is in line with the performance on DS_{Shape} and the Galaxy Tab S2 recordings by Schmitz et al. [71]. With increasing WIDTH, classification accuracies increase, and rotation errors decrease. However, a comparison of the Nexus 5 and Galaxy Tab results shows that the simulations tend to favor the Galaxy Tab's pictures. For the markers of the same WIDTH, accuracies drop 15.17% on average, and rotation errors increase on average by 6.53° for Nexus 5 recordings. The domain-specific models of Schmitz et al. [71] surpass our Nexus 5 models by far with on average 59.1% better classification accuracies and rotation errors that are on average 61.2° lower. We reason this because we trained our simulator with data recorded on a Galaxy Tab S2, which is why our Galaxy Tab recognizers outperform the Nexus 5 recognizers. Still, we have shown that we can simulate capacitive fiducials for other dot-pitches than 4.022mm to achieve classification accuracies $> 50\%$ for sufficiently large markers.

Table 5.1: Accuracies and rotation errors of our different recognizer networks. The first column shows the data which we simulated for training and testing our networks. The third and fourth columns denote the number of samples and classes in our test set. ZeroR is the baseline accuracy for predicting only the most frequent class. Columns RF depict the results of our random forest classifiers and regressors.

			Classification				**Rotation**		
					Baseline				**Baseline**
Experiment	**Samples**	**Classes**	**Accuracy**	**F1-Score**	**ZeroR**	**RF**	**MAE**	**SD**	**RF**
16h5 – 4mm only	139,652	10	93.5 %	0.93	10.5 %	9.8 %	9.8	16.0	86.3
16h5 – 6mm only	141,384	10	100.0 %	1.0	10.6 %	22.3 %	3.4	3.2	83.4
16h5 – 8mm only	145,460	10	100.0 %	1.0	11.5 %	45.4 %	2.2	1.7	69.7
36h11 – 4mm only	145,940	10	99.9 %	1.0	10.3 %	12.0 %	3.3	6.4	87.1
36h11 – 6mm only	145,456	10	100.0 %	1.0	10.5 %	36.7 %	1.3	1.0	57.9
36h11 – 8mm only	144,520	10	100.0 %	1.0	10.8 %	97.2 %	1.6	1.2	14.3
16h5 – all	426,496	30	97.3 %	0.97	3.9 %	24.7 %	6.8	10.6	81.1
36h11 – all	435,916	30	100.0 %	1.0	3.6 %	47.5 %	3.4	4.5	58.7
Random AprilTag 10	143,258	10	99.9 %	1.0	10.6 %	31.5 %	3.0	6.2	66.2
Random AprilTag 30	431,644	30	99.6 %	1.0	3.7 %	30.6 %	4.7	6.7	70.7
Random AprilTag 60	862,412	60	98.6 %	0.99	1.9 %	24.6 %	6.4	9.2	70.6
Shapes – 8mm only	123,812	10	9.2 %	0.04	11.1 %	11.2 %	90.4	52.3	89.2
Shapes – 12mm only	140,632	10	14.7 %	0.06	10.5 %	10.3 %	74.5	52.1	89.6
Shapes – 16mm only	138,524	10	42.6 %	0.4	10.6 %	9.8 %	48.9	48.0	86.5
External Shape Validity									
12mm – Galaxy [71]	4,924	10	14.9 %	0.11	10.3 %	10.5 %	81.0	53.0	83.2
16mm – Galaxy [71]	4,950	10	33.1 %	0.24	10.6 %	10.0 %	57.0	48.6	83.2
20mm – Galaxy [71]	4,950	10	85.9 %	0.86	10.6 %	13.9 %	21.8	26.1	83.1
8mm – Nexus 5 [71]	66,645	10	11.7 %	0.06	11.3 %	12.2 %	88.0	51.9	92.4
12mm – Nexus 5 [71]	64,795	10	12.8 %	0.09	11.1 %	10.2 %	86.7	51.8	86.5
16mm – Nexus 5 [71]	66,239	10	25.3 %	0.21	12.0 %	11.7 %	72.9	52.0	84.3
20mm – Nexus 5 [71]	64,032	10	50.3 %	0.49	11.1 %	13.2 %	64.4	54.4	85.3

6 Applications

In the following, we present several use cases of using or model in real-world applications. Here, we first discuss how to prepare the model for on-device deployment. Then we present four possible use cases using AprilTag fiducials.

6.1 Model Deployment

After freezing a trained recognizer, we exported it into a *tflite* file which we run with *TensorFlow Lite* on our Samsung Galaxy Tab S2. For exporting, we used an 8-bit quantization to compress the model file to a size of 1.79MB. First, we evaluated the effect of quantization and found that it resulted in no performance loss for our TYPE 16h5 all SIZES model. We can still classify 30 AprilTag markers with an accuracy of 97.3% and predict their rotation with an average error of 6.83° (SD = 10.6°). On the device, the prediction time for one sample was, on average, 42.15ms (SD = 3.62ms) calculated on a total of 2370 samples. This allows us to process markers in real-time as pre-processing (blob extraction and normalization) takes on average 15.45ms (SD = 3.62ms), and we pull capacitive images every $\sim$ 110ms.

Our experiments in Chapter 5 show that we can classify AprilTag fiducials with accuracies > 93%. We built an application that displays the class and the predicted rotation of detected AprilTags on the capacitive screen (see Figure 6.1). For filtering out touches by fingers, we calculated the area of blobs and excluded events if they were below a certain threshold. As a recognizer, we used the TYPE 16h5 all SIZES model. The app reliably displays the correct class and approximated rotation as soon as a valid blob is detected.

6.2 Example Applications

We explored multiple other examples where our approach can enhance the interaction space on capacitive touchscreens. We built four applications that utilize the full capabilities of our recognizer network architecture.

To support playful interaction such as discovering new information, we mapped semantic information of the tangible to digital information and built an application that we envision in a museum setting (see Figure 6.2 (a)). Especially children can benefit from such a hands-on interaction [1].

Tangibles can enrich the interactions for smart home environments, as we show with our tangible smart home application (see Figure 6.2 (b)). Here, users can place the tangibles onto the control panel and immediately receive the desired feedback. We used two tangibles that control light sources within a prototypical smart home by simply placing and sliding them on the control panel.

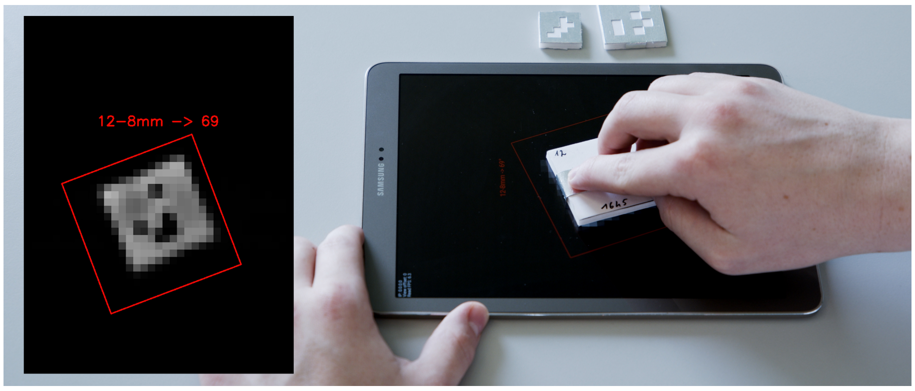


Figure 6.1: This sample application displays the capacitive matrix, including the blob boundaries, current classification result, and rotation prediction of fiducial tangibles on the screen.

To give haptic feedback to today's touch-enabled radios, we built a radio application that allows changing radio stations and volume by placing and rotating the respective station or volume tangible (see Figure 6.2 (c)).

Lastly, to support a collaborative gaming environment, we built a Minecraft-inspired D&D game, where users can place painted metal figurines attached to AprilTag fiducials onto the screen (see Figure 6.2 (d)). Based on the detected marker and figurine, its typical creature sound is played. We prototyped actions that as soon a monster is detected, a placed hero can attack it, or a hero can trade with detected merchants.

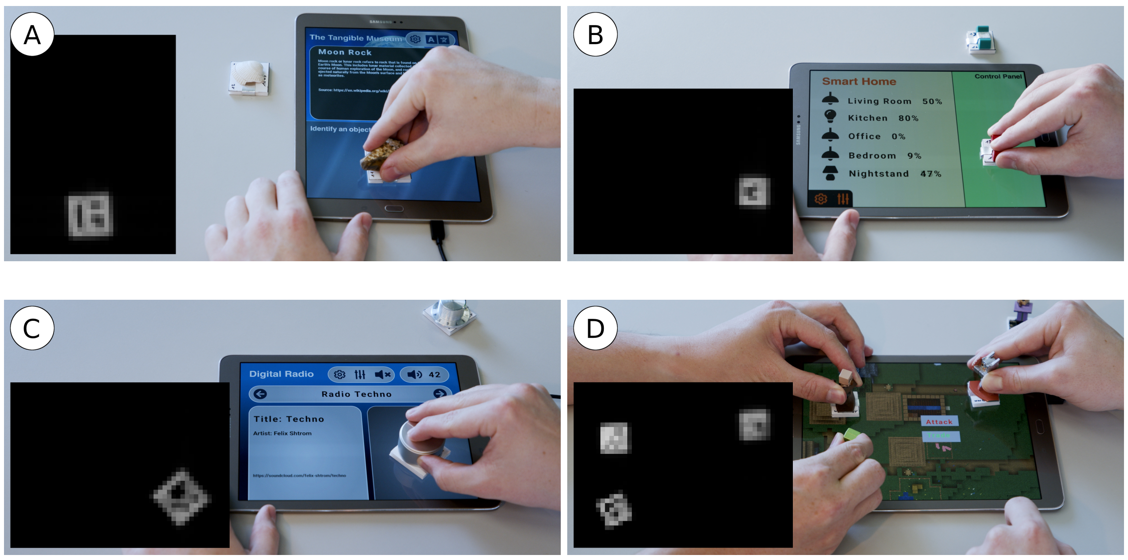


Figure 6.2: Demonstration of the capabilities of our proposed technique. We deployed different recognizer networks on a Samsung Galaxy Tab S2 and built four different applications that enable capacitive tangibles. (a) a tangible museum, (b) a tangible smart home, (c) a tangible radio, and (d) a tangible game.

7 Discussion

To avoid time-consuming data collection studies and to simplify the realization of TUIs on capacitive devices, we have developed a pipeline in which capacitive images are simulated and subsequently used to train recognizer models. Inspired by previous simulation techniques, e.g., [26, 99], we decided to use an cGAN, which learns to transfer the style of a capacitive sensor to sketches of conductive markers. Others in the domain of object recognition on capacitive screens required domain knowledge and hyperparameter tuning for recognizers to function sufficiently [40, 50, 73]. Our pipeline eliminates the need for both. The overall results are promising, showing a path towards tangibles on today's capacitive screens. Finally, to real-time test our recognizer, we exported them onto an end-device and built multiple apps featuring the direct output of the models. Using AprilTags, both classification and rotation prediction work exceptionally when testing our example applications.

We conditioned and trained our simulator solely on imprints and capacitive images of conductive AprilTag fiducials. This is reflected in the evaluation of our trained recognizers, where we achieved an average classification accuracy of 99.7% on unseen capacitive images for our simulator. The same holds for our rotation predictions on AprilTags, where our models can estimate the orientation of markers with an average error of 4.9° (SD = 1.4°). Overall these are promising results. On the other hand, we achieve the lowest classification accuracy and our highest rotation error with 93.5% and 9.8° (SD = 16°) for our smallest AprilTags of TYPE 16h5 SIZE 4mm . While 130,574 correctly classified capacitive images out of 139,652 total capacitive images are still acceptable, deploying this model onto an end-device might lead to users' frustration due to wrongly classified markers or falsely predicted rotations. We argue that this is because of the combination of the low-resolution capacitive matrix with a dot-pitch of 4.022mm and might also result from slight deviations from actual to recorded rotation. When our simulator network is conditioned on templates of fiducials in specific rotations, it applies the style of the capacitive sensor to this exact template. Thus, if our recorded ground truth rotation deviates from the actual rotation during training, the L1 loss penalizes the simulator, which leads to a shift from recorded to actual rotation. This can result in a rounding of the corners of our simulations and blurring the inner structures of the simulation (cf. Figure 5.3 *Recording* and *Simulation*). Especially for our smallest markers, the clearest representation of the internal structure is necessary for a reliable classification as they encode 16 bits within 256mm^2 or $\sim$ 16 capacitive pixels. This also explains why our accuracies for all other AprilTags are better as they either encode more bits (TYPE 36h11) or encode the same amount of bits within a larger capacitive area due to increasing SIZE. That statement is further supported by markers of TYPE 36h11 SIZE 4mm whose pixel size is equal to our worst-performing AprilTag, but they encode over twice the amount of bits which shows that the information density for small markers is crucial. Thus, our recognizer can extract more features, highlighting the need for design considerations when developing small conductive fiducial markers.

In addition, we would like to mention that the fabrication of our conductive markers could have also led to inconsistencies between capacitive recordings and simulations. Since we produced our markers by hand, we could not avoid slight deviations from the actual shape. Therefore, our capacitive recordings may not reflect the desired dimensions of the markers. However, with the ability to print conductive material using 3D printers within errors of submillimeter range, such deviations could be avoided. Printing conductive material to enable tangibles on capacitive touchscreens has shown to be effective since various restrictions compared to manufacturing by hand can be omitted [71]. Also, additional functionalities can be embedded within 3D printed tangibles [16, 72].

Our results suggest that for conductive AprilTags, our simulator can model the responses of the capacitive touch sensor. Nevertheless, when evaluating the results of our experiments on shapes, we can see that our simulations do not represent real sampled capacitive images as classification accuracies are $< 43\%$. There are multiple reasons why we can not simulate shapes as good as pixel-based fiducials like AprilTags. For example, we previously stated, our simulator network was only conditioned and trained on sketches and capacitive images of AprilTags. Thus, it is only natural that our simulator can not recreate the level of detail for conductive shapes than for conductive AprilTags. Additionally, as the WIDTHS of our shapes were 8, 12, and 16mm, their imprint rests on fewer capacitive pixels. The footprint of shapes with such WIDTHS corresponds to 11%, 25%, and 44% of the footprint for our smallest AprilTag. Therefore, even if our simulations are good enough, we argue that the Convolution layers of our recognizer network can not pick up enough information from the capacitive image. Otherwise, for shapes of WIDTH 20mm, we achieve a classification accuracy of 85.9%, which strengthens us in the assumption that our simulations are suitable for sufficiently large shapes.

While we performed most evaluations on capacitive data from one device, we also investigated how our simulations perform for data captured on other devices with a dot-pitch other than that of our device. Therefore, we used the shape data by Schmitz et al. [71] captured on a Nexus 5 that has a dot-pitch of 4.1mm. Again, we could better simulate large markers of WIDTH 20mm with an accuracy of 50.3% than 8mm, which we correctly classified in 11.7% of the cases. Overall our results for classification and rotation prediction are not nearly as good as for our shape data. However, it is hard to define whether the poor performance was due to incorrect simulations or the quality of the data set as the capacitive matrix of the Nexus 5 tends to produce noisy images. Since the authors did not specify how they filtered out the noise in their data to the state they used for training their recognizer, we could not pre-process their data to this state. Schmitz et al. [71] state that after their pre-processing steps, they had a usable 193,145 capacitive images, while after applying our basic pre-processing were left with 261,711 images. Thus, as we observe the same trend for higher accuracies with increasing WIDTH, we argue that our simulator can generate capacitive imprints for other devices However, the dataset quality had a strong impact on our results.

8 Conclusion

In this thesis, we have presented a toolkit for simulating capacitive footprints of conductive fiducials which are used to train a recognizer network. Our contributions comprise a simulator network, an cGAN, that simulates based on sketches of conductive fiducials the corresponding response of a capacitive touch sensor. Additionally, we provide a recognizer network structure capable of classifying conductive markers while simultaneously predicting the rotation with which the markers rest on the capacitive screen. We trained our simulator network on the capacitive data of 30 fiducial markers and evaluated its performance on over 90 unseen and structural different markers. In an extensive evaluation, we have shown that our pipeline can predict the correct classes for conductive AprilTag fiducials with an average accuracy of over 98%. On top of that, we can predict the correct orientation of AprilTag fiducials with an average error of 4.8°. Thus, to bring back effortless tangibles to capacitive touchscreens, we evaluated our recognizer models, solely trained on simulated data on a mobile device. As our models did not suffer any loss in performance, we could fully utilize their capabilities within multiple developed prototype applications. After an average training time of 3.4h, developers can export and use a trained recognizer model without the need for manual data collection, hyper-parameter tuning, and domain knowledge.

Where our results are promising on AprilTag fiducials, we have not achieved the same results for custom conductive shapes. As we have observed better classification accuracies and lower rotation errors for bigger shapes, we argue that our simulator network is not yet fully capable of simulating small-scale conductive shapes. Thus, future research needs also to consider small-scale shapes when designing and training a simulator network. This should increase the simulator's performance on such markers. In this work, we evaluated the performance of our simulator and recognizer on our recorded markers. However, the background of this work was to make the recording process of markers obsolete for future designers. Therefore, one should provide future designers and developers with a metric that highlights how likely it is for markers in their marker set to confuse their recognizer. Our simulator network performed a style transfer from sketches of conductive markers to the capacitive touch sensor's response. Since these responses are low resolution, our simulator also produces a low-resolution output. Research could change the simulation space, currently a mapping from high-resolution sketch to low-resolution capacitive images, to be a mapping from low-resolution capacitive images to high-resolution sketches. While the capacitive super-resolution algorithm by Mayer et al. [51] already works on tangibles, it requires moving objects across the screen. Streli and Holz [76] utilized a domain-specific GAN only to upsample finger touches and estimate contact shapes on capacitive screens. For example, researchers could integrate both solutions into a system that upscales any conductive object and allow designers to create custom detection metrics on upscaled images of touched objects.

Bibliography

[1] Alissa Antle. “Tangibles: Five Properties to Consider for Children”. In: *Proceedings of the 2007 Tangible User Interfaces in Context and Theory Workshop*. San Jose, California, 2007 (cit. on p. 41).

[2] Alissa N. Antle, Alyssa F. Wise, Amanda Hall, Saba Nowroozi, Perry Tan, Jillian Warren, Rachael Eckersley, Michelle Fan. “Youtopia: A Collaborative, Tangible, Multi-Touch, Sustainability Learning Activity”. In: *Proceedings of the 12th International Conference on Interaction Design and Children*. IDC ’13. New York, New York, USA: Association for Computing Machinery, 2013, pp. 565–568. ISBN: 9781450319188. DOI: 10.1145/2485760.2485866 (cit. on p. 14).

[3] Martin Arjovsky, Soumith Chintala, Léon Bottou. “Wasserstein GAN”. In: (2017). arXiv: 1701.07875 [stat.ML] (cit. on p. 17).

[4] Gary Barrett, Ryomei Omote. “Projected-Capacitive Touch Technology”. In: *Information Display* 26.3 (2010), pp. 16–21. DOI: 10.1002/j.2637-496X.2010.tb00229.x (cit. on p. 15).

[5] Patrick Baudisch, Torsten Becker, Frederik Rudeck. “Lumino: Tangible Blocks for Tabletop Computers Based on Glass Fiber Bundles”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2010, pp. 1165–1174. ISBN: 9781605589299. DOI: 10.1145/1753326.1753500 (cit. on p. 14).

[6] Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, Patrick Pérez. “This dataset does not exist: training models from generated images”. In: (2019). arXiv: 1911.02888 [cs.CV] (cit. on pp. 12, 18).

[7] Andrew Brock, Jeff Donahue, Karen Simonyan. *Large Scale GAN Training for High Fidelity Natural Image Synthesis*. 2019. arXiv: 1809.11096 [cs.LG] (cit. on p. 17).

[8] Liwei Chan, Stefanie Müller, Anne Roudaut, Patrick Baudisch. “CapStones and ZebraWidgets: Sensing Stacks of Building Blocks, Dials and Sliders on Capacitive Touch Screens”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 2189–2192. ISBN: 9781450310154. DOI: 10.1145/2207676.2208371 (cit. on p. 15).

[9] Peter Dalsgaard, Kim Halskov. “Tangible 3D Tabletops: Combining Tangible Tabletop Interaction and 3D Projection”. In: *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*. NordiCHI ’12. Copenhagen, Denmark: Association for Computing Machinery, 2012, pp. 109–118. ISBN: 9781450314824. DOI: 10.1145/2399016.2399033 (cit. on p. 14).

[10] George W. Fitzmaurice, William Buxton. “An Empirical Evaluation of Graspable User Interfaces: Towards Specialized, Space-Multiplexed Input”. In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI ’97. Atlanta, Georgia, USA: Association for Computing Machinery, 1997, pp. 43–50. ISBN: 0897918029. DOI: 10.1145/258549.258578 (cit. on p. 15).

[11] George W. Fitzmaurice, Hiroshi Ishii, William A. S. Buxton. “Bricks: Laying the Foundations for Graspable User Interfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 442–449. ISBN: 0201847051. DOI: 10.1145/223904.223964 (cit. on pp. 11, 13, 15).

[12] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. “Generative adversarial networks”. In: *arXiv* (2014). arXiv: 1406.2661 [stat.ML] (cit. on pp. 17, 18, 24).

[13] Timo Götzelmann, Daniel Schneider. “CapCodes: Capacitive 3D Printable Identification and On-Screen Tracking for Tangible Interaction”. In: *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*. NordiCHI ’16. Gothenburg, Sweden: Association for Computing Machinery, 2016. ISBN: 9781450347631. DOI: 10.1145/2971485.2971518 (cit. on p. 16).

[14] Sebastian Günther, Florian Müller, Martin Schmitz, Jan Riemann, Niloofar Dezfuli, Markus Funk, Dominik Schön, Max Mühlhäuser. “CheckMate: Exploring a Tangible Augmented Reality Interface for Remote Interaction”. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI EA ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–6. ISBN: 9781450356213. DOI: 10.1145/3170427.3188647 (cit. on p. 16).

[15] Sebastian Günther, Martin Schmitz, Florian Müller, Jan Riemann, Max Mühlhäuser. “BYO*: Utilizing 3D Printed Tangible Tools for Interaction on Interactive Surfaces”. In: *Proceedings of the 2017 ACM Workshop on Interacting with Smart Objects*. SmartObject ’17. Limassol, Cyprus: Association for Computing Machinery, 2017, pp. 21–26. ISBN: 9781450349024. DOI: 10.1145/3038450.3038456 (cit. on p. 16).

[16] Changyo Han, Katsufumi Matsui, Takeshi Naemura. “ForceStamps: Fiducial Markers for Pressure-Sensitive Touch Surfaces to Support Rapid Prototyping of Physical Control Interfaces”. In: *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI ’20. Sydney NSW, Australia: Association for Computing Machinery, 2020, pp. 273–285. ISBN: 9781450361071. DOI: 10.1145/3374920.3374924 (cit. on pp. 16, 44).

[17] Zekun Hao, Arun Mallya, Serge Belongie, Ming-Yu Liu. “GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds”. In: (2021). arXiv: 2104.07659 [cs.CV] (cit. on p. 17).

[18] G. E. Hinton, R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: https://science.sciencemag.org/content/313/5786/504.full.pdf (cit. on pp. 17, 25).

[19] Michael S. Horn, Erin Treacy Solovey, Robert J. K. Jacob. “Tangible Programming and Informal Science Learning: Making TUIs Work for Museums”. In: *Proceedings of the 7th International Conference on Interaction Design and Children*. IDC ’08. Chicago, Illinois: Association for Computing Machinery, 2008, pp. 194–201. ISBN: 9781595939944. DOI: 10.1145/1463689.1463756 (cit. on p. 13).

[20] Sharon Horwood, Jeromy Anglim. “Problematic smartphone usage and subjective and psychological well-being”. In: *Computers in Human Behavior* 97 (2019), pp. 44–50. ISSN: 0747-5632. DOI: 10.1016/j.chb.2019.02.028 (cit. on p. 11).

[21] Kohei Ikeda, Koji Tsukada. “CapacitiveMarker: Novel Interaction Method Using Visual Marker Integrated with Conductive Pattern”. In: *Proceedings of the 6th Augmented Human International Conference*. AH ’15. Singapore, Singapore: Association for Computing Machinery, 2015, pp. 225–226. ISBN: 9781450333498. DOI: 10.1145/2735711.2735783 (cit. on p. 16).

[22] Kaori Ikematsu, Itiro Siio. “Ohmic-Touch: Extending Touch Interaction by Indirect Touch through Resistive Objects”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–8. ISBN: 9781450356206. DOI: 10.1145/3173574.3174095 (cit. on p. 16).

[23] Sergey Ioffe, Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach, David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. arXiv: 1502.03167 (cit. on p. 25).

[24] Hiroshi Ishii. “Tangible Bits: Beyond Pixels”. In: *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*. TEI ’08. Bonn, Germany: Association for Computing Machinery, 2008, pp. xv–xxv. ISBN: 9781605580043. DOI: 10.1145/1347390.1347392 (cit. on p. 14).

[25] Hiroshi Ishii, Brygg Ullmer. “Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms”. In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI ’97. Atlanta, Georgia, USA: Association for Computing Machinery, 1997, pp. 234–241. ISBN: 0897918029. DOI: 10.1145/258549.258715 (cit. on pp. 11, 13, 18).

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR ’17. IEEE, 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632 (cit. on pp. 17, 24–27, 43).

[27] Robert J. K. Jacob, Hiroshi Ishii, Gian Pangaro, James Patten. “A Tangible Interface for Organizing Information Using a Grid”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’02. Minneapolis, Minnesota, USA: Association for Computing Machinery, 2002, pp. 339–346. ISBN: 1581134533. DOI: 10.1145/503376.503437 (cit. on p. 13).

[28] Ali Jahanian, Xavier Puig, Yonglong Tian, Phillip Isola. “Generative Models as a Data Source for Multiview Representation Learning”. In: (2021). arXiv: 2106.05258 [cs.CV] (cit. on pp. 12, 18).

[29] Sergi Jordà. “On Stage: the Reactable and other Musical Tangibles go Real”. In: *International Journal of Arts and Technology* 1 (2008), pp. 268–287. DOI: 10.1504/IJART.2008.022363 (cit. on p. 14).

[30] Sergi Jordà, Günter Geiger, Marcos Alonso, Martin Kaltenbrunner. “The ReacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces”. In: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. TEI ’07. Baton Rouge, Louisiana: Association for Computing Machinery, 2007, pp. 139–146. ISBN: 9781595936196. DOI: 10.1145/1226969.1226998 (cit. on pp. 11, 13, 14).

[31] Martin Kaltenbrunner, Ross Bencina. “ReacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction”. In: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. TEI ’07. Baton Rouge, Louisiana: Association for Computing Machinery, 2007, pp. 69–74. ISBN: 9781595936196. DOI: 10.1145/1226969.1226983 (cit. on pp. 13, 14).

[32] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, Timo Aila. “Alias-Free Generative Adversarial Networks”. In: (2021). arXiv: 2106.12423 [cs.CV] (cit. on p. 17).

[33] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila. “Analyzing and Improving the Image Quality of StyleGAN”. In: (2020). arXiv: 1912.04958 [cs.CV] (cit. on p. 17).

[34] Kunihiro Kato, Kaori Ikematsu, Yoshihiro Kawahara. “CAPath: 3D-Printed Interfaces with Conductive Points in Grid Layout to Extend Capacitive Touch Inputs”. In: *Proc. ACM Hum.-Comput. Interact.* 4.ISS (Nov. 2020). DOI: 10.1145/3427321 (cit. on p. 16).

[35] Kunihiro Kato, Homei Miyashita. “3D Printed Physical Interfaces That Can Extend Touch Devices”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST ’16 Adjunct. Tokyo, Japan: Association for Computing Machinery, 2016, pp. 47–49. ISBN: 9781450345316. DOI: 10.1145/2984751.2985700 (cit. on p. 16).

[36] Sven Kratz, Tilo Westermann, Michael Rohs, Georg Essl. “CapWidgets: Tangile Widgets versus Multi-Touch Controls on Mobile Devices”. In: *CHI ’11 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 1351–1356. ISBN: 9781450302685. DOI: 10.1145/1979742.1979773 (cit. on pp. 15, 18).

[37] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *University of Toronto* (May 2012) (cit. on pp. 11, 17).

[38] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386 (cit. on p. 11).

[39] Abinaya Kumar, Aishwarya Radjesh, Sven Mayer, Huy Viet Le. “Improving the Input Accuracy of Touchscreens Using Deep Learning”. In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI EA ’19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–6. ISBN: 9781450359719. DOI: 10.1145/3290607.3312928 (cit. on p. 28).

[40] Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, Niels Henze. "PalmTouch: Using the Palm as an Additional Input Modality on Commodity Smartphones". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3173934 (cit. on pp. 11, 43).

[41] Huy Viet Le, Sven Mayer, Niels Henze. "Investigating the Feasibility of Finger Identification on Capacitive Touchscreens Using Deep Learning". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI '19. Marina del Ray, California: ACM, 2019, pp. 637–649. ISBN: 978-1-4503-6272-6. DOI: 10.1145/3301275.3302295 (cit. on p. 11).

[42] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 17).

[43] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017. DOI: 10.1109/CVPR.2017.19 (cit. on p. 17).

[44] Jakob Leitner, Michael Haller. "Geckos: Combining Magnets and Pressure Images to Enable New Tangible-Object Design and Interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2011, pp. 2985–2994. ISBN: 9781450302289. DOI: 10.1145/1978942.1979385 (cit. on p. 14).

[45] Chuan Li, Michael Wand. "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks". In: *Computer Vision*. ECCV '16. Cham: Springer International Publishing, 2016, pp. 702–716. ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_43. arXiv: 1604.04382 [cs.CV] (cit. on pp. 17, 26).

[46] Rong-Hao Liang, Han-Chih Kuo, Liwei Chan, De-Nian Yang, Bing-Yu Chen. "GaussStones: Shielded Magnetic Tangibles for Multi-Token Interactions on Portable Displays". In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: Association for Computing Machinery, 2014, pp. 365–372. ISBN: 9781450330695. DOI: 10.1145/2642918.2647384 (cit. on pp. 15, 18).

[47] Andrew L Maas, Awni Y Hannun, Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proceedings of the International Conference on Machine Learning*. Vol. 30. ICML '13 1. 2013, p. 3 (cit. on p. 25).

[48] Karola Marky, Martin Schmitz, Verena Zimmermann, Martin Herbers, Kai Kunze, Max Mühlhäuser. "3D-Auth: Two-Factor Authentication with Personalized 3D-Printed Items". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–12. ISBN: 9781450367080. DOI: 10.1145/3313831.3376189 (cit. on p. 16).

[49] Michael Mathieu, Camille Couprie, Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: (2016). arXiv: 1511.05440 [cs.LG] (cit. on p. 24).

[50] Sven Mayer, Huy Viet Le, Niels Henze. “Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks”. In: *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ISS ’17. Brighton, United Kingdom: Association for Computing Machinery, 2017, pp. 220–229. ISBN: 9781450346917. DOI: 10.1145/3132272.3134130 (cit. on pp. 11, 19, 28, 43).

[51] Sven Mayer, Xiangyu Xu, Chris Harrison. “Super-Resolution Capacitive Touchscreens”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21. New York, New York, USA: Association for Computing Machinery, May 8, 2021. DOI: 10.1145/3411764.3445704. Forthcoming (cit. on pp. 11, 16, 18, 45).

[52] *Microsoft Corporation: PixelSense*. https://en.wikipedia.org/wiki/Microsoft_PixelSense. accessed August 26, 2021 (cit. on p. 14).

[53] Mehdi Mirza, Simon Osindero. “Conditional Generative Adversarial Nets”. In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784 (cit. on pp. 12, 17).

[54] Shakir Mohamed, Balaji Lakshminarayanan. “Learning in Implicit Generative Models”. In: (2017). arXiv: 1610.03483 [stat.ML] (cit. on p. 18).

[55] Hyoungsik Nam, Ki-Hyuk Seol, Junhee Lee, Hyeonseong Cho, Sang Jung. “Review of Capacitive Touchscreen Technologies: Overview, Research Trends, and Machine Learning Approaches”. In: *Sensors* 21 (July 2021), p. 4776. DOI: 10.3390/s21144776 (cit. on p. 15).

[56] Alex Olwal, Andrew D. Wilson. “SurfaceFusion: Unobtrusive Tracking of Everyday Objects in Tangible User Interfaces”. In: *Proceedings of Graphics Interface 2008*. GI ’08. Windsor, Ontario, Canada: Canadian Information Processing Society, 2008, pp. 235–242. ISBN: 9781568814230 (cit. on p. 14).

[57] Santiago Pascual, Antonio Bonafonte, Joan Serrà. *SEGAN: Speech Enhancement Generative Adversarial Network*. 2017. arXiv: 1703.09452 [cs.LG] (cit. on p. 17).

[58] James Patten, Hiroshi Ishii, Jim Hines, Gian Pangaro. “Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’01. Seattle, Washington, USA: Association for Computing Machinery, 2001, pp. 253–260. ISBN: 1581133278. DOI: 10.1145/365024.365112 (cit. on p. 13).

[59] Esben Warming Pedersen, Kasper Hornbæk. “Tangible Bots: Interaction with Active Tangibles in Tabletop Interfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2011, pp. 2975–2984. ISBN: 9781450302289. DOI: 10.1145/1978942.1979384 (cit. on pp. 11, 14).

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 31).

[61] Ken Perlin. “Improving Noise”. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 681–682. ISBN: 1581135211. DOI: 10.1145/566570.566636 (cit. on p. 29).

[62] Ben Piper, Carlo Ratti, Hiroshi Ishii. “Illuminating Clay: A 3-D Tangible Interface for Landscape Analysis”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’02. Minneapolis, Minnesota, USA: Association for Computing Machinery, 2002, pp. 355–362. ISBN: 1581134533. DOI: 10.1145/503376.503439 (cit. on p. 13).

[63] Alec Radford, Luke Metz, Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG] (cit. on p. 17).

[64] Gonzalo Ramos, Matthew Boulos, Ravin Balakrishnan. “Pressure Widgets”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2004, pp. 487–494. ISBN: 1581137028. DOI: 10.1145/985692.985754 (cit. on p. 11).

[65] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee. “Generative Adversarial Text to Image Synthesis”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan, Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1060–1069. arXiv: 1605.05396 (cit. on p. 17).

[66] Jun Rekimoto. “SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’02. Minneapolis, Minnesota, USA: Association for Computing Machinery, 2002, pp. 113–120. ISBN: 1581134533. DOI: 10.1145/503376.503397 (cit. on pp. 15, 18).

[67] Mitchel Resnick. “Behavior Construction Kits”. In: *Commun. ACM* 36.7 (July 1993), pp. 64–71. ISSN: 0001-0782. DOI: 10.1145/159544.159593 (cit. on p. 13).

[68] Olaf Ronneberger, Philipp Fischer, Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, Alejandro F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4 (cit. on pp. 17, 25).

[69] Dominik Schmidt, Ming Ki Chong, Hans Gellersen. “HandsDown: Hand-Contour-Based User Identification for Interactive Surfaces”. In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. NordiCHI ’10. Reykjavik, Iceland: Association for Computing Machinery, 2010, pp. 432–441. ISBN: 9781605589343. DOI: 10.1145/1868914.1868964 (cit. on p. 13).

[70] Martin Schmitz, Mohammadreza Khalilbeigi, Matthias Balwierz, Roman Lissermann, Max Mühlhäuser, Jürgen Steimle. “Capricate: A Fabrication Pipeline to Design and 3D Print Capacitive Touch Sensors for Interactive Objects”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software amp; Technology*. UIST ’15. Charlotte, NC, USA: Association for Computing Machinery, 2015, pp. 253–258. ISBN: 9781450337793. DOI: 10.1145/2807442.2807503 (cit. on p. 16).

[71] Martin Schmitz, Florian Müller, Max Mühlhäuser, Jan Riemann, Huy Viet Le. “Itsy-Bits: Fabrication and Recognition of 3D-Printed Tangibles with Small Footprints on Capacitive Touchscreens”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21. Yokohama, Japan: ACM, 2021. DOI: 10.1145/3411764.3445502 (cit. on pp. 11, 12, 16, 18, 19, 28, 36–39, 44).

[72] Martin Schmitz, Jürgen Steimle, Jochen Huber, Niloofar Dezfuli, Max Mühlhäuser. "Flexibles: Deformation-Aware 3D-Printed Tangibles for Capacitive Touchscreens". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 1001–1014. ISBN: 9781450346559. DOI: 10.1145/3025453.3025663 (cit. on pp. 16, 44).

[73] Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, Andreas Bulling. "KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens Using Deep Learning". In: *Proceedings of Mensch Und Computer 2019*. MuC'19. Hamburg, Germany: Association for Computing Machinery, 2019, pp. 387–397. ISBN: 9781450371988. DOI: 10.1145/3340764.3340767 (cit. on pp. 11, 28, 43).

[74] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, Ronald M. Summers. "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162 (cit. on p. 11).

[75] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958 (cit. on p. 24).

[76] Paul Streli, Christian Holz. "CapContact: Super-Resolution Contact Areas from Capacitive Touchscreens". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445621 (cit. on pp. 11, 17, 45).

[77] Hideyuki Suzuki, Hiroshi Kato. "An Educational Tool for Collaborative Learning: AlgoBlock". In: *Cognitive Studies: Bulletin of the Japanese Cognitive Science Society* 2.1 (1995), pp. 36–47. DOI: 10.11225/jcss.2.1_36 (cit. on p. 13).

[78] Satoshi Suzuki, Keiicji Abe. "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: 10.1016/0734-189X(85)90016-7 (cit. on p. 21).

[79] B. Ullmer, H. Ishii. "Emerging frameworks for tangible user interfaces". In: *IBM Systems Journal* 39.3.4 (2000), pp. 915–931. DOI: 10.1147/sj.393.0915 (cit. on pp. 14, 18).

[80] Brygg Ullmer, Hiroshi Ishii, Dylan Glas. "MediaBlocks: Physical Containers, Transports, and Controls for Online Media". In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 379–386. ISBN: 0897919998. DOI: 10.1145/280814.280940 (cit. on p. 13).

[81] John Underkoffler, Hiroshi Ishii. "Urp: A Luminous-Tangible Workbench for Urban Planning and Design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1999, pp. 386–393. ISBN: 0201485591. DOI: 10.1145/302979.303114 (cit. on p. 14).

[82] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, Haiyan Zhang. "Project Zanzibar: A Portable and Flexible Tangible Interaction Platform". In: *Proceedings of the 2018 CHI Conference*

on Human Factors in Computing Systems. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3174089 (cit. on p. 15).

[83] Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergård, Jan Borchers. "PERCs: Persistently Trackable Tangibles on Capacitive Multi-Touch Displays". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software amp; Technology*. UIST '15. Charlotte, NC, USA: Association for Computing Machinery, 2015, pp. 351–356. ISBN: 9781450337793. DOI: 10.1145/2807442.2807466 (cit. on p. 15).

[84] Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergård, Jan Borchers. "PUCs: Detecting Transparent, Passive Untouched Capacitive Widgets on Unmodified Multi-Touch Displays". In: *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*. ITS '13. St. Andrews, Scotland, United Kingdom: Association for Computing Machinery, 2013, pp. 101–104. ISBN: 9781450322713. DOI: 10.1145/2512349.2512791 (cit. on p. 15).

[85] John Wang, Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection". In: *RSJ International Conference on Intelligent Robots and Systems*. IROS '16. IEEE, 2016, pp. 4193–4198. DOI: 10.1109/IROS.2016.7759617 (cit. on pp. 12, 16).

[86] Xiaolong Wang, Abhinav Gupta. "Generative Image Modeling Using Style and Structure Adversarial Networks". In: *Computer Vision*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, Max Welling. ECCV 2016. Cham: Springer International Publishing, 2016, pp. 318–335. ISBN: 978-3-319-46493-0. DOI: 10.1007/978-3-319-46493-0_20 (cit. on p. 24).

[87] Roy Want, Kenneth P. Fishkin, Anuj Gujar, Beverly L. Harrison. "Bridging Physical and Virtual Worlds with Electronic Tags". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1999, pp. 370–377. ISBN: 0201485591. DOI: 10.1145/302979.303111 (cit. on p. 13).

[88] Mark Weiser. "The Computer for the 21^{st} Century". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3.3 (July 1999), pp. 3–11. ISSN: 1559-1662. DOI: 10.1145/329124.329126 (cit. on p. 13).

[89] Mark Weiser, John Seely Brown. "Designing calm technology". In: *PowerGrid Journal* 1.1 (1996), pp. 75–85 (cit. on p. 13).

[90] Lyndon White (https://stats.stackexchange.com/users/36769/lyndon-white). *Encoding Angle Data for Neural Network*. Cross Validated. (version: 2018-07-03). 2018 (cit. on p. 28).

[91] Andrew D. Wilson, Raman Sarin. "BlueTable: Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-Based Handshaking". In: *Proceedings of Graphics Interface 2007*. GI '07. Montreal, Canada: Association for Computing Machinery, 2007, pp. 119–125. ISBN: 9781568813370. DOI: 10.1145/1268517.1268539 (cit. on p. 13).

[92] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, Joshua B. Tenenbaum. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*. 2017. arXiv: 1610.07584 [cs.CV] (cit. on p. 17).

[93] Robert Xiao, Scott Hudson, Chris Harrison. “CapCam: Enabling Rapid, Ad-Hoc, Position-Tracked Interactions Between Devices”. In: *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ISS ’16. Niagara Falls, Ontario, Canada: Association for Computing Machinery, 2016, pp. 169–178. ISBN: 9781450342483. DOI: 10.1145/2992154.2992182 (cit. on p. 15).

[94] Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, Ming-Hsuan Yang. “Learning to Super-Resolve Blurry Face and Text Images”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017. DOI: 10.1109/ICCV.2017.36 (cit. on p. 17).

[95] Neng-Hao Yu, Li-Wei Chan, Seng Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Fang-I Hsiao, Lung-Pan Cheng, Mike Chen, Polly Huang, Yi-Ping Hung. “TUIC: Enabling Tangible Interaction on Capacitive Multi-Touch Displays”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 2995–3004. ISBN: 9781450302289. DOI: 10.1145/1978942.1979386 (cit. on pp. 15, 16).

[96] Neng-Hao Yu, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Meng-Han Lee, Mike Y. Chen, Yi-Ping Hung. “Clip-on Gadgets: Expanding Multi-Touch Interaction Area with Unpowered Tactile Controls”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: Association for Computing Machinery, 2011, pp. 367–372. ISBN: 9781450307161. DOI: 10.1145/2047196.2047243 (cit. on pp. 15, 18).

[97] Xin Yu, Fatih Porikli. “Ultra-Resolving Face Images by Discriminative Generative Networks”. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, Max Welling. Cham: Springer International Publishing, 2016, pp. 318–333. ISBN: 978-3-319-46454-1. DOI: 10.1007/978-3-319-46454-1_20 (cit. on p. 17).

[98] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, Sanja Fidler. “DatasetGAN: Efficient Labeled Data Factory With Minimal Human Effort”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 10145–10155. arXiv: 2104.06490 [cs.CV] (cit. on p. 18).

[99] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: vol. abs/1703.10593. 2017. arXiv: 1703.10593 (cit. on pp. 17, 24, 43).

[100] Oren Zuckerman, Ayelet Gal-Oz. “To TUI or Not to TUI: Evaluating Performance and Preference in Tangible vs. Graphical User Interfaces”. In: *Int. J. Hum.-Comput. Stud.* 71.7–8 (July 2013), pp. 803–820. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2013.04.003 (cit. on p. 15).

[101] Guillaume Zufferey, Patrick Jermann, Aurélien Lucchi, Pierre Dillenbourg. “TinkerSheets: Using Paper Forms to Control and Visualize Tangible Simulations”. In: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. TEI ’09. Cambridge, United Kingdom: Association for Computing Machinery, 2009, pp. 377–384. ISBN: 9781605584935. DOI: 10.1145/1517664.1517740 (cit. on p. 14).

All links were last followed on August 27, 2021.

Declaration