# Real-Time Lens Distortion with Deep Learning

Daniel Huang[1]

[1]Institute of Computational and Mathematical Engineering, Stanford

## Abstract

The success of machine learning in recent years relies heavily on high-quality and high-quantity data. For self-driving cars, obtaining real-life data is the best option. However, it is difficult to capture rare but important events and can be costly due to manual labor and traffic disruptions. To address this issue, a simulated environment is used to train AI, where a crucial subsystem is the camera sensor model that translates raw visual data of the physical world to the signals the model "sees". While simulated environments can generate undistorted images easily using 2D projections, distortions caused by the lens need to be post-processed to account for camera lens properties. Based on the project listed at [1], we propose a real-time deep-learning approach for lens distortion correction using a neural network that approximates radial and tangential lens distortion. We present experimental results that can be greatly improved with refinement of model architecture and hyper-parameters.

## Background: Terms and Notation

- **Radial Distortion** : Distortion caused by lens' radial symmetry and curvature
- **Tangential Distortion** : Distortion caused by lens' misalignment with the image plane
- **Distortion Center** : The point where the optical axis intersects the image plane
- **K** $= [k_1, k_2, \ldots, k_i]$ : Radial distortion parameters ($i = 3$ for our model)
- **P** $= [p_1, p_2, \ldots, k_j]$ : Tangential distortion parameters ($j = 2$ for our model)
- **XY**, $(x, y) \in \mathbb{R}^2_{[-1,1]}$ : Normalized undistorted image coordinates
- **XYd**, $(x, y) \in \mathbb{R}^2_{[-1,1]}$ : Normalized distorted image coordinates
- $\bar{\cdot}$ : Distortion-centered based coordinates
- $r^2 = \bar{x}^2 + \bar{y}^2$ : Squared distance from distortion center

## Background: Brown's Distortion Model

The Brown's distortion model describes radial and tangential lens distortion given calibrated camera parameters using an infinite polynomial series. The radial model is given by:

$$x_{d,r} = x + \bar{x}\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots\right)$$
$$y_{d,r} = y + \bar{y}\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots\right)$$

And the tangential model is given by:

$$x_{d,t} = x + \left[p_1(r^2 + 2\bar{x}^2) + 2p_2\bar{x}\bar{y}\right]\left(1 + p_3 r^2 + p_4 r^4 + \ldots\right)$$
$$y_{d,t} = y + \left[p_2(r^2 + 2\bar{y}^2) + 2p_1\bar{x}\bar{y}\right]\left(1 + p_3 r^2 + p_4 r^4 + \ldots\right)$$

$\Rightarrow$**Difficulty comes from inverting the distortion model (i.e. solving for XY given XYd).**

## Background: Previous Approaches

**Iterative Approaches** [4]

- Numerically solve for **XY** given **XYd** using Brown's distortion model using Newton's method or other iterative solvers
- Balance between speed and accuracy, is not suitable for real-time applications

**Polynomial Approximation of Exact Solutions** [3]

- Approximate inverse function is possible with an infinite polynomial series
- Number of "de-distortion" coefficients required increases dramatically for larger distortions
- Not generalizable to cameras with different distortion parameters

## Goals and Ideas

**Goals**

- *Real-time* : Able to distort images accurately
- *Generalizable* : Able to handle different distortion parameters
- *Robust* : Able to handle large distortions

**Potential Approaches**

→ *Deep Learning* : Use a neural network to learn the inverse function
- *GANs* : Use a generative adversarial network to learn the mapping from undistorted to distorted images

## Methodology

**Model Idea: Point Map Neural Networks (PMNN)**

★ Learn inverse of Brown's distortion function using neural network
- Input: Distorted image coordinates **XYd** (what pixels are required to form the distorted image)
- Output: Undistorted image coordinates **XY** (where to query the pixels from in the original image)
- Data: Can generate infinite training examples from Brown's distortion model

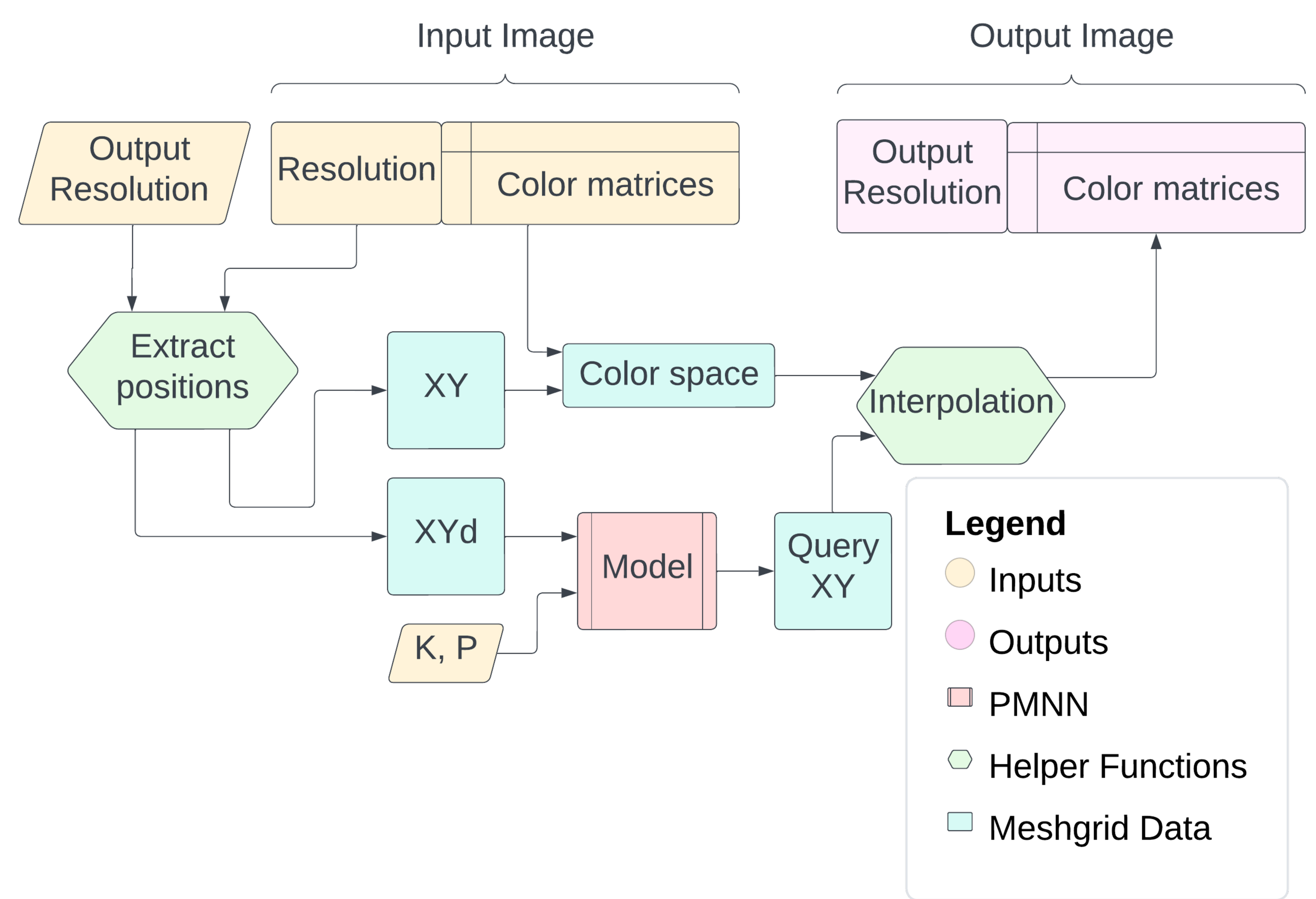


Figure 1. Block diagram of image distortion pipeline

**PMNN Architecture**

- *Layers* : $4 \times 16$ fully connected layers (small) or $6 \times 128$ fully connected layers (large)
- *Activation* : ReLU activation between each layer
- *Loss* : Mean absolute error $\mathtt{mean}(|XY - XY_p|_1)$ where $|\cdot|_1$ is the 1-norm and $XY_p$ is the predicted undistorted image coordinates
- *Optimizer* : Adam optimizer
- *Epochs* : Trained for 100 epochs, taking best validation loss
- *Dataset* : 2000 random sets of distortions $k_i, p_i \in (-0.1, 0.1)$ with uniformly random chosen $XY$ and $XYd$ coordinates. 70% training, 20% validation, 10% test

## Experiments

**Radial Distortion** $K = (0.1, 0.03, 0.005)$
**Ground Truth:** Gradient Descent on $XYd - \mathtt{distort}(XY)$, where $\mathtt{distort}(XY)$ distorts the image using Brown's distortion model

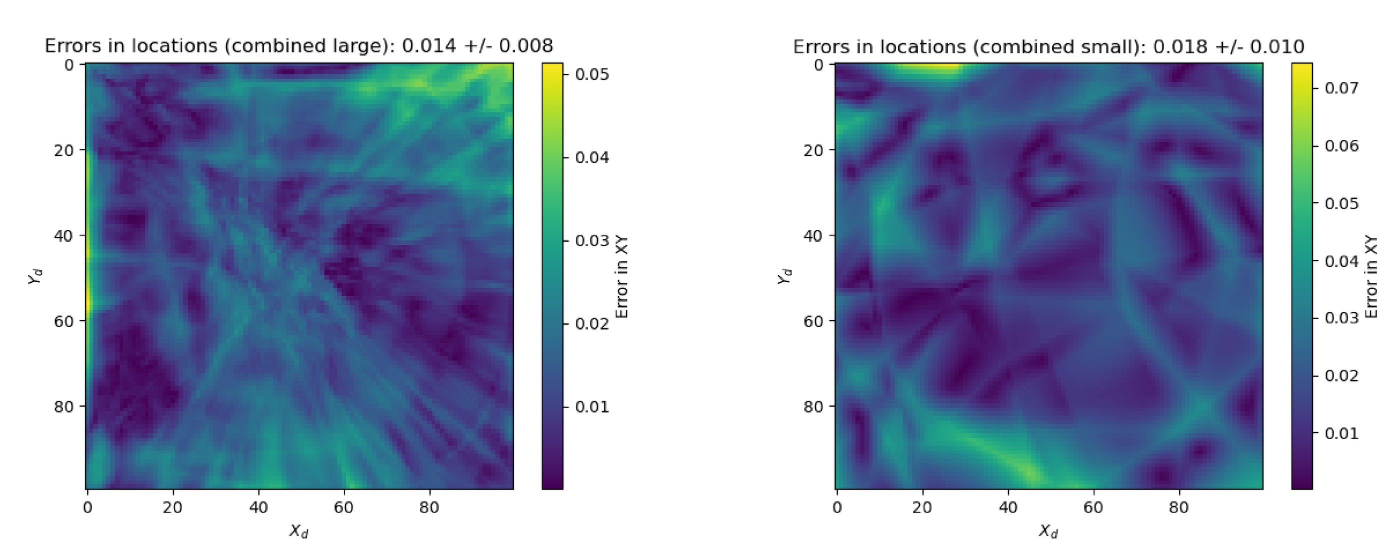


Figure 2. Heatmaps of error between ground truth and PMNN undistorted coordinates for large (left) and small (right) models

## Experiments (cont.)

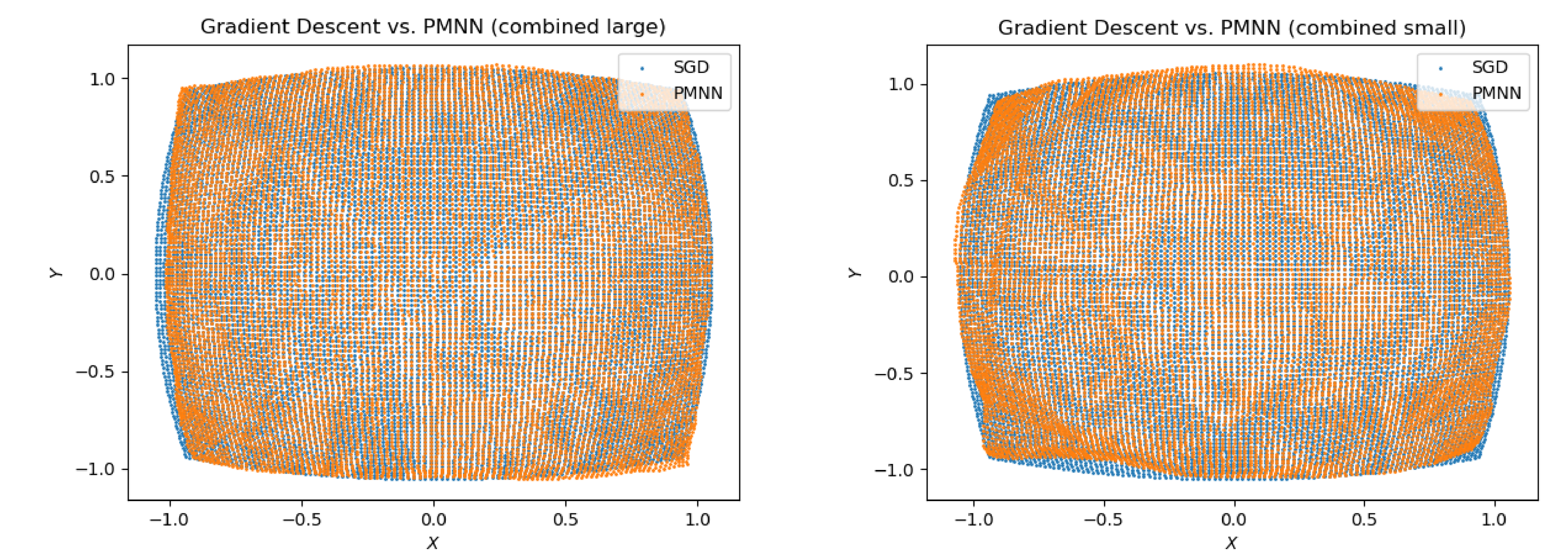


Figure 3. Point map comparison between ground truth and PMNN undistorted coordinates for large (left) and small (right) models

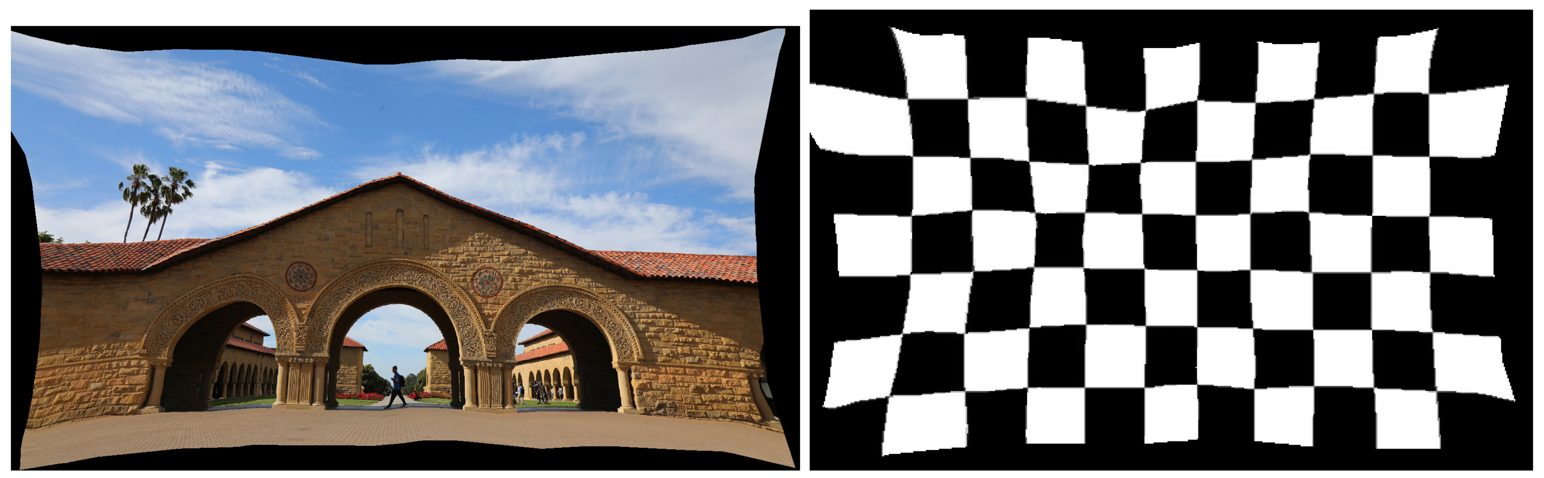

Figure 4. Stanford Main Quad and checkerboard images distorted using the small model

## Results and Discussion

- Neither model is capable of creating visually-consistent distortions (many artifacts)
- Large model slightly more accurate, but edges are problematic
- Much faster than gradient descent ($\approx$70 ms for PMNN on one core vs 17+ seconds for GD on multiple cores)
- Errors may come from inconsistencies between training data and expected usage

**Future Work**

- **Model Improvements** : Rapid post-processing to reduce noise in output
- **Iterative Models** : Design model with iterative refinement to improve accuracy [2]

## Acknowledgements

- **Industry Mentor** : Reza Fazel-Rezai, Ph.D., P.Eng., MathWorks
- **Faculty Mentor** : Alan Gous, Ph.D.
- **Xplore Mentor** : Kari Hanson

## References

[1] Mathworks excellence in innovation: Applying machine learning for the development of physical sensor models in game engine environment.
https://github.com/mathworks/MathWorks-Excellence-in-Innovation.

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas.
Learning to learn by gradient descent by gradient descent, November 2016.
arXiv:1606.04474 [cs].

[3] Pierre Drap and Julien Lefèvre.
An Exact Formula for Calculating Inverse Radial Lens Distortions.
*Sensors*, 16(6):807, June 2016.
Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[4] Jason P. de Villiers, F. Wilhelm Leuschner, and Ronelle Geldenhuys.
Centi-pixel accurate real-time inverse distortion correction.
In *Optomechatronic Technologies 2008*, volume 7266, pages 320–327. SPIE, November 2008.