Technical Appendix

Anonymous submission

Implementation Detail

In this section, we introduce the detailed methods of linearizing the projection distortion and the additional experiment result about the transferability of adversarial examples.

Proof of the direction constraint

Firstly we assume the z-axes of the world and camera coordinate system coincide (otherwise, we rotate the point cloud to make their z-axes coincide). Then we have the following theorem.

Theorem 1. When the camera and world coordinate systems' z-axes coincide, only perturbing on the z-axis has a very small influence on the corresponding pixel coordinate.

Proof. Because the z-axis of the world coordinate system and camera coordinate system coincides, the point's coordinate in the camera coordinate system can be deduced by

$$\begin{bmatrix} x_c \\ y_c \\ z'_c \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z + \Delta z \end{bmatrix} + T \quad (1)$$

According to the camera imaging model, the transition matrix from camera coordinate to image coordinate is

$$z'_c \begin{bmatrix} u'_c \\ v'_c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_c & 0 & 0 \\ 0 & f_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z'_c \end{bmatrix}. \quad (2)$$

where (u'_c, v'_c) is the point's corresponding pixel coordinate in I_c after the perturbation. f_c is the focal length, dx and dy are the pixel's physical size. Therefore, when $|\Delta z| < 10^{-2}|z|$, $u_c < 100, v_c < 100$, we have

$$u'_c = (\frac{f_c x_c}{dx})/(z + \Delta z) + u_0 \approx u_c \pm 1 \quad (3)$$

$$v'_c = (\frac{f_c y_c}{dy})/(z + \Delta z) + v_0 \approx v_c \pm 1 \quad (4)$$

Therefore, when we only add small perturbations on the z-axis, the pixels' coordinates in modulated images are almost uninfluenced. □

When the camera and world coordinate systems' z-axes do not coincide, we can suppose the perspective projection matrix is $A_c = K[R\ T]$. Then we just need to rotate the original point cloud by $P' = K \cdot R \cdot P$, then add noises on the z axis. The adversarial point cloud is

$$P^* = R^{-1}K^{-1}[K \cdot R \cdot P + [0, 0, \delta]]$$

The pixel coordinate is

$$\begin{aligned} z^*[x^*, y^*, 1]^T &= A_c \cdot [P^*, 1]^T \\ &= K[R\ T][P^*, 1]^T \\ &= K[R\ T][R^{-1}K^{-1}[KRP^T + [0, 0, \delta]^T]^T, 1]^T \\ &= [[KRP^T + [0, 0, \delta]^T] + KT]^T \\ &= A_c P^T + [0, 0, \delta]^T \\ &= (z + \delta)[\frac{z}{z + \delta}x, \frac{z}{z + \delta}y, 1]^T \end{aligned} \quad (5)$$

, when $\frac{z}{z+\delta} \approx 1$, only the depth is changed and the pixel coordinates are unchanged.

Modeling the projector distortion

In some literature, the distortion is modeled by an exponential function (Zhang 2015), which is formulated by $G_{actual} = G_{org}^{\gamma}$. In this case, we only need to estimate the γ parameters, then the pre-correction function of gray-scale distortion can be formulated by $G_{corr} = G_{org}^{1/\gamma}$. In the experiment, we find that using an exponential function may introduce small errors while using a cubic polynomial function can better formulate the distortion. Therefore, we suppose the distortion function is a cubic polynomial function $I_c = f(I_p) = aI_p^3 + bI_p^2 + cI_p + d$. We use an intermediate transformation $g(I_p)$ to linearize the distortion. That is, $I_c = f(g(I_p)) = \alpha I_p + \beta$. To simplify the calculation process, we propose a simple algorithm to find the intermediate pre-correction transformation. We firstly project a group of uniform gray-scale images with gray-scale G_{org}, and calculate the average gray-scale of human faces G_{actual} in the captured images. The algorithm of finding the intermediate pre-correction transformation is shown in Alg.1.

The fitting result is shown in Fig.1. The red line is the relationship between the original gray value and the actual gray value. The blue line is the ideal responding function. The green line is the pre-correction function, and the purple line is the linearized gray distortion function. The green circle

Algorithm 1: Find the pre-correction transformation

Input: Original gray-scale value G_{org} and captured actual gray-scale value G_{actual}.
Output: The pre-correction function $G_{corr} = g(G_{org})$
Initial: $g \leftarrow \min G_{org}, G_{corr} \leftarrow [\,]$.

1: Fit $G_{actual} = f(G_{org})$ by a cubic polynomial function f.
2: Find a linear function $y = \alpha x + \beta$ that pass through $(\min G_{org}, f(\min G_{org})$ and $(\max G_{org}, f(\max G_{org})$.
3: Find the corresponding pre-correction grayscale value:
4: **while** $g \leq \max G_{org}$ **do**
5: $\quad g_{real} \leftarrow \alpha \times g + \beta$
6: $\quad$ Solve g_{corr} in function $g_{real} = f(g_{corr}),\ s.t.\ g_{corr} \in [0, 255]$.
7: $\quad G_{corr}.append(g_{corr})$.
8: $\quad g \leftarrow g + 1$;
9: **end while**
10: Fitt $G_{corr} = g(G_{org})$ using a cubic polynomial function.
11: **return** g

in the graph is the pre-corrected gray value corresponding to the original gray value shown in the red circle.

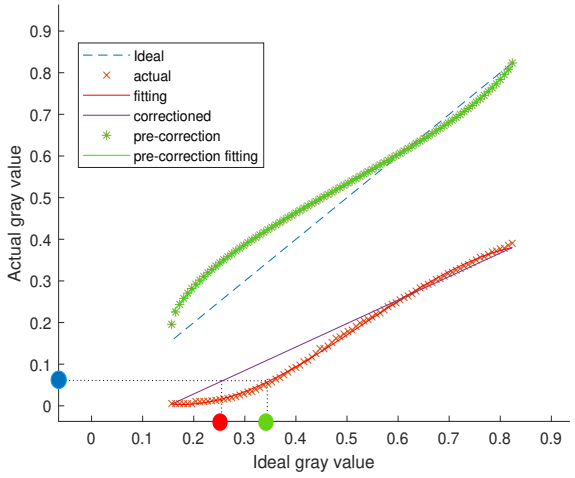

Figure 1: Gray scale value distortion result. The red line is the projector's responding function. The green line is the pre-corrected gray value.

We can also use tanh function to fit this distortion, as shown in Figure 2. We use MATLAB's *fit* function to get the hyperparameters.

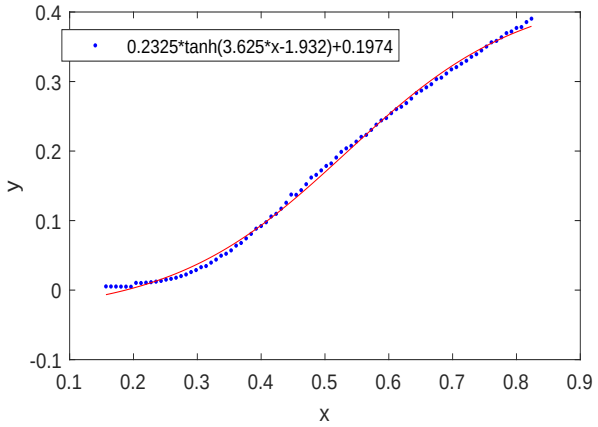

Figure 2: Fitting the projector distortion through tanh function.

Visualization of sensitivity maps

We visualize some sensitive maps in Figure 4. As we can see, the picture is highly sensitive in central and high-variance areas, which is consistent with human feeling.

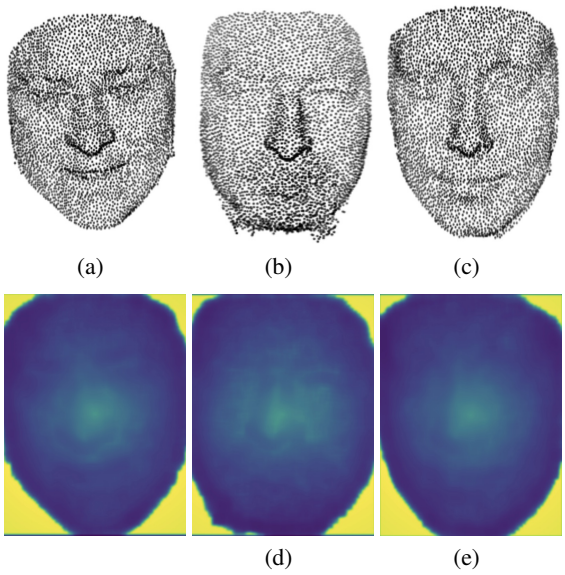

Figure 3: The face point cloud and sensitivity maps. The first row: the point cloud. The second row: the sensitivity maps

Visualization of the phase superposition attacks

As shown in Figure 4, we project adversarial perturbations onto the original faces. The perturbation is very small and hard to notice.

Transferability of adversarial examples

As mentioned in the paper, we find that 3D-TI loss can improve the transferability of 3D adversarial attacks, especially for dodging attacks. We have shown the transferability of

(a) (b) (c)

Figure 4: Visualizing the phase superposition attack. (a) the fringe image. (b) the perturbations (c) the modulated image.

dodging attacks using Chamfer+kNN distance without and with the 3D-TI module. In this subsection, we show the transferability of dodging attacks using Chamfer distance without and with the 3D-TI module. Table.1 and Table.2. We can see that transferability significantly improves.

subs. \ victim	PN	MSG	SSG	DN	CN
PointNet	1.00	0.05	0.10	0.09	0.12
PN++(MSG)	0.21	1.00	0.26	0.28	0.14
PN++(SSG)	0.25	0.43	1.00	0.32	0.23
DGCNN	0.24	0.37	0.22	1.00	0.11
CurveNet	0.25	0.37	0.38	0.40	1.00

Table 1: The transferability of the adversarial point clouds by original Chamfer loss. The horizontal column is the substitute model, the same below.

subs. \ victim	PN	MSG	SSG	DN	CN
PointNet	1.00	0.21	0.16	0.22	0.10
PN++(MSG)	0.27	1.00	0.26	0.35	0.23
PN++(SSG)	0.37	0.46	1.00	0.55	0.30
DGCNN	0.44	0.52	0.49	1.00	0.27
CurveNet	0.27	0.44	0.45	0.47	1.00

Table 2: The transferability of the adversarial point clouds by the 3D-TI Chamfer loss.

References

Zhang, S. 2015. Comparative study on passive and active projector nonlinear gamma calibration. *Applied optics*, 54(13): 3834–3841.