A General and Adaptive Robust Loss Function
Supplementary Material

Jonathan T. Barron
Google Research

1. Alternative Forms

The registration and clustering experiments in the paper require that we formulate our loss as an outlier process. Using the equivalence between robust loss minimization and outlier processes established by Black and Rangarajan [1], we can derive our loss's Ψ-function:

$$\Psi(z,\alpha) = \begin{cases} -\log(z) + z - 1 & \text{if } \alpha = 0 \\ z\log(z) - z + 1 & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha}\left(\left(1-\frac{\alpha}{2}\right)z^{\frac{\alpha}{(\alpha-2)}} + \frac{\alpha z}{2} - 1\right) & \text{if } \alpha < 2 \end{cases}$$

$$\rho(x,\alpha,c) = \min_{0\le z\le 1}\left(\frac{1}{2}\left({}^{x}/_{c}\right)^2 z + \Psi(z,\alpha)\right) \tag{1}$$

$\Psi(z,\alpha)$ is not defined when $\alpha \ge 2$ because for those values the loss is no longer robust, and so is not well described as a process that rejects outliers.

We can also derive our loss's weight function to be used during iteratively reweighted least squares [2, 5]:

$$\frac{1}{x}\frac{\partial \rho}{\partial x}(x,\alpha,c) = \begin{cases} \frac{1}{c^2} & \text{if } \alpha = 2 \\ \frac{2}{x^2+2c^2} & \text{if } \alpha = 0 \\ \frac{1}{c^2}\exp\left(-\frac{1}{2}\left({}^{x}/_{c}\right)^2\right) & \text{if } \alpha = -\infty \\ \frac{1}{c^2}\left(\frac{\left({}^{x}/_{c}\right)^2}{|\alpha-2|} + 1\right)^{\left({}^{\alpha}/_{2}-1\right)} & \text{otherwise} \end{cases} \tag{2}$$

Curiously, these IRLS weights resemble a non-normalized form of Student's t-distribution. These weights are not used in any of our experiments, but they are an intuitive way to demonstrate how reducing α attenuates the effect of outliers. A visualization of our loss's Ψ-functions and weight functions for different values of α can be seen in Figure 1.

2. Practical Implementation

The special cases in the definition of $\rho(\cdot)$ that are required because of the removable singularities of $f(\cdot)$ at $\alpha = 0$ and $\alpha = 2$ can make implementing our loss somewhat inconvenient. Additionally, $f(\cdot)$ is numerically unstable near these singularities, due to divisions by small values. Furthermore, many deep learning

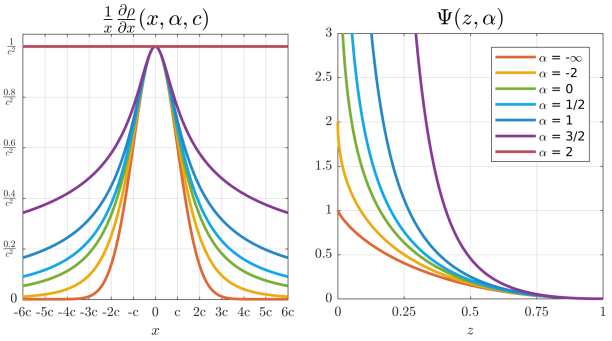

Figure 1: Our general loss's IRLS weight function (left) and Ψ-function (right) for different values of the shape parameter α.

frameworks handle special cases inefficiently by evaluating all cases of a conditional statement, even though only one case is needed. To circumvent these issues we can slightly modify our loss (and its gradient and Ψ-function) to guard against singularities and make implementation easier:

$$\rho(x,\alpha,c) = \frac{b}{d}\left(\left(\frac{\left({}^{x}/_{c}\right)^2}{b} + 1\right)^{\left({}^{d}/_{2}\right)} - 1\right)$$

$$\frac{\partial \rho}{\partial x}(x,\alpha,c) = \frac{x}{c^2}\left(\frac{\left({}^{x}/_{c}\right)^2}{b} + 1\right)^{\left({}^{d}/_{2}-1\right)}$$

$$\Psi(z,\alpha) = \begin{cases} \frac{b}{d}\left(\left(1-\frac{d}{2}\right)z^{\frac{d}{(d-2)}} + \frac{dz}{2} - 1\right) & \text{if } \alpha < 2 \\ 0 & \text{if } \alpha = 2 \end{cases}$$

$$b = |\alpha - 2| + \epsilon \qquad d = \begin{cases} \alpha + \epsilon & \text{if } \alpha \ge 0 \\ \alpha - \epsilon & \text{if } \alpha < 0 \end{cases}$$

Where ϵ is some small value, such as 10^{-5}. Note that even very small values of ϵ can cause significant inaccuracy between our true partition function $Z(\alpha)$ and the effective partition function of our approximate distribution when α is near 0, so this approximate implementation should be avoided when accurate values of $Z(\alpha)$ are necessary.

Implementing the negative log-likelihood of our general

distribution (ie, our adaptive loss) requires a tractable and differentiable approximation of its log partition function. Because the analytical form of $Z(\alpha)$ detailed in the paper is difficult to evaluate efficiently for any real number, and especially difficult to differentiate with respect to α, we approximate $\log(Z(\alpha))$ using cubic hermite spline interpolation in a transformed space. Efficiently approximating $\log(Z(\alpha))$ with a spline is difficult, as we would like a concise approximation that holds over the entire valid range $\alpha \geq 0$, and we would like to allocate more precision in our spline interpolation to values near $\alpha = 2$ (which is where $\log(Z(\alpha))$ varies most rapidly). To accomplish this, we first apply a monotonic nonlinearity to α that stretches values near $\alpha = 2$ (thereby increasing the density of spline knots in this region) and compresses values as $\alpha \gg 4$, for which we use:

$$\text{curve}(\alpha) = \begin{cases} \frac{9(\alpha-2)}{4|\alpha-2|+1} + \alpha + 2 & \text{if } \alpha < 4 \\ \frac{5}{18}\log(4\alpha - 15) + 8 & \text{otherwise} \end{cases} \quad (3)$$

This curve is roughly piecewise-linear in $[0, 4]$ with a slope of $\sim$1 at $\alpha = 0$ and $\alpha = 4$, but with a slope of $\sim$10 at $\alpha = 2$. When $\alpha > 4$ the curve becomes logarithmic. This function is continuously differentiable, as is required for our log-partition approximation to also be continuously differentiable.

We transform α with this nonlinearity, and then approximate $\log(Z(\alpha))$ in that transformed space using a spline with knots in the range of $[0, 12]$ evenly spaced apart by $1/1024$. Values for each knot are set to their true value, and tangents for each knot are set to minimize the squared error between the spline and the true log partition function. Because our spline knots are evenly spaced in this transformed space, spline interpolation can be performed in constant time with respect to the number of spline knots. For all values of α this approximation is accurate to within 10^{-6}, which appears to be sufficient for our purposes. Our nonlinearity and our spline approximation to the true partition function for small values of α can be seen in Figure 2.

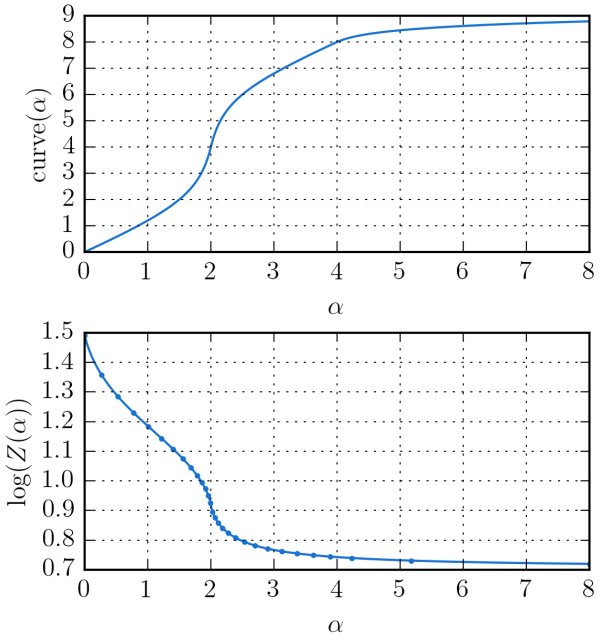

Figure 2: Because our distribution's log partition function $\log(Z(\alpha))$ is difficult to evaluate for arbitrary inputs, we approximate it using cubic hermite spline interpolation in a transformed space: first we curve α by a continuously differentiable nonlinearity that increases knot density near $\alpha = 2$ and decreases knot density when $\alpha > 4$ (top) and then we fit an evenly-sampled cubic hermite spline in that curved space (bottom). The dots shown in the bottom plot are a subset of the knots used by our cubic spline, and are presented here to demonstrate how this approach allocates spline knots with respect to α.

3. Motivation and Derivation

Our loss function is derived from the "generalized Charbonnier" loss [8], which itself builds upon the Charbonnier loss function [3]. To better motivate the construction of our loss function, and to clarify its relationship to prior work, here we work through how our loss function was constructed.

Generalized Charbonnier loss can be defined as:

$$d(x, \alpha, c) = \left(x^2 + c^2\right)^{\alpha/2} \quad (4)$$

Here we use a slightly different parametrization from [8] and use $\alpha/2$ as the exponent instead of just α. This makes the generalized Charbonnier somewhat easier to reason about with respect to standard loss functions: $d(x, 2, c)$ resembles L2 loss, $d(x, 1, c)$ resembles L1 loss, etc.

We can reparametrize generalized Charbonnier loss as:

$$d(x, \alpha, c) = c^\alpha \left(\left(x/c\right)^2 + 1\right)^{\alpha/2} \quad (5)$$

We omit the c^α scale factor, which gives us a loss that is scale invariant with respect to c:

$$g(x, \alpha, c) = \left(\left(x/c\right)^2 + 1\right)^{\alpha/2} \quad (6)$$

$$\forall_{k>0} \quad g(kx, \alpha, kc) = g(x, \alpha, c) \quad (7)$$

This lets us view the c "padding" variable as a "scale" parameter, similar to other common robust loss functions. Additionally, only after dropping this scale factor does setting α to a negative value yield a family of meaningful robust loss functions, such as Geman-McClure loss.

But this loss function still has several unintuitive properties: the loss is non-zero when $x = 0$ (assuming a non-zero value of c), and the curvature of the quadratic "bowl" near

$x = 0$ varies as a function of c and α. We therefore construct a shifted and scaled version of Equation 6 that does not have these properties:

$$\frac{g\left(x, \alpha, c\right) - g\left(0, \alpha, c\right)}{c^2 g''\left(0, \alpha, c\right)} = \frac{1}{\alpha}\left(\left(\left({}^{x}/_{c}\right)^2 + 1\right)^{{}^{\alpha}/_{2}} - 1\right) \quad (8)$$

This loss generalizes L2, Cauchy, and Geman-McClure loss, but it has the unfortunate side-effect of flattening out to 0 when $\alpha \ll 0$, thereby prohibiting many annealing strategies. This can be addressed by modifying the $1/\alpha$ scaling to approach 1 instead of 0 when $\alpha \ll 0$ by introducing another scaling that cancels out the division by α. To preserve the scale-invariance of Equation 7, this scaling also needs to be applied to the $\left({}^{x}/_{c}\right)^2$ term in the loss. This scaling also needs to maintain the monotonicity of our loss with respect to α so as to make annealing possible. There are several scalings that satisfy this property, so we select one that is efficient to evaluation and which keeps our loss function smooth (ie, having derivatives of all orders everywhere) with respect to x, α, and c, which is $|\alpha - 2|$. This gives us our final loss function:

$$f\left(x, \alpha, c\right) = \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{\left({}^{x}/_{c}\right)^2}{|\alpha - 2|} + 1\right)^{{}^{\alpha}/_{2}} - 1\right) \quad (9)$$

Using $|\alpha - 2|$ satisfies all of our criteria, though it does introduce a removable singularity into our loss function at $\alpha = 2$ and reduces numerical stability near $\alpha = 2$.

4. Additional Properties

Here we enumerate additional properties of our loss function that were not used in our experiments.

At the origin the IRLS weight of our loss is $\frac{1}{c^2}$:

$$\frac{1}{x}\frac{\partial \rho}{\partial x}\left(0, \alpha, c\right) = \frac{1}{c^2} \quad (10)$$

For all values of α, when $|x|$ is small with respect to c the loss is well-approximated by a quadratic bowl:

$$\rho\left(x, \alpha, c\right) \approx \frac{1}{2}\left({}^{x}/_{c}\right)^2 \quad \text{if } |x| < c \quad (11)$$

Because the second derivative of the loss is maximized at $x = 0$, this quadratic approximation tells us that the second derivative is bounded from above:

$$\frac{\partial^2 \rho}{\partial x^2}(x, \alpha, c) \leq \frac{1}{c^2} \quad (12)$$

When α is negative the loss approaches a constant as $|x|$ approaches infinity, letting us bound the loss:

$$\forall_{x,c}\ \rho\left(x, \alpha, c\right) \leq \frac{\alpha - 2}{\alpha} \quad \text{if } \alpha < 0 \quad (13)$$

The loss's Ψ-function increases monotonically with respect to α when $\alpha < 2$ for all values of z in $[0, 1]$:

$$\frac{\partial \Psi}{\partial \alpha}\left(z, \alpha\right) \geq 0 \quad \text{if } 0 \leq z \leq 1 \quad (14)$$

The roots of the second derivative of $\rho\left(x, \alpha, c\right)$ are:

$$x = \pm c\sqrt{\frac{\alpha - 2}{\alpha - 1}} \quad (15)$$

This tells us at what value of x the loss begins to redescend. This point has a magnitude of c when $\alpha = -\infty$, and that magnitude increases as α increases. The root is undefined when $\alpha \geq 1$, as our loss is redescending iff $\alpha < 1$. Our loss is strictly convex iff $\alpha \geq 1$, non-convex iff $\alpha < 1$, and pseudoconvex for all values of α.

5. Wavelet Implementation

Two of our experiments impose our loss on images reparametrized with the Cohen-Daubechies-Feauveau (CDF) 9/7 wavelet decomposition [4]. The analysis filters used for these experiments are:

lowpass	highpass
0.852698679009	0.788485616406
0.377402855613	-0.418092273222
-0.110624404418	-0.040689417609
-0.023849465020	0.064538882629
0.037828455507	

Here the origin coefficient of the filter is listed first, and the rest of the filter is symmetric. The synthesis filters are defined as usual, by reversing the sign of alternating wavelet coefficients in the analysis filters. The lowpass filter sums to $\sqrt{2}$, which means that image intensities are doubled at each scale of the wavelet decomposition, and that the magnitude of an image is preserved in its wavelet decomposition. Boundary conditions are "reflecting", or half-sample symmetric.

6. Variational Autoencoders

Our VAE experiments were performed using the code included in the TensorFlow Probability codebase at `http://github.com/tensorflow/probability/blob/master/tensorflow_probability/examples/vae.py`. This code was designed for binarized MNIST data, so adapting it to the real-valued color images in CelebA [6] required the following changes:

- Changing the input and output image resolution from $(28, 28, 1)$ to $(64, 64, 3)$.
- Increasing the number of training steps from 5000 to 50000, as CelebA is significantly larger than MNIST.

- Delaying the start of cosine decay of the learning rate until the final 10000 training iterations.

- Changing the CNN architecture from a 5-layer network with 5-tap and 7-tap filters with interleaved strides of 1 and 2 (which maps from a 28×28 image to a vector) to a 6-layer network consisting of all 5-tap filters with strides of 2 (which maps from a 64×64 input to a vector). The number of hidden units was left unchanged, and the one extra layer we added at the end of our decoder (and beginning of our decoder) was given the same number of hidden units as the layer before it.

- In our "DCT + YUV" and "Wavelets + YUV" models, before imposing our model's posterior we apply an RGB-to-YUV transformation and then a per-channel DCT or wavelet transformation to the YUV images, and then invert these transformations to visualize each sampled image. In the "Pixels + RGB" model this transformation and its inverse are the identity function.

- As discussed in the paper, for each output coefficient (pixel value, DCT coefficient, or wavelet coefficient) we add a scale variable (σ when using normal distributions, c when using our general distributions) and a shape variable α (when using our general distribution).

We made as few changes to the reference code as possible so as to keep our model architecture as simple as possible, as our goal is not to produce state-of-the-art image synthesis results for some task, but is instead to simply demonstrate the value of our general distribution in isolation.

CelebA [6] images are processed by extracting a square 160×160 image region at the center of each 178×218 image and downsampling it to 64×64 by a factor of $2.5\times$ using TensorFlow's bilinear interpolation implementation. Pixel intensities are scaled to $[0, 1]$.

In the main paper we demonstrated that using our general distribution to independently model the robustness of each coefficient of our image representation works better than assuming a Cauchy ($\alpha = 0$) or normal distribution ($\alpha = 2$) for all coefficients (as those two distributions lie within our general distribution). To further demonstrate the value of *independently* modeling the robustness of each individual coefficient, we ran a more thorough experiment in which we densely sampled values for α in $[0, 2]$ that are used for *all* coefficients. In Figure 3 we visualize the validation set ELBO for each fixed value of α compared to an independently-adapted model. As we can see, though quality can be improved by selecting a value for α in between 0 and 2, no single global setting of the shape parameter matches the performance achieved by allowing each coefficient's shape parameter to automatically adapt itself to the training data. This observation is consistent with earlier results on adaptive heavy-tailed distributions for image data [7].

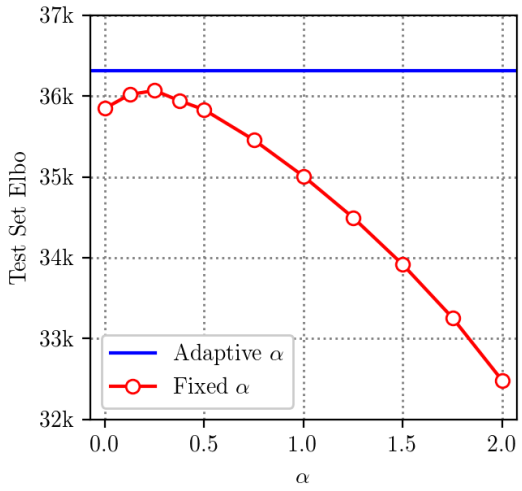

Figure 3: Here we compare the validation set ELBO of our adaptive "Wavelets + YUV" VAE model with the ELBO achieved when setting all wavelet coefficients to have the same fixed shape parameter α. We see that allowing our distribution to individually adapt its shape parameter to each coefficient outperforms any single fixed shape parameter.

In our Student's t-distribution experiments, we parametrize each "degrees of freedom" parameter as the exponentiation of some latent free parameter:

$$\nu^{(i)} = \exp\left(\nu_\ell^{(i)}\right) \tag{16}$$

where all $\nu_\ell^{(i)}$ are initialized to 0. Technically, these experiments are performed with the "Generalized Student's t-distribution", meaning that we have an additional scale parameter $\sigma^{(i)}$ that is divided into x before computing the log-likelihood and is accounted for in the partition function. These scale parameters are parametrized identically to the $c^{(i)}$ parameters used by our general distribution.

Comparing likelihoods across our different image representations requires that the "Wavelets + YUV" and "DCT + YUV" representations be normalized to match the "Pixels + RGB" representation. We therefore construct the linear transformations used for the "Wavelets + YUV" and "DCT + YUV" spaces to have determinants of 1 as per the change of variable formula (that is, both transformations are in the "special linear group"). Our wavelet construction in Section 5 satisfies this criteria, and we use the orthonormal version of the DCT which also satisfies this criteria. However, the standard RGB to YUV conversion matrix does not have a determinant of 1, so we scale it by the inverse of the cube root of the standard conversion matrix, thereby forcing its

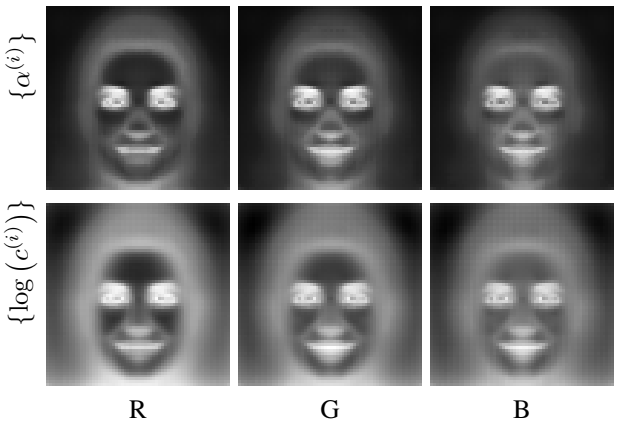

Figure 4: The final shape and scale parameters $\{\alpha^{(i)}\}$ and $\{c^{(i)}\}$ for our "Pixels + RGB" VAE after training has converged. We visualize α with black=0 and white=2 and $\log(c)$ with black=$\log(0.002)$ and white=$\log(0.02)$.

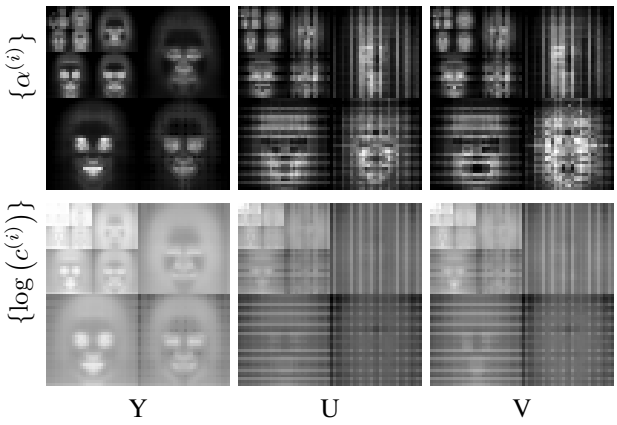

Figure 5: The final shape and scale parameters $\{\alpha^{(i)}\}$ and $\{c^{(i)}\}$ for our "Wavelets + YUV" VAE after training has converged. We visualize α with black=0 and white=2 and $\log(c)$ with black=$\log(0.00002)$ and white=$\log(0.2)$.

determinant to be 1. The resulting matrix is:

$$\begin{bmatrix} 0.47249 & 0.92759 & 0.18015 \\ -0.23252 & -0.45648 & 0.68900 \\ 0.97180 & -0.81376 & -0.15804 \end{bmatrix}$$

Naturally, its inverse maps from YUV to RGB.

Because our model can adapt the shape and scale parameters of our general distribution to each output coefficient, after training we can inspect the shapes and scales that have emerged during training, and from them gain insight into how optimization has modeled our training data. In Figures 4 and 5 we visualize the shape and scale parameters for our "Pixels + RGB" and "Wavelets + YUV" VAEs respectively. Our "Pixels" model is easy to visualize as each output coefficient simply corresponds to a pixel in a channel, and our "Wavelets" model can be visualized by flattening each wavelet scale and orientation into an image (our DCT-based model is difficult to visualize in any intuitive way). In both models we observe that training has determined that these face images should be modeled using normal-like distributions near the eyes and mouth, presumably because these structures are consistent and repeatable on human faces, and Cauchy-like distributions on the background and in flat regions of skin. Though our "Pixels + RGB" model has estimated similar distributions for each color channel, our "Wavelets + YUV" model has estimated very different behavior for luma and chroma: more Cauchy-like behavior is expected in luma variation, especially at fine frequencies, while chroma variation is modeled as being closer to a normal distribution across all scales.

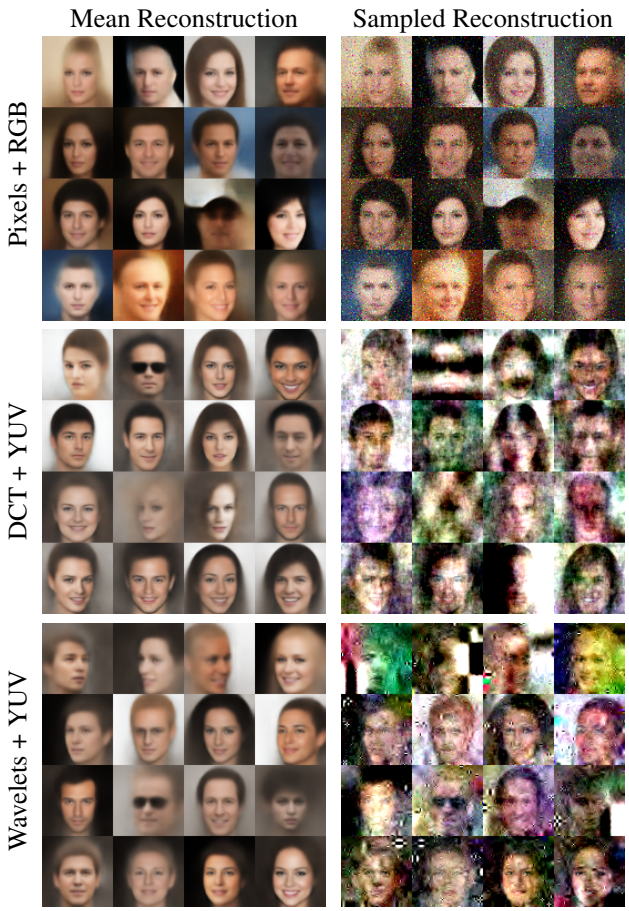

Figure 6: As is common practice, the VAE samples shown in this paper are samples from the latent space (left) but not from the final conditional distribution (right). Here we contrast decoded means and samples from VAEs using our different output spaces, all using our general distribution.

See Figure 8 for additional samples from our models,

and see Figure 9 for reconstructions from our models on validation-set images. As is common practice, the samples and reconstructions in those figures and in the paper are the means of the output distributions of the decoder, not samples from those distributions. That is, we draw samples from the latent encoded space and then decode them, but we do not draw samples in our output space. Samples drawn from these output distributions tend to look noisy and irregular across all distributions and image representations, but they provide a good intuition of how our general distribution behaves in each image representation, so in Figure 6 we present side-by-side visualizations of decoded means and samples.

7. Unsupervised Monocular Depth Estimation

Our unsupervised monocular depth estimation experiments use the code from `https://github.com/tinghuiz/SfMLearner`, which appears to correspond to the "Ours (w/o explainability)" model from Table 1 of [10]. The only changes we made to this code were: replacing its loss function with our own, reducing the number of training iterations from 200000 to 100000 (training converges faster when using our loss function) and disabling the smoothness term and multi-scale side predictions used by [10], as neither yielded much benefit when combined with our new loss function and they complicated experimentation by introducing hyperparameters. Because the reconstruction loss in [10] is the sum of the means of the losses imposed at each scale in a D-level pyramid of side predictions, we use a D level normalized wavelet decomposition (wherein images in $[0, 1]$ result in wavelet coefficients in $[0, 1]$) and then scale each coefficient's loss by 2^d, where d is the coefficient's level.

In Figure 7 we visualize the final shape parameters for each output coefficient that were converged upon during training. These results provide some insight into why our adaptive model produces improved results compared to the ablations of our model in which we use a single fixed or annealed value for α for all output coefficients. From the low α values in the luma channel we can infer that training has decided that luma variation often has outliers, and from the high α values in the chroma channel we can infer that chroma variation rarely has outliers. Horizontal luma variation (upper right) tends to have larger α values than vertical luma variation (lower left), perhaps because depth in this dataset is largely due to horizontal motion, and so horizontal gradients tend to provide more depth information than vertical gradients. Looking at the sides and the bottom of all scales and channels we see that the model expects more outliers in these regions, which is likely due to boundary effects: these areas often contain consistent errors due to there not being a matching pixel in the alternate view.

In Figures 10 and 11 we present many more results from the test split of the KITTI dataset, in which we compare our "adaptive" model's output to the baseline model and the ground-truth depth. The improvement we see is substantial and consistent across a variety of scenes.

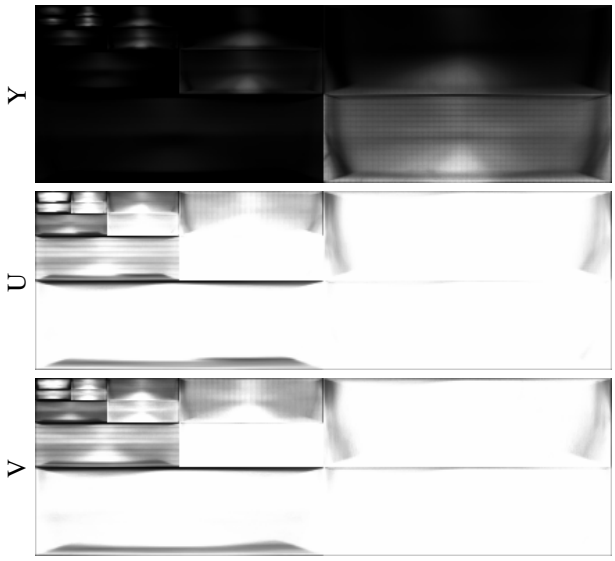

Figure 7: The final shape parameters α for our unsupervised monocular depth estimation model trained on KITTI data. The parameters are visualized in the same "YUV + Wavelet" output space as was used during training, where black is $\alpha = 0$ and white is $\alpha = 2$.

8. Fast Global Registration

Our registration results were produced using the code release corresponding to [9]. Because the numbers presented in [9] have low precision, we reproduced the performance of the baseline FGR algorithm using this code. This code included some evaluation details that were omitted from the paper that we determined through correspondence with the author: for each input, FGR is run 20 times with random initialization and the median error is reported. We use this procedure when reproducing the baseline performance of [9] and when evaluating our own models.

References

[1] Michael J. Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 1996.

[2] Richard H. Byrd and David A. Pyne. Convergence of the iteratively reweighted least-squares algorithm for robust regression. Technical report, Dept. of Mathematical Sciences, The Johns Hopkins University, 1979.

[3] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. *ICIP*, 1994.

[4] Albert Cohen, Ingrid Daubechies, and J-C Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 1992.

[5] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, 1986.

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[7] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE TIP*, 2003.

[8] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. *CVPR*, 2010.

[9] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. *ECCV*, 2016.

[10] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CVPR*, 2017.

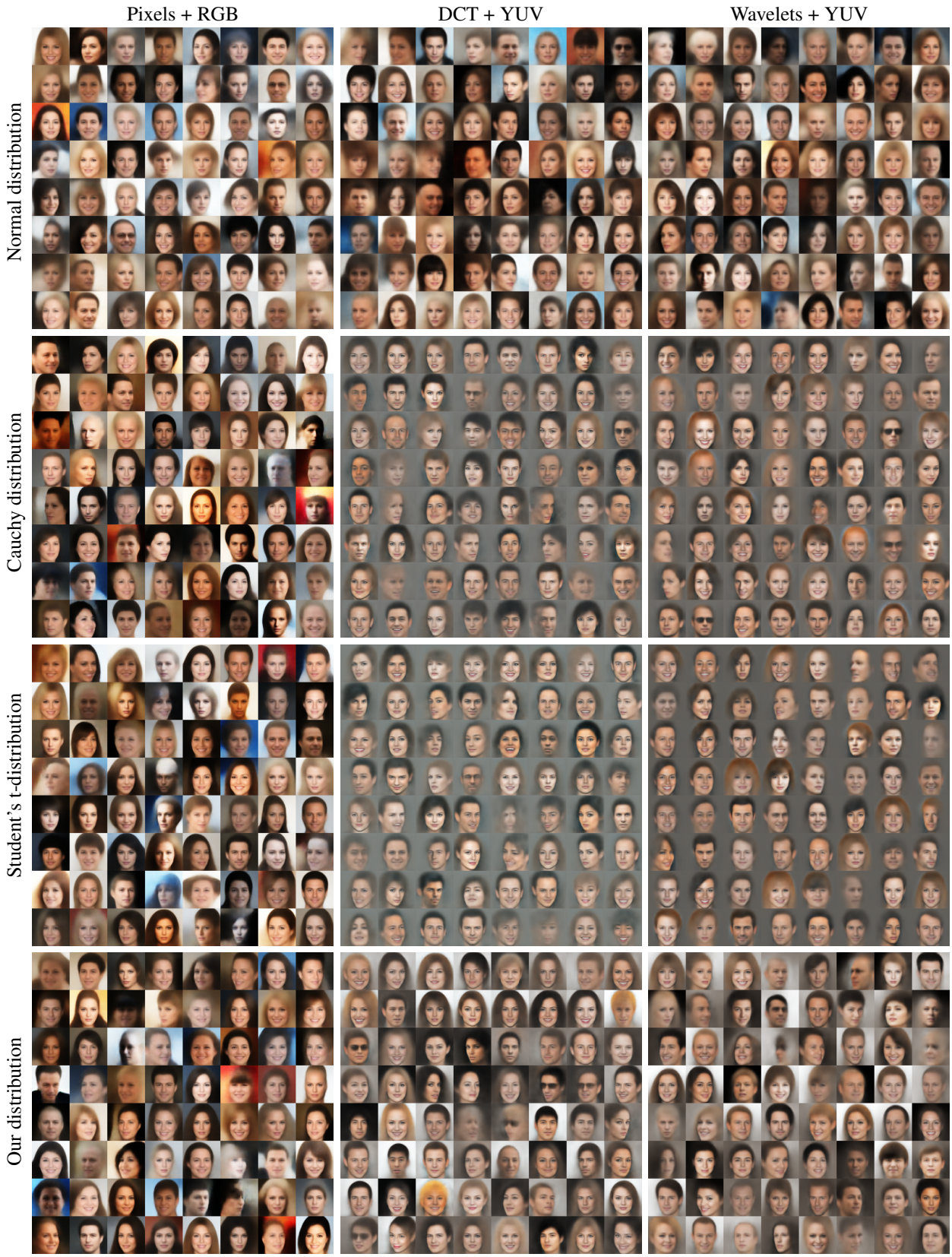

Figure 8: Random samples (more precisely, means of the output distributions decoded from random samples in our latent space) from our family of trained variational autoencoders.

Figure 9: Reconstructions from our family of trained variational autoencoders, in which we use one of three different image representations for modeling images (super-columns) and use either normal, Cauchy, Student's t, or our general distributions for modeling the coefficients of each representation (sub-columns). The leftmost column shows the images which are used as input to each autoencoder. Reconstructions from models using general distributions tend to be sharper and more detailed than reconstructions from the corresponding model that uses normal distributions, particularly for the DCT or wavelet representations, though this difference is less pronounced than what is seen when comparing samples from these models. The DCT and wavelet models trained with Cauchy distributions or Student's t-distributions systematically fail to preserve the background of the input image, as was noted when observing samples from these distributions.

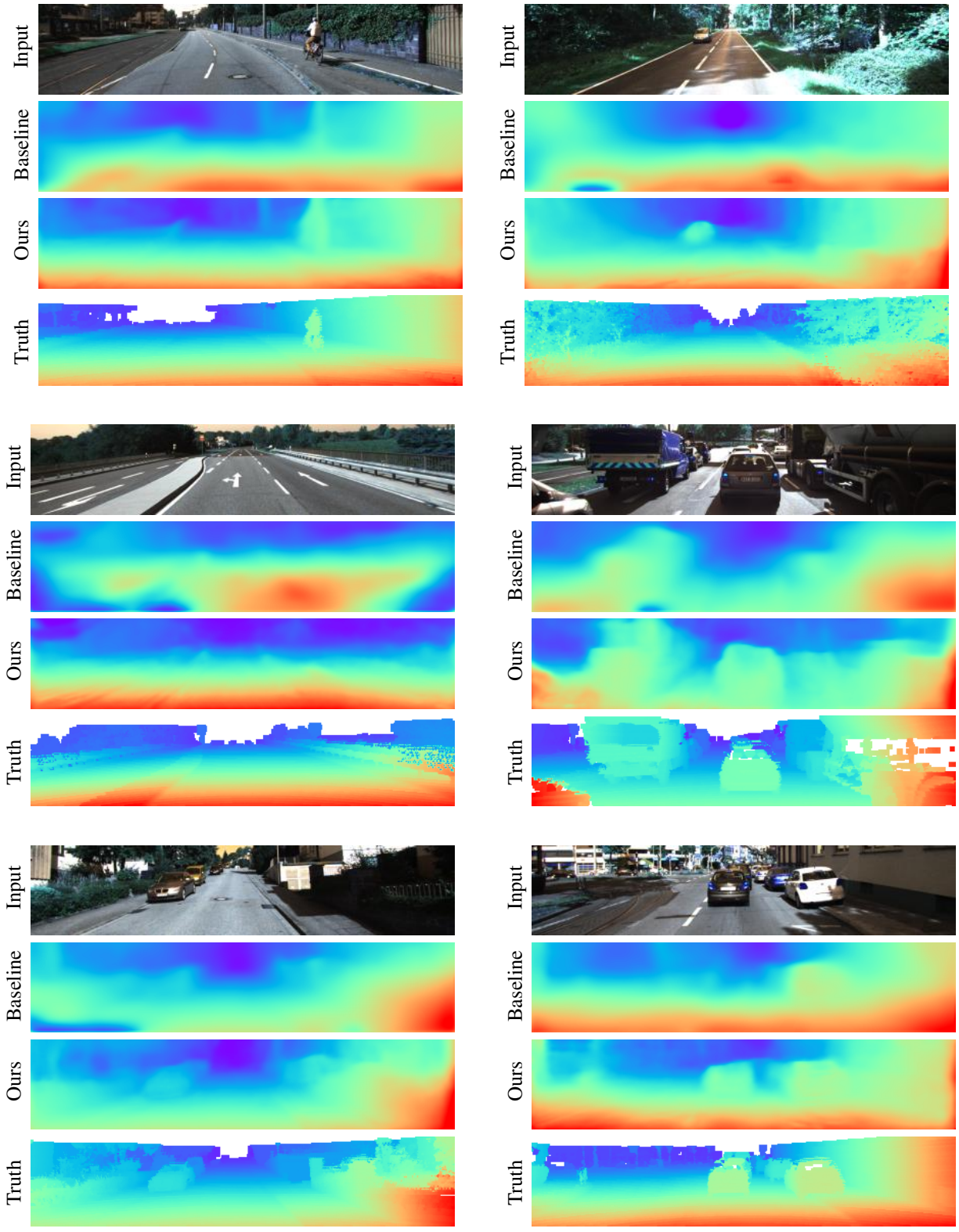

Figure 10: Monocular depth estimation results on the KITTI benchmark using the "Baseline" network of [10] and our own variant in which we replace the network's loss function with our own adaptive loss over wavelet coefficients. Changing only the loss function results in significantly improved depth estimates.

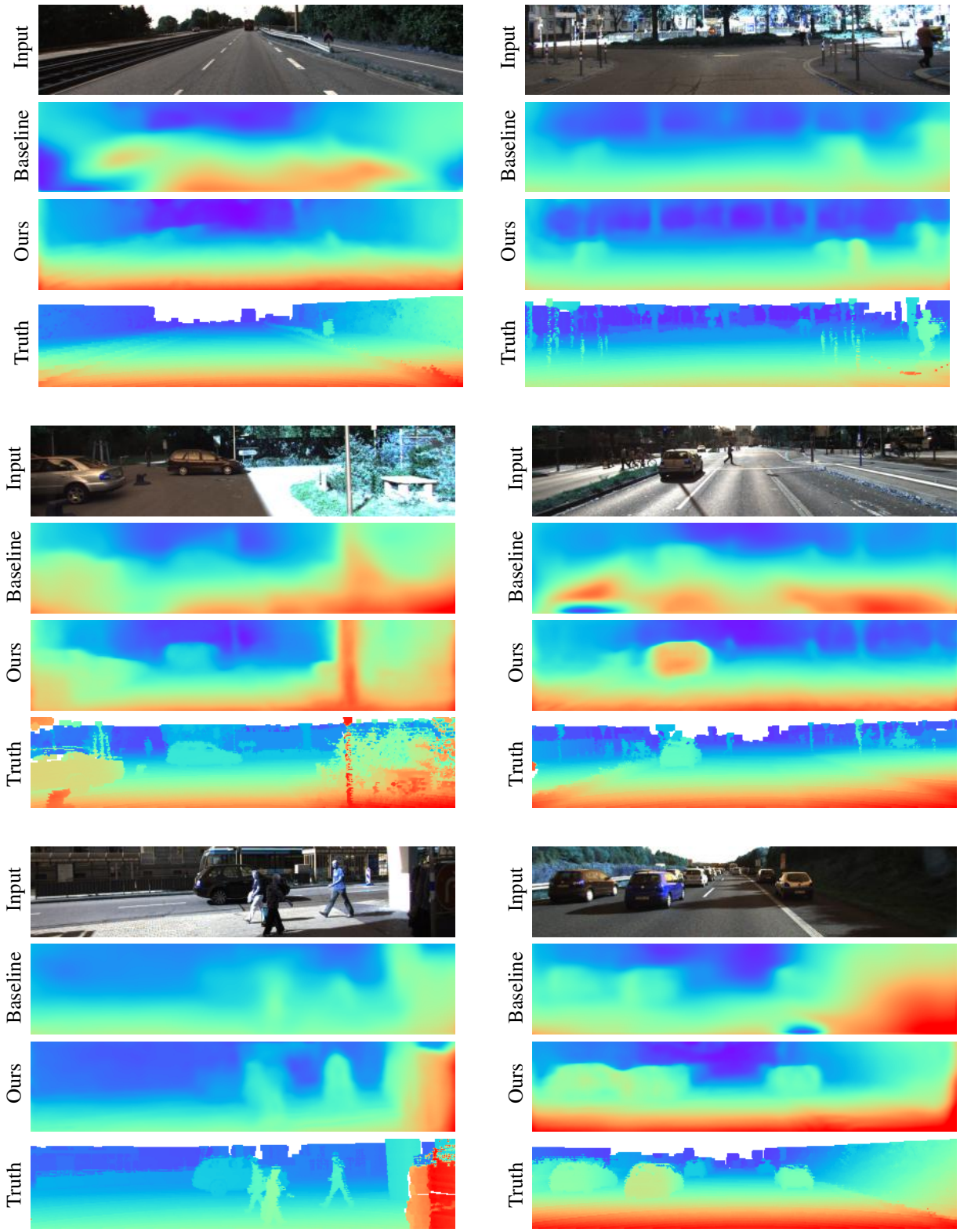

Figure 11: Additional monocular depth estimation results, in the same format as Figure 10.