Creating a Forensic Database of Shoeprints from Online Shoe Tread Photos *(Supplementary Material)*

Samia Shafique[1] Bailey Kong[2] Shu Kong[3*] Charless Fowlkes[1*]
[1]University of California, Irvine [2]Ronin Institute [3]Texas A&M University
sshafiqu@uci.edu bailey.kong@ronininstitute.org fowlkes@ics.uci.edu shu@tamu.edu
https://github.com/Samia067/ShoeRinsics

Outline

We propose to create a forensic database of shoeprints by leveraging shoe-tread imagery collected by online retailers. To do so, we strive to predict depth maps for these photos, thresholding whihc generates shoeprints used to match a query print (collected from a crime scene). We propose a novel method, *ShoeRinsics*, to learn depth estimation from synthetic data (along with intrinsic components) and real data (with no annotations). *SheoRinsics* incorporates synthetic-to-real domain adaptation and intrinsic image decomposition techniques to mitigate domain gaps. We validate our method with a defined evaluation protocol that measures the degree of match between predicted depth and ground-truth shoeprints (collected in a lab environment). Results convincingly demonstrate that *ShoeRinsics* remarkably outperforms state-of-the-art methods for shoe-tread depth prediction. In this supplementary document, we discuss the following topics:

- Section 1 details the process of matching a ground-truth shoeprint to predicted depth for evaluation.
- Section 2 shows qualitative analysis of our *ShoeRinsics*. We visualize of all predictions of *ShoeRinsics* on real-FID-val images downloaded from the internet in Section 2.1 and compare to RIN [9] in Section 2.2.
- Section 3 compares performance of *ShoeRinsics* with related work for each individual image from real-val and real-FID-val.
- Section 4 provides details on our synthetic dataset generation. The process of depth map generation is described in Section 4.1 and the light environemnts used are visualized in Section 4.2.
- Section 5 describes how we photograph shoe-treads and collect their prints to create a validation set (real-val) for quantitative evaluation.

*Authors share senior authorship.

Algorithm 1 Metric of Depth-Print Matching

Input: predicted depth $\hat{X}_R^d$, ground-truth shoeprint S^*, and mask m
Initialize best-matching IoU $v_{max} = 0$
determine per-pixel local depth, $d_l = \frac{\text{blurred depth}}{\text{blurred mask}}$

for $s \in [0.1, 0.11, 0.12, \ldots, 2]$ **do**
  if IoU$(\hat{X}_R^d < sd_l, S^*) > v_{max}$ **then**
    $v_{max} = \text{IoU}(\hat{X}_R^d < sd_l, S^*)$,
    set best-matching shoeprint $S_{best} = \hat{X}_R^d < sd_l$
  end if
end for

set p_{95} = value at the 95th percentile in sorted $\hat{X}_R^d$
for $t_{nc} \in [0.1p_{95}, 0.1p_{95} + 0.01, 0.1p_{95} + 0.02, \ldots, p_{95}]$ **do**
  determine shoeprint $S_{t_{nc}} = S_{best}$ AND $(\hat{X}_R^d < t_{nc})$
  if IoU$(S_{t_{nc}}, S^*) > v_{max}$ **then**
    $v_{max} = \text{IoU}(S_{t_{nc}}, S^*), \quad S_{best} = S_{t_{nc}}$
  end if
end for

set p_{05} = value at the 5th percentile in sorted $\hat{X}_R^d$
for $t_c \in [p_{05}, p_{05} + 0.1, p_{05} + 0.2, \ldots, 30p_{05}]$ **do**
  determine shoeprint $S_{t_c} = S_{best}$ OR $(\hat{X}_R^d < t_c)$
  if IoU$(S_{t_c}, S^*) > v_{max}$ **then**
    $v_{max} = \text{IoU}(S_{t_c}, S^*)$
  end if
end for
return best-matching IoU v_{max}

- Section 6 details the architecture of each component of our *ShoeRinsics*.
- Section 7 discusses the pseudo albedo generated for real shoe-tread images and compares them to the albedo predicted by *ShoeRinsics*.

1. Depth-Print Matching in Evaluation

We get the shoeprint prediction by thresholding the predicted depth $\hat{X}_R^d$ of a shoe-tread image. However, different thresholds can produce different predicted shoeprints. So, we develop a threshold-free metric for evaluating how well

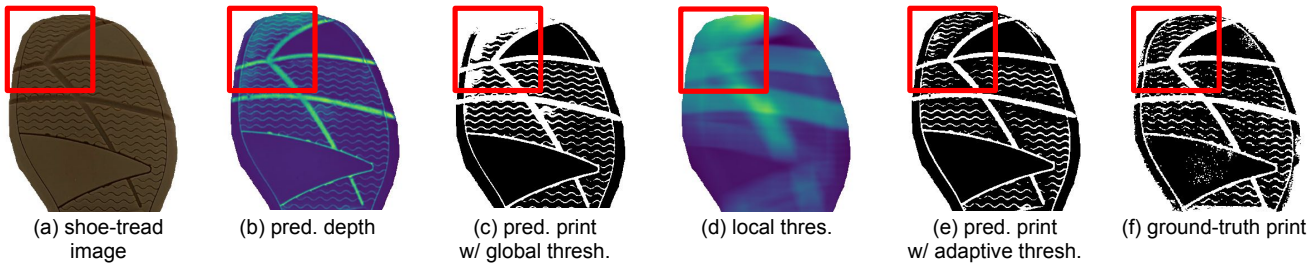

Figure 1. Effect of adaptive threshold. Given a shoe-tread image (a), we predict its depth (b). Notice that the front of the shoe is curved up slightly (highlighted by the red boxes). Thus, a global threshold fails to capture the print properly. Compare the print prediction using a global threshold (c) with the ground-truth print (f). A solution is to use an adaptive threshold instead. As such, we first blur the predicted depth to get the local mean depth d_l. Using sd_l as the local threshold (d) where s is an appropriate constant for scaling, we get predicted print (e) which is much closer to the ground-truth (f).

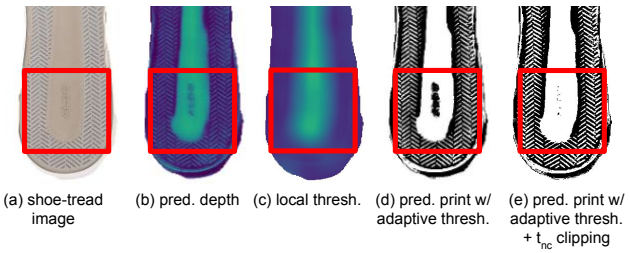

Figure 2. Effect of non-contact threshold t_{nc}. A shoe-tread image (a), the corresponding depth prediction (b), the local threshold (c), and the predicted print prediction with adaptive threshold (d) is shown. We can see that using only the adaptive threshold can cause errors in large areas of non-contact surface as shown by the red boxes. Assume non-contact surfaces have high depth values. Although the logo is correctly predicted to have a high depth value, the local threshold also happens to be high in the region and causes adaptive thresholding to undesirably predict the logo to leave a print. To correct this, we find an appropriate non-contact threshold t_{nc} for which regions where predicted depth is greater than t_{nc} does not leave a print. The resulting print is shown in (e).

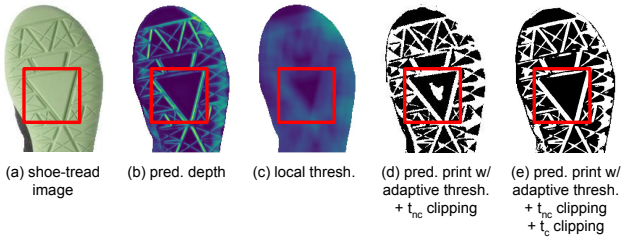

Figure 3. Effect of contact threshold t_c. We visualize a sample shoe-tread image (a), the predicted depth (b), the local threshold (c), and the print prediction after using adaptive thresholing with t_{nc} clipping (d). Parts of a large contact surface incorrectly leaves no print as shown by the red boxes because the local threshold is very low in the region (assuming contact areas have low depth value). To correct this, we determine an appropriate contact threshold t_c such that regions where predicted depth is less than t_c always leave a print. (e) demonstrates the final result.

our predicted depth matches a ground-truth shoeprint. Ideally, we want to threshold the depth prediction in a way that produces a shoeprint prediction that most closely matches the ground-truth shoeprint. We summarize our method in Algorithm 1.

Global thresholding vs. adaptive thresholding. Using a global threshold for print prediction from depth prediction is troublesome since errors can creep into regions where the shoe-tread curves upwards (for example, in the front of the shoe). Fig. 1 illustrates this scenario with a sample shoe from real-val. In such cases, although the shoe is curved upwards, it still leaves a print when someone walks wearing those shoes. This is because the weight of the person flattens out the shoe. Also, the physical motions of walking causes the curved parts to come in contact with the ground. Assuming high depth values correspond to non-contact surfaces, a portion of the shoe that curves up would have high depth values and a global threshold might incorrectly indicate that region does not leave a print. We can solve this issue by using adaptive thresholding instead.

Details of adaptive thresholding. To perform adaptive thresholding, we first determine a per-pixel local average d_l for the predicted depth. This is achieved by blurring the predicted depth $\hat{X}_R^d$ with a large square kernel of size 45×45. For comparison, our shoe-tread image and predicted depth map resolution is 405×765. We note that boundaries and invalid depth values outside the mask may cause artifacts in d_l. To negate this effect, we set $d_l = \frac{d_l}{m_l}$ where m_l is the per-pixel local average for the mask computed in a similar manner.

Next, we set our best-matching shoeprint $S_{best} = \hat{X}_R^d < sd_l$ for some scalar multiplier s. Theoretically, we want to sweep over all possible values of s and find the one which gives the highest IoU between $\hat{X}_R^d < sd_l$ and the ground-truth print S^*. Practically, we sample values from range $[0.1, 2]$ at intervals of 0.01.

Threshold t_{nc} for large non-contact regions. Although, our current best-matching shoeprint estimation is good enough for most cases, it may have issues for very large non-contact surfaces. Fig. 2 illustrates how a large

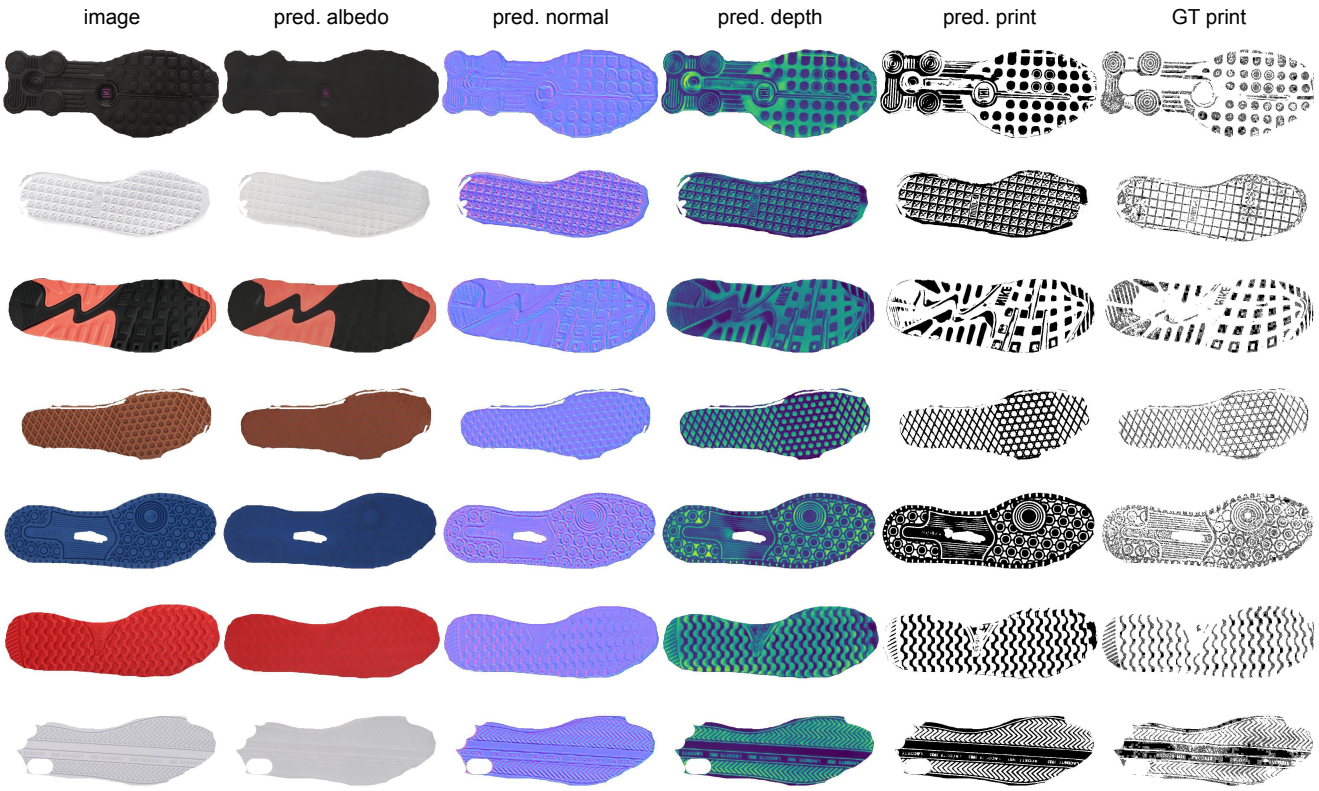

Figure 4. Visualization of predicted shoeprints, as well as intrinsics, by our *ShoeRinsics* on real-FID-val. Real-FID-val consists of images of real shoe-treads downloaded from online retailers and corresponding ground-truth shoeprints. Visually, we can see our method works quite well w.r.t both shoeprint prediction and intrinsic decomposition (albedo, normal, and depth).

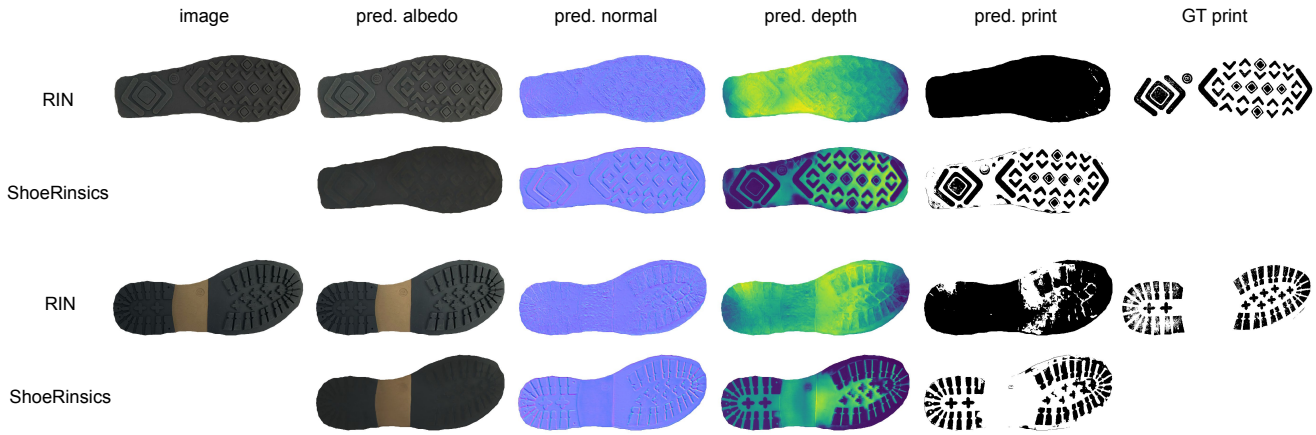

Figure 5. Qualitative comparison between RIN [9] and *ShoeRinsics* on real-val. Along with input images and ground-truth shoeprint, we show albedo, normal, depth, and print prediction. Note that RIN does not directly produce depth predictions. We obtain them by integrating their normal predictions using the well-established Frankot Chellappa algorithm [4]. As we can see, RIN produces poor quality albedo and normal predictions, presumably because it does not explicitly perform domain adaptation. The albedo predictions retain much of the shading information and the normal predictions are noisy. The substandard normal predictions in RIN lead to unsatisfactory depth and print predictions. Comparatively, *ShoeRinsics* is able to produce more likely albedo, normal, and depth predictions and thus predict shoeprints which are much closer to the ground-truth.

non-contact region can cause the local threshold sd_l to be very high. This in turn can lead to incorrectly predicting some areas to leave a print (such as the logo in Fig. 2). We fix this by identifying the threshold t_{nc} which gives the highest IoU between ground-truth shoeprint S^* and S_{best} AND $(\hat{X}_R^d < t_{nc})$ and update our shoeprint predic-

tion S_{best}. We find it sufficient to determine t_{nc} by sampling values in range $[0.1p_{95}, p_{95}]$ at an interval of 0.01 where p_{95} is the 95^{th} percentile of sorted $\hat{X}_R^d$ values.

Threshold t_c for large contact regions. A similar problem and solution apply for large contact surfaces as demonstrated in Fig. 3. In such regions, the local threshold is very low and can result in "holes" in our predicted print. Our solution is to find threshold t_c for which the IoU between ground-truth shoeprint S^* and S_{best} OR $(\hat{X}_R^d < t_c)$ is highest. We can find an adequate value for t_c by sampling numbers from range $[p_{05}, 30p_{05}]$ at intervals of 0.1 where p_{05} is the 5^{th} percentile of sorted $\hat{X}_R^d$ values.

Generic thresholds for print prediction. To determine shoeprint predictions from real images without ground-truth print, we set $s = 1$, $t_{nc} = p_{97}$, and $t_c = p_{03}$ where p_x is the x^{th} percentile of sorted $\hat{X}_R^d$ values.

2. Qualitative Results of *ShoeRinsics* on Real Shoe-Treads

In this section, we perform additional qualitative analysis of our *ShoeRinsics*. We visualize our albedo, normal, depth, and print predictions (Fig. 4) as well as compare shoeprint predictions to that of RIN [9] (Fig. 5).

2.1. Visualization of Predictions on Real-FID-val

One of the datasets we collected, real-FID-val, consists of images of real shoe-treads downloaded from online retailers with corresponding ground-truth shoeprints. We visualize our intrinsic predictions (albedo, normal, and depth) and compare print predictions to ground-truth shoeprints on the real-FID-val dataset in Figure 4. We see that the intrinsic predictions are visually pleasing and the predicted print closely resembles the ground-truth shoeprint.

2.2. Comparison with RIN

RIN [9] learns from unlabeled real images using intrinsic image decomposition. It breaks down images into albedo, normal and light. We integrate their normal predictions to obtain a depth prediction using the well-established Frankot Chellappa algorithm [4]. Thresholding this depth prediction gives us the shoeprint prediction which we compare to the ground-truth shoeprint. In Figure 5, we compare the albedo, normal, depth, and shoeprint predictions of RIN on real-val with that of *ShoeRinsics*. We find that RIN performs poorly on real shoes, presumably because it does not explicitly perform domain adaptation. Even though our focus is on the depth prediction, our albedo and normal predictions visually look better than the predictions made by RIN. Albedo predictions from RIN retain much of the shape information. More importantly, noisy normal predictions from RIN integrate to give low quality depth predictions and thus unsatisfactory shoeprint predictions.

Table 1. Comparison of Intersection over Union (IoU) values achieved by *ShoeRinsics* and related work on each example in real-val. The best IoU per shoe example is written in bold and the second-best is underlined. We can see that *ShoeRinsics w/ aug* is the clear winner while *ShoeRinsics* is the second-best.

Shoe ID	RIN [9]	ADDA [10]	UDAB [6]	CyCADA [7]	ShoeRinsics	ShoeRinsics w/ aug
0001-L	25.7	29.9	34.1	38.8	41.4	**43.5**
0001-R	24.2	22.8	29.0	33.0	34.3	**36.8**
0002-L	27.8	29.3	27.8	**35.5**	29.6	32.4
0002-R	26.4	28.8	28.5	**36.9**	33.7	35.0
0003-L	20.3	24.5	30.2	32.3	37.7	**37.9**
0003-R	21.4	22.7	25.8	28.7	33.0	**33.7**
0004-L	26.3	31.0	31.0	38.0	39.6	**39.7**
0004-R	23.3	28.5	30.1	32.6	**37.5**	35.9
0005-L	29.2	50.4	56.0	59.2	**60.5**	59.9
0005-R	27.1	54.3	60.0	56.6	**62.4**	60.6
0006-L	31.5	36.2	36.3	37.7	42.2	**43.2**
0006-R	30.0	34.7	36.0	38.5	38.0	**40.4**
0007-L	19.6	19.3	19.4	17.1	22.8	**63.3**
0007-R	23.1	24.0	24.0	22.6	32.5	**42.4**
0009-L	48.8	48.8	48.9	**58.4**	55.8	49.7
0009-R	47.3	50.5	48.9	55.6	**56.1**	47.9
0010-L	52.5	56.2	54.8	56.4	**66.6**	61.0
0010-R	46.8	49.4	46.5	**53.5**	53.2	52.9
0011-L	22.4	24.6	25.0	25.0	25.0	**25.1**
0011-R	25.7	**28.9**	28.2	**28.9**	28.5	28.8
0012-L	32.1	77.5	68.2	75.8	76.2	**83.3**
0012-R	30.5	72.3	69.6	72.5	69.1	**76.9**
0013-L	24.2	**35.7**	31.7	31.5	31.2	32.8
0013-R	27.5	**40.9**	36.6	38.2	37.9	39.0
0014-L	13.6	23.9	21.2	18.8	27.7	**31.4**
0014-R	15.3	29.4	29.0	22.1	32.4	**37.8**
0015-L	24.9	**36.8**	29.8	35.0	35.3	34.4
0015-R	27.4	45.5	41.1	41.8	53.1	**53.2**
0016-L	21.7	63.4	61.1	66.0	63.9	**68.6**
0016-R	21.4	61.3	60.5	65.3	63.6	**67.8**
0017-L	24.8	47.6	47.3	56.3	55.3	**57.1**
0017-R	26.4	47.8	54.2	**60.2**	59.2	57.3
0018-L	34.8	35.3	37.3	46.4	**51.8**	50.6
0018-R	37.9	38.6	38.0	48.9	**54.1**	52.8
0019-L	66.4	69.2	72.0	73.5	71.4	**75.4**
0019-R	66.1	68.7	73.2	75.2	72.4	**76.0**
Average	30.4	41.4	41.4	44.8	46.8	**49.0**

3. Further Details of Quantitative Analysis

We compare methods using our defined metric based on Intersection over Union (IoU). We analyse the IoU values for each of the shoe examples in real-val (Table 1) and real-FID-val (Table 2) to further demonstrate that *ShoeRinsics* outperforms the state-of-the-art domain adaptation and intrinsic image decomposition methods. We can see from the Tables that *ShoeRinsics w/ aug* performs the best, followed by *ShoeRinsics* in the second position.

4. Synthetic Data Preparation

To train our model, we need shoe-sole images with paired ground-truth albedo, depth, normal and light information. Publicly available datasets that include shoe objects (among other categories) [1] either do not focus on the shoe-sole and/or do not provide full decomposition into shape, albedo, and lighting. Thus, we introduce our own synthetic dataset, syn-train. For this purpose, we synthesize depth maps, albedo maps, and lighting environments. We observe

Table 2. Comparison of Intersection over Union (IoU) values achieved by *ShoeRinsics* and related work on each example in real-FID-val. The best IoU per shoe example is written in bold and the second-best is underlined. We can see that *ShoeRinsics w/ aug* is the clear winner while *ShoeRinsics* is the second-best.

Shoe ID	RIN [9]	ADDA [10]	UDAB [6]	CyCADA [7]	ShoeRinsics	ShoeRinsics w/ aug
1	**33.0**	28.5	28.5	30.1	31.8	32.6
3	18.8	18.8	26.7	**35.9**	33.7	33.4
4	20.3	21.6	27.7	29.2	27.9	**29.7**
5	23.8	24.1	26.9	28.3	**29.9**	29.5
8	11.9	14.4	18.3	**21.4**	19.3	20.7
9	21.0	20.6	27.3	31.5	31.8	**32.3**
10	15.8	21.5	16.4	**23.8**	22.2	22.7
11	25.5	32.1	34.3	34.0	35.1	**36.5**
12	30.7	29.5	27.7	31.5	**32.5**	32.4
13	33.3	30.1	32.1	34.0	33.2	**34.4**
16	28.9	30.6	37.7	**52.4**	51.7	51.9
17	24.8	22.5	33.0	**35.9**	29.9	29.6
23	28.3	29.0	30.7	**32.8**	31.0	31.9
32	36.0	43.0	42.6	**47.0**	46.3	46.5
33	28.3	28.3	28.0	27.3	29.4	**30.0**
35	34.3	40.5	40.1	40.6	40.6	**41.2**
45	31.2	30.8	33.3	32.8	36.9	**37.2**
47	24.8	24.1	24.0	24.0	**24.9**	**24.9**
53	11.7	11.8	11.7	12.1	**13.7**	13.1
54	22.0	22.0	22.0	22.1	**29.1**	29.0
55	30.6	28.7	29.4	30.1	31.6	**31.8**
56	19.0	**19.3**	19.2	19.1	19.0	19.0
62	33.6	33.9	36.3	36.8	**38.1**	**38.1**
72	28.2	28.1	28.2	**29.0**	28.2	28.2
74	32.8	34.1	35.5	34.7	36.3	**36.8**
82	44.2	36.8	41.3	45.5	45.3	**45.7**
1040	42.0	37.8	40.3	45.4	46.0	**47.1**
1041	27.6	36.7	35.5	35.3	38.0	**38.1**
1044	19.5	20.7	20.6	21.5	23.9	**24.3**
1047	29.1	29.2	30.2	31.2	**31.4**	**31.4**
1048	23.5	27.7	28.5	28.1	28.0	**28.6**
1049	26.7	21.6	25.3	27.7	27.8	**28.2**
1050	26.4	25.4	26.4	26.1	**26.5**	26.4
1058	24.2	36.5	**38.8**	35.2	36.7	38.0
1062	19.0	21.0	23.5	25.6	**29.1**	28.6
1064	18.0	20.7	23.4	21.6	24.4	**24.6**
1071	11.7	16.5	19.2	**19.8**	18.3	18.3
1076	24.3	30.0	30.1	33.6	**34.5**	33.6
1079	26.4	28.4	29.4	29.3	31.3	**31.4**
1088	29.2	27.4	29.9	33.5	34.0	**34.2**
1095	26.8	29.7	28.7	37.7	38.1	**41.4**
Average	26.0	27.2	29.0	31.1	31.6	**32.0**

that commercial shoe tread photographs are taken under very diffuse lighting conditions where the primary variations in surface brightness are driven by global illumination effects rather than surface normal orientation (e.g., grooves appear darker). This necessitates the use of a physically-based rendering engine [8] rather than simple local shading models. We discuss details of depth map generation in Section 4.1 and visualize the different light environment maps in Section 4.2

4.1. Depth Map Generation

We use an existing shoeprint dataset [11] collected in a controlled lab environment. Sample shoeprints are displayed in Fig. 6. We convert the 2D shoeprints to 3D depth maps by adding fake depth values to each point on the print. We generate 10-15 different depth maps from each of the

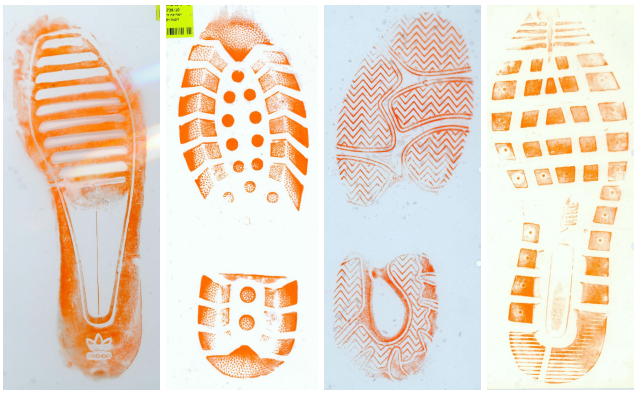

Figure 6. Examples from an existing shoeprint dataset [11]. We use these prints as a starting point for synthetic depth map generation. Note that even though these are shoeprints collected under a controlled lab environment, the images are still quite noisy. This necessitates some preprocessing before these can be used for synthetic depth map generation.

387 shoeprints available in [11]. Fig. 7 highlights major steps in depth map generation from shoeprint images. Details of depth map generation is provided below.

Removing noise. Raw shoeprint data is noisy (as shown in Fig 6). We employ two tricks to filter noise. First, we compute a mask for the shoeprint to remove notes and dirt in the background from consideration. Given that the shoeprints are orange colored and the background does not contain any of that color, we determine the mask using the concave hull of the orange colored regions. Second, we filter noise by applying a Gaussian blur followed by a sigmoid function on the gray-scale shoeprint.

Adding a realistic touch. At this stage, our depth map mainly consists of the two extreme values representing contact and non-contact surfaces. To incorporate some texture, we add a moderated amount of high frequency details (obtained from subtracting the blurred depth from the original gray-scale shoeprint). Next, we optionally add slanted bevels to our depth map to make the tread blocks look more natural. We further add a local curvature to the non-contact surfaces to give them some dimension. Essentially, we square the euclidean distance transform of the depth image and add the smoothed out result to our depth map. Finally, we also add a global curvature along the edge of the shoe-tread to attain the natural upward curvature that is common in many shoes.

4.2. Visualization of Light Environments

We display the different light environments in our synthetic dataset (syn-train) in Figure 8 and also provide a video to better visualize the lighting effects. We have a total of 17 different light environments in our dataset. One consists of diffuse white light. Eight light configurations consist of a white light bulb providing directional light (from

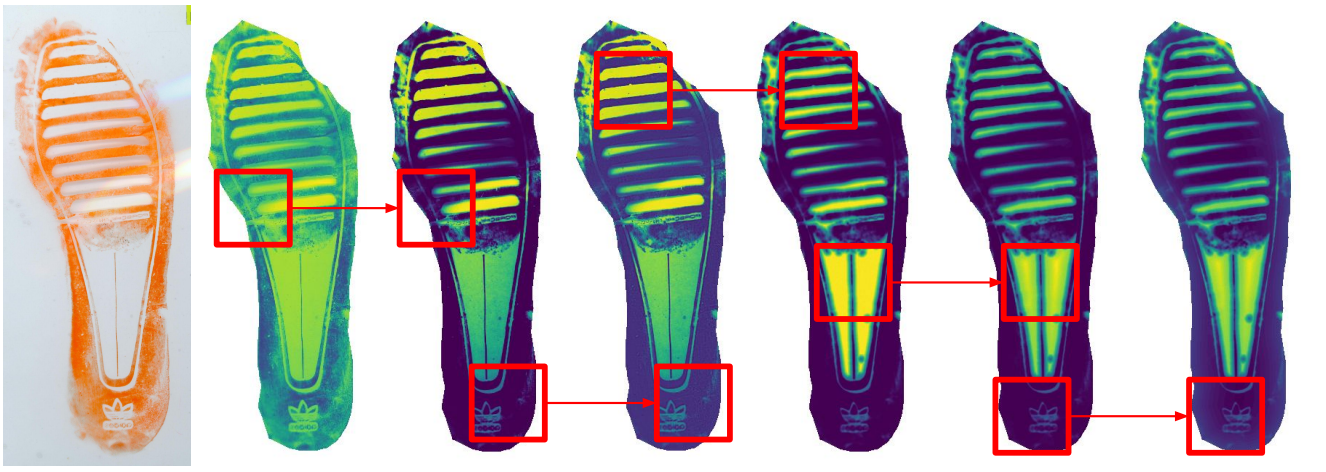

Figure 7. We illustrate major steps in depth map generation with an example. We first filter noise in a shoeprint image (a) by masking out the background (b) and applying a Gaussian blur followed by a sigmoid function (c) on the shoeprint image. Then, we add in some moderated amount high frequency details from the shoeprint image as textures (d). To make our depth maps more realistic, we optionally add slanted bevels (e), local curvatures for non-contact surfaces (f), and a global curvature along the edge of the shoe-tread (g).

8 different directions) in addition to the diffuse white light. The other eight light configurations consists of two white light bulbs at a 120° angle to each other in addition to the diffuse white light. For each light configuration, we visualize a shiny sphere in place of the shoe demonstrating the light placements. Additionally, we render an example synthetic shoe under all the different light conditions to show the effect of each of them. We see different light environments create different shadows on on the shoe-tread blocks.

5. Real Data Preparation for Evaluation

To quantitatively evaluate and compare methods, create real-val which consists of paired shoe-tread images and ground-truth prints for real shoes.

Photographing shoe-treads. Real-val contains new-athlectic, used-athletic, and new-formal shoes. The new shoes are collected from thrift stores which often sell new or very lightly used shoes. The used shoes are worn-out athletic shoes donated by volunteers. We first clean all the shoe-treads with soap and water and let them dry. Next, we photograph the shoe-treads in a brightly lit environment similar to that of a professional photography setting. We put together 5 square light panels to create a light box and place the shoe on a holder inside the light box. We also illuminate the shoes using a ring light on top.

Preparing ground-truth shoeprint. After photographing the shoes we proceed to collecting their prints. We use a process called *block printing technique* which is widely used in forensics to collect lab shoeprint impressions [2]. With the shoe resting on the holder, we paint the shoe-tread with a thin layer of relief ink using a roller. Forgoing the roller and simply using a paint brush would cause ink blobs to get stuck in the nooks and crannies of the shoe-tread leading to blotchy prints. While the ink is still wet, we quickly press a slightly absorbent white paper onto the shoe-tread using a roller. The use of the roller distributes pressure throughout the paper and thus produces more uniform prints. We collect 2-3 sets of prints for each shoe, each time painting the shoe with a new layer of ink. Notice how these individual prints are not identical and contain areas of uneven coverage. To get a smoother result, we align all the prints to the shoe-tread image (using thin-plate spline [3] and point correspondences between the shoe-tread image and the collected prints) and average them. The average is a more complete and evenly colored print. Finally, the average print is thresholded to get our binary ground-truth shoeprint.

6. Further Implementation Details

Decomposer $\mathcal{F}$. Our decomposer consists of a classic encoder-decoder structure with skip connections. We use separate decoders for albedo, normal, depth, and light predictions. All of the encoded input is passed to each of these decoders. The light decoder consists of residual blocks followed by a final convolution layer which outputs 17 numbers representing the probability of predicting the 17 light types in our synthetic training dataset. We use the output of the second last layer of the albedo, normal, and depth decoder as the corresponding features. For light features, we use the 17 light probabilities.

Renderer $\mathcal{R}$. The renderer has a mirrored structure as the decomposer. It has separate encoders for albedo, depth,

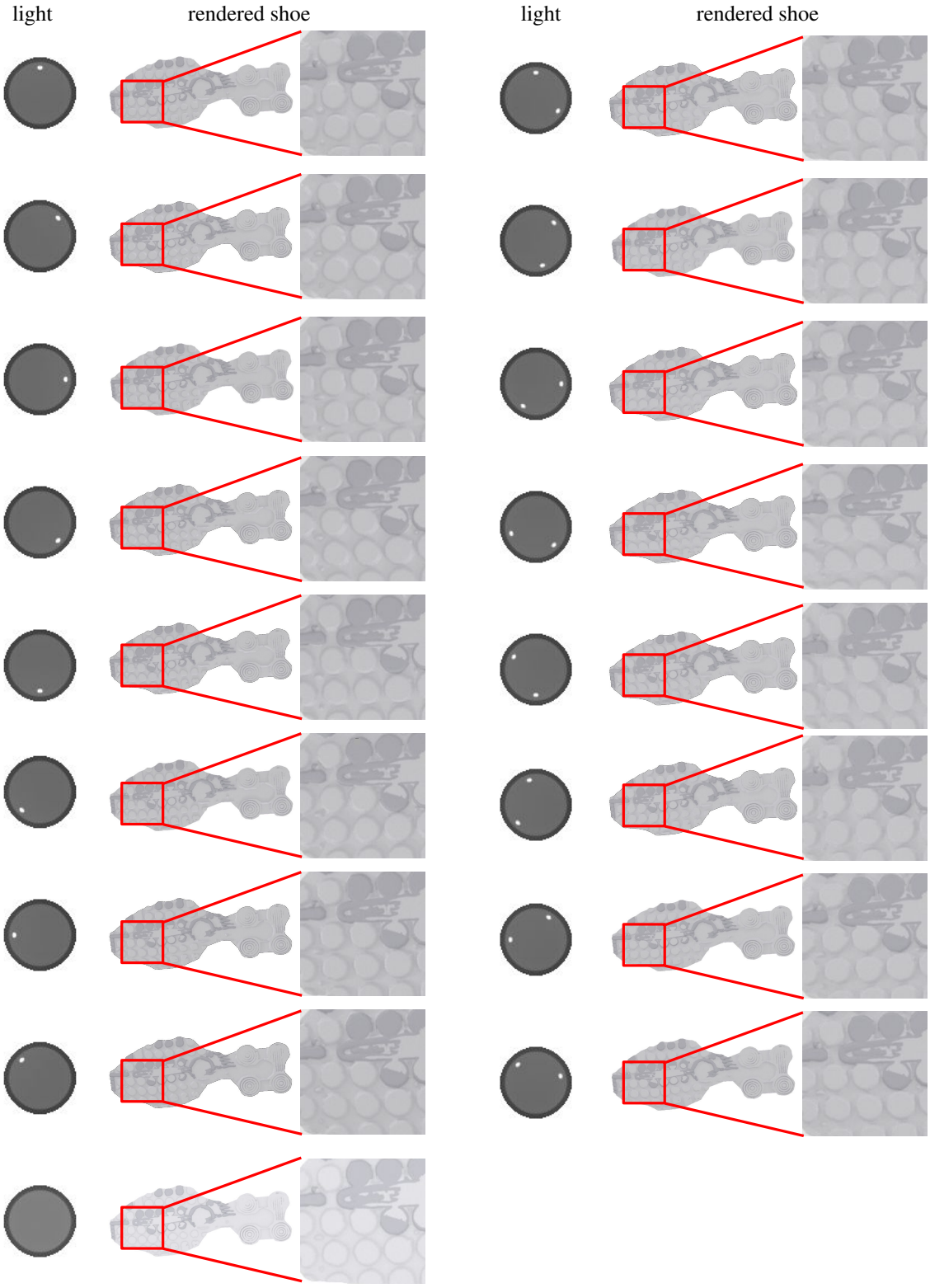

Figure 8. Visualization of the 17 different light types in our synthetic dataset (syn-train). We show a shiny sphere representing the light in the environment and a shoe rendered under that light condition. Our light environments consist of diffuse white light in addition to 0 to 2 light bulbs for directional light. Different light sources produce different shadows on the shoe-tread blocks. Please refer to the attached video for a better visualization.

normal, and light. The light encoder takes in a one-hot array representing the light configuration. The encoded information from each of the encoders is concatenated and passed to the decoder which predicts the synthetic or real shoe-tread image.

Connecting the decomposer and renderer. When passing decomposer outputs to the renderer in our main pipeline, we ensure that the decomposer outputs look similar to the synthetic albedo, depth, normal, and light used to train the renderer. We set the background (i.e., parts outside the shoe-tread) pixel values to 1 in the albedo, depth, and normal predictions. We also use the Hard-Gumble trick to represent the light predictions as one-hot vectors instead of fractional probabilities for the renderer. This ensures that a path exists for gradient back-propagation through the light decoder while providing a one-hot representation for the light probabilities.

Image translators $\mathcal{G}_{S\to R}$ and $\mathcal{G}_{R\to S}$. We use a ResNet backbone for the image translators. The two generators have the exact same structure. They consist of 2 convolution layers with stride 2, followed by 9 residual blocks, and finally 2 convolution layers coupled with nearest 2D upsampling layers with a scale factor of 2. The convolution layers and residual blocks in the generators are interspersed with batch normalization and the leakyReLU activation function.

Image discriminators $\mathcal{D}_S$ and $\mathcal{D}_R$. The discriminators used to learn image translation are PatchGANs and consist of 4 convolution layers with stride 2 followed by 2 convolution layers with stride 1. Similar to the image translators, the discriminators also have batch normalization and the leakyReLU activation function interspersed among the convolution layers and residual blocks.

Feature discriminator $\mathcal{D}_{feat}$. The discriminator for feature alignment takes in the concatenation of the albedo, normal, and depth features as one input, and the light features as a second input. These features are processed in two separate branches and the results are concatenated in the final output. Each of these branches consist of 3 convolution layers. The branch for the albedo, normal, and depth features uses a kernel size of 3 (to encode some context), while the branch for light features use a kernel size of 1.

7. Discussion on the Pseudo Albedo

We provide pseudo supervision on the albedo prediction of real images. Fig. 9 shows examples of pseudo albedo and the albedo predictions made by *ShoeRinsics* on real shoe-tread images. The following is a discussion on pseudo albedo generation and how pseudo albedo differs from predicted albedo.

7.1. Pseudo Albedo Generation

Creating pseudo albedo segments. We first group the pixels in the real image using the mean-shift algorithm [5].

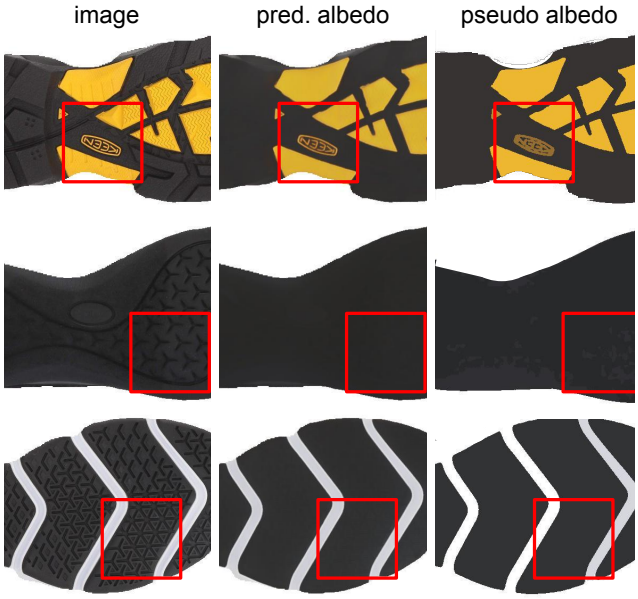

Figure 9. Visualization of difference between predicted albedo and pseudo albedo. Given that the albedo for shoe-treads consists mostly of piece-wise constant segments, we use the mean shift clustering algorithm [5] to determine pseudo albedo. *ShoeRinsics* learns to predict albedo for real shoe-treads using the psuedo albedo as ground-truth. We do not use pseudo albedo directly instead of the albedo prediction because it is not perfect ground truth and contains deviating segment boundaries (row 1), over-segmentation (row 2), and incorrect albedo labels for segments (row 3). Our *ShoeRinsics* learns to fix these errors.

To generate the pseudo albedo labels, we work with the LAB color space since it is easier to distinguish hue (A and B) from brightness (L) in this color space. Additionally, to ensure that shading does not interfere with pixel grouping, we scale the L channel by a factor of 0.15. Note that ignoring L altogether would make it difficult to distinguish between black and white. It turns out that we do not need to work on full resolution images for pixel grouping. So, we first downsize our real-shoe images to 67×150 for faster computation. After running mean-shift on the resulting shoe-tread pixels, we get an initial segmentation of the pixels in the real image. We define the color of each segment as the average color across that segment.

Refining pseudo albedo segments. The initial segmentation is very grainy as expected. Thus, we proceed to iteratively refine the segments for a maximum of 10 iterations. For each iteration we merge 'nearby' segments and update the color of the segments to reflect the segment updates. To merge segments, we find segments which are small in size and close to another segment both physically (share segment boundary) and numerically (have similar segment color). After merging segments, we update the color of the resulting segment as the average color across all the pixels in the new segment. We break the iterative refinement loop

when we reach an iteration where the segmentation does not receive any updates or when the maximum iteration count (10) has been reached. Since this is a time-consuming process, we predetermine the pseudo albedo for all real shoes and save them to be used directly during training.

7.2. Comparing Pseudo Albedo to Predicted Albedo

It may seem counter-intuitive to learn to predict albedo when we can simply determine the corresponding ground-truth pseudo albedo. However, as we can see in Fig. 9, pseudo albedo is only approximate and can contain deviating segment boundaries (row 1), over-segmentation (row 2), and incorrect albedo labels for segments (row 3). *ShoeRinsics* learns to fix these errors when trained using pseudo albedo as ground-truth.

References

[1] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7822–7831, 2021.

[2] William J Bodziak. *Footwear impression evidence: detection, recovery, and examination*. CRC Press, 2017.

[3] Jean Duchon. Splines minimizing rotation-invariant seminorms in sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.

[4] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 10(4):439–451, 1988.

[5] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

[6] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, 07–09 Jul 2015. PMLR.

[7] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 10–15 Jul 2018.

[8] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[9] Michael Janner, Jiajun Wu, Tejas D. Kulkarni, Ilker Yildirim, and Joshua B. Tenenbaum. Self-supervised intrinsic image decomposition. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5938–5948, Red Hook, NY, USA, 2017. Curran Associates Inc.

[10] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, 2017.

[11] Yoram Yekutieli, Yaron Shor, Sarena Wiesner, and Tsadok Tsach. Expert assisting computerized system for evaluating the degree of certainty in 2d shoeprints. *The US Department of Justice: Washington, DC, USA*, 2012.