Improving RANSAC-Based Segmentation Through CNN Encapsulation

Dustin Morley and Hassan Foroosh
University of Central Florida
Orlando, FL 32816
dustinrmorley@knights.ucf.edu, foroosh@cs.ucf.edu

Abstract

In this work, we present a method for improving a random sample consensus (RANSAC) based image segmentation algorithm by encapsulating it within a convolutional neural network (CNN). The improvements are gained by gradient descent training on the set of pre-RANSAC filtering and thresholding operations using a novel RANSAC-based loss function, which is geared toward optimizing the strength of the correct model relative to the most convincing false model. Thus, it can be said that our loss function trains the network on metrics that directly dictate the success or failure of the final segmentation rather than metrics that are merely correlated to the success or failure. We demonstrate successful application of this method to a RANSAC method for identifying the pupil boundary in images from the CASIA-IrisV3 iris recognition data set, and we expect that this method could be successfully applied to any RANSAC-based segmentation algorithm.

1. Introduction

Convolutional neural networks (CNN) have revolutionized the field of computer vision over the course of the past few years. This recent revolution had its ultimate origins in the specific area of object recognition in two-dimensional images, and then quickly spread to other areas such as semantic segmentation. As part of the natural evolution of the methodology, early work on utilizing CNNs for segmentation maintained as much similarity as possible to the successful object recognition approaches. Among other things, this led to the still commonly used approach of training a CNN to classify individual patches from images rather than operating on the entire image at once. Recent works on segmentation [15] have begun to move away from this for a variety of reasons, including efficiency. Indeed, the approach of operating on the entire image at once in CNN-based segmentation bears much clearer resemblance to segmentation pipelines which do not involve deep learning.

Model-specific segmentation problems, defined as a segmentation problem in which some straightforward mathematical form for the boundary of the desired object(s) is known beforehand, have yet to be explored with CNNs to the extent that other segmentation problems have. One popular approach to model-specific segmentation problems is

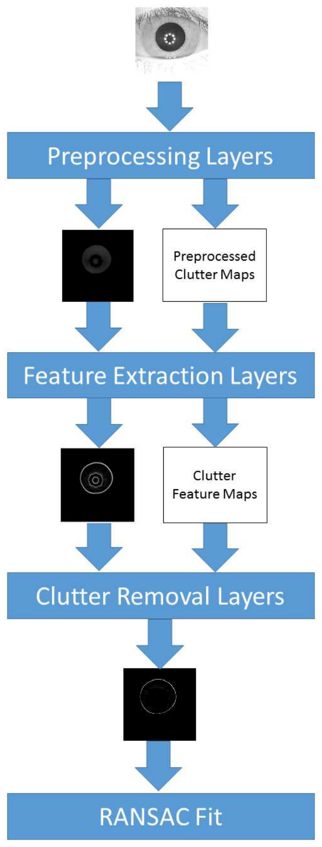

Figure 1. Our method for improving RANSAC segmentation performance by CNN encapsulation, shown with idealized intermediate outputs for an example problem of pupil segmentation. Following encapsulation of the algorithm into a CNN, the network is finetuned with a RANSAC based loss function.

to use RANSAC to enforce the mathematical form, as this method is extremely robust to outliers. In these approaches, there are usually filtering and thresholding steps that occur on the original input to generate the input for the RANSAC algorithm, and these steps traditionally do not utilize machine learning for optimization. We seek to demonstrate that these approaches (or at least significant pieces of them) can in general be directly encapsulated into a CNN ”as-is”, and that upon doing so the parameters can be fine-tuned through backpropagation using a novel error function which is directly tied to the propensity of RANSAC to choose the true segmentation over any false segmentation. Another interesting aspect of doing this is that a CNN constructed for a model-specific segmentation problem will generally be significantly smaller than the CNN architectures currently participating in the modern deep learning revolution. Thus, our work offers some validation of how well CNN concepts and techniques generalize to smaller problem sizes.

We apply our CNN formulation to the problem of pupil segmentation in images of human eyes. This is a problem with important applications to biometric identification [2][3][27] and ophthalmic surgery [1][16] that has been well-studied with classical computer vision approaches, which are capable of achieving a very high success rate on this problem due to the contrast between the pupil and iris being quite good under normal infrared imaging conditions. The fact that gradient strength is a key underlying assumption in these algorithms directly implies that it should be possible to exchange parts of these algorithms for a CNN and achieve better performance. We explore this directly by first constructing an algorithm along the lines of typical classical computer vision approaches (specifically, a combination of thresholding, edge detection, and filtering out extraneous edges), directly converting this algorithm into a CNN (by directly copying convolutional filters, using combinations of filter biases and ReLU layers for thresholding, and adding custom layers for additional calculations where necessary), and then executing training epochs to further fine-tune the constructed CNN.

In summary, our work makes the following contributions: we present a novel framework for model-specific segmentation that unifies CNN and RANSAC approaches using a loss function based on RANSAC outputs; we demonstrate success in using our framework to fine-tune a functional RANSAC segmentation algorithm through CNN training; we demonstrate robustness of our method through a multiplicity of experiments; and we demonstrate successful utilization of a CNN for a problem type and size that is very different from typical CNN work, thus providing significant additional validation of the adaptability and generalizability of the CNN framework.

2. Related Work

CNNs have been a topic of active research and discussion, particularly since significant performance gains on image classification were first reported [12]. Although the fundamentals of the CNN technique can be said to have already existed for a few decades [13], it has only been in recent years that CPUs and GPUs have advanced far enough to allow these techniques to be applied to large sets of typically sized images. Since this time, there have been important ongoing discussions about topics like how trained CNNs can be interpreted [30] and how well they generalize to other data sets or even different tasks [21]. CNNs have also been successfully applied to a variety of vision tasks besides classification, such as segmentation [8][15], super-resolution [4], and edge detection [28].

An important concept that has come to light with CNNs in recent years is the idea of fine-tuning, normally referring to the practice of taking as a starting point a CNN which has been pre-trained for some task and data set and applying it to a different task and/or data set. Success in doing so is well documented [28][21], with the pre-trained nets having at least reasonable performance right off the bat (due to the fact that filters within a pre-trained CNN exhibit positive responses to a large variety of useful features) and fantastic performance following training. Our work slots into this area in general, but with the important distinction of starting from a manually designed ”simple” CNN rather than a pre-trained deep CNN. We were unable to find any instances in the literature of other researchers attempting this task.

Another interesting development in CNNs that has emerged with the variety of problems they are being used to tackle is the utilization of a wide variety of loss functions for training. One example of growing interest is the use of structured loss for precise locations of objects [31][23]. As another example, Shen et al. [22] proposed a unique loss function for contour detection based on the idea that a false negative in contour detection is a more significant error than mislabeling the ”type” of contour. These examples demonstrate the importance of defining a loss function that is aligned as closely as possible with the most important metrics for the problem at hand. This philosophy is what inspired us to experiment with a unique RANSAC-based loss function (see section 3.4) for a model-specific segmentation problem, as this allowed us to pinpoint the loss function directly onto the success rate of the final emergent algorithm. This is in direct contrast to typical CNN segmentation approaches in which object boundaries are not direct outputs and must be found by applying additional algorithms to the CNN output (*e.g.* the CNN might output some kind of probability map from which boundary information must be extracted via algorithms like graph cut or RANSAC).

RANSAC [5][19] has been applied to many different problems, ranging from robotics [24][29] to biomedical image processing [26][17][11][20]. Its many advantages include robustness to outliers and ease of implementation. However, like all estimation methods, its performance has dependence on the input. If all inliers are present and there is no set of outliers forming a strong instance of the model being fit to, RANSAC is virtually guaranteed to identify the correct model given enough iterations. If some inliers are missing, RANSAC output can have accuracy issues. If there are a lot of outliers present, or if a large subset of outliers just so happen to form a model instance, RANSAC can experience a catastrophic failure of selecting a model formed from outliers rather than inliers. Therefore, whenever RANSAC is in use, it is important to optimize the input as much as possible. We seek to demonstrate that convolutional neural networks can be an effective tool for accomplishing this optimization task.

3. Method

In this section we describe our approach in detail. Our CNN contains four phases: preprocessing, feature extraction, clutter removal, and RANSAC model fitting. The first three phases all contain convolutional kernel weights and biases that can be optimized through network training. The weights and biases can also be initialized prior to training based on existing insights from other successful approaches to the problem at hand. We would argue that the ease of initialization with our approach makes it ideal for industrial applications where the algorithms currently being used already perform very well, as this allows for a starting point that largely (if not entirely) preserves the original performance prior to any machine learning. The final phase of our CNN is a RANSAC layer, which performs straightforward RANSAC model fitting in the forward pass and computes a novel RANSAC-based loss function in the backward pass.

3.1. Preprocessing

For the class of problems to which our method is applicable, the preprocessing phase can involve any combination of smoothing, rescaling, and thresholding. In CNN terms, smoothing and rescaling are convolutional operations, while thresholding can be performed by adding biases to the kernel outputs and then passing the output through a rectified linear unit (ReLU) layer. The significance of this phase has a lot of dependence on the regularity of the intensity profile of the object to be segmented relative to that of the background throughout the data samples. For more irregular intensity profiles, this layer would either have to be less aggressive, or include a sizable multiplicity of kernel/bias combinations.

3.2. Feature Extraction

The goal of the feature extraction phase is to construct feature maps from the outputs of the preprocessing phase. For a successful segmentation, the union of these outputs should ideally contain the full set of boundary points for the object of interest, with the amount of false positives being minimal and/or easily reduced by the clutter removal phase. The backbone of feature extraction is ultimately a set of convolutional filters, with the main source of diversity in different feature extraction methods lying in the set of filters used and the way in which their outputs are ultimately combined. For example, edge features have directional dependence to them, meaning that a single filter cannot capture all edges of an object. Therefore, a simple edge detection approach is to use one filter to extract horizontal edge strength and one filter to extract vertical edge strength, and then build a complete edge map from the Euclidean norm of the two resultant edge maps. To be sure, this Euclidean normalization of two feature maps is not an operation traditionally found in CNNs, but there is no reason why it couldn't be given the appropriate context, as the euclidean norm is indeed differentiable with respect to its inputs. We demonstrate successful application of this fact in our experiments. In particular, we insert a custom layer into a CNN that performs the following forward and backward calculations on two input channels denoted g_x and g_y given a loss function L:

$$h = \sqrt{g_x^2 + g_y^2} \tag{1}$$

$$\frac{\partial L}{\partial g_x} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial g_x} = \frac{\partial L}{\partial h}\left(\frac{g_x}{h}\right) \tag{2}$$

$$\frac{\partial L}{\partial g_y} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial g_y} = \frac{\partial L}{\partial h}\left(\frac{g_y}{h}\right) \tag{3}$$

On the other hand, by utilizing more filters, one could just as well have a full bank of filters for different edge orientations and combine them with one of several possible methods, including Euclidean norm across all outputs, max across all outputs, average of all outputs, or another layer of convolutional filters applied to the outputs (which collapses to the averaging option in the limit that a single filter is used with the center value equal to 1 for all input channels and all other values set to zero). Any differentiable operation is fair game in a CNN.

3.3. Clutter Removal

Once features have been extracted, it is often necessary to subject the features to some kind of pruning process, consisting of any combination of throwing away weak features, removing certain feature classes altogether, or successive application of additional filters to the feature maps. In a CNN, ReLU layers are the most straightforward way to throw away weak features. Regarding the other types of operations, a CNN can be constructed to have multiple largely independent channels entering this phase, which is significant because it is then possible to interpret some of these channels as focused on obtaining high feature strength for boundary pixels of the object to be segmented with the other channels instead focused on obtaining high feature strength for other objects or artifacts in the image. In this framework, some kind of weighted subtraction of the second kind of channels from the first kind should yield a good final map for the desired object boundary. This idea of a weighted subtraction can of course straightforwardly be executed with convolutional kernels, which can simultaneously apply other interesting operations on the channels (such as smoothing) prior to the subtraction. Alternatively, undesired objects or artifacts can also be filtered out ahead of time in some cases. For example, many segmentation pipelines in iris recognition remove LED reflections as one of the earliest steps [7][14]. A CNN embodying this design philosophy is simply one in which the first few layers produce an output which is (ideally) the original image but with the undesired artifacts removed. We performed CNN experiments utilizing each of these approaches to artifact removal (in fact, they are not mutually exclusive), but we focused more effort on the first approach due to the implementation being much more straightforward.

3.4. RANSAC Fitting and Backpropagation

Perhaps the most unique aspect of our work is the novel RANSAC-based loss function we employ in our CNN. RANSAC [5][19] is a model fitting technique that is extremely robust to noise, as outliers have no impact on the final shape provided the input has enough signal strength for the desired shape. It is also an extremely generic technique, applicable to any modeling problem where a fixed number of data points define an instance of the model. Our RANSAC implementation for pupil detection operates directly on the output Z of the previous CNN layer according to the following steps (assuming a circular model): construct a list of all points (x, y) where $Z(x, y) > 0$; select three of these points at random and construct the unique circle C passing through these points; compute a score for that circle based on the values of Z at points sufficiently close to the circle, but assigning a score of 0 if the circle violates known geometric constraints; repeat the random point sampling and circle scoring steps for a fixed number of iterations, maintaining (and eventually returning) the highest scoring circle.

We now turn to a very interesting question: what causes RANSAC to fail? Certainly, if $Z = 0\ \forall(x, y) \in C^*$ with C^* denoting the true circle, RANSAC will surely fail. Indeed, the input values at the points along the true circle are clearly a critical factor. But how high do these values actually have to be? How low do the other values actually have to be? The answer is that if the points satisfying the true model all have positive values, the only way RANSAC can actually fail (assuming a sufficient number of iterations) to return the true model is if a more convincing alternate model is present in the data. This means that not all false positives in Z are equally important, as a set of false positives that don't fit a single model instance (*i.e.* randomly scattered points) are considerably less likely to cause issue than a set of false positives which do fit a model instance. In the case of iris images, there are other structures present in the image besides the pupil which form a circle: the eyelids, the outer iris boundary, and the ring of LEDs inside the pupil. Thus, the key factor in whether RANSAC succeeds or fails provided decent representation of true positives is the strength of the strongest "impostor" model instance. For this reason, we propose a loss function centered on the ratio between the RANSAC scores of the strongest impostor and the true model, together with additive terms to penalize false negatives and false positives (thus, the error function completely ignores true negatives). Explicitly, our loss function is the following:

$$L = \log\left(\frac{1+S^{'}}{1+S^{*}}\right) - \alpha \sum_{\substack{(x,y)\in C^* \\ Z(x,y)\leq 0}} Z(x,y) + \beta \sum_{\substack{(x,y)\notin C^*\cup C^{'} \\ Z(x,y)>0}} Z(x,y) \quad (4)$$

With $S^{'}$ and S^* the scores of the strongest impostor $C^{'}$ and the true model C^* respectively, with scores computed by the following:

$$S = \sum_{\substack{(x,y)\in C \\ Z(x,y)>0}} Z(x,y) \quad (5)$$

This loss function can easily be differentiated with respect to each point in Z, as follows:

$$\frac{\partial L}{\partial z_i} = \begin{cases} \frac{-1}{1+S^*} & : (x_i, y_i) \in C^*, z_i \geq 0 \\ \frac{1}{1+S^{'}} & : (x_i, y_i) \in C^{'}, z_i \geq 0 \\ -\alpha & : (x_i, y_i) \in C^*, z_i \leq 0 \\ \beta & : (x_i, y_i) \notin C^* \cup C^{'}, z_i > 0 \\ 0 & : otherwise \end{cases} \quad (6)$$

Our loss function has some interesting aspects. If a strong impostor is present within the data, this loss function will drive down the values of the points comprising the impostor. If there are no particularly strong impostors in the data, this aspect of the loss function transitions toward applying a harsher penalty to an arbitrary subset of false positives on a stochastic basis. Additionally, false negatives are always penalized.

3.5. Parameters

Our method does contain some parameters which must be specified up front (*i.e.*, parameters that are not learned or optimized directly from the data). Two of these are the weights α and β applied globally to false negatives and positives (respectively) in the loss function. β does not need to be very large; in fact, it can even be zero, as this just means only false positives detected as impostors by the RANSAC layer will contribute to backpropagation. α is a more important parameter, as setting $\alpha > 0$ is the only way to penalize false negatives. Unless otherwise specified, we used values $\alpha = 1$ and $\beta = 0.01$ for our experiments. The other important parameters are those involved in the RANSAC algorithm. This includes the tolerance for model membership in computing the scores, the number of RANSAC iterations in relation to the number of points provided as input, the criterion for labeling a proposed model as an impostor, and potential constraints for rejecting models that grossly violate feasible geometries for the object in question. An important point about the tolerance parameter in score computation is that the tolerance used for the forward and backward passes of the CNN does not necessarily have to be the same; for example, using a smaller value for the backward pass has the effect of being a bit more conservative with weight updates. Unless otherwise specified, we generally used a tolerance of 2 pixels in the forward pass and 1 pixel in the backward pass. We also applied an extremely loose upper bound on pupil radii (roughly 5 times the average radius in the data set) as a constraint. Finally, the number of RANSAC iterations was set to the number of input data points divided by 5, but capped at a maximum value of 2000.

4. Experiments

We perform several experiments using images from the CASIA-IrisV3 data set [1], which contains more than 2000 iris images from more than 249 subjects (including images of both the left and right eye for most subjects). Ground truth segmentations for this data set are publicly available [9]. We are not the first to experiment with CNNs on images of the eye - see, for example, [6] and [10] - however, as far as we are aware, no other published works evaluate segmentation CNNs with a CASIA data set. The CNN we construct for these experiments is extremely tiny, containing only a few thousand free parameters. All of our experiments were performed in MATLAB utilizing Matconvnet [25]. We did not utilize a GPU in our experiments, which was not really a problem due to the size of the CNNs (computational speed in our experiments is upwards of 5 images per second for forward pass only and 1 to 2 images per second for both forward and backward passes). For analysis of the significance of small errors in pupil localization in iris recognition and iris registration, the reader is referred to [18] (recognition) and [16] (registration).

[1]http://biometrics.idealtest.org

4.1. Base Configuration Definition

Our base network architecture has a total of 3 convolution layers. The first convolution layer contains two filters operating on the grayscale image (size $H \times W$), the first being initialized to an inverted Gaussian smoothing filter with a large positive bias and the second being initialized to a smoothing filter with a moderate negative bias. The outputs are then fed to a ReLU layer, with the result that the nonzero pixels in the first output channel of this ReLU layer belong almost exclusively to the pupil. The next convolution layer is initialized to extract horizontal and vertical edges from the first input channel using basic Sobel-type filters, while also convolving a family of four different orientations of a Gabor wavelet designed to have a strong response to the LED reflections - hence, a total of 6 output channels. These are then fed to a customized layer, which computes the euclidean norm of the first two input channels (see section 3.2) and extracts the max value over the other four channels at each pixel location to produce a second output map. These outputs are then fed to another ReLU layer, and then to a third convolution layer which is initialized to a weighted subtraction of the "clutter" channel from the "signal" channel with very heavy smoothing applied to the clutter channel. Ideally, this layer produces output which contains the entire pupil boundary and nothing else (see Figure 1). This output is then fed to our RANSAC layer as discussed in section 3.4. We train the network with a total of 35 epochs using a batch size of 30 images, momentum of 0.9, and weight decay of 0.0005, with a learning rate of 10^{-6} for the first 15 epochs and 10^{-7} for the next 20 epochs.

4.2. Base Configuration Results

The results of this experiment with 1051 training images and 1577 testing images are shown in Tables 1 and 2, and further illustrated in Figures 2 and 3. Table 1 shows the marginal but significant accuracy gains that were made in the ability to correctly identify the pupil center and radius through CNN training, while Figure 2 illustrates that the nature of much of this gain actually came in the form of removing directional bias. Additionally, Table 2 and Figure 3 show the efficacy of the final edge maps before and after training. The network after training had a huge increase in average recall with only a small decrease in average precision, good for a slightly higher average F1 score. Equally important are the dramatically lower standard deviations for these metrics, which show that the network became much more robust and more repeatable after training. Another important result is that no overfitting was observed; when evaluated on the training set, the center and radius errors are only slightly different (1.04 ± 0.54 and 0.48 ± 0.36), with the average precision, recall, and F1 score virtually identical (each within 0.002 of the corresponding test set value).

At this point, it is appropriate to wonder what exactly

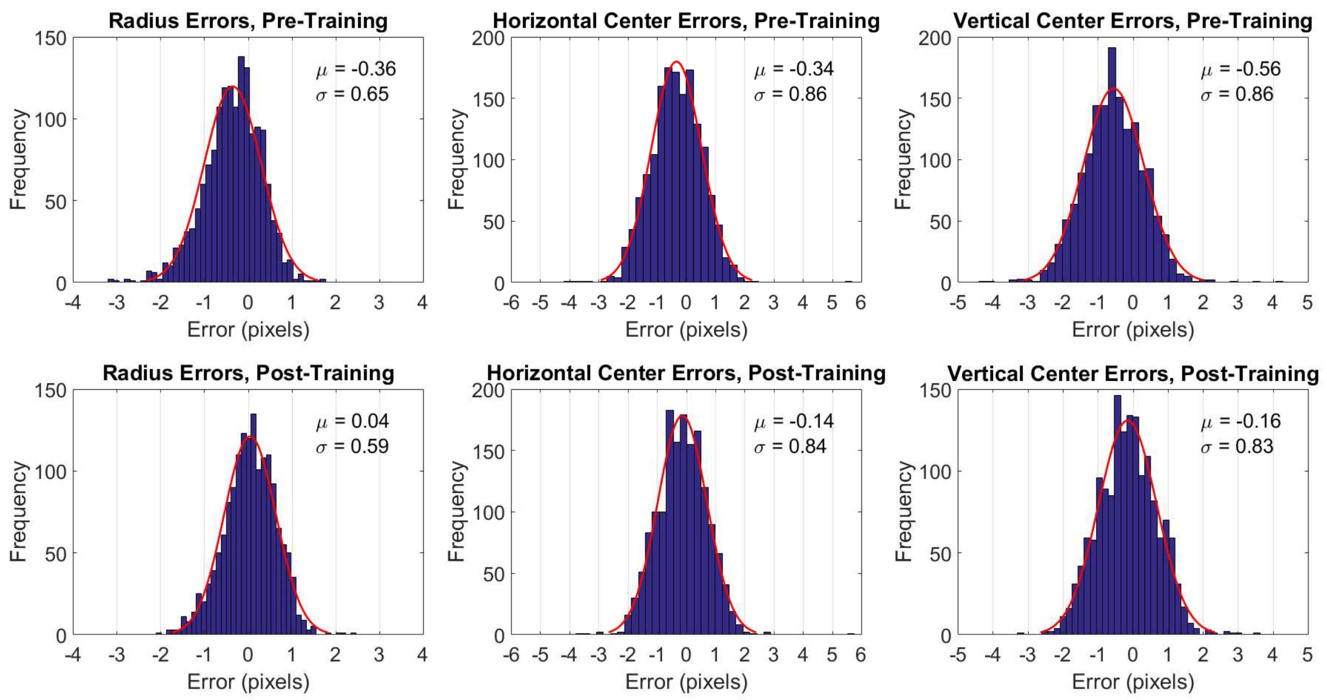

Figure 2. Pupil segmentation error distributions before (top) and after (bottom) training, using our base configuration with parameters set as described in section 3.5. For all three geometric parameters governing the best fit circle for the pupil, the initial algorithm produced measurable biases in one direction or another, and optimizing the algorithm through CNN training significantly reduced these biases.

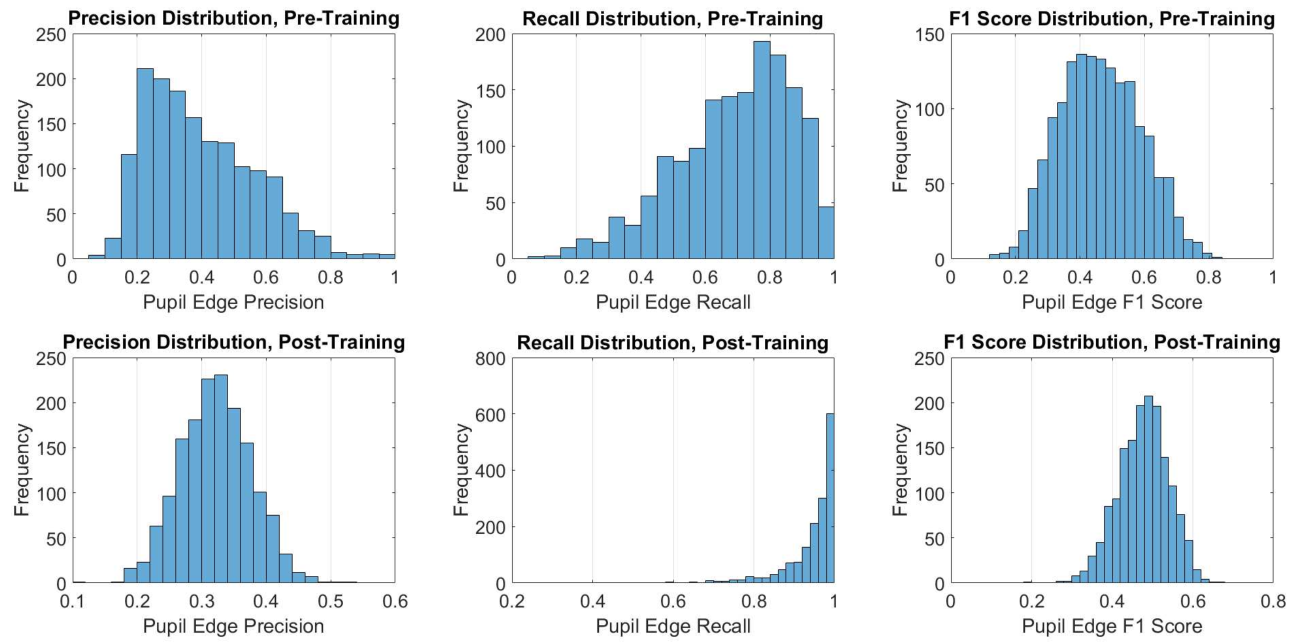

Figure 3. Pupil edge precision, recall, and F1 score distributions before (top) and after (bottom) training, using our base configuration with parameters set as described in section 3.5. Training led to dramatically improved recall and much more controlled precision (albeit with a slightly decreased average precision), thus producing F1 scores that are slightly higher on average with a much tighter distribution.

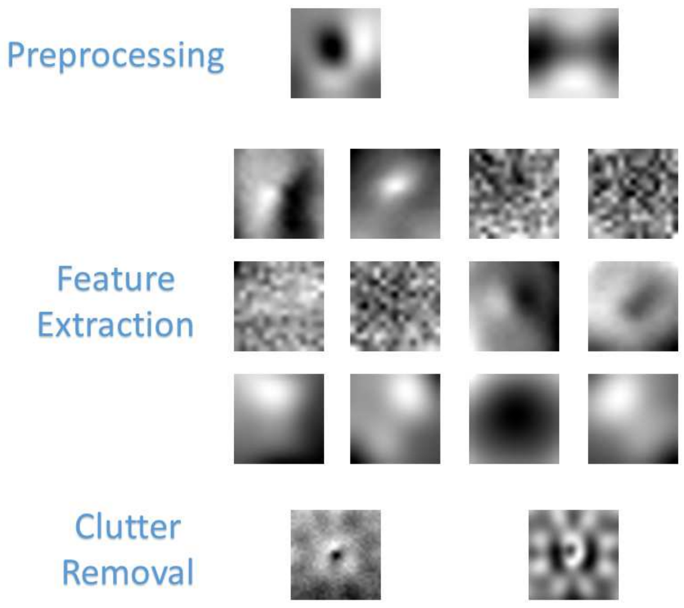

Figure 4. Learned alterations to the convolutional filters in the network, shown in the spirit of [30]. Most are extremely smooth functions, implying that these alterations are taking advantage of meaningful patterns in the images. It is interesting that the "horizontal gradient" perturbation (shown in the top left corner of the Feature Extraction group) appears to take on different characteristics at the top of the filter from the middle and bottom of the filter; one could speculate that this has something to do with some images having low-hanging eyelashes near the upper pupil boundary. It is also interesting how much more complex (yet still incredibly symmetric) the perturbations to the deepest two filters are compared to the others. These two filters are responsible for the final "subtraction" of the clutter map from the feature map to produce the final edge map.

the network learned in order to achieve these improvements. The network contains a total of 16 two-dimensional convolutional filters (some of which comprise multiple channels of a single filter operating on multichannel input) - 2 in the first convolutional layer, 12 in the second, and 2 in the third. 10 of these were specifically initialized as part of encapsulating the classical edge detection and filtering operations within the CNN, while the others were zeroed out (initialized to extremely tiny random numbers prior to training, but set to exactly zero for all "pre-training" performance evaluations). The differences between the post-training filters and the pre-training filters take the form of very smooth functions for 11 out of the 16 filters, implying that the improved results following training do indeed stem from leveraging meaningful patterns in the images. The learned filter alterations are shown in Figure 4.

4.3. Hyperparameter Variation

We performed additional experiments with varied hyperparameters. One such variation was setting the backpropagation RANSAC tolerance to 2 pixels (instead of 1). The results under this variation were very similar to the original results, the final net being more sensitive than in the prior case (average precision, recall, and F1 score of 0.24,

Measure	Center	Radius
Initial	1.20 ± 0.69	0.57 ± 0.48
Post-Training	1.06 ± 0.57	0.47 ± 0.36
Difference	0.15 ± 0.45	0.10 ± 0.42

Table 1. Accuracy results for our base configuration with parameters set as described in section 3.5. The CNN was initialized to a pretty good performance prior to any training, and this performance was improved by training on our RANSAC loss function. The results were calculated over a test set of 1577 images, with the post-training CNN having been trained on 1051 different images. See Figure 5 for some specific examples of challenging images.

Measure	Precision	Recall	F1 Score
Initial	0.40 ± 0.17	0.69 ± 0.18	0.46 ± 0.12
Post-Training	0.32 ± 0.05	0.94 ± 0.07	0.48 ± 0.06

Table 2. Edge map evaluation for our base configuration with parameters set as described in section 3.5. Training on the RANSAC loss function resulted in greatly improved recall of pupil edges with a minor decrease in mean precision, resulting in a better average F1 score. Equally important are the reduced spreads for each of these metrics after training.

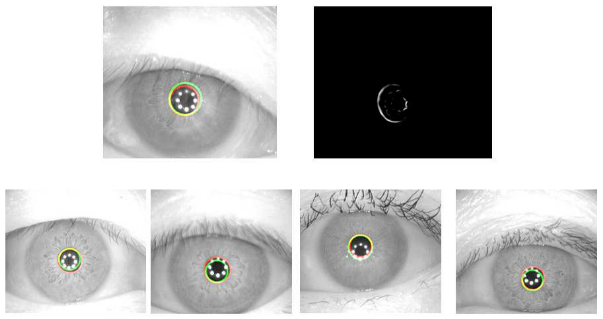

Figure 5. Illustration of challenging images in the data set. The top row shows the result and the corresponding edge map for the sole image for which the trained network makes an obvious error (green circle is network output, red circle is ground truth). The bottom row similarly shows errors made by the net prior to training, which no longer occur after training.

0.98, and 0.38) with only slightly degraded accuracy (center and radius errors 1.07 ± 0.57 and 0.50 ± 0.38). We also tried reducing the RANSAC imposter threshold, defined in terms of the sum of squared geometric parameter errors being greater than the threshold, from 80 to 15, and obtained results virtually identical to the original results (center and radius errors of 1.05 ± 0.57 and 0.47 ± 0.36; average precision, recall, and F1 score of 0.34, 0.94, and 0.50). The third variation we tried was setting $\beta = 0$, such that the only false positives that contributed to backpropagation were those belonging to detected impostors. This resulted in a much more sensitive net (average precision, recall, and F1 score of 0.028, 0.997, and 0.055) with accuracy that was still better than the initial net but clearly not as good as our other results (center and radius errors 1.14 ± 0.67 and 0.51 ± 0.46).

4.4. Alternate Configurations

We ran an experiment with all pre-initialization related to clutter removal removed. The net after training still achieved performance very close to that of the base configuration in terms of final edge map evaluation (average precision, recall, and F1 score of 0.30, 0.95, and 0.46), however in terms of finding the pupil there was one catastrophic failure in which RANSAC identified a circle that went through a combination of eyelid edges and LED edges that had not been filtered out successfully (in other words, a convincing impostor that the net failed to suppress). Excluding this case, the center and radius errors were 1.08 ± 0.65 and 0.51 ± 0.45. We repeated this experiment with extra layers added to the front of the net designed to attempt to remove the LED reflections from the image and pass the result along to the rest of the net. After training, this configuration actually produced the best edge map metrics of all our experiments, with average precision, recall, and F1 score of 0.44, 0.98, and 0.60 (the respective standard deviations were 0.07, 0.04, and 0.07), although no improvements were seen in the accuracy metrics relative to the base configuration (center and radius errors 1.06 ± 0.57 and 0.54 ± 0.42). However, it is important to point out that this configuration is not quite as good as the base configuration pre-training (center and radius errors 1.26 ± 0.70 and 0.69 ± 0.56). Thus the amount of improvement gained through training is actually somewhat more significant than for the base configuration (differences in center and radius error of 0.2 ± 0.46 and 0.15 ± 0.35).

4.5. Reduced Training Set

Encouraged by the complete lack of overfitting observed in our experiments, we explored utilizing a reduced training size. The training set was reduced by half - thus, only 525 images were utilized. We doubled the number of epochs at each learning rate such that the total amount of parameter updates remained fixed for each learning rate. Using the same test set utilized for all other experiments, the accuracy results were virtually unchanged from the base configuration (center and radius errors 1.06 ± 0.57 and 0.48 ± 0.37) with the final edge maps being slightly less sensitive (average precision, recall, and F1 score of 0.34, 0.94, and 0.50).

5. Conclusion

In this work, we successfully embedded a high-performing RANSAC segmentation algorithm for a practical problem into a CNN by hand, and achieved even better performance by fine-tuning the constructed CNN with backpropagation. The fine-tuning utilized a novel loss function based on the strongest "imposter" set detectable by RANSAC so as to directly train on what ultimately impacted segmentation performance. Our work strengthens the case for CNNs as a robust problem solving approach applicable to a wide variety of problem types and sizes. We believe that our approach of CNN encapsulation and fine-tuning with our RANSAC loss function has general application to any computer vision problem where RANSAC has been proven to be a successful method, and we look forward to experimentally investigating this in the future.

Acknowledgements

This work was supported in part by the National Science Foundation under grant #IIS-121294.

References

[1] D. A. Chernyak. Iris-based cyclotorsional image alignment method for wavefront registration. *IEEE Transactions on Biomedical Engineering*, 52(12):2032–2040, 2005.

[2] J. Daugman. How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):21–30, 2004.

[3] J. Daugman. New methods in iris recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(5):1167–1175, 2007.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] W. Fuhl, T. Santini, G. Kasneci, and E. Kasneci. Pupilnet: Convolutional neural networks for robust pupil detection. *arXiv preprint arXiv:1601.04902*, 2016.

[7] A. Gangwar, A. Joshi, A. Singh, F. Alonso-Fernandez, and J. Bigun. Irisseg: A fast and robust iris segmentation framework for non-ideal iris images. In *Biometrics (ICB), 2016 International Conference on*, pages 1–8. IEEE, 2016.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] H. Hofbauer, F. Alonso-Fernandez, P. Wild, J. Bigun, and A. Uhl. A ground truth for iris segmentation. In *22nd International Conference on Pattern Recognition, ICPR, Stockholm, Sweden, August 24-28, 2014*, pages 527–532. IEEE Computer Society, 2014.

[10] C. L. L. Jerry and M. Eizenman. Convolutional neural networks for eye detection in remote gaze estimation systems. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1. Citeseer, 2008.

[11] S. K. Kim, H.-J. Kong, J.-M. Seo, B. J. Cho, K. H. Park, J. M. Hwang, D.-M. Kim, H. Chung, and H. C. Kim. Segmentation of optic nerve head using warping and ransac. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 900–903. IEEE, 2007.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[14] D. Li, D. Winfield, and D. J. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 79–79. IEEE, 2005.

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] D. Morley and H. Foroosh. Computing cyclotorsion in refractive cataract surgery. *IEEE Transactions on Biomedical Engineering*, 63(10):2155–2168, 2016.

[17] C. Papalazarou, P. M. Rongen, and P. H. de With. Multiple model estimation for the detection of curvilinear segments in medical x-ray images using sparse-plus-dense-ransac. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2484–2487. IEEE, 2010.

[18] H. Proença and L. A. Alexandre. Iris recognition: Analysis of the error rates regarding the accuracy of the segmentation stage. *Image and vision computing*, 28(1):202–206, 2010.

[19] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. Usac: a universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2013.

[20] R. Rocha, A. Campilho, J. Silva, E. Azevedo, and R. Santos. Segmentation of ultrasound images of the carotid using ransac and cubic splines. *Computer methods and programs in biomedicine*, 101(1):94–106, 2011.

[21] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[22] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3982–3991, 2015.

[23] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging*, 35(5):1196–1206, 2016.

[24] K. Tanaka and E. Kondo. Incremental ransac for online relocation in large dynamic environments. In *2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 68–75. IEEE, 2006.

[25] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

[26] M. Waine, C. Rossa, R. Sloboda, N. Usmani, and M. Tavakoli. 3d shape visualization of curved needles in tissue from 2d ultrasound images using ransac. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4723–4728. IEEE, 2015.

[27] R. P. Wildes. Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, 85(9):1348–1363, 1997.

[28] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[29] S.-W. Yang, C.-C. Wang, and C.-H. Chang. Ransac matching: Simultaneous registration and segmentation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1905–1912. IEEE, 2010.

[30] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[31] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks

via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258, 2015.