DeepFormableTag: End-to-end Generation and Recognition of Deformable Fiducial Markers

MUSTAFA B. YALDIZ, ANDREAS MEULEMAN, HYEONJOONG JANG, HYUNHO HA, and MIN H. KIM, KAIST, South Korea

(a) Detection of our multiple markers on deformed surfaces   (b) Structured light   (c) Motion tracking   (d) Tracking with motion blur
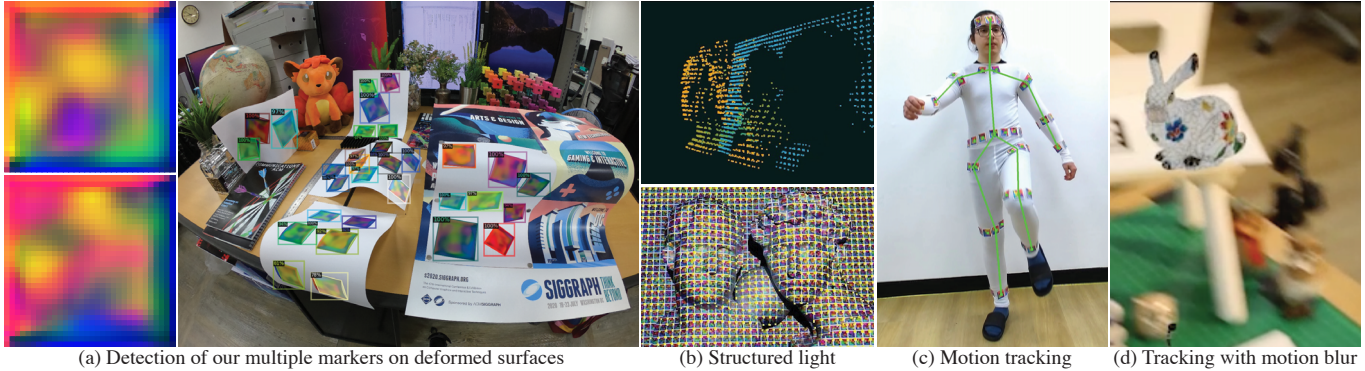
Fig. 1. (a) shows our learned deformable markers at the leftmost column and presents detection of multiple deformable markers on curved surfaces. All the deformed markers are detected successfully. (b) presents the captured results of 3D scanning (top) using a 2D array of multiple fiducial markers as structured light patterns (bottom). (c) shows motion tracking of human body motion using our deformable markers. (d) demonstrates rendering of a virtual object on the detected markers with strong motion blur. Refer to the supplemental demo video for more results.

Fiducial markers have been broadly used to identify objects or embed messages that can be detected by a camera. Primarily, existing detection methods assume that markers are printed on ideally planar surfaces. The size of a message or identification code is limited by the spatial resolution of binary patterns in a marker. Markers often fail to be recognized due to various imaging artifacts of optical/perspective distortion and motion blur. To overcome these limitations, we propose a novel deformable fiducial marker system that consists of three main parts: First, a fiducial marker generator creates a set of free-form color patterns to encode significantly large-scale information in unique visual codes. Second, a differentiable image simulator creates a training dataset of photorealistic scene images with the deformed markers, being rendered during optimization in a differentiable manner. The rendered images include realistic shading with specular reflection, optical distortion, defocus and motion blur, color alteration, imaging noise, and shape deformation of markers. Lastly, a trained marker detector seeks the regions of interest and recognizes multiple marker patterns simultaneously via inverse deformation transformation. The deformable marker creator and detector networks are jointly optimized via the differentiable photorealistic renderer in an end-to-end manner, allowing us to robustly recognize a wide range of deformable markers with high accuracy. Our deformable marker system is capable of decoding 36-bit messages successfully at ~29 fps with severe shape deformation. Results validate that our system significantly outperforms the traditional and data-driven marker methods. Our learning-based

marker system opens up new interesting applications of fiducial markers, including cost-effective motion capture of the human body, active 3D scanning using our fiducial markers' array as structured light patterns, and robust augmented reality rendering of virtual objects on dynamic surfaces.

CCS Concepts: • **Computing methodologies** → **Object detection**; **Scene understanding**; **Tracking**.

Additional Key Words and Phrases: Fiducial marker system, object detection, tracking, deep learning

1 INTRODUCTION

Fiducial markers with binary patterns, such as QR code [Denso Wave 1994], ARTag [Fiala 2005], ARToolKit [Kato and Billinghurst 1999], AprilTag [Olson 2011], ArUco [Munoz-Salinas 2012], and ChArUco [OpenCV 2020], have been used broadly to encode identification codes or messages within a square pattern, enabling object recognition, camera tracking, embedding messages in various computer graphics, vision, and robotics applications.

Despite the usefulness of these fiducial systems, they still show several limitations. First, all the existing markers assume to be printed on *ideally planar and rigid surfaces*, and thus existing marker systems are incapable of detecting severely deformed marker images. Second, marker recognition often fails due to various imaging artifacts of optical/perspective distortion and motion blur when markers are captured in real-world environments. Lastly, the size of a message that can be embedded in markers is limited by the spatial resolution of a binary or color pattern.

Authors' address: Mustafa B. Yaldiz; Andreas Meuleman; Hyeonjoong Jang; Hyunho Ha; Min H. Kim, KAIST, School of Computing, Daejeon, 34141, South Korea, minhkim@kaist.ac.kr.

Even though various hand-crafted visual features have been proposed alternatively in the previous studies [DeGol et al. 2017; Munoz-Salinas 2012; Olson 2011; OpenCV 2020; Wang and Olson 2016] to enhance the capability of encoding large-scale information and the performance of detection, *deformation* of fiducial markers has been rarely discussed due to the hardness of challenges. Additionally, several assumptions have been made in imaging conditions as well, e.g., there is no optical distortion and motion blur. Unfortunately, when capturing physically printed markers with a camera in the real-world, these assumptions do not always hold, often resulting in unreliable localization, detection, and decoding of marker messages.

Recently, learning-based approaches have been proposed to improve the performance of fiducial marker systems through a set of learnable features [Grinchuk et al. 2016; Peace et al. 2020] or providing a learning-based enhancement of an existing marker detection method [Hu et al. 2019]. However, the former method also trains the network in an ideal setup where fiducial markers are located at the center region of training images or perfectly sit on ideal planar surfaces, missing the important localization capability. The latter method detects and refines fiducial corners and the camera pose, relying on a traditional marker system, ChArUco [OpenCV 2020] that assumes ideal imaging scenarios. To the best of our knowledge, there is no robust and practical solution to tackle the deformed marker detection problem without sacrificing messaging capability and detection performance.

In this work, we propose a novel deformable fiducial marker system specially designed for generating and detecting deformable markers in an end-to-end manner. Our learning-based method consists of three main parts: First, a fiducial marker generation network creates learnable fiducial marker patterns to enhance the large-scale message-embedding capability, which can be attached to real-world free-form surfaces. Second, a novel differentiable rendering framework creates a realistic training dataset during optimization, where a set of fiducial markers are simulated with surface deformation and realistic imaging conditions, including specular reflection, perspective/lens distortion, defocus and motion blur, color alteration, and imaging noise simulated in a differentiable manner. Lastly, a marker detection network consists of a localizer network and a novel marker decoder network that samples markers' features with respect to deformation. Then, it finally decodes embedded messages within a frame. Our marker generator and detector are jointly trained by means of the automatically generated *photorealistic* training dataset so that it can detect multiple markers' codes successfully even on severely deformed surfaces under various environments. Our differentiable rendering framework that creates immersive markers for real scenes achieves high-quality realism to minimize potential domain gap in the training dataset, outperforming existing learning-based approaches that merely superimpose markers on photographs.

Our end-to-end marker system is capable of creating a very large number of messages as unique fiducial markers, for instance, theoretically feasible up to ~68.7 billion (=2^{36}) of 36-bit binary messages with a resolution of 32-by-32 pixels, which can be recognized very robustly and successfully with high bit accuracy at ~29 fps. Results validate that our fiducial marker system outperforms the traditional and learning-based fiducial marker systems, with robust performance under severely ill-conditioned imaging setup. Our

deformable fiducial marker system opens three interesting applications of fiducial markers. We demonstrate (1) cost-effective motion capture of the human body with deformable markers, (2) single-shot 3D scanning with deformable markers' array used as structured light patterns, and (3) robust augmented reality rendering of virtual objects on dynamic surfaces. Figure 1 demonstrates our learnable marker examples and applications using our markers. All code, dataset, and demo are published online to ensure reproducibility (link: https://github.com/KAIST-VCLAB/DeepFormableTag).

2 RELATED WORK

Traditional markers. Conventional fiducial marker methods [Fiala 2005; Kato and Billinghurst 1999; Munoz-Salinas 2012; Olson 2011; OpenCV 2020] consist of two main stages: marker creation and detection. In the marker generation stage, unique binary identification codes are generated with consideration of detection errors, and maximum inter-marker code distance. Binary codes in a square shape are used to create payload inside a black border of markers. The square shape has been preferred to a circular shape [Bergamasco et al. 2011; Naimark and Foxlin 2002; Xu and Dudek 2011] or irregular shape [Bencina et al. 2005]. This is because four known point correspondences, such as corners, help estimating the marker poses. Also, color markers with red and green blocks have been proposed for more efficient detection [DeGol et al. 2017]. In the detection stage, a decoder algorithm detects and recognizes the fiducial markers in an image. The general approach for the detection algorithms performs quadrilateral detection mainly through an edge detection technique to identify boundaries of the marker. After the quad-detection process, each marker is warped through an estimated homography, and the payload is decoded with thresholding.

These traditional methods have been developed further with goals to reduce false negatives, to detect smaller and denser tags, to achieve more robustness against occlusions [Garrido-Jurado et al. 2014; Kallwies et al. 2020; Krogius et al. 2019; Romero-Ramirez et al. 2018; Wang and Olson 2016]. However, several limitations still remain. First, the amount of encoded information is limited by the spatial resolution of the binary patterns [Garrido-Jurado et al. 2016; Olson 2011], reducing the number of possible markers to several thousands. Second, the dimensionality of real-world perturbations and actual capturing conditions are significantly higher than what the current marker methods assume. For instance, shadow and specular reflection combined with the perspective distortion hinder the successful detection of markers. Designed patterns and recognition algorithms often cannot function robustly when imaging conditions are poor. Recently, Hu et al. [2019] proposed to train a convolutional neural network, specifically a variant of super-point network [DeTone et al. 2018], to improve the identification capability of the existing ChArUco board marker corners [OpenCV 2020] and the camera pose in harsh conditions. They demonstrate the improved detection results, while they only focus on the ChArUco board corner detection as a whole, as opposed to individual ArUco marker detection. Their corner refinement network requires separate training and needs recollection of a dataset for different marker methods.
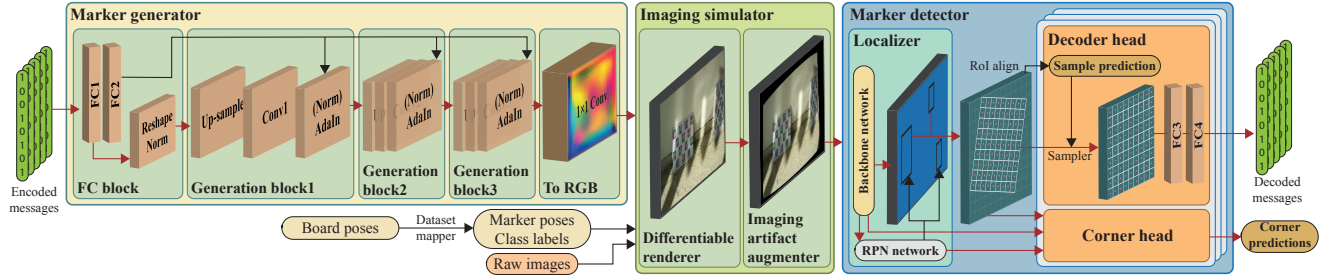
Fig. 2. Overview. Our end-to-end marker encoding/decoding system consists of three main parts: First, the marker generator creates unique fiducial marker patterns from input messages. Second, the image simulator renders photorealistic scenes where the markers are simulated with high fidelity through a differentiable rendering framework. The rendered images are also augmented to mimic real-world imaging scenarios. Lastly, our marker detector consists of a localizer that proposes regions of interest of each marker, where our decoder network infers a binary message via inverse transformation of deformation.

In addition to square-shaped markers, dot-pattern based markers are developed to detect the deformation of the surfaces [Narita et al. 2016; Uchiyama and Marchand 2011]. In order to be detected successfully, each marker needs to occupy a very large portion of object surfaces. For these reasons, detection of a large number of markers is rather limited, for instance, Narita et al. [2016] presented that only two markers at a time are detected as the maximum number.

Learning-based markers. There have been several efforts to enhance the performance of fiducial marker detection using deep learning. An approach of creating learning-based markers was proposed by Grinchuk et al. [2016]. They synthesize markers from binary input codes and render them on random image patches with transformations and distortions. They then recognize the decoded message. This work is a seminal conceptual study that focuses on only creating learnable markers. It misses other critical components, for instance, capability of localization, which are required to be used in practice. And thus, these trained markers should be superimposed at the center of images, and this method cannot detect deformed shapes of markers. There is a concurrent work within this scope. Peace et al. [2020] attempted to overcome practicability of the existing learnable markers. However, the number of their binary codes is limited to a very small number of 30, and the performance is insufficient as 13 fps for the real-world tracking applications. As like other existing methods, they assume that fiducial markers are printed on perfectly planar surfaces so that they cannot detect severely deformed fiducial markers. In contrast, our system is a complete deformable marker method, which is learned with *shape deformation of markers* in an end-to-end manner. It can be used in various real-world application scenarios in real time.

Stegenography. Steganography and watermarking methods have been used to embed hidden messages to an image and to decode them through certain perturbations. Recent methods utilize learning-based encoding-decoding of messages with a small perceptual difference in a perturbed image [Baluja 2017; Hayes and Danezis 2017; Tang et al. 2017; Wu et al. 2018]. HiDDeN [Zhu et al. 2018] augments messages by applying color distortions and noise to the encoded image, assuming perfect alignment during the decoding process without any spatial transformations. Wengrowski and Dana [2019] capture a dataset to model a camera-display transfer function in order to add distortions to input images. However, none of these

methods account for the localization problem of encoded messages in input images. Recently, Stegastamp [Tancik et al. 2020] starts to include perspective warp to the perturbation pipeline. However, their detection method presents a degraded performance in the real experiments due to the separate architecture of the segmentation network.

3 DEFORMABLE FIDUCIAL MARKER SYSTEM

3.1 Overview

Our deformable fiducial marker system stands on end-to-end training of message encoding and decoding in three folds. First, our marker generator creates unique marker patterns from input binary messages. Second, the created fiducial markers are then supplied to our photorealistic image simulator that consists of a differential renderer and the imaging artifact augmenter. The differentiable renderer simulates the photorealistic appearance of the markers. The augmenter simulates various imaging artifacts of deformation, perspective/lens distortion, motion blur, compression artifacts, and a variety of illumination conditions. Lastly, our marker detector is trained with those images. It consists of a localizer to obtain regions of interest and our marker decoder that detects corners and decodes messages via deformation-aware transformation. During inference, the identification of markers' binary messages is evaluated with the dictionary of markers. Figure 2 provides an overview.

3.2 Learnable Marker Generator

Design insight. In order to create a set of ideal fiducial markers, there are several challenges and trade-offs that need to be resolved. Fiducial markers should contain a *rich variety of appearance* to be able to encode and decode *a large number of messages*. At the same time, a group of fiducial markers should look *similar* among themselves in a way to be robustly detected by a marker detection method. On the other hand, from the perspective of the detector, the marker patterns should look *unique* against natural objects' appearance in the real-world in order to be detected clearly. Otherwise, a detector would fail with false-positive identification.

Traditional fiducial markers consist of black/white (or color) patterns enclosed by a border. Marker patterns are analytically designed to achieve the aforementioned goals. However, the number of such hand-crafted patterns is very limited to hundreds, and still, these patterns cannot guarantee robust performances under deformation,

distortion, and motion blur. Recent learning-based approaches are still tested with a limited number of marker patterns, for instance, 30 patterns [Peace et al. 2020].

To achieve the capability of embedding a large number of messages, we are inspired by steganography, a practice of concealing a message within another message [Johnson and Jajodia 1998]. Our design insight is based on binary encoding and decoding of embedded messages. However, different from traditional steganography, our method learns alternative representations of binary codes, which are associated with the latent representations of real-time object detection network jointly trained in the end-to-end manner. Our approach can embed a very large number of binary messages that grow exponentially by the size of the message/number of bits (two to the power of the number of bits).

Network architecture. To achieve the strongly distinguishable appearance of a rich variety of marker patterns, we begin with a baseline generative model, StyleGan [Karras et al. 2019]. The model includes the adaptive instance normalization operator (AdaIn), which learns to align the mean and variance of feature channels. Our goal is to create visual markers in a certain spatial resolution from binary messages in the dictionary. We found that the AdaIn operator helps transformation from the binary message domain to the real-world domain for the markers for better detection and decoding.

We adopt the generation block of StyleGan with the AdaIn operator with a difference. The original generative model is trained on images with progressively higher resolution based on [Karras et al. 2018]. Since we use a relatively small resolution (32×32) compared to natural images, we do not include the progressive resolution approach.

In detail, we apply a fully-connected (FC) linear transformation to the input binary messages. We then normalize these transformed features and apply an activation function. We resize the features from the FC layer into a four-by-four shape. As a next, we control the appearance of the markers through convolutional generation blocks. Each generation block applies upsampling at first followed by convolution and normalization operations. Style controlling weights of the AdaIn are computed through applying the FC layer transformations further to the transformed message features. We use the Leaky ReLu activation function after convolutional layers. The outputs of marker generation networks are finally applied to sigmoid function to normalize the output values within [0, 1] range in three color channels for rendering. Refer to the leftmost yellow block of Figure 2 for an overview of the marker generator network.

3.3 Differentiable Image Simulator

Design insight. Our differentiable image simulator is designed with two insights. First, the existing learning-based approaches [Hu et al. 2019; Peace et al. 2020] have trained the marker generator or detector by superimposing markers on photographs from the MS COCO [Lin et al. 2014] or ImageNet [Deng et al. 2009] datasets. Markers are placed at random positions, ignoring scene geometry and illumination. See Figure 3 for an example. The realism of the superimposed training dataset is far lower than actual images that include fiducial markers in the real-world scenes. The domain gap between the randomly superimposed images and natural images

(a) Random superimposed rendering   (b) Our photorealistic rendering

Fig. 3. (a) An example of randomly superimposed training images with our markers on a COCO dataset, commonly practiced in state-of-the-art methods. Original image courtesy of the Mennonite Church USA Archives. (b) Our photorealistic rendering of our training dataset.

(a) Raw   (b) Preprocessed   (c) Markers

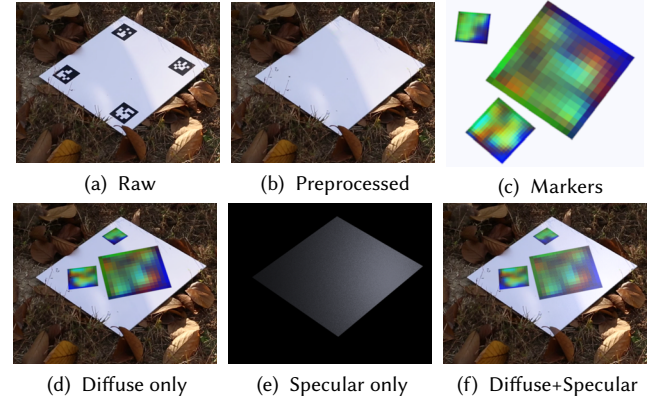(d) Diffuse only   (e) Specular only   (f) Diffuse+Specular

Fig. 4. Rendering components. Raw input images (a) are preprocessed by removing initial markers used for the board detection (b). (d) shows diffuse-only rendering of the created marker (c). (e) presents simulated the specular component of the marker. (f) shows the final rendering image.
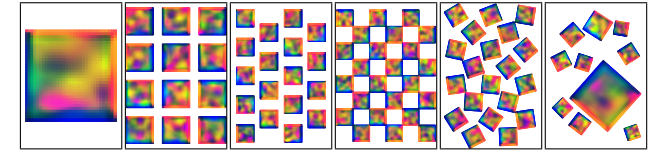

Fig. 5. Marker configuration examples. We place markers on the boards with various alignments during training.

of fiducial markers could degrade training performance severely. It requires further efforts of domain adaptation. It is verified later through an ablation study in Table 2. To mitigate the domain gap in training, our approach is to create *photorealistic rendering images* for training, where the generated markers are seamlessly rendered by physically-based rendering composed in real photographs. Second, our method has increased the messaging capability in an exponential scale by means of color patterns. However, it becomes more critical to learn the appearance variation of color patterns in the real-world environment because a color code might be misinterpreted as a different code depending on environments. To mitigate these two challenges, we devise a differentiable image simulator for *photorealistic* rendering of the generated fiducial markers.

3.3.1 Differentiable Rendering. In our end-to-end optimization framework, we need a large number of photorealistic training images, where the generated markers are placed in real-world scenes at runtime. If we render the generated fiducial markers through a physically-based rendering framework, the rendering process may

take a several hours to obtain a single high-resolution image. Instead, we introduce a practical solution to create realistic marker images efficiently. In brief, we first prepare thousands of real-world video frames under different illumination, where multiple white boards as space holder are placed. Then, we render the generated markers on the board with diffuse shading of scene illumination and synthetic specular reflection. Lastly, our augmentation operators simulate various imaging artifacts of deformation, projection, noise, illumination, etc.

Preprocessing and placement. For our main training pipeline, we collect a real-image dataset, so-called *placement dataset.* It consists videos of planar white boards in real scenes from simple to complex scenes with many objects (see Figure 4a). We capture video frames of 140 real-world scenes under various illumination conditions indoor and outdoor with two DSLR cameras (Canon EOS 5Ds equipped with 22mm and 50mm lenses, respectively). To capture high-quality photographs without imaging artifacts, such as motion blur, we capture videos at a short shutter rate using two 3-axis gimbals. Note that the white boards include the initial Aruco markers [Munoz-Salinas 2012] on the corners, which are used for automatic localization of the white board of space holder. We detect the white board's location and orientation from the initial markers, and estimate its homography for the marker placement later on. To provide a spotless surface for rendering, we remove the initial Aruco markers using two inpainting algorithms [Liu et al. 2018; Yu et al. 2018] (see Figure 4b).

During training, we create 96 unique messages at each iteration, and feed them to the generator. For each board, we select one of the preset layout configurations (as shown in Figure 5). Next, we place markers by randomly selecting from 96 markers according to the boards homography and layout. Using 96 unique markers per iteration provided better convergence, as opposed to using unique markers for each location. In addition, identification indices of the markers later provide us ground-truth labels to calculate the average precision score.

Rendering of markers. For the planar board, we use diffuse white paper evenly attached on an aluminum plate so that surface exhibits mostly Lambertian reflection. Assuming paper surface m_p and marker's surface m_t have constant BRDFs: $f_{\mathbf{x},\vec{n},m_p}(\vec{w},\vec{v}) = \frac{\rho_p}{\pi}$, $f_{\mathbf{x},\vec{n},m_t}(\vec{w},\vec{v}) = \frac{\rho_t}{\pi}$, for the camera direction $\vec{v}$, radiance from surface point $\mathbf{x}$ in 3D is: $L(\mathbf{x},\vec{v}) = \frac{\rho}{\pi} \int_{\Omega} L(\vec{w},\mathbf{x})cos(\theta)d\omega$. Therefore, the reflected radiance of the surface is proportional to the diffuse reflectance ratio between the target rendering surface and the paper surface. The reflected radiance can be computed as: $L_{m_t}(\mathbf{x},\vec{v}) = L_{m_p}(\mathbf{x},\vec{v})\frac{\rho_t}{\rho_p}$. The marker colors are then multiplied by corresponding surface pixels and divided by the surface's proportional diffuse reflectance (Figure 4d). We calculate color in linear color space, then gamma correct it before providing it to the detection network.

Real-world paper and printer ink often present specular reflections. To simulate shininess of real-world paper, we introduce specular rendering (Figure 4e). We use the surface normals, viewing directions, and surface pixel values to create specular effects. We select the light direction from perfect reflection direction of the brightest point within the board. And we define the light's color

(a) Deformation (b) Perspective (c) Radial dist. (d) Motion blur

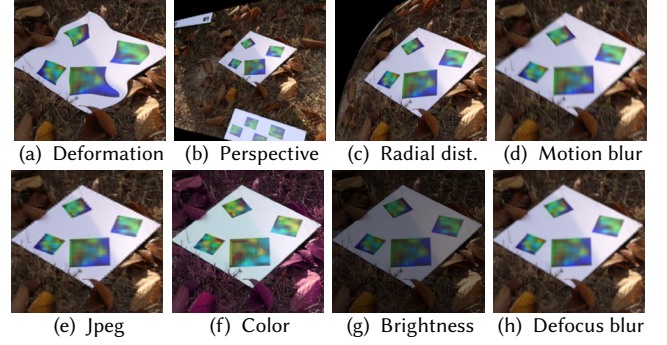(e) Jpeg (f) Color (g) Brightness (h) Defocus blur

Fig. 6. Examples of imaging artifacts. Deformation, perspective and radial distortions provide geometric augmentations, while the rest provides color-based augmentations of motion blur, JPEG compression artifacts, illumination color change, scene brightness change, defocus artifacts, etc.

as the normalized color of the brightest point, with a randomized intensity power value, scaled by the boards overall brightness. We use the Cook-Torrance microfacet BRDF model with the GGX microfacet distribution function [Walter et al. 2007] to render specular highlight on surfaces. For the specular component, we fix roughness and specular albedo across the surface with small normal perturbations. This approach shows viable renderings of specular effects that occur in the wild (see Figures 3b and 4f for rendering results used for training).

3.3.2 Simulation of imaging artifacts. As discussed earlier, many edge cases exist for recognition. Hence, it is not enough to simply feed rendered images with labels to the detection pipelines. Therefore, we devise an imaging artifact simulator for differentiable augmentations of rendered images to generalize for the difficult conditions. Figure 6 presents examples of artifacts.

Deformation. To simulate scenarios of deformed and non-flat placement of markers, we warp the rendered image using *thin-plate splines (TPS)* [Duchon 1977] at training time. We intentionally deform the surfaces in an algorithmic manner, rather than capturing predefined non-planar surfaces. This decision is taken to avoid any potential risk of the network's overfitting to a limited number of predefined shapes. Also, preparing real objects of various shapes is cumbersome.

In detail, we first define control points in the target image as a uniform grid. We then shift them, each in a random direction and amplitude, to obtain the control points in the source image. Finally, we used TPS to create a dense sampling grid following the control points. This process allows for varied deformation and distortion while the properties of TPS ensure smoothness. See Figure 6a for an example.

It is worth noting that while we apply these non-linear geometric distortions, we had to recalculate ground-truth labels such as marker corners. We also store and recalculate uniform marker sampling locations, which are later used to teach the network to explicitly invert all the nonlinear transformations applied to the marker.

Perspective and lens distortion. To cover a more variety of perspective distortions, we applied random homography to the image with bilinear sampling. We also simulate radial distortion following
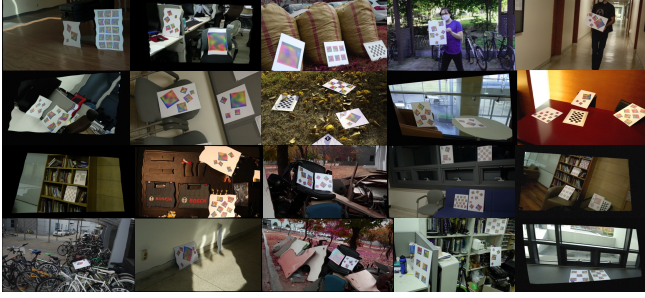
Fig. 7. Examples of our training dataset with imaging artifact augmentation generated at runtime.

the Brown–Conrady model [Fryer and Brown 1986] with three random coefficients and random focal length and distortion center. In particular, we focus on barrel distortion (negative coefficients) as the popular wide field of view lenses are usually prone to it.

Color and brightness changes: Since our method uses color features, it is important to have a realistic variety of colors and brightness of the environment. Environment illumination conditions, such as specular reflections, surface roughness, shadows, and scarce light exposure, affect the measured irradiance. We cover specular reflection with different roughness values, and shadows in the rendering section. For enhanced robustness of the detector under ill-conditioned illumination, we apply random brightness re-scale to the whole images. In addition, we introduce non-linear transformations through random gamma correction and also shift hues for better generalization to different lighting environments.

Defocus and motion blur, noise, and compression artifacts: We simulate imaging artifacts related to the camera: defocus assuming a random-sized circular aperture and constant depth over the image and motion blur of variable magnitudes and directions. We also introduce Gaussian noise with a random standard deviation to mimic CMOS sensor noise with different ISO parameters. We finally reproduce compression artifacts through differentiable JPEG approximation [Zhu et al. 2018].

After augmentations are applied, we apply gamma correction to the augmented colors convert back to sRGB values, and clamp them to the valid range for the detection network. During training, we create random binary messages of given bits to generate fiducial markers. Until finishing 12 epochs, we create about 561,912 unique images with all augmentations. See Figure 7 for examples of our training dataset.

3.4 Learnable Marker Detection

Design insight. Our ultimate goal for marker detection is to detect *multiple* markers within the image and decode binary messages of each marker *in real time.* To develop a novel real-time marker detector, we are inspired by the two-stage Faster-RCNN approach [Ren et al. 2015] that employs backbone features exclusively for both the region proposal and classification tasks. In addition, we need our marker detector to be scale-invariant. Therefore, we design two main RoI heads that are shared across the different scales of the backbone features: (1) corner detection and (2) message decoding heads. To utilize backbone features efficiently, we only spatially transform
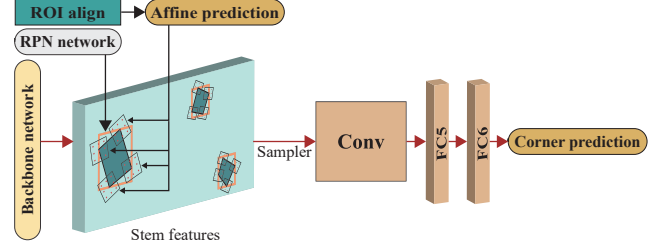
Fig. 8. Corner head. This diagram depicts the detailed architecture of our corner detection network shown in Figure 2. Four affine matrices predict to sample from low-level features of the corner regions. Then the sampled regions are processed to predict relative corner coordinates in affine frame. Finally, affine matrices are used again to calculate corners in image coordinates.

them by inverse warping through two-stages of resampling, and avoid costly convolution operation. This makes our decoding network invariant to the non-linear deformation transformation, while keeping the network depth shallow.

Network architecture. We exclusively make use of features from the backbone network of object detection. First, features are used by the region proposal network (RPN) for obtaining the regions of interest (RoIs) and its initial bounding box proposal. Second, using the bounding box proposals, features are resampled by the RoI align [He et al. 2017] operation. Lastly, the pooled features are directly used for (1) predicting corners by estimating affine transformation matrices and (2) decoding messages via deformation-aware sampling. Our network architecture is shown in the rightmost column of Figure 2.

We include Feature Pyramid Network (FPN) in our network to aid possible scale differences further [Lin et al. 2017]. With the consideration of real-time performance of detection, we choose VovNet [Lee et al. 2019] as a backbone network.

Corner head. Corners of markers are the critical information for various applications of fiducial markers, enabling object tracking and camera pose estimation. However, corners are a type of information that does not include any high-level semantic meaning, and thus when we use high-level features for training the corner detector, it could not perform very well. Accordingly, we decided to use the low-level stem features from the backbone network directly, rather than high-level FPN features. See Figure 8 for a detailed architecture of our corner detector. For robust detection of corners, we estimate affine transformations of sampling window for each corner with respect to the center origin of the normalized RoI domain. We then sample the stem features through the transformation and then apply a convolutional and two fully connected layers to predict the corner location. We use the predicted corners to calculate corner loss which is formulated as

$$\mathcal{L}_{\text{corner}} = \frac{1}{8N_{\text{total}}} \|x_{\text{gt}} - x_{\text{predict}}\|_1, \qquad (1)$$

where N_{total} is the total number of regions, x_{gt} and x_{predict} present pixel coordinates of ground-truth and prediction, respectively, and the loss computed for only foreground regions.

Decoding head. The spatial transformer network [Jaderberg et al. 2015] been used broadly for object classification in order to enhance classification performance by inferring optimal inverse transformation. However, it lacks explicit knowledge of geometric transformations. In contrast, we actively leverage the explicit knowledge of geometric transformation applied to markers to estimate inverse transformation of deformation.

To this end, we first calculate uniform sampling locations in the marker domain. Every time a transformation is applied, we recalculate these locations. We apply homography transformation for rendering, then in the augmentation stage, we apply affine augmentation, lens distortion, and thin-plate spline warping consecutively and recalculate sampling locations. We assume that if the network can predict these locations correctly, it understands the forward warping applied, and can inverse them to normalize markers back to generated marker domain. The sampling resolution is set to 9×9, for the 12×12 pooled RoI region. In addition, we set a sampling loss from calculated ground truth feature sample locations for decoding, which can be formulated as

$$\mathcal{L}_{\text{sample}} = \frac{1}{2N_{\text{total}}N_{\text{sample}}}\|x_{\text{gt}} - x_{\text{predict}}\|_1, \tag{2}$$

where N_{sample} is the total number of sample locations (9×9).

As a next step, we resample from the pooled RoI features using predicted sampling locations with bilinear sampling. Then we apply fully connected layers to the geometrically normalized marker features. For our comparison study of traditional markers, the cross-entropy loss is used to predict the marker class. For our learned markers, we have decoding loss, formulated as following:

$$\mathcal{L}_{\text{decode}} = \frac{1}{N_{\text{total}}N_{\text{bits}}}\|m_{\text{encoded}} - m_{\text{decoded}}\|_2^2, \tag{3}$$

where N_{bits} is the total number of bits in a binary message, m_{encoded} and m_{decoded} present the encoded binary message of ground-truth and the decoded message, respectively. We also predict objectness for each RoI. We use binary cross entropy as an objectness loss $\mathcal{L}_{\text{obj}}$, which later used for non-maximal suppression.

Our final loss term is formulated as follows:

$$\mathcal{L}_{\text{total}} = (\mathcal{L}_{\text{rpn}_{\text{class}}} + \mathcal{L}_{\text{rpn}_{\text{loc}}}) + \mathcal{L}_{\text{sample}} + \tag{4}$$
$$0.1\mathcal{L}_{\text{corner}} + (0.5\mathcal{L}_{\text{obj}} + 10.0\mathcal{L}_{\text{decode}}),$$

where $\mathcal{L}_{\text{rpn}}$ loss comes from region proposal network's objectness and bounding box prediction, which is unchanged from the original detection network [Wu et al. 2019].

Marker identification. Once we train the end-to-end system for encoding/decoding messages, we are ready to identify markers using our marker detector network. From the decoder network, we receive a binary message, a 36-bit dimensional feature vector of binary codes, and check the identification by comparing the binary codes of each message with the dictionary. If the confidence level of matching is lower than 80%, we reject the identification. The impact of this threshold on accuracy is evaluated in Section 5.

3.5 Implementation Details

We implement our method with PyTorch on the base architecture of Detectron2 [Wu et al. 2019]. We train our models on eight NVIDIA Titan RTX GPU with Intel Xeon Silver 4214 CPU.

We employ a stochastic gradient descent with momentum and gradient clipping. Our model is trained with 35,000 steps (~12 epochs) with learning rate decayed to 1/10 after 20,000 and 30,000 steps. We do not change the default training parameter settings of Detectron2. Please refer to supplementary material for more details on network architecture.

4 ABLATION STUDY

For comprehensive ablation experiments, we choose our synthetic dataset including the ground truth values of corners and messages.

Evaluation metric. To evaluate the accuracy of marker detection results, we employ a popular metric for object detection, average precision (AP)[1] instead of object proposal proxy metrics, following other detection works [Ren et al. 2015]. We make use of Detectron2 implementation to calculate metrics.

4.1 Marker Generation

We train our marker generation network with different options in the marker generation architecture. Figure 9 compares the visual markers qualitatively, and Table 1 evaluates AP scores, message decoding accuracy, false positive (FP) rate, and corner errors. As shown in the leftmost part of Figure 2, our marker generator consists of five main blocks: FC block, three sub-generation blocks, and color conversion block. Each sub-generation block consists of upsampling, convolution, and normalization. We evaluate the impact of different normalization options: no normalization, pixel-wise feature normalization, channel-wise AdaIn normalization with replication padding,

---

[1] Average precision (AP) computes the average precision value (about how accurate the predictions are) for recall value (how good the detector is finding the positive examples) in the percentage scale. AP is averaged over all markers in our experiments.
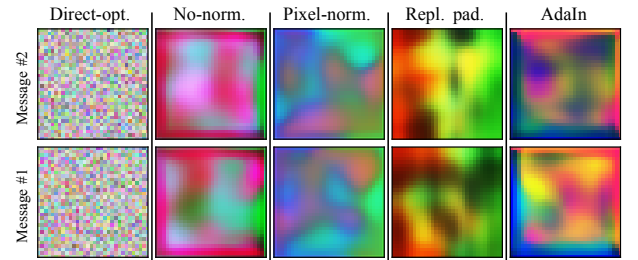

Fig. 9. Qualitative comparison using different types of optimization and normalization in our marker generator network.

Table 1. Quantitative comparison of the generator network options in terms of initialization and normalization. The red numbers indicate the highest performance, and the blue numbers represent the second-best performance.

Method	Marker Detection			Corner RMSE
	AP	Decoding accuracy	FP Rate	
Direct optimization	75.75	-	0.0088	0.6699
No-norm. generator	94.25	96.12	0.0008	0.3808
Pixel-norm. generator	94.35	98.05	0.0018	0.3965
AdaIn generator repl. padding	94.44	95.65	0.0008	0.3653
AdaIn generator w/ zero pad.	95.10	98.18	0.0014	0.3643

Table 2. To evaluate the performance of our image simulator independently from other parts of our system, we render Aruco markers as shown in Figure 10 and train the same decoder network. This table shows *AP* metric evaluation of each option of our image simulator. Our detector model is trained on different rendering methods, indicated as columns. Simulated and real-world test datasets are used to evaluate which indicated as two main rows. Each sub-row and column indicate the augmentations used to train/test the models. As the simulated and real test datasets yields high accordance, simulated images are used to evaluate the architectural decisions of the network. Real-world testing rows indicate that our full simulation pipeline with training augmentations yield the best results across all testing categories. Including color and geometric augmentations helps the generalization of the models across all datasets. Note that the overall scores on fully augmented test datasets is still low as imaging artifacts impose challenges to detection and due to the absence of our learnable markers.

Test \ Train	Baseline Aruco	Superimp. COCO	Placement w/o shading				Diffuse shading only				Diffuse shading + spec.			
			No	Color	Geo	Full	No	Color	Geo	Full	No	Color	Geo	Full
Simulated No	70.41	29.76	02.47	57.40	04.05	60.72	10.32	75.24	19.74	76.95	79.13	91.07	87.47	92.80
Color	37.55	16.54	00.40	32.97	00.40	34.56	05.72	68.00	11.27	71.15	45.61	70.06	43.37	72.19
Geo.	35.82	22.98	01.16	48.02	02.60	53.79	05.11	64.24	14.75	69.80	54.51	77.65	70.50	82.44
Full	22.89	13.50	00.28	27.70	00.22	30.23	03.11	54.25	08.08	59.09	33.84	56.19	34.95	59.78
Real No	67.60	35.59	02.89	51.42	04.94	52.30	27.18	76.92	55.54	80.52	57.96	85.24	68.72	87.66
Color	36.20	17.39	00.02	25.88	00.07	26.29	11.97	55.10	23.47	58.35	27.04	60.77	33.40	62.45
Geo.	34.60	26.62	01.06	41.02	03.19	44.88	15.21	64.31	42.30	70.94	41.42	71.51	54.70	76.73
Full	22.29	14.43	00.00	21.63	00.00	22.91	07.95	44.17	18.69	48.49	27.47	48.87	27.14	51.92

and AdaIn normalization with zero padding. Also, we compare them with direct optimization, which is to calculate gradients of randomly initialized weights for optimization of marker patterns.

Quantitatively, we find that the AdaIn normalization with zero padding outperforms all other options in terms of AP score and decoding accuracy in marker detection. It also outperforms others in terms of corner detection accuracy. We, therefore, use the AdaIn normalization with zero padding as our method. Qualitatively, we observe that the learned markers produce common features, such as blue patterns at the left-bottom corner, and thus preserving appearance similarity. We found that the rich variety of marker appearances on our selected generation method supports our design insights described in Section 3.2.

4.2 Image Simulation

To evaluate our image simulator, we train our detection network with different rendering configurations. Note that we employ Aruco markers to exclude the impact of the marker generator network. We then test our models on both the captured boards with random Aruco markers and a rendered image dataset. The rendered images are simulated by our renderer with the same placement configuration as the real-world dataset (see Figure 10c), creating a simulated replica of the real-world test dataset. The model trained on our diffuse and specular rendering combined with geometric and color augmentations shows the best results for both synthetic and real-world evaluation datasets (Table 2). In addition, we apply augmentation to both test datasets, making the detection task significantly more challenging. Table 2 reveals that training with augmentation successfully improves the robustness of the detection networks, for all rendering methods. Evaluation results indicate that the simulated and the rendered test datasets yield consistent gap with mean difference of 2.455 and standard deviation of 10.066. This shows that rendering can be used for further ablation studies.

4.3 Marker Detection

Backbone networks. We experiment on different backbone networks, to select the most suitable architecture for our real-time marker detection system. We select networks with feature pyramid structure to aid better for scale differences, mainly ResNet [He et al. 2016], and VoVNet [Lee et al. 2019] with different depths. Prior

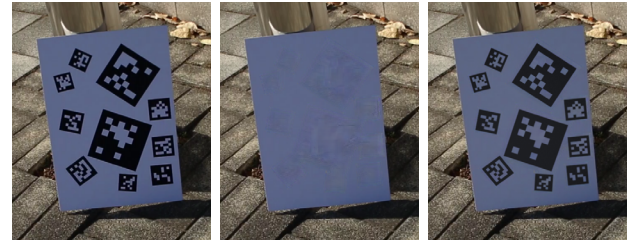(a) Captured　　　(b) Inpainted　　　(c) Simulated

Fig. 10. Rendering pipeline for creating Aruco images for image simulation ablation (Table 2). We first capture boards with random Aruco markers (a), we then simulate the same markers at the same location through our inpainting (b) and rendering (c) approach.

Table 3. We evaluate the impact of different backbone network architectures for marker detection performance. We test two different levels of ResNet with FPN, and VoVNet. Despite the shallower depth of the network, we found that VoVNet19 DW-FPNLite [Lee et al. 2019] is the best.

Method	AP	AP_S	AP_M	AP_L	Decoding accuracy	FP-Rate	FPS
ResNet50 FPN	92.40	88.83	99.51	100	97.91	0.0014	17.0
ResNet100 FPN	90.45	86.00	99.25	100	97.68	0.0018	12.5
VoVNet39 FPN	94.38	92.22	98.85	100	98.40	0.0022	16.5
VoVNet19 DW-FPNLite	95.10	93.06	99.43	100	98.18	0.0014	29.0

to our experiments, we expected deeper networks to perform better at recognition tasks. On the contrary, we observe that shallow VoVNet19 DW-FPNLite provides better performance than the others. See Table 3 for comparison.

Message embedding capacity. To evaluate the message embedding capability and impact of the number of bits on decoding accuracy, we train models with three different message lengths of 18, 36, and 64 bits. We place different markers to the image using our rendering pipeline as shown in Figure 11a. We place 70 markers at each iteration (see Figure 11b). We test all possible markers for the 18-bit model and test ~1 billion (=2^{30}) random markers for 36- and 64-bit models. This number of samples is limited by the computing resource in our setup (Section 3.5). These experiments took 12 days. For fair validation of the 36- and 64-bit models, we employ stratified sampling of bit messages to cover all marker feature space with ~4.2 million markers sampled for 256 subgroups. Statistical results provided in Table 4 validate that the generated markers are encoded

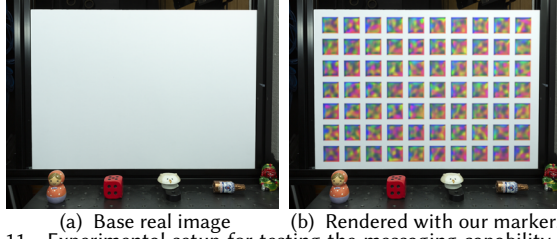(a) Base real image   (b) Rendered with our markers

Fig. 11. Experimental setup for testing the messaging capability of our method. We capture an empty white board (a) to render our fiducial markers (b) through iterations of encoding/decoding of messages.

Table 4. Decoding accuracy results over markers sampled from three different models of 16-, 36-, and 64-bit models. Percentages of 0-bit error (all bits in each message are correctly predicted) and 1-bit error are also presented.

Model	Mean decoding accuracy	Standard deviation	0-bit error	1-bit error
16 bits	99.998	0.1143	99.97%	0.03%
36 bits	99.921	0.7605	98.51%	00.75%
64 bits	99.558	1.0787	80.39%	14.00%

and decoded successfully with high accuracy within the bit range of messages.

RoI heads. We investigate a variety of RoI heads on detection and decoding performance with our learned markers. We found that naïve convolutional decoding head without any learned spatial transformation fails to converge, and thus we instead employ an affine-based spatial transformer network [Jaderberg et al. 2015] as a baseline. Next, we develop our special decoding head that uses supervision signals to inverse the transformations. In addition, to provide better corner accuracy, our corner head learns to adaptively sample the corner regions, and predict the corner locations. Table 5 shows that our guided transformer improves the decoding accuracy significantly by only resampling the backbone features without expensive operation. It keeps our network efficient, and satisfies our design insights described in Section 3.4. In addition, the affine-based sampled low-level backbone features, utilized by the corner head, improve sub-pixel corner prediction accuracy with a significant margin.

Table 5. RoI head architecture validation. We compare the marker detection accuracy and corner prediction accuracy. Our guided transformer and affine-based RoI head outperform the baseline methods.

Method	AP	Decoding accuracy	FP-Rate	Corner RMSE
Baseline corner head + spatial transformer	83.69	95.41	0.0022	1.0426
Baseline corner head + guided transformer	89.80	97.09	0.0015	0.7768
Affine-based corner head + guided transformer	95.10	98.18	0.0014	0.3643

4.4 Deformation

Our marker detection network consists of two main blocks: the guided transformer network and the corner prediction network. Our detection network is also trained with TPS-based augmentation. In this study, we evaluate the relative impact of our detection network and the TPS-based augmentation in training using the traditional Aruco markers. We built an experimental setup, where Aruco and our markers trained with/without TPS augmentation are printed on paper and deformed as shown in Figure 12.

Figure 13 compares detection performances with different configurations, compared with the baseline OpenCV detector with Aruco

(a) Detection with Aruco   (b) Detection without TPS   (c) Detection with TPS

Fig. 12. We build an experimental setup to evaluate the impact of our RoI head network and TPS-based augmentation for detecting markers.
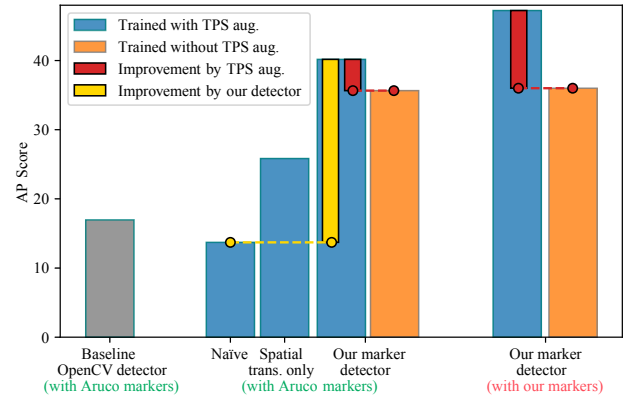

Fig. 13. Using the experimental setup shown in Figure 12, we compare the impact of our marker detection network and our training with TPS-based augmentation on detection accuracy with deformed fiducial markers. As shown in results using Aruco markers in the middle, our detection network contributes more than our TPS-based augmentation for robust detection performance of deformed markers. When our learnable markers are applied additionally to our detector method, our system outperforms all the baseline methods significantly.

markers. In the middle, the four different bars present AP scores using four different configurations with Aruco markers. The first blue bar shows the AP score using only a vanilla VoVNet19 network and Aruco markers. The simple detection network does not account for any spatial deformation, resulting in a lower performance than the baseline method. The second blue bar presents an improvement achieved by applying an existing affine-based spatial transformer network [Jaderberg et al. 2015] in the RoI head. The third blue bar shows the performance of our detector network with Aruco markers, being trained with TPS-based augmentation. The fourth orange bar compares the result by the same network but trained without TPS-based augmentation. As indicated by the additional yellow and red thin bars, when we employ the fixed patterns of Aruco markers, the improvement by our detection network is more significant than that by our TPS-based augmentation in training.

In addition, we compare these Aruco results with our learnable marker results. Unfortunately, the training process of our marker detector without any spatial transformation network does not converge so that we cannot compare the result. Instead, we compare the precision results of our detection network trained with/without TPS-based deformation. As shown in the rightmost part of the plot, when our method is trained with our learnable markers and TPS-based augmentation, our marker detector outperforms both the baseline detector and the Aruco-based trained network significantly.

5 RESULTS

Markers' appearance over iterations. We evaluate increasing AP scores over every 2,500 iterations. For every 5,000 iterations, we also evaluate the visual appearance of one of the optimized visual markers. Figure 14 presents testing AP scores with our synthetic test dataset, and also visualizes the marker's appearances over the training iterations. Starting from 5,0000 iterations, the network achieves reliable AP scores. The appearance of the optimized marker becomes stabilized after 25,000 to 30,000 iterations. With more iterations, the accuracy increases asymptotically and the marker generator uses more saturated colors and defines clearer lines at the boundaries.
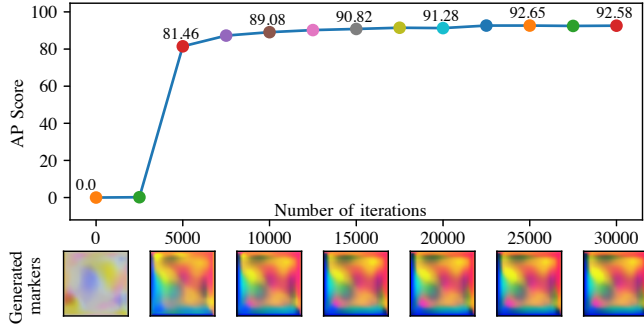
Fig. 14. Marker appearance changes over iterations. With more iterations, the accuracy increases asymptotically and the marker generator uses more saturated colors.

Impact of the confidence level. Once we estimate a binary message code from a detected marker, we apply confidence filtering to determine the marker's identification. In our current system, the confidence level is a hyperparameter in our detection mechanism. We evaluate the impact of the confidence level in terms of AP score with our real-world test dataset. There is a tradeoff between the AP score and confidence level. When we increase the level of confidence, the overall AP score decreases. When the confidence level decreases, it potentially increases the risks of having more false positives. We, therefore, choose 80% for the confidence level in our experiments. See Figure 15.
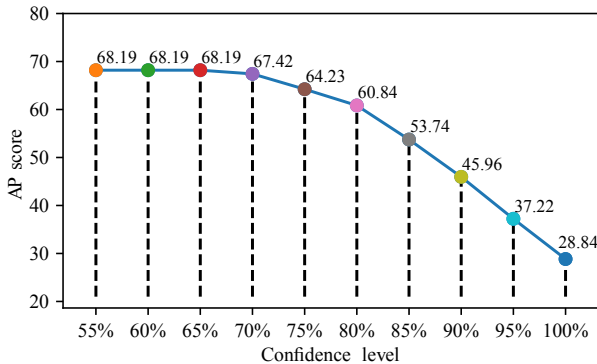

Fig. 15. Impact of the confidence level on the AP score tested with the real-world test dataset. We choose 80% as the confidence level threshold for our experiments.
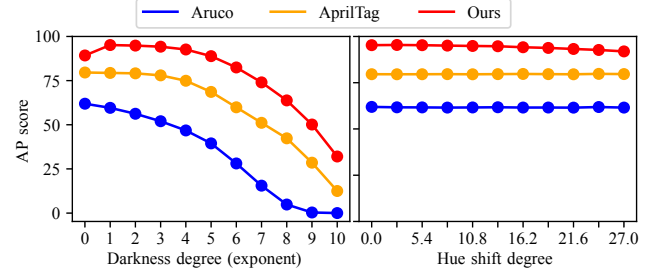

Fig. 16. To evaluate the robustness of three methods: Aruco, AprilTag, and ours, we create brightness changes and hue shifts synthetically and compare precision performances. Brightness is scaled by 0.6^k, with k ranging from 0 to 10 in the linear intensity domain. The hue values of pixel colors are shifted by 0 to 27 degrees in HSV color space. Our method outperforms the baseline methods within the range of scene illumination conditions.

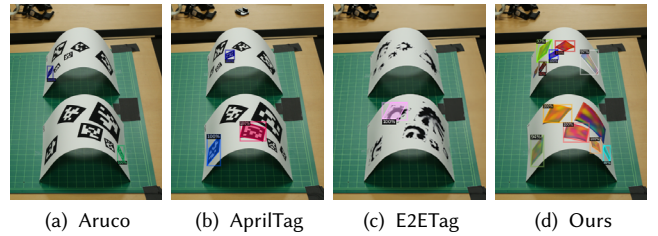(a) Aruco   (b) AprilTag   (c) E2ETag   (d) Ours

Fig. 17. Setup for deformation evaluation. We bend the paper with the same distances (10 cm) for all methods to analyze the robustness of each method to placement on non-flat surfaces.

Robustness against brightness and hue changes. Instead of binary black-and-white features, we create color features using our generative network, and thus its performance might be affected by brightness or hue changes of scene illumination. We, therefore, evaluate the performance of our detection network by synthetically varying the level of brightness and hue of images. We compare our method's AP scores with those of binary methods: Aruco [Munoz-Salinas 2012] and AprilTag [Wang and Olson 2016]. For brightness changes, we scale the level of pixel intensity by 0.6^k, with k ranging from 0 to 10 in the linear intensity domain. For hue changes, we shift the hue values of pixel colors by 0 to 27 degrees in HSV color space. As shown in Figure 16, our method still outperforms these two baseline methods of Aruco (implemented by OpenCV) and AprilTag (implemented by the authors).

Real-world deformation experiments. To evaluate the performance of these four methods on deformed surfaces, we design a controlled environment for a marker deformation experiment. We fix one side of the paper on a flat table and move the other side to bend the sheet and create a consistent saddle shape (see Figure 17). We prepare 54 manually annotated images for testing. We plot a graph of the displacement against the detection accuracy (more displacement meaning a more deformed surface). Figure 18 shows that our method is able to detect markers with great robustness on deformed surfaces. Figure 19 validates that our markers outperforms the other methods in detecting the identification of deformed markers.

We compare the detection rate of our learning-based deformable markers with two existing deformable markers based on dot patterns [Narita et al. 2016; Uchiyama and Marchand 2011]. For fairness of

(a) AprilTag on deformed surfaces

(b) Our markers on deformed surfaces

Fig. 18. Qualitative comparison of our deformable markers against AprilTag on deformed surfaces. While AprilTag fails on deformed markers, our method detects all the markers on deformed surfaces.
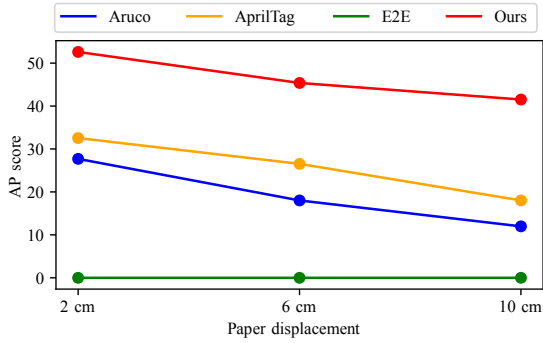

Fig. 19. AP score against different deformations. A larger displacement yields a stronger paper curvature. Our method significantly outperforms others when the the markers are placed on non-planar surfaces.

comparison, we replicate an experimental setup (Figure 20) with the same physical configuration presented in the original paper [Narita et al. 2016]. Then, we compare our detection performance on convex/concave curved surfaces against the performance reported in [Narita et al. 2016]. Figure 21 validates that our method outperforms these two methods for both cases of deformations.

Real-world flat surface comparison. We evaluate the performance of Aruco [Munoz-Salinas 2012] , AprilTag [Wang and Olson 2016], E2ETag [Peace et al. 2020] and our method on the real-world test dataset. Note that we compare our work against methods capable of multi-marker detection and localization since it is crucial for the applications we propose. For example, as the learnable visual markers [Grinchuk et al. 2016] do not localize and therefore cannot be included in our comparison. For fair comparison and to evaluate the real-world capabilities of each system, we print markers of the selected other works and capture them in the same conditions. For a given scene, we capture a set of markers created by four different methods. Among the four shots, we ensure the perfectly matching marker alignment, surroundings and lighting environments to evaluate each method equally. In the test scenes, we change illumination

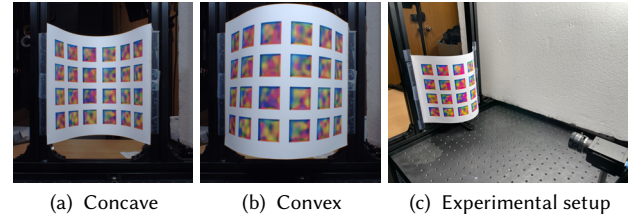

(a) Concave (b) Convex (c) Experimental setup

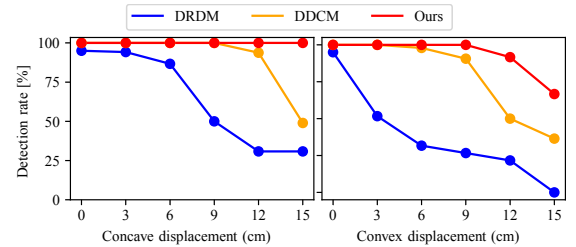Fig. 20. Experimental setup for comparing deformable markers.

Fig. 21. Detection rate against the deformation levels of concave and convex shapes compared with DRDM [Uchiyama and Marchand 2011], DDCM [Narita et al. 2016].

and camera parameters to cover varied scenarios. We then manually annotate the markers. We do not apply any augmentations during testing. We conducted the first experiment of markers placed on planar surfaces. We captured and manually annotated 73 images in total per marker system. Our real-world dataset includes various light sources with correlated color temperatures with a mean of 6,636 K (standard deviation: 4,542 K), ranging from 1,989 K to 37,406 K calculated from sRGB signals in images.

Figure 22 and Table 6 verify that our tags outperform the others on flat surfaces. We note particularly low accuracy from E2ETag [Peace et al. 2020]. This is explained by a different layout compared to their training dataset: they place their markers with simple superimposition and constant white boundaries while we place them on larger white paper sheets. In addition, their classification suffers greatly from the presence of multiple E2ETags in a single scene. We also ran our method at a lower resolution due to implementation
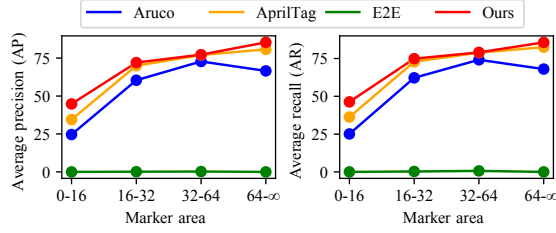
Fig. 22. Average precision and recall plots with respect to markers' areas in pixel domain on our flat real-world dataset. Our method outperforms all the compared methods in terms of all sizes of markers.

Table 6. We evaluate the performance of three state-of-the-art methods against ours on a real-world test dataset manually annotated. Tags are placed on planar surfaces. We use the same spatial resolution of 1216×800 for Aruco, AprilTag, and ours. Due to the implementation limits of E2ETag, we supply input images in a reduced resolution of 640×426. Our method outperforms classical and learned approaches on this planar dataset. The other learned method, E2ETag [Peace et al. 2020] presents low accuracy when several markers are captured in the same shot due to difficulties in classification.

Method	AP	FP-Rate	FPS
Aruco [2012]	50.19	0.0000	31
AprilTag [2016]	57.58	0.0000	19
E2ETag [2020]	00.04	0.8625	13
Ours (Aruco markers)	55.09	0.0575	29
Ours (Learned markers)	60.84	0.0000	29

requirements, further harming its detection performance, especially for small markers.

6 APPLICATIONS

In this section we propose three main applications that are allowed by our new deformable marker system. Refer to the supplemental video for all results in this section.

6.1 Structured Light 3D Imaging

For this application, we capture our markers projected densely in a scene by a projector with a stereo camera. One of the advantages of our marker system is the capability of producing and detecting a very large number of fiducial markers. For structured light 3D scanning, we project a large number of unique markers at each vertex on a 2D grid. Using the correspondence matches by the fiducial markers, we triangulate the matching corners from both views to estimate 3D geometry. It is worth noting that the scanning resolution with our marker system is merely limited by the resolution of the camera and projector used, rather than the messaging capability. Our results in Figure 23 show that, without explicit change in training augmentations, our markers are able to accurately estimate a large number of 3D points even when using a projector on curved surfaces.

6.2 Motion Capture of Deformable Surfaces

Assuming planar placement is a hard constraint, and prevents many additional applications. To achieve robustness for deformable markers, we apply strong geometric augmentations and we show in Section 5 that our marker system enables much more robust detection of markers on deformable surfaces. As an application, we place our markers on cloth, we use our markers like a feature descriptor
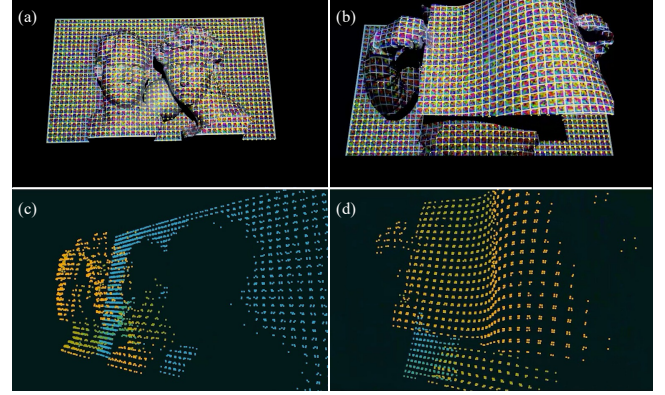
Fig. 23. Application of estimating depth using multiple deformable markers as structured lighting. (a) and (b) are input images, where structured light patterns of multiple deformable markers are projected on free-form objects. (c) and (d) render point clouds at different viewpoints, which are estimated via stereo using our structured light patterns recognized by our detector.
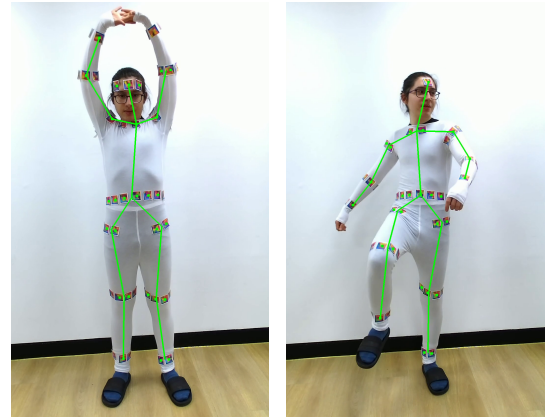
(a) Motion capture frame #1   (b) Motion capture frame #2
Fig. 24. Application of human motion capture using our markers.

that we triangulate through a stereo camera. For each estimated label points, we calculate the joint points in 3D by averaging the label points in same joint group. Figure 24 shows that our markers provide successful pose estimation.

6.3 Augmented Reality Rendering

When the fiducial marker sizes and camera intrinsic parameters are known, relative marker pose can be estimated. While this application is already allowed by existing markers, they fail under extreme imaging conditions such as motion blur and dark illumination. Especially in the video setting, where the camera and markers are moving with changing illumination, our method provides smooth and consistent detection of the markers. Figure 25 shows that, in the presence of motion blur, the virtual object is placed more reliably thanks to the robust detection performance of our method. We provide qualitative comparison with AprilTag in the supplementary video. Also, we provide quantitative results of the video in Table 7.

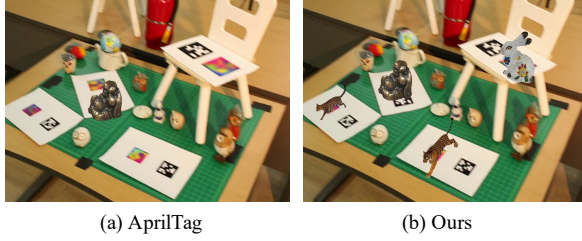|	
(a) AprilTag	(b) Ours

Fig. 25. Application of augmented reality rendering of virtual objects on the detected fiducial markers. (a) shows AR rendering results using AprilTag. In this video frame, fiducial markers are failed to be detected due to strong motion blur by shaking the camera. Only one marker is detected successfully among four markers positions. (b) shows the rendering result using our markers. Our AR rendering does not flicker over frames, showing all four virtual objects successfully.

Table 7. Quantitative detection rates of the supplementary video.

Experiment	Time	AprilTag [2016]	Ours
Smooth camera motion	01:30-01:38	99.94%	100.00%
Dynamic camera motion	01:38-01:55	76.11%	99.83%
Dim illumination	01:56-02:13	89.81%	99.70%
Halogen illumination	02:14-02:19	98.40%	100.00%
Deformed markers	02:19-02:44	44.21%	92.58%

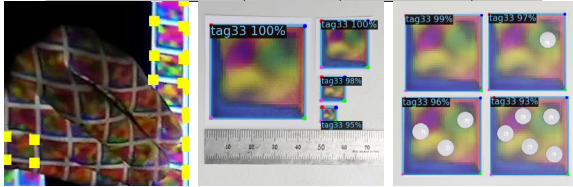| (a) Color background	(b) Small resolution	(c) Occlusion

Fig. 26. Failure cases of our method with markers projected on a hand, captured in a very small resolution, and occluded by front objects. The numbers in (b) and (c) indicate the confidence levels of the decoded messages.

7 DISCUSSION

Colored marker. The capability of producing a large number of deformable fiducial markers constitutes the core of our method. It is possible by using the 3-channel colored markers. However, there is a tradeoff when using the colored markers. Color appearance and contrast of the captured markers depends on illumination condition. Therefore, achieving the robust performance of color fiducial markers requires strong augmentation when training the network. Our results verify that our color fiducial marker system can outperform monochromatic systems. However, we do not intend robustness against extreme color changes in this work as color channels are designed to embed information to the markers. We observe that markers perform well under the hue-shift levels within the range of the training hue shift (Figure 16), but it could fail when markers are projected or printed on colored surfaces (Figure 26a).

Error detection. Error detection mechanism has been used for verification of decoded messages by checking the sanity of the inferred messages with cost of the messaging resource. During real-time inference, we reject a fiducial marker's identification that is lower than 80% match of the 36-bit dimension of the binary feature vector. As we minimize the domain gap successfully in training encoder/decoder networks and achieve the highest AP scores and competitively low false positives, our system does not include a sophisticated error detection scheme. This reveals itself rarely with

small markers with wrongly detected bits (Figure 26b). However, the robustness of marker identification could be improved further by additionally plugging in an existing error detection mechanism (e.g., minimum distance coding, parity bit, checksum, etc.) to our framework, following the current coding systems [Munoz-Salinas 2012; Olson 2011].

Learning with augmentation. During our training stage, we notice that sometimes the produced data is too difficult or impossible to detect for the algorithm. This affects training stability, and sometimes causes exploding loss and gradients. To prevent this, we fine-tune our parameters to produce a dataset, such as minimum and maximum strength values for the augmentations, minimum and maximum marker size in image space for rendering, and specular intensity range.

Specular reflection. For the specular reflection intensity, we calculate a maximum value that can make brightest point on the board to barely saturate after added to the diffuse component. We also select a random power within the range of zero to the maximum value found. This ensures that the specular component does not fully saturate the surface. Although these efforts increase stability of training, they could not fully prevent difficult edge cases to be produced. We, therefore, use adaptive loss functions. To achieve this, we keep the running statistics (mean and standard deviation) of the loss value. And we clamp the values that are bigger than $\mu + 3\sigma$ of calculated statistics. This prevents explosion of loss or gradients when edge cases are produced.

Augmentation of further imaging artifacts. If certain environmental conditions can be provided as a differentiable function to the network, such as atmospheric haze or underwater, further robustness could be achieved.

Marker occlusion. Partial occlusion of a fiducial marker is a traditional problem to achieve robust performance in detecting deformable markers. Currently, a feature that compensates form some occluded codes within a fiducial marker is missing from our current system, and sometimes degrades the successful performance of localization or decoding (Figure 26c). It is an interesting direction for our future work.

8 CONCLUSION

We have developed a novel deformable fiducial marker method through an end-to-end optimization of the marker generator and detector network via photorealistic differentiable rendering. Our method outperforms state-of-the-art detection capabilities with both synthetic and real-world image data. In particular, our method presents strength in detecting deformed fiducial markers and markers with strong motion blur. In addition, the number of messages that can be embedded in a marker is significantly larger than existing methods. It allows us to detect a large number of multiple markers simultaneously in real-time, enabling novel applications such as structured light 3D imaging and human motion capture. We anticipate that our method can provide a more robust performance of augmented reality rendering with various real-world imaging scenarios.

ACKNOWLEDGMENTS

REFERENCES

Shumeet Baluja. 2017. Hiding images in plain sight: Deep steganography. In *The Conference and Workshop on Neural Information Processing Systems*. 2069–2079.

Ross Bencina, Martin Kaltenbrunner, and Sergi Jorda. 2005. Improved topological fiducial tracking in the reactivision system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 99–99.

Filippo Bergamasco, Andrea Albarelli, Emanuele Rodola, and Andrea Torsello. 2011. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 113–120.

Joseph DeGol, Timothy Bretl, and Derek Hoiem. 2017. ChromaTag: a colored marker and fast detection algorithm. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 1472–1481.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 248–255.

Denso Wave. 1994. Quick Response (QR) code. https://d1wqtxts1xzle7.cloudfront.net/51791265/Three_QR_Code.pdf

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2018. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 224–236.

Jean Duchon. 1977. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In *Constructive Theory of Functions of Several Variables*, Walter Schempp and Karl Zeller (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 85–100.

Mark Fiala. 2005. ARTag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2. IEEE, 590–596.

John G Fryer and Duane C Brown. 1986. Lens distortion for close-range photogrammetry. *Photogrammetric engineering and remote sensing* 52, 1 (1986), 51–58.

Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292.

Sergio Garrido-Jurado, Rafael Munoz-Salinas, Francisco José Madrid-Cuevas, and Rafael Medina-Carnicer. 2016. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition* 51 (2016), 481–491.

Oleg Grinchuk, Vadim Lebedev, and Victor Lempitsky. 2016. Learnable visual markers. In *The Conference and Workshop on Neural Information Processing Systems*. 4143–4151.

Jamie Hayes and George Danezis. 2017. Generating steganographic images via adversarial training. In *The Conference and Workshop on Neural Information Processing Systems*. 1954–1963.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2961–2969.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

Danying Hu, Daniel DeTone, and Tomasz Malisiewicz. 2019. Deep charuco: Dark charuco marker pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8436–8444.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. In *The Conference and Workshop on Neural Information Processing Systems*. 2017–2025. http://papers.nips.cc/paper/5854-spatial-transformer-networks

Neil F Johnson and Sushil Jajodia. 1998. Exploring steganography: Seeing the unseen. *Computer* 31, 2 (1998), 26–34.

Jan Kallwies, Bianca Forkel, and Hans-Joachim Wuensche. 2020. Determining and Improving the Localization Accuracy of AprilTag Detection. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8288–8294.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hk99zCeAb

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4401–4410.

Hirokazu Kato and Mark Billinghurst. 1999. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. IEEE, 85–94.

Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. 2019. Flexible Layouts for Fiducial Tags.. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1898–1903.

Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. 2019. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2117–2125.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the European conference on computer vision (ECCV)*. Springer, 740–755.

Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*. 85–100.

Rafael Munoz-Salinas. 2012. Aruco: a minimal library for augmented reality applications based on opencv. *Universidad de Córdoba* (2012).

Leonid Naimark and Eric Foxlin. 2002. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE, 27–36.

Gaku Narita, Yoshihiro Watanabe, and Masatoshi Ishikawa. 2016. Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker. *IEEE transactions on visualization and computer graphics* 23, 3 (2016), 1235–1248.

Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 3400–3407.

OpenCV. 2020. Open Source Computer Vision Library. https://opencv.org/. Version 4.2.0.

John Peace, Eric Psota, Yanfeng Liu, and Lance C. Pérez. 2020. E2ETag: An End-to-End Trainable Method for Generating and Detecting Fiducial Markers. In *British Machine Vision Conference (BMVC)*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *The Conference and Workshop on Neural Information Processing Systems*. 91–99.

Francisco J Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. 2018. Speeded up detection of squared fiducial markers. *Image and vision Computing* 76 (2018), 38–47.

Matthew Tancik, Ben Mildenhall, and Ren Ng. 2020. Stegastamp: Invisible hyperlinks in physical photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2117–2126.

Weixuan Tang, Shunquan Tan, Bin Li, and Jiwu Huang. 2017. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters* 24, 10 (2017), 1547–1551.

Hideaki Uchiyama and Eric Marchand. 2011. Deformable random dot markers. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 237–238.

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. *Rendering techniques* 2007 (2007), 18th.

John Wang and Edwin Olson. 2016. AprilTag 2: Efficient and robust fiducial detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4193–4198.

Eric Wengrowski and Kristin Dana. 2019. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1515–1524.

Pin Wu, Yang Yang, and Xiaoqiang Li. 2018. Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet* 10, 6 (2018), 54.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Anqi Xu and Gregory Dudek. 2011. Fourier tag: A smoothly degradable fiducial marker system with configurable payload capacity. In *Canadian Conference on Computer and Robot Vision*. IEEE, 40–47.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5505–5514.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*. 657–672.