Conductive Fiducial Tangibles for Everyone: A Data Simulation-Based Toolkit Using Deep Learning

BENEDICT STEUERLEIN, University of Stuttgart, Germany
SVEN MAYER, LMU Munich, Germany

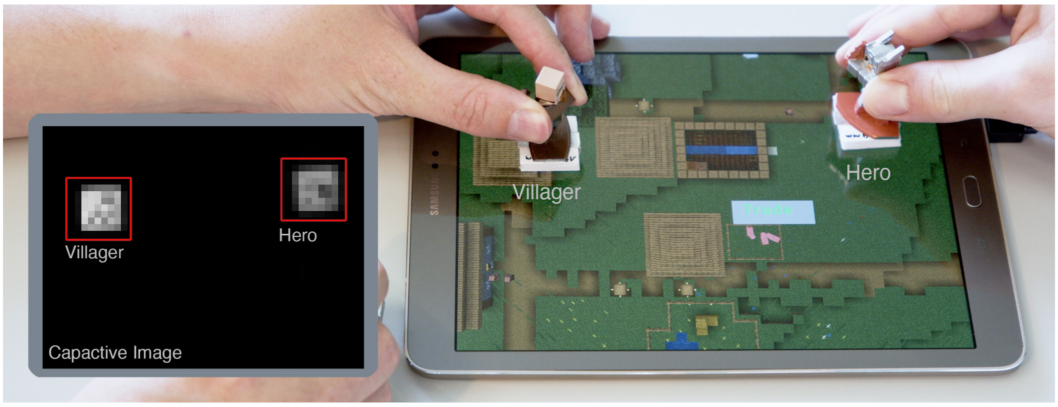


Fig. 1. A Minecraft-inspired Dungeons & Dragons game using our recognizer model to determine the character placed on the screen. On the lower left, we show the corresponding capacitive image.

While tangibles enrich the interaction with touchscreens, with projected capacitive screens being mainstream, the recognition possibilities of tangibles are nearly lost. Deep learning approaches to improve the recognition of conductive triangles require collecting huge amounts of data and domain-specific knowledge for hyperparameter tuning. To overcome this drawback, we present a toolkit that allows everyone to train a deep learning tangible recognizer based on simulated data. Our toolkit uses a pre-trained Generative Adversarial Network to simulate the imprint of fiducial tangibles, which we then use to train a deployable recognizer based on our pre-defined neuronal network architecture. Our evaluation shows that our approach can recognize fiducial tangibles such as AprilTags with an average accuracy of 99.3% and an average rotation error of only 4.9°. Thus, our toolkit is a plug-and-play solution requiring no domain knowledge and no data collection but allows designers to use deep learning approaches in their design process.

CCS Concepts: • **Human-centered computing** → **Touch screens**; **Human computer interaction (HCI)**.

Additional Key Words and Phrases: capacitive sensing, conductive tangibles, datasets, neural networks

ACM Reference Format:
Benedict Steuerlein and Sven Mayer. 2022. Conductive Fiducial Tangibles for Everyone: A Data Simulation-Based Toolkit Using Deep Learning. *Proc. ACM Hum.-Comput. Interact.* 6, MHCI, Article 183 (September 2022), 22 pages. https://doi.org/10.1145/3546718

Authors' addresses: Benedict Steuerlein, University of Stuttgart, Stuttgart, 70174, Germany, st111340@stud.uni-stuttgart.de; Sven Mayer, LMU Munich, Munich, 80337, Germany, info@sven-mayer.com.

2573-0142/2022/9-ART183 $15.00
https://doi.org/10.1145/3546718

1 INTRODUCTION

Tangibles have long been shown to enrich the interaction between humans and computers. On the other hand, direct input using touchscreens is the dominant way for most people to interact with their devices. Combining tangible interaction and touch input into a single interface is a long-standing challenge. While first-generation tabletops were able to detect fingers and tangibles using camera systems, they were bulky and, thus, have basically been fully replaced by projective-capacitive touchscreens [7, 61]. Over the last years, capacitive sensors have evolved from tracking only the fingertip to multi-purpose sensing devices, e.g., [13, 17, 29]. Hence, the software stack of today's touchscreens is fine-tuned to extract fingers touching a screen from the low-resolution "capacitive image," making them incapable of recognizing structural information needed to detect tangibles placed on them. However, the sensor itself picks up on some of the structural information, which must, therefore, be present in the capacitive image.

Bringing back conductive tangibles to capacitive touchscreens is an unsolved challenge [40, 47, 58]. The most promising approach to make use of the capacitive image is to use deep neural networks to extract the information, e.g., [31, 39, 47]. Here, the capacitive image is passed into a model to extract higher-level information from it. However, training supervised deep neural networks requires a lot of application-specific training data, cf. [15, 18, 21, 26]. In the human-computer interaction (HCI) domain, data collected in user studies are used to train a recognizer, which is time-consuming and allows only to train one application-specific model, cf. [31, 47, 51]. Thus, today designers lack the flexibility to design for their use-cases and are required to train deep neural networks.

The aim of our toolkit is to overcome the time-consuming data collection needed to train a deep learning conductive fiducial recognizer and to allow machine learning (ML) novices to recognize their newly designed fiducial markers on touchscreens. For great usability with a low entry barrier, our toolkit uses the 2D imprint of fiducial markers as input to generate simulated imprints to train a recognizer model. The final model can be directly deployed on a touch device. For this, we use a *simulator network* to generate capacitive images from the 2D sketches. As a *simulator network*, we use a conditional Generative Adversarial Network (cGAN) [41], a technique used already for image generation, e.g., Tonolini et al. [53]. This allows generating images that represent the hypothetical imprint of a conductive tangible and, therefore, replacing the data collection study needed in prior work. While a data collection study becomes obsolete with the uses of the *simulator network*, designers of fiducial tangibles would still need to train a domain-specific recognizer using the generated data, e.g., Schmitz et al. [47]. To also overcome this issue, we present a second deep learning network to finally also recognize fiducial tangibles using our general *recognizer architecture*. For this, we used the *simulator network* as a pre-trained model to simulate data and train the recognizers tailored to the designer's needs. Thus, train a domain-specific recognizer without data collation study and deep learning knowledge.

With this paper, we present a toolkit allowing everyone to train their own conductive fiducial tangible recognizer. We make the first step toward a *simulator network* to simulate capacitive data. We show that we can train a *simulator network* to generate the capacitive imprint of fiducial markers (e.g., AprilTags) placed on projective-capacitive touchscreens. We show that our *simulator network*, trained only on 10 different marker templates in 3 pitch sizes, is sufficient to generate an arbitrary fiducial tangible imprint, giving designers full flexibility to shape their products. On our test dataset, the simulator achieves a mean pixel error of 7.6. We further show that our recognizer architecture can be used in various settings using AprilTags with an accuracy $> 93\%$ (M = 99.3%, SD = .9%) and average rotation errors of only 4.9° (SD = 7.5°) without additional hyperparameter tuning when a fiducial recognizer. Our toolkit, comprising the pre-trained model and the recognizer architecture, allows ML novices, e.g., designers, to effectively design, train, and deploy deep learning fiducial tangible recognizers on today's touchscreens, which was previously not possible. Thus, we allow designers to create various 2D fiducial imprints, and, in return, they will get a ready-to-deploy model. Finally, we showcase the potential of our toolkit by presenting a set of applications.

2 RELATED WORK

We review three important areas to enable everyone to build their own deep learning recognizer for conductive fiducial markers. First, we review important work in the domain of tangible user interfaces, highlighting the importance of object tracking on today's capacitive screens. Second, we review the work of capacitive sensing upon which we will build. Lastly, we review deep learning approaches, especially GANs, enabling us to simulate capacitive data.

2.1 Tangible User Interfaces

Many of the early on-screen tangibles utilize the technologies used back in the 2000s in the tabletop systems, which were infrared-based and even RGB camera-based. Both were not fine-tuned to primarily track finger touches; additionally, they had high resolution, which is beneficial to track an object's imprint on the screen either through visual markers (e.g., Kaltenbrunner and Bencina [23]) or object contours (e.g., Wilson and Sarin [66]). This enabled a seamless integration of Tangible User Interfaces (TUIs), e.g., [5, 6, 20, 43, 54–56]. Such approaches include using the Touch API of the smartphone to detect markers, e.g., [9, 70]. However, these are all limited by low information density per TUI; otherwise, the controller detects them as one single touch. On the other hand, with projected capacitive screens being the dominant sending capability of touchscreens tracking, TUIs are also feasible on mobile devices, e.g., [40, 47].

The main challenges with these systems were not the accuracy and performance but portability. The camera projector setups were overall bulky and with the low-resolution capacitive touchscreen becoming mainstream for finger tracking, tabletops also moved toward small capacitive sensing technologies. However, this diminished the possibility of tracking the imprint of tangibles placed on the screen accurately. Until today, researchers struggle to bring back the plentiful interaction possibilities we have seen on the early tabletops. Over the last decade, a wide range of projects have attempted to bring the tracking capabilities back. As the capacitive sensor is tuned toward finger detection, detecting multiple pin-like markers can also be done at low cost for both the recognition and tangible production, cf. [65, 67]. Other approaches have used low-density passive conductive fiducials [3, 14, 25, 34, 48, 58, 59] and contour recognition [57, 68], both in close alignment with the approaches for infrared and RGB tracking.

Today, we can manufacture conductive fiducial tangibles with a 3D printer using conductive materials, e.g., [11, 12, 24]. However, their information density is mostly very low. In an effort to improve recognition, Mayer et al. [40] proposed a geometric super-resolution approach to overcome the limitations of the low-resolution capacitive sensor to detect fiducial markers by multi-stacking multiple frames. Moreover, Schmitz et al. [47] proposed a deep learning model for use-case-specific fiducial recognition. Both approaches help restore conductive fiducial tangibles to today's capacitive screens.

2.2 Extended Capacitive Touch Sensing

Capacitive sensing in HCI is generally nothing new, cf. Grosse-Puppendahl et al. [10]; however, in recent years, we have seen an increase in performance, especially due to deep learning techniques and the use of raw capacitive images. In the simplest case, the raw capacitive image is used to improve the touch location [27, 51]. Various authors have investigated possibilities to extend the input space for fingers and hands to be detected with more information than the simple x/y position of the touch itself. The more prominent investigation is toward finger orientation [39, 42, 69], enabling a wide variety of interactions previously not possible. Other investigations track the finger type [30], finger parts [49], or multi-finger gestures [29, 31].

Beyond improved finger identification, researchers have also envisioned detecting hand and body parts. For instance, Le et al. [28] used the capacitive image to detect the palm of the hand. Moreover, Holz et al. [17] pushed this idea to even recognize other body parts, such as the ear of the user. Later, Guo et al. [13] used this idea to enable user identification. Recently, Choi et al. [4] extended this idea to recover the full-hand posture using inverse kinematics based on the imprint of the hand, enabling a wide range of interaction possibilities.

Lately, we have seen a trend toward restoring conductive tangibles to capacitive screens based on the raw capacitive image, which contrasts tracking techniques used previously for TUIs as described above. Here, Mayer et al. [40] presented a multi-frame super-resolution approach to generate a higher resolution imprint of the tangible for improved recognition. Finally and most aligned with this work, Schmitz et al. [47] presented a use-case-specific deep learning model to recognize up to 30 tangible markers.

2.3 Networks for Sample Generation

In 2014, Goodfellow et al. [8] presented the idea of Generative Adversarial Networks (GANs) where they showed the idea of two neural networks playing against each other in order to improve. Later, Mirza and Osindero [41] introduced conditional Generative Adversarial Networks (cGAN), a special type of GANs that allows controlling the output. In detail, with cGANs, developers can direct the model to procure an output from a particular class based on the random vector itself. Influential for the presented work is especially the Image-to-Image translation paper by Isola et al. [21]. They presented a style transfer technique [63], turning one image into another while keeping the overall structure the same but changing the visual appearance. Moreover, one promising application domain for image-to-image translation GANs is synthesizing new, unseen data [2, 71], which is an approach that we will focus on in the following.

In the HCI domain, Streli and Holz [51] implemented a Wasserstein GAN [1] to upsample low-resolution capacitive images of finger touches to make adjacent touches more distinguishable. Moreover, Murray-Smith et al. [42] used a Variational Autoencoder (VAE) for data generation allowing them to forward model data, cf. Tonolini et al. [53]. However, to train their VAE, they collect ground truth data using a data collection study to stabilize their models. This is in contrast to the GAN approach by Streli and Holz [51], which, however, only works for fingers. Thus, we use the variation of the GAN approach to achieve forward modeling. In detail, we use a cGAN [41] with Encoder-Decoder structure, which allows us to be flexible and also to condition the data generation toward a specific fiducial and, therefore, build a recognizer without any additional fiducial marker collection study.

3 METHOD

Assuming a trained *recognizer network* (*R*), we can simply deploy *R* to a touch device and recognize the element and even the rotation or size with which it is placed on the screen, see Figure 2c. We propose training a *simulator network* to generate samples with the same data characteristics, see Figure 2a. The *simulator network* can then be used to generate arbitrary fiducial tangible imprints. This output can then be used to train a recognizer model, see Figure 2b. The trained recognizer can then be deployed and used for inference in interactive scenarios on touchscreen devices, Figure 2c. Such a framework has already been applied successfully in other domains such as human face generation [53].

In detail, the goal is to simulate capacitive images based on the shape and form of the fiducial marker (*Template*, see Figure 2) using a *simulator network*, and then train a *recognizer network* based

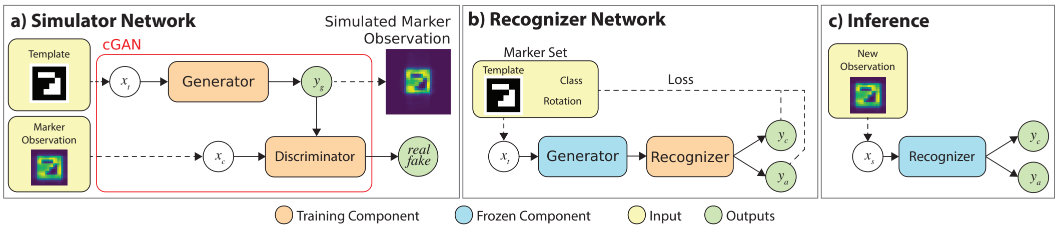


Fig. 2. Architecture of our (a) simulator network, the training process of our recognizer network (b), and the used recognizer R in a deployed state (c).

on simulated data. As a simulator network, we propose using a conditional Generative Adversarial Network (cGAN) [41] to condition the output to reflect the different classes of which the recognizer R should later on classify. A cGAN itself consists of two components, a generator G to generate data and a discriminator D, which judges if the quality of the simulated data is good. D is only used during the adversarial training of the two components. G is well trained if D cannot distinguish between fake and real images, see Figure 2a. The generator G can then be used to simulate capacitive data. If G generalizes, then it can be used to train R solely on simulated data; thus, not needing a data collection study.

From a practical point of view, a designer or developer envisions a *Marker Set* (M) consisting of a number of templates x_t for their fiducial tangible application. Then a developer needs R to deploy in a real application and give the marker class as well as the orientation on the screen. Thus, R is a mapping $R(x_s) \longrightarrow \hat{y}_c, \hat{y}_a$. To train R, we used capacitive simulated data y_g with various orientations obtained from G. To train the initial *simulator network* (consisting of G and D), we need paired data of templates x_t and capacitive recordings of the template x_c. Pair data $\{x_t, x_c\}$ need to be recorded only once to train the models G and D. Based on this, we can train the *simulator network* to generate a new sample $\hat{y}_g$; thus, G is a mapping $G(x_t) \longrightarrow \hat{y}_g$.

3.1 Simulator Network

A traditional Generative Adversarial Network (GAN) [8] does not allow us to condition the output based on the *Templates* (x_t) given by the *Marker Set* (M). However, a cGAN [41] allows to condition the output not on a random vector r but based on a given input. Thus, this is a mapping: $G\{x, r\} \longrightarrow \hat{y}_g$, see Figure 2a. On the other hand, D is trained to determine if the ground truth image x_c or the newly generated image $\hat{y}_g$ is fake.

3.1.1 Simulator Network Objective Function. 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, the generator G wants to produce better quality images. Traditional GANs and various cGANs use a noise vector z to produce output that is not deterministic [63]. However, often the model is able to learn to ignore the noise [21, 38]. One common approach to overcome this issue is to embed Dropout layers [50] into the model structure during training to generate non-deterministic output, which will allow the generation of a wider range of outputs [21], in our case, simulated conductive markers x_g. As we do replace the typical noise vector z with the Dropout layer, our adversarial loss $L_{cGAN}(G, D)$ can be described as the following:

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

Moreover, prior work has shown that it is beneficial to not only rely on the discriminator loss but also on the traditional loss [21], e.g., L1 loss. Thus, we add the L1 loss (pixel-wise loss) of G, defined as:

$$\mathcal{L}_{L1}(G) = \mathrm{E}_{x_t,x_c}[\|x_c - G(x_t)\|_1] \tag{2}$$

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

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda * \mathcal{L}_{L1}(G). \tag{3}$$

3.1.2 Generator Network Architecture. The objective of our model can be described as a style transfer objective. Borrowing techniques from style transfer models, such as Pix2Pix [21], we use a special Encoder-Decoder network [16] with skip connections called U-Net [46], allowing us to keep any input shape given by the *Template*. However, when feeding it through the generator G, the model will add the characteristics of the low-resolution capacitive sensor, thus applying the style transfer. Encoder-Decoder networks [16] downsample the image into a latent space (a representation of the input in a lower-dimensional space). From the latent space, the network then upsamples the data while adding the "new style" – the characteristics of the sensor. To not only rely on the information in the latent space, skip connections between the layers can be added, referred to as U-Net [46]. This generally has been shown to yield better results, cf. Isola et al. [21]. Figure 2a presents the final network architecture.

3.1.3 Discriminator Network Architecture. The overall structure of our discriminator is shown in Figure 2b. Generally, the model is a simple Convolutional Neural Network (CNN) combined with a batch norm [19]. To counteract blurry images, we use a Markovian discriminator (PatchGAN), which has been shown to counteract blurry images by model design [21, 32]. This affects the output layer of the model returning $N \times N$ patches. Moreover, PatchGANs have fewer parameters and, therefore, are faster to train and run.

3.2 Recognizer Network

Our recognizer is a traditional CNN model which based on a capacitive sample x_s predicts the class $\hat{y}_c$ and the orientation $\hat{y}_a$; thus, $R(x_s) \longrightarrow \hat{y}_c, \hat{y}_a$. Such a model is common in processing capacitive images, e.g., [27, 29, 47, 49].

The output for the class prediction $\hat{y}_c$ is a n long vector, representing the one-hot notion of the n classes for the different models a designer can envision. We predict the sine and cosine components ($\hat{y}_{sin}$ and $\hat{y}_{cos}$) of the angle $\hat{y}_a$ for better performance as suggested by White [64]. While this is similar to Schmitz et al. [47], however, they only calculated the components post-prediction. Thus, we calculate the orientation angle a based on: $a = atan2(\hat{y}_{sin}, \hat{y}_{cos})$.

3.2.1 Loss Function. For our rotational regression branch, the network tries to minimize the error between y_a and $\hat{y}_a$ by minimizing the root-mean-square error (RMSE) $\mathcal{L}_{Angle}$. In detail, we minimize the distance between the two components $y_{sin} - \hat{y}_{sin}$ and $y_{cos} - \hat{y}_{cos}$, in the following manner:

$$\mathcal{L}_{Angle}(R) = \sqrt{\frac{1}{b}\sum_{i=1}^{b}(0.5 * ((y_{sin,i} - \hat{y}_{sin,i})^2 + (y_{cos,i} - \hat{y}_{cos,i})^2))}, \tag{4}$$

where y_{sin} and y_{cos} are the ground trough angles of any input x_s and b is the batch size. The loss for our class output branch is the multi-class cross-entropy loss $\mathcal{L}_{Class}$. The final objective of our recognizer is described by

$$R^* = arg\ \min_R \mathcal{L}_{Angle}(R) + \mathcal{L}_{Class}(R). \tag{5}$$

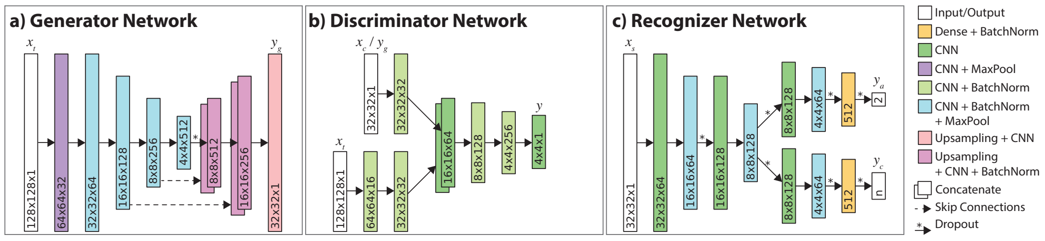


Fig. 3. Structures of our three deep neuronal models.

4 MODEL TRAINING

While our *Simulator Network* only needs to be trained once, we need a small dataset first. Thus, we first collect data to train the *Simulator Network* using AprilTag markers [62] and a larger set to train and test various *recognizer models*. Only then can we perform pre-processing, data augmentation, and hyperparameter tuning.

4.1 Data Collection

To train and test our models, we fabricated a wide range of fiducial markers, cf. Figure 4. We recorded 2 Types of markers, 36h11 AprilTags and 16h5 AprilTags, each with 3 different pitch Sizes per pixel (4, 6, and 8 mm), resulting in 6 different sizes: 16h5 AprilTags with 24, 36, and 48 mm tag width, and 36h11 AprilTags with 32, 48, and 64 mm tag width (see Figure 4). For each combination, we recorded 15 markers, resulting in $2 \times 3 \times 15 = 90$ markers. For generalization purposes, we also recorded 10 different custom *shapes*, see Figure 5a. Each custom shape was fabricated with 3 Widths (8, 12, and 16 mm), resulting in 30 additional markers. We made all markers from aluminum foil glued onto of a 4 mm high foam board. For capacitive sensing, we added a $\sim 1cm$ wide tap that has to be touched (3D-printed markers will include them in the core of the print, e.g., [37, 47]).

For data collection, we use a Samsung Galaxy Tab S2 tablet (9.7") lying flat on a table and plugged into the recording PC. The tablet has 49×37 capacitive pixels (6.33PPI, 4.0127 mm dot-pitch). Commercial devices rarely expose the capacitive image via a public API. Thus, we deployed a custom kernel driver to directly communicate with the Synaptics touch controller via I^2C as done many times before [27, 29, 40]. Here, we achieve a frame rate of ~ 9FPS (~ 110 ms per frame). In line with prior work, e.g., [39, 47], we used an optical motion capture system (OptiTrack-V120:Trio recording at 120 Hz) with millimeter accuracy to record the orientation of the marker. Using three markers attached to a custom apparatus, we were able to track the orientation of the tangibles relative to the tablet.

During our data collection, we recorded a total of 448,261 capacitive frames over the course of 13 h and 41 min. We recorded each marker for an average duration of 6 min 51 s (SD = 33 s).

4.2 Pre-Processing & Data Augmentation

While we timestamped the capacitive images and ground truth orientation of the fiducial markers gained from OptiTrack, the system has a latency of 8.33 ms. Thus, we manually aligned sub-latency shifts by visually inspecting the orientation and change of the capacitive image.

During pre-processing, we identified all capacitive blobs (an imprint of the fiducial marker) by finding the contours [52] on a thresholded version of the capacitive image. Here, we removed all images without any fiducial markers present. Next, we cropped a 32×32 patch around the center of the blob. This allowed us to recognize fiducial markers with a diameter of up to 128 mm (32 dots * 4.0127 mm dot-pitch). For data augmentation, we rotated each sample 3 times by 90°,

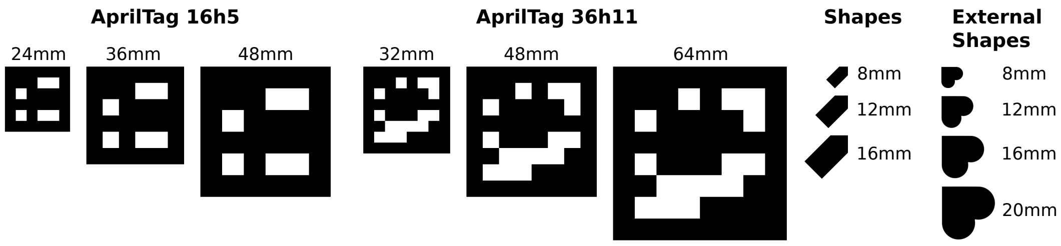


Fig. 4. Size comparison of the differently sized markers in our dataset.

resulting in 3 times the amount of initial data. Thus, we were left with $1,699,044$ samples ($655,392$ 36h11 AprilTags, $640,684$ 16h5 AprilTags, and $402,968$ shapes). Finally, we normalized the input images between -1 and 1 to help the training process and to overcome different capacitive ground conditions that affect the sensing as discussed in prior work, e.g., Choi et al. [4].

4.3 Datasets

We call the final set of data we obtained after our pre-processing and data augmentation step DS (see Section 4.2). The shape recording we put into a separate dataset DS_{Shape}. We use all 36h11 AprilTags and 16h5 AprilTags for training, validation, and testing. These markers we split into two sets: one for the GAN training DS_{GAN} and one to train the recognizer DS_R. We split the data so that one AprilTag is only 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 36h11 ids and 5 different 16h5 ids each in 3 sizes. Therefore, the DS_R set contains the remaining 10 and 10 ids each in 3 sizes, respectively.

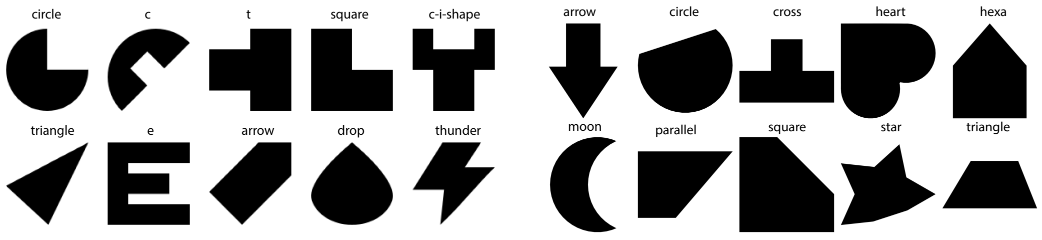
Subsequently, the marker set DS_{GAN} was randomly split into training (DS_{GAN}^{train}) and validation (DS_{GAN}^{val}) subsets using a $70\% : 30\%$ split ($303,565 : 130,099$ samples). While this could potentially lead to overfitting, the latent space reduces the information to counteract overfitting. Moreover, as DS_R does not contain the same markers, we argue that the *simulator network* has to generalize beyond its seen 30 different markers for the recognizer to work with completely new markers. In fact, the *recognizer models* will highlight any weakness of the *simulator network* and the *recognizer models* will act as a test model with its unseen data DS_R set for testing.

As the *recognizer models* will only be trained on simulated data y_g, the set DS_R does not need to be split. Thus, 100% (862,412 samples) of this set can be used to test the quality of the *recognizer models*.

4.4 Simulator Network Training & Tuning

Overall, the simulator network structure is inspired by style transfer papers such as Isola et al. [21]. Combined with a trial-and-error method and hyperparameters tuning, we derived the following model, which is beneficial for conductive fiducial simulation. The task of the generator network is to generate 32×32 capacitive images y_g. While we initially aimed to generate y_g based on a 32×32 representation of the template x_t, we found it beneficial to start with more information. Thus, the generator expects a template to be four times the size of its low-resolution counterpart; therefore, the input template x_t has to be 128×128. To increase the variance of our training set DS_{GAN}, we randomly shifted the samples x_c by ± 1 in both x and y directions, and x_t is shifted by ± 4 due to the scaling.

4.4.1 Generator Model. Figure 3b represents the final model architecture of the generator network with its 5,942,369 parameters. The model starts with a downsampling component as the input is



(a) Shapes in our DS_{Shape} dataset (b) Shapes by Schmitz et al. [47]

Fig. 5. Image depicting the shapes used for testing the simulator network on generating shapes other than AprilTag fiducials: (a) shapes used to generate DS_{Shape} and (b) shapes by Schmitz et al. [47].

larger than the output. The central component is an Encoder-Decoder network [16] with skip connections called U-Net [46]. To avoid deterministic outputs of the model, we employ a Dropout layer [50] after the latent space reduction. The model was trained with an Adam optimizer with a learning rate of .0002 and momentum parameter $\beta_1 = .5$, $\beta_2 = .999$. We used a batch size of 128 to train the model. The training time for the generator model was 66 hours on an Nvidia Tesla V100.

4.4.2 Discriminator Model. The final network architecture of the discriminator model is depicted in Figure 3b with its 736,337 parameters. Here, each CNN layer is followed by a batch normalization layer. All CNN layers use a 3×3 kernel. All CNN layers but the concatenate layer and output layer use a stride of 2. The others use a stride of 1. As activation functions, we chose LeakyReLU [36] for all layers but the output layer. The output layer, the PatchGAN output, uses a linear activation function. All other layer parameters are set to standard. The model was trained with an Adam optimizer with a learning rate of .0002 and momentum parameter $\beta_1 = .5$, $\beta_2 = .999$. We used a batch size of 128 to train the model.

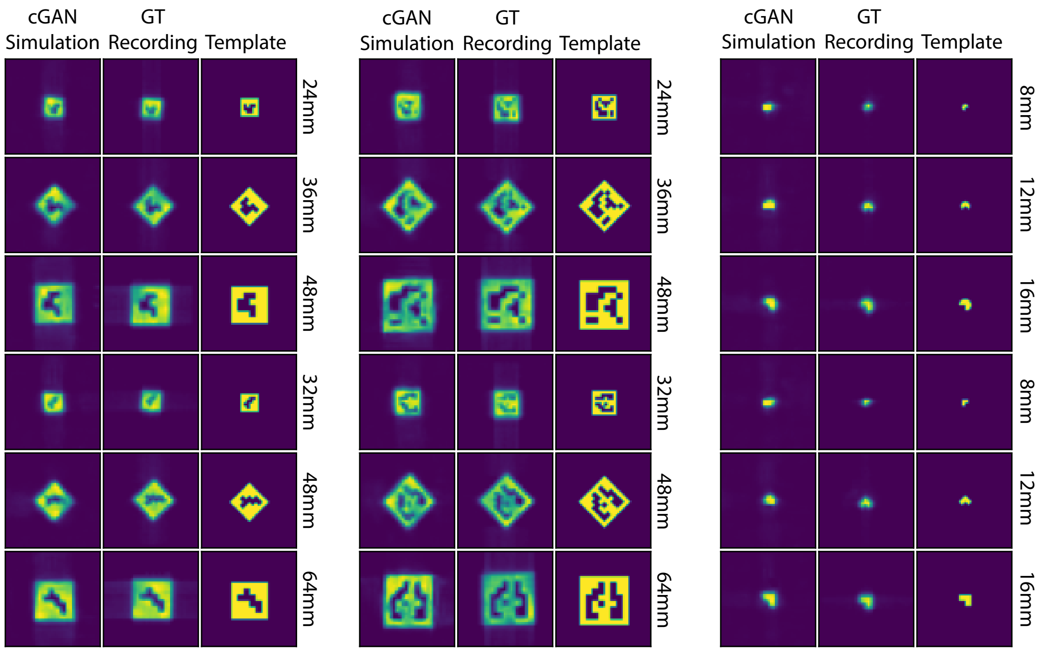
4.5 Recognizer Network Training & Tuning

The various recognizer models presented in the following are solely trained on generated capacitive data gained from our *generator network*. In any marker set M, the markers are only represented with one specific orientation; thus, for training, we rotate the input template x_t round itself (1° steps in $[0, 360)°$). Additionally, to increase the variation, we apply shifts on the input templates x_t as we did during GAN training (x_t is shifted by ± 4 due to the scaling). Finally, to add even more variation to the input for the recognizer, we added Perlin noise [44] to y_g (the output of the generator).

We used the trial-and-error method combined with a smaller grid search for hyperparameters tuning. The model was trained with an Adam optimizer with a learning rate of .001. We used a batch size of 64 to train the model. However, we applied early stopping with a patience of 20; however, maximally trained for 400 epochs. The training time for a single recognizer was between .07 and 7.38 h on an Nvidia Tesla V100; however, the time is highly impacted by the class count ($n = 2 : .07$ h, $n = 10 : 1.17$ h, $n = 30 : 3.42$ h, and $n = 60 : 7.38$ h).

5 EVALUATION

In the following, we present a series of evaluations showcasing our simulation tool's accuracy and the final recognizer quality. Here, for the simulator network, we use the unused dataset DS_R. Moreover, we evaluate our approach on our shape markers DS_{Shapes} and the shapes provided by Schmitz et al. [47] for better external validity. Finally, we use the data recorded by Schmitz et al. [47] on a Nexus 5 to showcase the quality when using a different device for deployment. This is of



(a) AprilTags 16h5 (b) AprilTags 36h11 (c) Shapes

Fig. 6. Each column shows a geometrical template, a recorded capacitive blob, and a simulation result of the respective geometric template: a) $16h5_{13,14}$ in 24, 36, and 48 mm width; b) $36h11_{38,55}$ in 32, 48, and 64 mm width; and c) $Shape_{circle,square}$ in 8, 12, and 16 mm width. These were not used for training the simulator network.

special interest as the dot-pitch of the Nexus 5 is different: 4.1 mm instead of 4.022 mm. The overall results are composited in Table 1. For all results, we provide a Random Forest baseline, see Table 1. However, as in prior work, the baseline does not outperform our models, cf. [29, 47].

5.1 Simulator Network Evaluation

Our simulator model generates 2,048 images per second (0.000488 s per image) on an Nvidia Tesla V100 with a batch size of 512 images. In terms of quality, on our training dataset, the average error is 7.6 out of 255 ($SD = 18.0$). This is a mean discrepancy of 3.0%. For the validation and test sets, the results are similar at 3.0% and 3.1%, respectively ($M = 7.6$, $SD = 18.1$, and $M = 7.0$, $SD = 18.4$). Thus, we conclude that our simulator is not prone to overfitting. For a visual quality comparison, please see Figure 6.

5.2 Marker Size and Form Effect

We used the different subsets of our dataset DS_R to train 6 different recognizers to understand the effect of Type × Size. Thus, we independently trained models for the three 36h11 AprilTags (32, 48, and 64 mm) and the three 16h5 AprilTags (24, 36, and 48 mm). We trained each model with the 10 unseen markers.

The results show that all models achieve accuracies over 93.5% with an average of 98.9%, see Figure 7a. There is no statistically significant difference for Type between our tested AprilTags

Table 1. Overview of evaluated models including the evaluation of shapes and data by Schmitz et al. [47]

			Classification				Rotation		
					Baseline				Baseline
Experiment	**Samples**	**Classes**	**Acc.**	**F1**	**ZeroR**	**RF**	**MAE**	**SD**	**RF**
16h5 – 24mm only	139, 652	10	93.5 %	0.93	10.5 %	9.8 %	9.8	16.0	86.3
16h5 – 36mm only	141, 384	10	100.0 %	1.0	10.6 %	22.3 %	3.4	3.2	83.4
16h5 – 48mm only	145, 460	10	100.0 %	1.0	11.5 %	45.4 %	2.2	1.7	69.7
36h11 – 32mm only	145, 940	10	99.9 %	1.0	10.3 %	12.0 %	3.3	6.4	87.1
36h11 – 48mm only	145, 456	10	100.0 %	1.0	10.5 %	36.7 %	1.3	1.0	57.9
36h11 – 64mm 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 [47]	4, 924	10	14.9 %	0.11	10.3 %	10.5 %	81.0	53.0	83.2
16mm – Galaxy [47]	4, 950	10	33.1 %	0.24	10.6 %	10.0 %	57.0	48.6	83.2
20mm – Galaxy [47]	4, 950	10	85.9 %	0.86	10.6 %	13.9 %	21.8	26.1	83.1
8mm – Nexus 5 [47]	66, 645	10	11.7 %	0.06	11.3 %	12.2 %	88.0	51.9	92.4
12mm – Nexus 5 [47]	64, 795	10	12.8 %	0.09	11.1 %	10.2 %	86.7	51.8	86.5
16mm – Nexus 5 [47]	66, 239	10	25.3 %	0.21	12.0 %	11.7 %	72.9	52.0	84.3
20mm – Nexus 5 [47]	64, 032	10	50.3 %	0.49	11.1 %	13.2 %	64.4	54.4	85.3

36h11 and 16h5 ($t(6) = -.98$, $p = .381$). However, the accuracy is lower than 100% only for the 24 mm models.

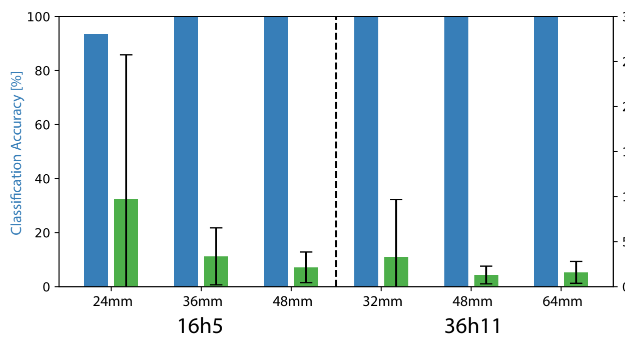
The average rotation error is only 3.8°, varying between 1.3° and 9.8°. Also, here, AprilTags had no significant impact on the rotation ($t(6) = 1.5$, $p = .191$). Again unsurprisingly, the 24 mm models perform slightly worse than the two larger markers.

5.3 AprilTag Recognizer

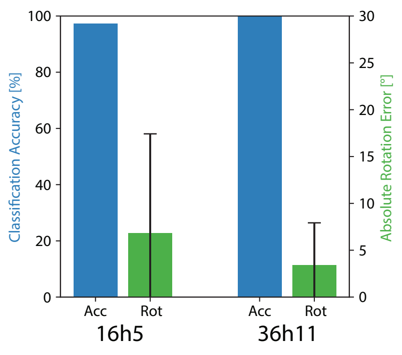
In real life, more than 10 classes are often needed to enable use-cases. Thus, we next trained two models to recognize all AprilTags 36h11 and AprilTags 16h5 of our DS_R dataset. Thus, both models have 30 classes. Overall, the accuracy and rotation error did not drop over only 10 classes. The 30-classes models have an average accuracy of 98.7% and an average rotation error of 5.7°, see Figure 7b.

5.4 Effect of Marker Count on Quality

To generally understand the quality and robustness of our recognizer model architecture, we performed validation on the number of classes captured by the recognizer. Here, we trained models with 2 to 60 different classes based on our DS_R dataset, which comprised 60 different markers. We randomly picked n markers out of the data set and trained a model. To show the robustness of our results, we trained each model with n classes three times with different random picks.



(a) Model results by AprilTag TYPE and SIZE



(b) Model results by AprilTag TYPE

Fig. 7. Model results of classification accuracy and absolute rotation errors concerning the recognizer pipeline: (a) results for the six recognizers trained on markers that are in the same TYPE (16h5 AprilTag vs. 36h11 AprilTag) and have the same SIZE, and (b) results of two recognizers trained on all markers in one TYPE.

The average accuracy is 99.31% (SD = .67). The model accuracy has overall a slight downward trend with respect to the class count; using a linear fit, we show $R^2 = .58$ ($slope = -.020$), see Figure 8. However, this results in a great performance even with 60 classes (98.6%). In terms of rotation error, the rotation error increased with more classes; using a linear fit, we show $R^2 = .89$ ($slope = .056$), see Figure 8. Here, we achieved an average rotation error of 5.51° (SD = 8.3°) with the most classes ($n = 60$) resulting in a rotation error of 6.4° (SD = 9.2°). Results are shown in Figure 8.

5.5 Shape Markers

So far, our architecture has no problems recognizing the structure from the raw capacitive sensor. Thus, to push our model to its limits, we recorded the DS_{Shapes} dataset (see Figure 5a) in line with Schmitz et al. [47]. The shape's bounding box was only 8, 12, and 16 mm. This is 89%, 75%, and 56% smaller than our smallest AprilTag in imprint (16h5 − 24 mm: $24 * 24mm = 576mm^2$).

We trained three recognizers, one for each marker size, with all 10 markers included. The accuracy for the smallest shapes (8 mm) dropped drastically to chance level on our DS_{Shapes} dataset; see Table 1. However, as soon as the marker becomes larger, e.g., 12 mm and 16 mm in WIDTH, the model can again pick up on the details encoded in the fiducial marker, bringing the accuracy up to 14% and 43%, respectively.

Next, we used the shapes provided by Schmitz et al. [47], which were also recorded on a Samsung Galaxy Tab S2 tablet. Again, we trained three recognizers, one for each marker size, with all 10 markers included. Their dataset provides 12, 16, and 20 mm markers that are already 4 mm larger. Especially for the 20 mm markers, we see a drastic improvement, allowing us to recognize now 86% of the markers correctly, see Table 1. Comparing our 12 and 16 mm shapes to the ones of Schmitz et al. [47], we see that ours perform a bit better, but we argue this is due to the shape design.

When comparing our results using simulated data to the domain-specific model by Schmitz et al. [47], we see a reduction in performance. On average, we are 49% worse in accuracy and 35.4° less accurate in detecting the rotation. Unsurprisingly our results are not as good as the domain-specific model. However, the results between training externally recorded data and our data are very much in line. This suggests that the training of the recognizer is not the cause of the drop in quality. Rather our simulator network does not deliver good simulation data for small objects; however, this is also unsurprising as the simulator network has never seen such small markers.

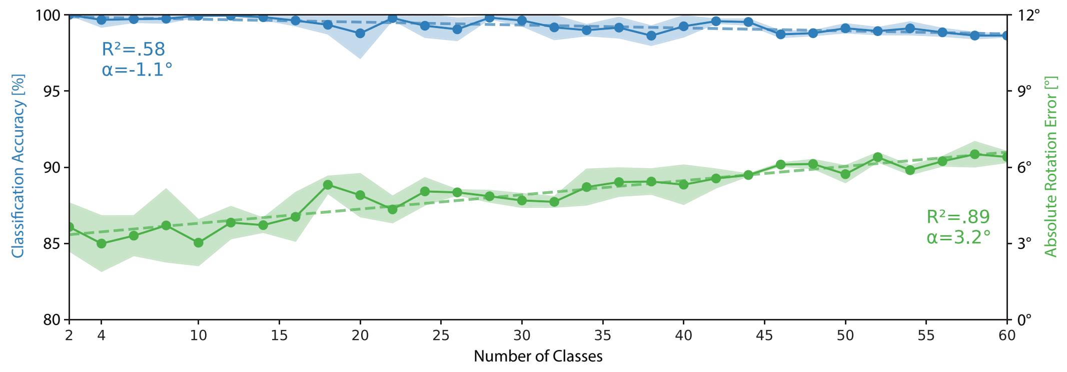


Fig. 8. Test results for increasing the number of randomly sampled classes from DS_R. In blue is the average classification accuracy displayed, and in green is the average absolute rotation errors. We trained each step independently a total of three times. Additionally, the standard deviations are displayed.

5.6 Nexus 5 Data - External Validity

As our approach is tested only with one device and one screen type (4.022 dot-pitch), we next investigate if our simulator network can also simulate data for different dot-pitches with appropriate marker scaling. Therefore, we use the second half of the data provided by Schmitz et al. [47], which contains shape markers recorded on a Nexus 5. Again, our model struggles to recognize smaller shapes; however, we see the same trend as with Samsung Galaxy Tab S2 data, where larger markers can be recognized and rotation estimated, see Table 1.

5.7 Potential Fiducial Marker Set

The largest marker set we tested is the AprilTag 36h11 marker set, containing 587 unique markers. Here, we tested marker sizes of $32 \times 32mm$. The marker set AprilTag 16h5 allows to create smaller markers, such as $24 \times 24mm$, however, the unique marker count is reduced to 30 unique markers. Both allow for error correction of at least 2 bits. However, totally different dot patterns are possible. To understand the boundaries of our approach, we need to study both marker size and data density.

First, we need to determine the largest possible marker that can be recognized. The input to the recognizer is 32×32 pixels and we assume a sensor size of $4 \times 4mm$ – the average size of a capacitive pixel on today's devices. Thus, the real world footprint is $32 \times 32 * 4mm = 128 \times 128mm$; however we also need to account for markers places 45° rotated onto the screen. Thus, $128/\sqrt{2} \approx 90mm$ is the longest edge of a square that can be fed into the recognizer – the largest marker that can be recognized using our implementation is $90 \times 90mm$.

As a next step, we need to define the smallest dot size for the pattern. From our results, we know that $4mm$ is still doable with a high accuracy. When using this as the dot pattern size, we arrive at a dot pattern with $90 \times 90mm/4mm \approx 22 \times 22$. This theoretically means we can use $22 \times 22 = 484bit \approx 60byte$ to store information in the largest possible marker footprint.

For simplicity, we turn now to QR codes as they now fit into the footprint given. The smallest-sized QR code is *Version 1 QR codes* with a size of 21×21. Such a QR code can store 152 bits and has an error correction of 7% (level L); with a higher error correction, this reduces down to 72 bits with 30% error recovery rate (level H). Even with 30% correction that allows for $2^{72} = 4.7 * 10^{21}$ unique markers with at bit/pixel size of $4 \times 4mm$ and a marker size of $90 \times 90mm$. When using specialized dot patterns like AprilTags, even more unique markers are possible with the given marker size as less bits are taken up by the QR coding scheme.

5.8 Model Run Time

We deployed the final recognizer model with 8-bit quantization on a Samsung Galaxy Tab S2 tablet and ran the recognition 2,370 times. First, we evaluated the effect of the quantization, which resulted in no performance loss for the test 16h5 models with 30 classes (97.3% and rotation error 6.83, SD = 10.6). On average, the prediction time was 42.15 ms (SD = 6.52ms). Additionally, the pre-processing (blob extraction and normalization) takes on average 15.45 ms (SD = 3.62ms). As we pull the capacitive image every $\sim$ 110ms, we can process the marker in real time on the device.

6 TOOLKIT & APPLICATIONS

With this work, we aim to enable everyone to build their own conductive fiducial recognizer without data collection and ML knowledge. Therefore, we provide a toolkit allowing us to generate a use-case-specific model based on fiducial templates. Moreover, we present a set of possible example applications.

6.1 Toolkit

The toolkit allows everyone to train their own use-case-specific deep learning model. As such, users of the toolkit only need to provide 2D sketches of the imprint of their fiducial markers and then the toolkit can automatically deliver a plug-and-play recognizer to be deployed on phones and tablets. This will give designers the opportunity to design games such as chess, which has 32 pieces, games with many more pieces like memory but also museum applications to display many different objects. As such, the toolkit provides a pre-trained simulator model (the simulator model described above, see Figure 3a), allowing everyone to simulate capacitive data used to train the recognizer model. The simulated capacitive data is then used to train our recognizer model based on our general recognizer architecture (the recognizer architecture described above, see Figure 3c). As we did not perform additional hyperparameter tuning when training the numerous models in our evaluation, we argue that the recognizer model is ready to be trained for various fiducial markers. Thus, the user only needs to provide the toolkit with the templates of the fiducial markers as shown in Figure 6 and the toolkit will then automatically train an appropriate recognizer model.

The toolkit requires python 3.5+ and uses TensorFlow 2.5+ as ML back-end, and provides the user with a single tf-lite file[1], which is ready to be deployed on a large variety of operating systems such as Android, and iOS but also Embedded Linux (e.g., Raspberry Pi) and even microcontrollers.

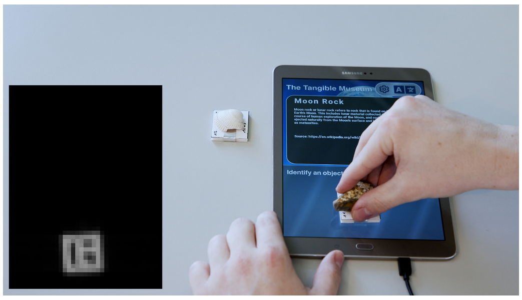
Our toolkit will be released as an open-source toolkit. We additionally provide the scripts to train the models used in the toolkit. Finally, we released the collected capacitive image data to allow others to build upon our results and offer a benchmark for future evaluations. Our open-source contribution is available for download via https://github.com/mimuc/Conductive-Fiducial-Marker-Simulation-Toolkit.

6.2 Applications

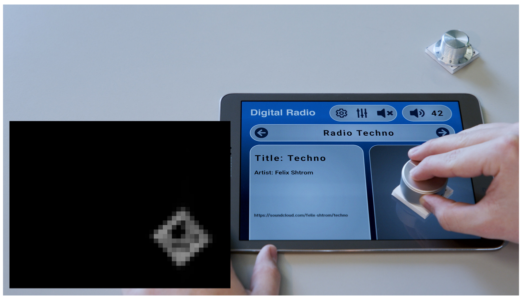
Based on our toolkit pipeline, we build four example use-cases in order to explore the wide range of possibilities to bring tangibility back to today's touchscreens. We built four showcase applications and manufactured fiducial tags attached to our use-case-specific objects. While we manufactured them using conductive foil, they can easily be 3D-printed using conductive materials, e.g., [11, 12, 24]. This allows for a large number of objects as we find in applications such as the museum app and chess (32 board pieces).

Discovery & Learning Tangibles in museums are generally nothing new, e.g., Ma et al. [35] and tangibls have long been used for learning, cf. Li et al. [33]. However, allowing users to discover

[1]TensorFlow Lite https://www.tensorflow.org/lite



(a) Tangible Museum



(b) Tangible Radio

Fig. 9. Two applications with the respective capacitive input. a) Tangible museums use-case where each artifact has a code on the bottom for recognition. b) The radio is manipulated using tangible controllers.

objects on their own devices is not possible yet. Thus, to overcome this limitation, our setup can enable museums and teachers to build tangibles. Here, accompanying apps can use by everyone to explore the objects themselves. Our museum demonstrator application allows placing objects on the screen to get additional information, see Figure 9a.

Radio Inspired by earlier work music creation [22], we built a tangible-enabled radio, see Figure 9b. In detail, one tangible enables the user to scroll through the different radio stations in line with old-school FM or AM radios. Additionally, a second tangible now allows the user to manipulate the volume as typically present on car radios.

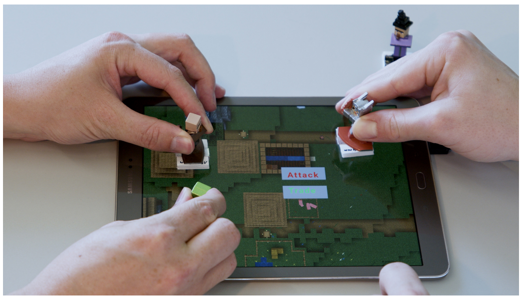
Dungeons & Dragons Game Our toolkit allows game developers to bring tangibles back into everyone's home. While a game can consist of traditional figures such as in chess and other games, the board can now be interactive itself as it allows tracking game figures on personal devices. This brings back tangibility to digital board games and allows for collaboration, fostering engagement and fun [45, 72]. Here, we present a multiplayer Minecraft-inspired Dungeons & Dragons game, see Figure 10a.

Smart Home While we see screens in smart home environments, the directness of the haptic touch of a switch is still dominant. With our tangible light switches on capacities screen, we combine the best parts of both worlds: a) the flexibility of the touchscreen and b) the tangibility of physical knobs, see Voelker et al. [60]. Therefore, we present a smart home interface that is controlled by different physical sliders, see Figure 10b.

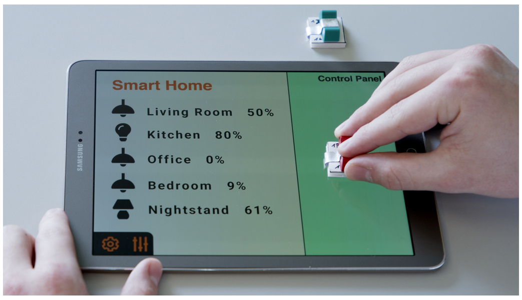
7 DISCUSSION

Over the last few years, we have seen great development in the domain of capacitive sensing. However, today we still do not see widespread use. We argue that one major problem is the missing ease of use of the latest development, e.g., recognition using deep learning. Our toolkit allows designers to design arbitrary markers that fit their needs while avoiding time-consuming data collection studies and simplifying the realization of TUIs on capacitive devices. We opted to use a cGAN, which learns to transfer the style of a capacitive sensor to sketches of conductive markers. With this, our tool eliminates the need for data collection and hyperparameter tuning, both elements that are hard for designers. Thus, we improve the ease of use of capacitive fiducial tangibles.

AprilTag Fiducials. 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



(a) Tangible Game



(b) Smart Home Controller

Fig. 10. a) Minecraft-inspired dungeons and dragons game using multiple fiducial recognized tangibles at the same time, which is even playable with two players. b) Smart home control panel where lights can be controlled directly with their respective tangible counterpart.

our simulator with an average error of 4.9° (SD = 1.4°). While our results are overall promising, we achieve the lowest classification accuracy and our highest rotation error with 93.5% and 9.8° (SD = 16°) for our smallest 24 mm AprilTags. One impact factor could be the combination of the low-resolution capacitive matrix with a dot-pitch of 4.022 mm, which might also result in slight deviations from actual to recorded rotation. This can result in a rounding of the corners of our simulations and blurring the inner structures of the simulation (see Figure 6a *GT Recording* and *cGAN Simulation*). When comparing our model to prior work, Mayer et al. [40] used the raw capacitive image to allow for tangible fiducial tracking. Mayer et al. [40] achieved between 66.2% and 99.0% on average on differently sized AprilTags. Thus, our general marker generation approach outperforms a specialized approach.

Results Small Marker. In detail, the results by Mayer et al. [40] already showed a performance loss for markers with high information density. Therefore, we purposefully included markers with a high information density to analyze if this problem persists with our approach, see Figure 4. For our smallest AprilTags (Type 16h5 Size 24 mm) a clear representation of the internal structure is necessary for a reliable classification as they encode 16 bits within 256 mm^2 or $\sim$ 16 capacitive pixels. This also explains why our accuracy for all other AprilTags is 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 32 mm, 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. This further demonstrates that a deep learning generator approach outperforms the super-resolution approach by Mayer et al. [40]. However, their argument is that higher visual accuracy will result in a higher recognition rate with any recognizer. Thus, using super-resolution approaches additionally in our network will likely also boost our performance in a future version.

Results Shapes. 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 accuracy is < 43%. Because we exclusively conditioned our simulator on imprints of AprilTags, the model lacks the fine-grained capabilities needed for the conductive shapes. This becomes even more clear when looking at the respective imprints of the shapes. Here, the Widths of our shapes were 8, 12, and 16 m and their imprints corresponded, respectively, to only 11%, 25%, and 44% of that of our smallest AprilTag. Comparing our results to Schmitz et al. [47], which achieved between 88.86% and 99.51% on 3 models supporting 10 classes our model performs worse. Our general marker generation approach does not sufficiently support small shape markers. To improve the simulator to capture such fine-grained details, we propose training the simulator on a wider variety of data. As we recorded such data, this can easily be done in the next step.

Size of the Capacitive Matrix. To gather performance metrics on capacitive data of other devices, we used the shape data by Schmitz et al. [47]. Large markers of Width 20 mm outperformed small 8 mm shape markers with an accuracy of 50.3% compared to 11.7%. Overall, our classification and rotation prediction results are not nearly as good as for shape data. However, it is hard to define whether the poor performance was due to incorrect simulations or the quality of the dataset as the capacitive matrix of the Nexus 5 tends to produce noisy images. Note that these shapes are small and, thus, suffer from the same problem as discussed above. Thus, as we observe the same trend for higher accuracy with increasing Width, we argue that our simulator can generate capacitive imprints for other devices.

Marker Fabrication and Quality. The fabrication of our conductive markers might have led to inconsistencies between capacitive recordings and simulations. Because we produced our markers by hand, we could not avoid slight deviations from the actual shape. However, with sub-millimeter errors and the ability to utilize conductive material, 3D printers have shown to be effective for printing tangibles [37, 47]. On the other hand, together with our pipeline, this opens up new prototyping capabilities as now not only can tangibles be modeled within minutes, but also the model can be trained without time-consuming data collection.

Limitations. While all devices internally have the capacitive matrix stored to extract the position of the fingers, the capacitive matrix is typically not exposed to the OS layer. For this reason, we applied kernel modification to our test device, as others, e.g., Schmitz et al. [47]. Moreover, the Nexus 5 and the Samsung Galaxy Tab S2 kernels are publicly available. Thus, while it is possible to retrieve the data, today's designers would need to additionally apply this modification; while in the future, manufacturers can expose this easily, today, extra effort is required.

8 CONCLUSION

To bring back effortless conductive fiducial tangibles to capacitive touchscreens, we presented a toolkit for training a recognizer network solely on simulated data, thus allowing everyone to train their own recognizer. In detail, we contribute a simulator cGAN network that simulates the response of a capacitive touch sensor based on sketches of fiducial markers. We pre-trained the simulator network on the capacitive data of 30 fiducial markers. 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. Using our toolkit, we found that we can classify conductive AprilTag fiducials with an average accuracy of over 98%. Moreover, we can predict the correct orientation of AprilTag fiducials with an average error of 4.8°. Finally, our toolkit does not require domain knowledge or a data collection study. All crucial elements for designers to develop robust interactions with tangibles on today's touch devices are included. Thus, our tool will help designers and practitioners to develop new fiducial markers and bring back tangible interaction to today's capacitive devices.

Currently, we see a performance loss when simulating fine-grained details of markers. Thus, as a next step, we want to investigate approaches to simulate also fine-grained details by training the simulator with more small tangibles. Moreover, we see great potential in using super-resolution techniques to improve the recognition performance; however, the downside will be a recognition delay, cf. Mayer et al. [40]. Moreover, to support the designer further, a heuristic model could already show the designer during design time how well the model will likely recognize the real-world counterparts of the sketches.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 214–223. arXiv:1701.07875 [stat.ML]

[2] Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. 2020. This dataset does not exist: training models from generated images. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, New York, NY, USA, 1–5. https://doi.org/10.1109/ICASSP40776.2020.9053146 arXiv:1911.02888 [cs.CV]

[3] Liwei Chan, Stefanie Müller, Anne Roudaut, and Patrick Baudisch. 2012. 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* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2189–2192. https://doi.org/10.1145/2207676.2208371

[4] Frederick Choi, Sven Mayer, and Chris Harrison. 2021. 3D Hand Pose Estimation on Conventional Capacitive Touchscreens. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*. Association for Computing Machinery, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/3447526.3472045

[5] Peter Dalsgaard and Kim Halskov. 2012. 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* (Copenhagen, Denmark) *(NordiCHI '12)*. Association for Computing Machinery, New York, NY, USA, 109–118. https://doi.org/10.1145/2399016.2399033

[6] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. 1995. Bricks: Laying the Foundations for Graspable User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 442–449. https://doi.org/10.1145/223904.223964

[7] Tom Goldstein and Stanley Osher. 2009. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences 2* 2 (2009), 323–343. https://doi.org/10.1137/080725891

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv* (2014). arXiv:1406.2661 [stat.ML]

[9] Timo Götzelmann and Daniel Schneider. 2016. CapCodes: Capacitive 3D Printable Identification and On-Screen Tracking for Tangible Interaction. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction* (Gothenburg, Sweden) *(NordiCHI '16)*. Association for Computing Machinery, New York, NY, USA, Article 32, 4 pages. https://doi.org/10.1145/2971485.2971518

[10] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. 2017. Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3293–3315. https://doi.org/10.1145/3025453.3025808

[11] Sebastian Günther, Florian Müller, Martin Schmitz, Jan Riemann, Niloofar Dezfuli, Markus Funk, Dominik Schön, and Max Mühlhäuser. 2018. CheckMate: Exploring a Tangible Augmented Reality Interface for Remote Interaction. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI EA '18)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3170427.3188647

[12] Sebastian Günther, Martin Schmitz, Florian Müller, Jan Riemann, and Max Mühlhäuser. 2017. BYO*: Utilizing 3D Printed Tangible Tools for Interaction on Interactive Surfaces. In *Proceedings of the 2017 ACM Workshop on Interacting with Smart Objects* (Limassol, Cyprus) *(SmartObject '17)*. Association for Computing Machinery, New York, NY, USA, 21–26. https://doi.org/10.1145/3038450.3038456

[13] Anhong Guo, Robert Xiao, and Chris Harrison. 2015. CapAuth: Identifying and Differentiating User Handprints on Commodity Capacitive Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (Madeira, Portugal) *(ITS '15)*. Association for Computing Machinery, New York, NY, USA, 59–62. https://doi.org/10.1145/2817721.2817722

[14] Changyo Han, Ryo Takahashi, Yuchi Yahagi, and Takeshi Naemura. 2020. PneuModule: Using Inflatable Pin Arrays for Reconfigurable Physical Controls on Pressure-Sensitive Touch Surfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376838

[15] Eiji Hayashi, Jaime Lien, Nicholas Gillian, Leonardo Giusti, Dave Weber, Jin Yamanaka, Lauren Bedal, and Ivan Poupyrev. 2021. RadarNet: Efficient Gesture Recognition Technique Utilizing a Miniature Radar Sensor. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 5, 14 pages. https://doi.org/10.1145/3411764.3445367

[16] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504–507. https://doi.org/10.1126/science.1127647

[17] Christian Holz, Senaka Buthpitiya, and Marius Knaust. 2015. Bodyprint: Biometric User Identification on Mobile Devices Using the Capacitive Touchscreen to Scan Body Parts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3011–3014. https://doi.org/10.1145/2702123.2702518

[18] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. 2019. Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV '19)*. IEEE, New York, NY, USA, 1314–1324. https://doi.org/10.1109/ICCV.2019.00140

[19] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*. PMLR, Lille, France, 448–456. arXiv:1502.03167 [cs.LG]

[20] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '97)*. Association for Computing Machinery, New York, NY, USA, 234–241. https://doi.org/10.1145/258549.258715

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17)*. IEEE, New York, NY, USA, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

[22] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. 2007. 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* (Baton Rouge, Louisiana) *(TEI '07)*. Association for Computing Machinery, New York, NY, USA, 139–146. https://doi.org/10.1145/1226969.1226998

[23] Martin Kaltenbrunner and Ross Bencina. 2007. ReacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (Baton Rouge, Louisiana) *(TEI '07)*. Association for Computing Machinery, New York, NY, USA, 69–74. https://doi.org/10.1145/1226969.1226983

[24] Kunihiro Kato, Kaori Ikematsu, and Yoshihiro Kawahara. 2020. CAPath: 3D-Printed Interfaces with Conductive Points in Grid Layout to Extend Capacitive Touch Inputs. *Proc. ACM Hum.-Comput. Interact.* 4, ISS, Article 193 (nov 2020), 17 pages. https://doi.org/10.1145/3427321

[25] Sven Kratz, Tilo Westermann, Michael Rohs, and Georg Essl. 2011. CapWidgets: Tangile Widgets versus Multi-Touch Controls on Mobile Devices. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI EA '11)*. Association for Computing Machinery, New York, NY, USA, 1351–1356. https://doi.org/10.1145/1979742.1979773

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. https://doi.org/10.1145/3065386

[27] Abinaya Kumar, Aishwarya Radjesh, Sven Mayer, and Huy Viet Le. 2019. Improving the Input Accuracy of Touchscreens Using Deep Learning. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312928

[28] Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, and Niels Henze. 2018. 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 (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173934

[29] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 779–792. https://doi.org/10.1145/3242587.3242605

[30] Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the Feasibility of Finger Identification on Capacitive Touchscreens Using Deep Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 637–649. https://doi.org/10.1145/3301275.3302295

[31] Huy Viet Le, Sven Mayer, Maximilian Weiß, Jonas Vogelsang, Henrike Weingärtner, and Niels Henze. 2020. Shortcut Gestures for Mobile Text Editing on Fully Touch Sensitive Smartphones. *ACM Trans. Comput.-Hum. Interact.* 27, 5, Article 33 (Aug. 2020), 38 pages. https://doi.org/10.1145/3396233

[32] Chuan Li and Michael Wand. 2016. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. In *Computer Vision (ECCV '16)*. Springer International Publishing, Cham, 702–716. https://doi.org/10.1007/978-3-319-46487-9_43 arXiv:1604.04382 [cs.CV]

[33] Yanhong Li, Meng Liang, Julian Preissing, Nadine Bachl, Michelle Melina Dutoit, Thomas Weber, Sven Mayer, and Heinrich Hussmann. 2022. A Meta-Analysis of Tangible Learning Studies from the TEI Conference. In *Sixteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Daejeon, Republic of Korea) *(TEI '22)*. Association for Computing Machinery, New York, NY, USA, Article 7, 17 pages. https://doi.org/10.1145/3490149.3501313

[34] Rong-Hao Liang, Han-Chih Kuo, Liwei Chan, De-Nian Yang, and Bing-Yu Chen. 2014. 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* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 365–372. https://doi.org/10.1145/2642918.2647384

[35] Joyce Ma, Lisa Sindorf, Isaac Liao, and Jennifer Frazier. 2015. Using a Tangible Versus a Multi-Touch Graphical User Interface to Support Data Exploration at a Museum Exhibit. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction* (Stanford, California, USA) *(TEI '15)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2677199.2680555

[36] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning (ICML '13, Vol. 18)*. MLResearchPress, 3.

[37] Karola Marky, Martin Schmitz, Verena Zimmermann, Martin Herbers, Kai Kunze, and Max Mühlhäuser. 2020. 3D-Auth: Two-Factor Authentication with Personalized 3D-Printed Items. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376189

[38] Michael Mathieu, Camille Couprie, and Yann LeCun. 2016. Deep multi-scale video prediction beyond mean square error. (2016). arXiv:1511.05440 [cs.LG]

[39] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 220–229. https://doi.org/10.1145/3132272.3134130

[40] Sven Mayer, Xiangyu Xu, and Chris Harrison. 2021. Super-Resolution Capacitive Touchscreens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 10 pages. https://doi.org/10.1145/3411764.3445703

[41] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014), 8. arXiv:1411.1784 [cs.LG]

[42] Roderick Murray-Smith, John H. Williamson, Andrew Ramsay, Francesco Tonolini, Simon Rogers, and Antoine Loriette. 2021. Forward and Inverse models in HCI: Physical simulation and deep learning for inferring 3D finger pose. (2021). arXiv:2109.03366 [cs.HC]

[43] Alex Olwal and Andrew D. Wilson. 2008. SurfaceFusion: Unobtrusive Tracking of Everyday Objects in Tangible User Interfaces. In *Proceedings of Graphics Interface 2008* (Windsor, Ontario, Canada) *(GI '08)*. Canadian Information Processing Society, CAN, 235–242.

[44] Ken Perlin. 2002. Improving Noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (San Antonio, Texas) *(SIGGRAPH '02)*. Association for Computing Machinery, New York, NY, USA, 681–682. https://doi.org/10.1145/566570.566636

[45] Eckard Riedenklau, Thomas Hermann, and Helge Ritter. 2012. An Integrated Multi-Modal Actuated Tangible User Interface for Distributed Collaborative Planning. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (Kingston, Ontario, Canada) *(TEI '12)*. Association for Computing Machinery, New York, NY, USA, 169–174. https://doi.org/10.1145/2148131.2148167

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, Cham, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

[47] Martin Schmitz, Florian Müller, Max Mühlhäuser, Jan Riemann, and Huy Viet Viet Le. 2021. 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 (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 419, 12 pages. https://doi.org/10.1145/3411764.3445502

[48] Martin Schmitz, Jürgen Steimle, Jochen Huber, Niloofar Dezfuli, and Max Mühlhäuser. 2017. Flexibles: Deformation-Aware 3D-Printed Tangibles for Capacitive Touchscreens. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1001–1014. https://doi.org/10.1145/3025453.3025663

[49] Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, and Andreas Bulling. 2019. KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens Using Deep Learning. In *Proceedings of Mensch Und Computer 2019* (Hamburg, Germany) *(MuC'19)*. Association for Computing Machinery, New York, NY, USA, 387–397. https://doi.org/10.1145/3340764.3340767

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[51] Paul Streli and Christian Holz. 2021. CapContact: Super-Resolution Contact Areas from Capacitive Touchscreens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 289, 14 pages. https://doi.org/10.1145/3411764.3445621

[52] Satoshi Suzuki and Keiicji Abe. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30, 1 (1985), 32–46. https://doi.org/10.1016/0734-189X(85)90016-7

[53] Francesco Tonolini, Jack Radford, Alex Turpin, Daniele Faccio, and Roderick Murray-Smith. 2020. Variational Inference for Computational Imaging Inverse Problems. *Journal of Machine Learning Research* 21, 179 (2020), 1–46. arXiv:1904.06264 [cs.LG] http://jmlr.org/papers/v21/20-151.html

[54] Brygg Ullmer and Hiroshi Ishii. 1997. The MetaDESK: Models and Prototypes for Tangible User Interfaces. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology* (Banff, Alberta, Canada) *(UIST '97)*. Association for Computing Machinery, New York, NY, USA, 223–232. https://doi.org/10.1145/263407.263551

[55] Brygg Ullmer and Hiroshi Ishii. 2000. Emerging Frameworks for Tangible User Interfaces. *IBM Systems Journal* 39, 3.4 (2000), 915–931. https://doi.org/10.1147/sj.393.0915

[56] John Underkoffler and Hiroshi Ishii. 1999. Urp: A Luminous-Tangible Workbench for Urban Planning and Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99)*. Association for Computing Machinery, New York, NY, USA, 386–393. https://doi.org/10.1145/302979.303114

[57] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project Zanzibar: A Portable and Flexible Tangible Interaction Platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174089

[58] Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergård, and Jan Borchers. 2015. PERCs: Persistently Trackable Tangibles on Capacitive Multi-Touch Displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software amp; Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 351–356. https://doi.org/10.1145/2807442.2807466

[59] Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergård, and Jan Borchers. 2013. 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* (St. Andrews, Scotland, United Kingdom) *(ITS '13)*. Association for Computing Machinery, New York, NY, USA, 101–104. https://doi.org/10.1145/2512349.2512791

[60] Simon Voelker, Kjell Ivar Øvergård, Chat Wacharamanotham, and Jan Borchers. 2015. Knobology Revisited: A Comparison of User Performance between Tangible and Virtual Rotary Knobs. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (Madeira, Portugal) *(ITS '15)*. Association for Computing Machinery, New York, NY, USA, 35–38. https://doi.org/10.1145/2817721.2817725

[61] Geoff Walker. 2012. A review of technologies for sensing contact location on the surface of a display. *Journal of the Society for Information Display* 20, 8 (2012), 413–440. https://doi.org/10.1002/jsid.100

[62] John Wang and Edwin Olson. 2016. AprilTag 2: Efficient and robust fiducial detection. In *RSJ International Conference on Intelligent Robots and Systems (IROS '16)*. IEEE, New York, NY, USA, 4193–4198. https://doi.org/10.1109/IROS.2016.7759617

[63] Xiaolong Wang and Abhinav Gupta. 2016. Generative Image Modeling Using Style and Structure Adversarial Networks. In *Computer Vision (ECCV 2016)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 318–335. https://doi.org/10.1007/978-3-319-46493-0_20

[64] Lyndon White. 2018. Encoding Angle Data for Neural Network. Cross Validated. https://stats.stackexchange.com/q/218547 (version: 2018-07-03).

[65] Alexander Wiethoff, Hanna Schneider, Michael Rohs, Andreas Butz, and Saul Greenberg. 2012. Sketch-a-TUI: Low Cost Prototyping of Tangible Interactions Using Cardboard and Conductive Ink. In *Proceedings of the Sixth International*

Conference on Tangible, Embedded and Embodied Interaction (Kingston, Ontario, Canada) *(TEI '12)*. Association for Computing Machinery, New York, NY, USA, 309–312. https://doi.org/10.1145/2148131.2148196

[66] Andrew D. Wilson and Raman Sarin. 2007. BlueTable: Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-Based Handshaking. In *Proceedings of Graphics Interface 2007* (Montreal, Canada) *(GI '07)*. Association for Computing Machinery, New York, NY, USA, 119–125. https://doi.org/10.1145/1268517.1268539

[67] Katrin Wolf, Stefan Schneegass, Niels Henze, Dominik Weber, Valentin Schwind, Pascal Knierim, Sven Mayer, Tilman Dingler, Yomna Abdelrahman, Thomas Kubitza, Markus Funk, Anja Mebus, and Albrecht Schmidt. 2015. TUIs in the Large: Using Paper Tangibles with Mobile Devices. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 1579–1584. https://doi.org/10.1145/2702613.2732863

[68] Robert Xiao, Scott Hudson, and Chris Harrison. 2016. CapCam: Enabling Rapid, Ad-Hoc, Position-Tracked Interactions Between Devices. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces* (Niagara Falls, Ontario, Canada) *(ISS '16)*. Association for Computing Machinery, New York, NY, USA, 169–178. https://doi.org/10.1145/2992154.2992182

[69] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (Madeira, Portugal) *(ITS '15)*. Association for Computing Machinery, New York, NY, USA, 47–50. https://doi.org/10.1145/2817721.2817737

[70] 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, and Yi-Ping Hung. 2011. TUIC: Enabling Tangible Interaction on Capacitive Multi-Touch Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2995–3004. https://doi.org/10.1145/1978942.1979386

[71] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. 2021. DatasetGAN: Efficient Labeled Data Factory With Minimal Human Effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New York, NY, USA, 10145–10155. https://doi.org/10.1109/CVPR46437.2021.01001 arXiv:2104.06490 [cs.CV]

[72] Oren Zuckerman, Ofir Sadka, Ron Gissin, and Hadas Erel. 2021. TUI as Social Entity: A Study of Joint-Actuation and Turn-Taking-Actuation in Actuated-Interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 507, 12 pages. https://doi.org/10.1145/3411764.3445468

Received February 2022; revised May 2022; accepted June 2022