Vision algorithms for guiding the Automated NonDestructive Inspector of aging aircraft skins

Ian Lane Davis and M. W. Siegel

The Robotics Institute, School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213 USA

ABSTRACT

Under the FAA Aging Aircraft Research Program we are developing robots to deploy conventional and, later, new-concept NDI sensors for commercial aircraft skin inspection. Our prototype robot, the Automated NonDestructive Inspector (ANDI), holds to the aircraft skin with vacuum assisted suction cups, scans an eddy current sensor, and translates across the aircraft skin via linear actuators. Color CCD video cameras will be used to align the robot with a series of rivets we wish to inspect using NDI inspection sensors. In a previous paper we provided a background scenario and described two different solutions to the alignment problem: a model-based system built around edge detection and a trainable neural network system. In this paper, we revisit the background and previous research and detail the first steps taken towards a method that will combine the neural and the model based systems: a neural edge detector.

1.0 BACKGROUND

1.1 Project Goals

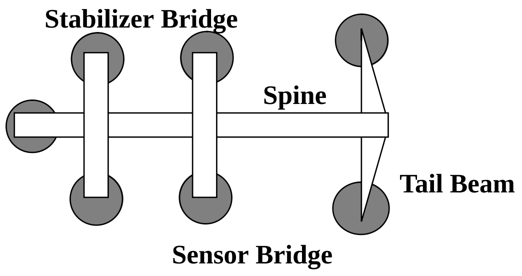
The initial goal of the Automated NonDestructive Inspector (ANDI) project is to help the human inspector to work safely, rapidly, and effectively. This will be accomplished by providing the human inspector with a tool that eliminates the need to work in awkward positions or at tedious tasks, and by ensuring that measurements will be accurate and repeatable. A later goal is to improve the productivity of inspection activity by giving the inspector computer-based instrument monitoring and visual observation tools integrated with powerful database and trend-spotting tools that provide support for making judgements and decisions. Our initial plans include giving the human inspector, who is watching and supervising directly and via TV monitors, a data stream that is at least as reliable as the data stream generated in “hands-on” inspection, to acquire data more rapidly, to identify and mark areas of interest distinctly but removably, and to archive data automatically.

1.2 Mechanical Design

1.2.1 Hardware

ANDI affixes itself to the aircraft skin with vacuum assisted suction cups. In the envisioned fully functional system, electric power, air for the vacuum ejectors, pneumatic actuators, and air-bearings, and input and output signals are transported via an umbilical attached to a safety-tether that hangs from the safety-rail above each aircraft in the maintenance hangar. At the start of an inspection sequence, ANDI is manually positioned close to a manufacturing reference position such that ANDI’s main translation axis is aligned approximately along the first line of rivets to be inspected. The primary eddy current sensors (ANDI now deploys an SE Systems SMART-EDDY 3.0 with a Nortec SPO-1958 pitch-catch sliding probe), secondary sensors (e.g., cameras driving monitors on the inspector’s console), and system sensors (e.g., alignment, guidance, and navigation aids) are then used to guide the machine alternately in scans over lines of rivets, skin joints, etc, and in walks to and alignments with subsequent sections.

The robot’s body is cruciform in design. The main frame consists of a tail beam rigidly attached to a spine in a T-shape. Two “bridges” are mounted crosswise on the spine in the same plane as the tail. Each of these bridges is joined to the spine through a linear actuator that allows the bridge to translate along the length of the spine. Furthermore, each bridge has a lead screw-driven linear actuator perpendicular to the spine, and has a rotational degree of freedom about its point of attachment to the spine. This rotation is normally locked. Suction cups whose vacuum is maintained by pneumatic aspirators are mounted at the free end of the spine, each end of the tail, and at both ends of the two bridges. Walking is standard beam-walking[1]. With the spine affixed to the aircraft, the bridges release their suction cups and are propelled via the linear actuators to the end of the spine. Once the bridges are affixed to the aircraft again, the spine and tail suction cups release and the spine moves through the bridges in the same direction. This process repeats to enable walking.



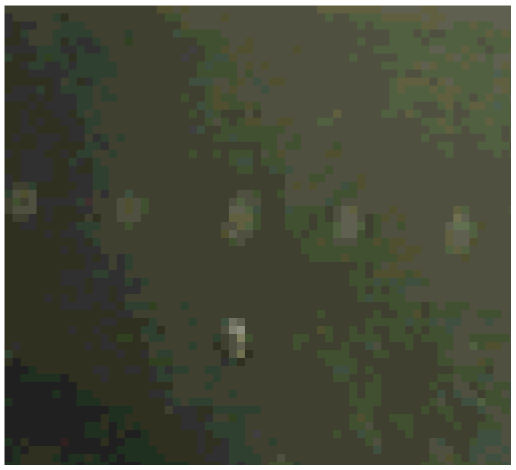


FIGURE 1. (a) Top-view diagram of ANDI hardware. Eddy current sensor is mounted on the lower (with respect to this diagram) end of the sensor bridge. The dark discs represent the suction cups. (b) Sample low resolution image taken from test data (only four bits of the eight bit grey-scale can be printed).

1.3 Vision Problem

Our first milestone is to provide the human operator a tool to eliminate the need to be in physically awkward positions on the aircraft. Towards this end, some parts of an eventually comprehensive vision system are needed immediately, e.g., components that help the control system to follow a rivet line or skin joint. The vision system under design has four cameras: one for close-up inspection (field-of-view of one rivet), one fore and one aft for alignment of the robot with lines of rivets or skin joints (field-of-view of several rivets), and one shared by proprioception and navigation systems. Design issues relating to the real-world environment include dealing with uncontrolled lighting, eliminating (or perhaps exploiting) specular reflections, and locating metal landmarks on a metal background. We envision a hybrid image understanding system with traditional algorithms for feature enhancement and neural networks for additional interpretation. The first part of the vision system that needs to be explored is the alignment system. ANDI must be able to align itself automatically with a series of rivets to be inspected. Initial efforts were reported in a previous paper[2].

The fore and aft cameras both have fields of view that include four to six rivets in a line, and both are dedicated to the alignment task. Thus, we can align the robot if we can 1) calculate a line running through the rivets seen in each camera and 2) relate the line constructed in each image to a robot-based coordinate system. Since the cameras are both rigidly attached to the robot, the problem is to construct a line passing through the centers of the appropriate rivets in the field of view of each alignment camera, where “appropriate” is defined as being those rivets that lie on the line to be scanned by the NDI inspection devices carried by the robot.

2.0 VISUAL ALIGNMENT

2.1 Image Understanding Issues

The issues involved in detecting rivets on the aircraft skin are numerous. The problem boils down to detecting metal features on a metal background. In a video camera image, we can see reflections of lights, people, and machines in the aircraft hangar. The environment is uncontrolled as numerous independent inspections and repairs are occurring at the same time as our NDI inspection. Another major influence on our image quality is that the aircraft that we are inspecting are, by definition, aging. This means that the skin is covered with numerous scratches, scuff marks, and patches on the metal, as well as being interrupted by the doors, windows, and wings that we expect to find on an aircraft. Any surface flaw in the metal, even if not large or deep enough to be worrisome from an inspection viewpoint, will show up on our image.

One important issue in understanding these images is that there are different shapes of rivets with which we must contend. Some are flat-head rivets and some are raised-head rivets, and our image understanding system must be able to detect either sort of rivet. In addition, rivet placement tolerances are large enough that often even the naked eye can see that they are neither evenly space nor co-linear. Usually they are sufficiently co-linear that a pitch-catch eddy current probe can be scanned along a mean linear path. However a pencil-probe, with its tighter localization, might have to be scanned along a path that deviates from the mean line to accommodate the actual rivet deviations. The vision system has to do an adequate job of finding the mean line and, when necessary, of characterizing the actual rivet deviations.

2.2 Problem Definition

For the early experiments that we are reporting, images were taken of a simulated aircraft panel (3.0 m by 1.8 m) that is primarily used for tests of the robot mechanism. A sequence of images was taken by manually scanning a handheld 8 mm video camera along one row of rivets. The videotape was then transferred to 3/4inch tape and digitized with a Matrox digitization board onto a Sun workstation. The images used to generate results for this paper are all from a sixty second sequence of images on the videotape. Each frame was digitized into a high resolution (480 x 512) color image and then converted by averaging 8 x 8 pixel blocks into a low resolution (60 x 64) image for real-time processing. All of the methods for locating rivets that we discuss here operated on the order of one image every few seconds (including digitization and conversion to low resolution) without optimization. These unoptimized rivet line inspection algorithms are not a system bottleneck. Converting to low resolution has the added advantages of eliminating some noise and reducing the size of a typical rivet in the image to a small enough size that any standard image processing operator can cover an entire rivet without adding too much computation.

The images used have both flat-head and raised-head rivets, have outlying rivets (not on the main scan line), are not necessarily centered over the line of rivets, and have uncontrolled lighting. The implications are that our rivet detection must be robust enough to find both types of rivets, we must fit our line to the rivets without letting the outlying rivets unduly influence the line location, we cannot use *a priori* knowledge of the rivet positions to serve as a starting point for the line fitting, and we must expect not only nasty specular effects, but possibly steep intensity gradients across the whole of each image. In the real inspection task, we will use controlled lighting and we will have some idea as to the location of the rivets from our last image, but by leaving these two parameters free, we feel that we can develop methods that will be more robust to deviations from our assumptions about lighting and position.

All three methods explored were tested on the same 40 images taken from a 60 second sequence of videotape. All three methods start from the low resolution color images (8 bits per pixel per color) and generate intensity images from which connected regions that correspond to potential rivets are extracted. A robust line fitting method[3] is then used to fit a line to the target rivets. The same images, connected region extraction algorithm, and robust line fitting algorithm are used for all methods. We are comparing only the rivet finding and localizing algorithms.

For ease of understanding, we have defined four failure modes for the algorithms. The first mode is “bad rivets failure”: if we found two or more false rivets collinear with a real rivet or three or more collinear with only each other, there is no reason that the robust line fitting would choose the real rivets over the false line of rivets. The second mode is “good rivets failure”: if the algorithm did not find at least three of the real rivets, we cannot (due to an artifact of our line fitting algorithm) fit a line properly, though this is easily remedied when there are exactly two rivets found. The third mode is “poor ratio failure”: if we found enough good rivets but still some bad rivets, the line which is fit to the good rivets might be badly skewed. The final mode which we have defined is “bad line fitting”: this mode implies that although we found only proper rivets, our robust line fitting method failed to be quite so robust and skewed the line slightly towards the outlying rivet(s). This often occurs when the outlying rivet is at the end of a series of rivets, so that a line that passed through the outlier is not at such a large angle from the line we wish to find.

2.3 Methods

2.3.1 Common Operators

The nice thing about most of the background material on which we find rivets is that it is largely flat and “feature-sparse”. The bad thing about it is that it’s polished metal, so we get reflections of hangars, inspectors, and parts of the robot. Any common image understanding operator, including edge detectors, variance detectors, and any number of feature detectors - finds most rivets. It also finds everything else with the same scale as rivets. Thus, the problem is best thought of not as “how do we find rivet stuff?” but rather “how do we avoid finding other stuff?” The answer is that we start with methods that turn up as few false positives as possible and missing as few real rivets as possible, and then we use our prior knowledge of what a rivet is supposed to look like to eliminate false rivets. Thus, known things we find that have the same scale as rivets, but also have other distinguishing features can be eliminated. Unknown things that lack significant features of rivets can also be eliminated. Finally, we know that the rivets we are concerned with will be along a common line.

2.3.2 Model Based Edge Detection Method

The first rivet finding method that we discuss here attempts to exploit rather than eliminate specular reflections. The down side of specular reflections is self-evident: we generate false features on the metal background material. The up side, fortunately, requires two brief observations. First, no matter where the light is coming from (with the pathological exception of directly normal to the surface, which could not happen, since the camera is there), if there is a break in the metal skin, one edge of the metal or the other will reflect the light relatively brightly and the other edge will remain darker. Second, rivets can be described as circular breaks in the metal skin. Thus, if we look for sharp changes in the intensity gradients of an image, the edges of the rivets should jump right out at us. Sadly, scratches are also breaks in the metal skin, and dents and light reflected off of the metal frequently look like breaks in the aircraft skin. Even looking for circular edges is difficult because common reflections include reflections of light bulbs which are often circular. Fortunately, we also know how big rivets should appear to us. Thus, we wish to fold our knowledge of the general size and shape of rivets into our edge detection algorithm in order to make the rivet finder a better discriminator between rivets and other features.

The details of this method are straightforward. We start with a low resolution image such as the one shown in Figure 1b. Then we run a standard 9 x 9 Canny edge detection operator[4] over the image separately in each color band. The Canny operator computes the intensity gradients of the images and suppresses those which are not local maxima. The reason for using the operator in each of the three color bands is that actual breaks in the metal, such as rivets, give strong edge readings in all bands, and most reflections don’t. Therefore, we take the results of each of the three edge detection scans and AND them together, so that a pixel is defined to be part of an edge if and only if it is part of an edge in all three bands. As we can see in Figure 2a, this method eliminates a considerable amount of noise.

As we can see the edges around the rivets are not fully connected in all three color bands in all cases. Using just the combined image from all three color bands we grow the regions of edge by using an eight connected grassfire transformation[5]. The primary effect of this is to join regions that are separated by one or two pixels. This completes the circles for all of the rivets in the image shown and in all of the rivets in the test images which have sufficient illumination. A secondary effect at this resolution is that the circles around the rivets are filled in and the features we have found are solid “blobs.”

At this point, each image is a 60 x 64 binary image, and a recursive connected region extraction method[6]is used to pull all of the found blobs out of the image. This connected region extraction (which is the same as used later by the neural network method) compiles certain basic statistics about each of the blobs. We to try to discriminate rivet blobs from other feature blobs using three of these statistics. One is simply the area of the blob in pixels. The second is the aspect ratio of the blob (length/height). The third is the percentage of the bounding rectangle of the blob which is filled. The criteria for rivets is area between 25 and 60 pixels, aspect ratio between 0.55 and 1.7, and fill percentage between 0.45 and 1.0. All of these ranges were chosen by trial and error. The centroids of all blobs that survive these thresholds are then fed into the robust line fitter and a line is fit through their centers as in Figure 2b. The robust line fitter allows us not only to ignore outlying rivets, but also throws out some false rivets.

On 40 test images, we had five “bad rivets” failures, four “bad line fitting” failures, and one “bad ratio” failure. Clearly, the parameters of each stage of this method could be tweaked exhaustively and painstakingly to work better over this set of images. In general, however, the method as-is has been found to be very good at finding real rivets, though not quite as good at distinguishing them from other features. The sum total of the knowledge which we bring to this method is that rivets have edges, they are not too little and not too big, they tend to be fairly filled in, and they are roughly squarish (actually circular, but we don’t exploit the distinction at this resolution). This gets us partway there, but our model of *rivethood* (the state of being a rivet) is only defined loosely by the knowledge we are exploiting.

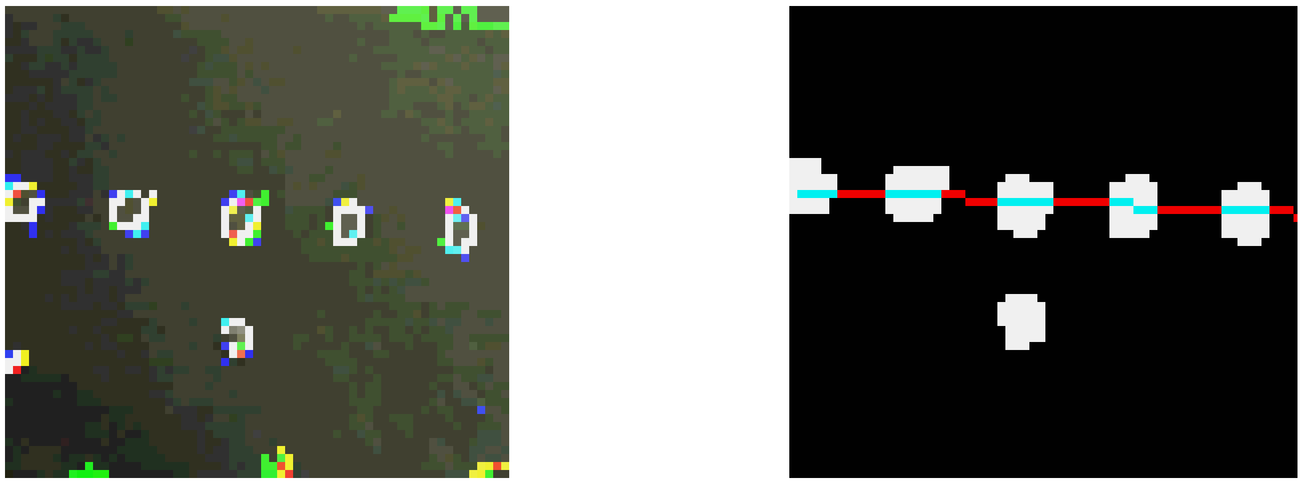


FIGURE 2. (a) Results of running Canny edge detector on all three color bands separately. The five bright regions in a row in the middle of the image and the one bright region in the middle below the third region in the row are rivets. (b) Result of heuristic blob evaluation and robust line fitter; the outlying rivet does not affect the line.

2.3.3 Neural Method

This looseness of feature definition brings us to the second method, training a custom operator via a neural network. What motivates this method is that we wish to take full advantage of our own ability to preceive rivets in an image. And the best way for us to define what a rivet looks like is to take a sample of images and use our full cognitive powers (or as much as is necessary!) to say “here are the rivets.” We then show the neural network a spanning set of examples of the two sets of $rivets$ (feautres that are rivets) and $\neg rivets$ (features that are not rivets) in image space and let the training algorithm iteratively create a mapping from that image space to the space containing the two sets. The set of $rivets$ must be fairly representative of the rivets that we find in all of our images and the set of $\neg rivets$ should include examples of all the different kinds of false rivets.

With a sufficiently large and varied set of training points and a sufficiently small number of processing units in the network architecture, the neural network will learn to generalize by extracting the significant features of rivets and the signatures of false rivets. It should be able to identify almost any part of a test image as belonging to the set of either $rivets$ or $\neg rivets$. If the training set is too small, the trained network may not be able to positively identify all of the rivets which we show it. If the number of processing units in the network is too high, the network will generalize poorly and find many false positives. Once we have a reasonable training set and network architecture, we iteratively train the network to produce the correct mapping and stop training when the learning algorithm is no longer improving the mapping.

The input processing units of the neural network are small retinas that sample subwindows of the image in question. The output of the network is a single continuous valued number that tells us degree of rivethood, much as the output of a Canny edge detector tells us degree of edgeness.

Our network architecture is what we call an "operator architecture," which is a standard backpropagation[7] three layer feedforward network with a single output unit. We decided through trial and error that five hidden units were sufficient to do a decent mapping approximation. Since we did not want to build into the network the assumption that the features which add up to rivethood are color independent, we used one input retina for each color band (red, green, and blue). The operator window size (input retina size) is 7 x 7 pixels in each color band, and a typical rivet just fills this retina. The output layer is a single unit. The input units are fully connected to the hidden units, that, are fully connected to the output unit. We have a 147 (three 7 x 7 retinas) -> 5 -> 1 network with 746 connections (weights to be adjusted during learning). A diagram of the network is shown in Figure 3.

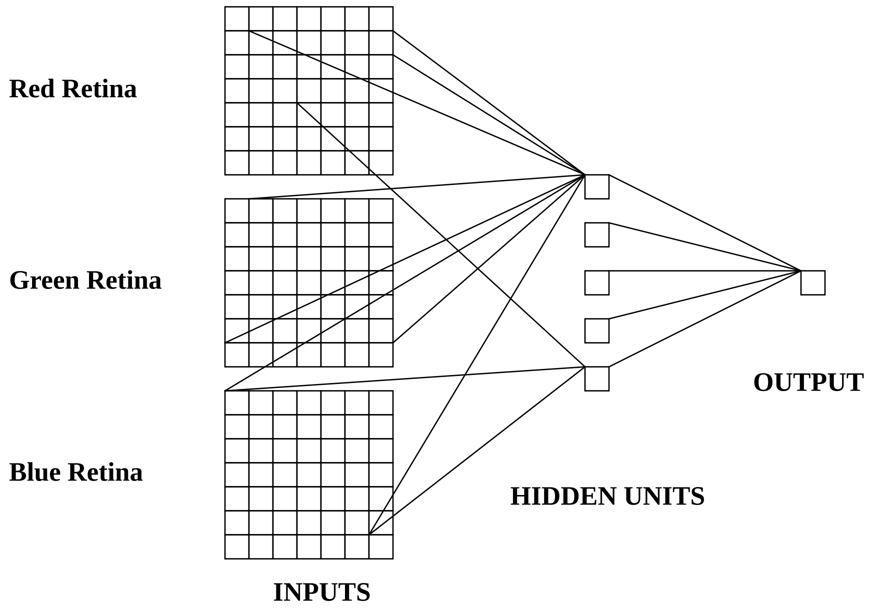


FIGURE 3. Architecture of operator network. Actual network is "fully connected." 147 -> 5 -> 1

Our training set consisted of windows taken from 43 images that were not part of the test set. They came from a slightly later section of videotape, but were taken with that same lighting and at the same distance from the same test panel as the images from the training set. Each image was used for 23 training pairs consisting of an input window and a statement of whether this window looks at a rivet or not. The rivets were identified in each image by eye. Roughly 55% of all input pairs showed a rivet or a part of a rivet. All images were given a target value of either 100% $rivet$ or 100% $\neg rivet$. Note that the training set included some false positives and some false negatives because once a rivet was identified we defined a disc about the center of the rivet as containing "rivet stuff" and occasionally a rivet would extend beyond the disc or would not perfectly fill the disc due to image distortion (often caused by the camera being tilted). Thus, we had a total of 989 "windows" from 43 images.

Due to the relatively large training set containing some false positives and negatives, 10000 epochs (complete passes through the training set, including iterative mapping adjustment) were used to train the network to reliability. On the available Sparc II IPX workstation this endeavor took less than thirty minutes. The total number of training sets seen was 9,890,000, and each training pair required one forward and one backward propagation of activation through the network.

For the test images, we used the fully trained multi-layer neural operator to scan over each pixel of each image (except those within 3 pixels of the edge of the image. This required 3132 forward passes of activation propagation through the network per image. This generated an intensity image in which high intensity corresponds to rivethood and low intensity corresponds to not rivethood. This image was thresholded and then connected regions were extracted as with the edge detection method. Each image represents a test set of 58x54x40 or 125,280 instances for a total of 5,011,200 test cases (see Figure 4 for sample).

The network performed well. In the 40 test images, the neural based image understanding chain failed to identify the proper line in two images due to "good rivet" failure and on two because of "bad ratio" failure. When an intermediate grassfire transformation stage was added to connect noisy regions of high rivethood by growing them, we failed on two due to "bad rivet" failures, on one because of "bad ratio", and once because of "bad line fitting" failure.

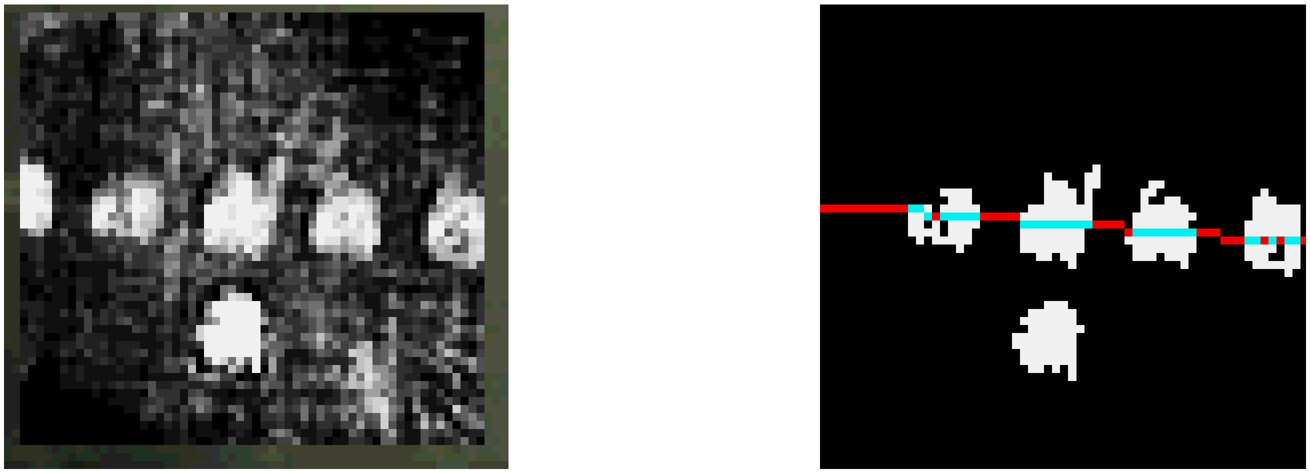


FIGURE 4. (a) Raw intensity image of degree of rivethood calculated by neural network (b) Line fitting with rivets found by neural network method

2.3.4 Neural Edge Detector

We can see the advantages inherent in each of the two methods described so far and reported previously, and the next step is to combine the two methods. We wish to develop a method that will allow us to take direct advantage of our prior knowledge that edges are useful features while still allowing us to *learn* the relevant features of the rivets that we wish to find. Towards this end we are investigating putting our prior knowledge of task and domain into a neural network that can then be combined with the neural scheme described in the previous section. Our goal is to train a neural network to recognize edges in an image and then to use the internal representations learned by that network as additional corroborating inputs to another network trained to find rivets (as the network in section 2.3.3).

This first step (and the step that is newly described in this paper) is to train a neural network to find edges in our task domain as well as the Canny operator finds edges in our task domain. The disadvantage of a neural operator is that we must train it. The first advantage is that we can tailor the operator to a specific task of edge detection. More importantly, we develop the internal neural representations (patterns of weights to the hidden units) of edges which we can in the future use in other task-dependent networks such as a modification of the rivet finder network described in Section 2.3.3. The architecture for our edge detection network is similar to the architecture in the rivet finding network. We use an operator architecture in which the input to the network is a retina that holds part of the low resolution image, and whose output is a single node that gives us an estimation of *edgehood*.

To train a neural network to find general edges, we do not need the three retina input that we used in the rivet detection network. To remain “compatible” with the Canny operator (same inputs and outputs), we have only a single 7 x 7 input retina. Through trial and error, seven hidden units was found to be adequate. The architecture can be seen in Figure 5.

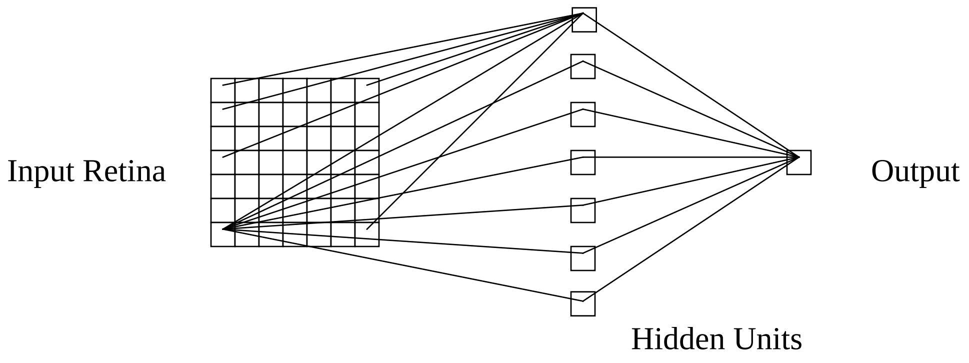


FIGURE 5. Neural edge detector architecture. 49 -> 7 -> 1 network (network is fully connected).

Construction of the training set is the main issue in the design of the neural edge detector. For purposes of comparison we desire the network to find the same edges as the Canny operator, so we could train the network by inputting random images and setting the target output to be the output from a Canny operator run over the same output. Although this route would be sufficient for developing internal neural representations (which is our main goal), we believe that we could develop a more effective edge detector by training with images in which we know all the edges.

The most obvious images in which we know the edges are the images that were used as the training set for the neural rivet detector described in the previous section. We could train from these, but we chose not to since we might be learning more about rivets than about edges, which defeats the purposes of learning specific features (that include a hoped for decrease in training time and a greater confidence in the successful learning of the network).

Our training set for the neural edge detector was constructed as a series of artificial edge images. We generated a training set of nine hundred 7 x 7 images, of which roughly 25% contained no edge at all, and had just a blank background of a random intensity. All of the remaining images had an edge that was positioned randomly. In half of these the edge was a line drawn over a background of a random intensity, and in the other half the edge was the meeting of two patches of different intensities at a line. In two-thirds of the "edge" images the edge was close enough to the center of the image for the output of the operator to be targeted as "edge" and in the other third the edge was far from the center and the output was targeted as "no edge." Therefore, approximately half of the images were positive instances and half were negative instances. Gaussian noise was added to each training image in an attempt to make the operator more robust in the presence of noise in the input images (each pixel's intensity could vary either positively of negatively up to 4% from the background intensity).

Training progressed for 2000 epochs, at which point the learning algorithm ceased to improve the network's performance on the training set. Performance of the network as an edge detector can be described qualitatively as excellent. In this first qualitative part of the analysis, the network was run over several images from the rivet sequence used in the rest of the paper and on several other file images. The results were quite similar to the results for a standard edge detection operator such as the Canny operator. The neural edge detector exhibited the tendency to find longer edges and be less sensitive to noise (or short length edges). This is most likely a result of the straight line edges used in training.

To provide some sort of quantitative comparison between the neural edge detector and the Canny edge detector, we simple plugged the neural edge detector into the algorithm for finding rivets through the Canny edge detector (Section 2.3.2) in place of the Canny operator. Since the two operators found slightly different types of edges, the criteria for rivet based on edges were modified slightly. The new criteria were area between 40 and 100 pixels, aspect ratio between 0.55 and 1.7 (same as before), and fill percentage between 0.60 and 1.0. The differences in criteria emphasize the fact that the neural edge detector found long and distinct edges more easily than the Canny operator and tended to give positive results at more pixels around the true edge (not always desirable behavior, but not inherently bad). See Figure 6a and 6b for a side by side comparison of the edge detectors on an image.

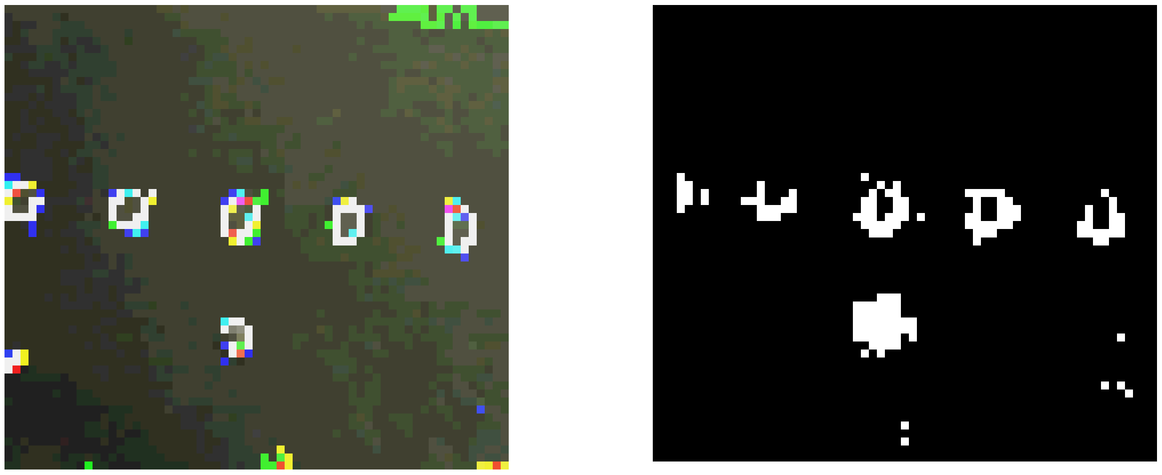


FIGURE 6. (a) Edges found with Canny operator. (b) Edges found with neural edge detector.

In our usual 40 test images, we had one "bad rivet" failure, one "bad ratio" failure, and five "bad line fitting" failure. Overall, we had three fewer failures on the test images than with the Canny edge detector. This implies that the sort of edges we trained to find corresponded very well to the sorts of edges that we might find around rivets in our test images. Furthermore, it implies that we can *learn* to find edges well enough to perform the rivet detection task on par with the results of the standard edge detection operators. Figure 7 shows rivets found with the neural edge detector. Having a neural edge detector that performs at least as well as the model based edge detector at the given task means we have encoded the information about edges neurally and can in the future use the neural representations of edges in networks that combine our knowledge of what rivets are with our knowledge that edges are important part of rivets, and we can hope to improve our rivet finding capabilities.

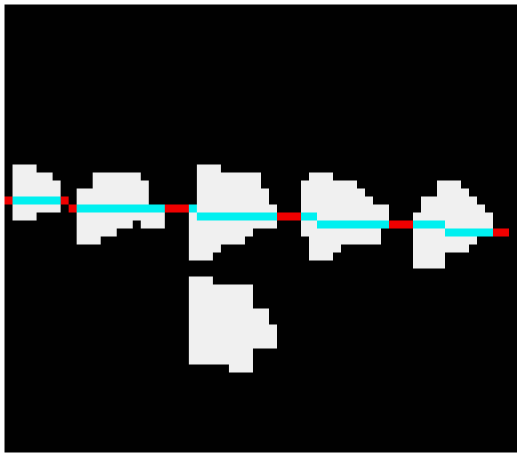


FIGURE 7. Rivets successfully found with neural edge detector.

3.0 CONCLUSIONS

In summary, we can see that both the edge detection model and the neural network method offer us the beginnings of a solution for the alignment problem for the ANDI robot. Both of these first two methods worked well with poor images that included specular reflections, scratches, and outlying rivets. Various obvious ways for improving the results of either of these two methods might offer themselves up: controlled lighting, Kalman filtering[8] on the last known line position, and focussing on the important second order information by ignoring the mean intensity and first order intensity gradient of the images, to list a few examples. But we wish to affect a more profound improvement and feel that with ongoing effort we can combine the best parts of the model based method with the best parts of the neural method.

The lesson of the edge detection method is that we can bring prior knowledge of important features to an image understanding problem such as this and apply simple techniques to generate useful information. The lesson of the neural network approach is that we can automatically extract common features of known examples of $rivet$ and $\neg rivet$. In this paper we have shown that we can train a neural network to detect specific image features; in this case, we trained a neural network to detect edges at a similar level of performance as was achieved by the Canny edge detector at the same task on the same images.

That shown, we now pursue the at-first daunting goal of combining the advantages of the model based method with those of the neural method by addressing the hopefully reduced problem of integrating the internal representations of one network (edge detection) into another network (rivet detection). By narrowly directing the learning in one network for extracting specific features and then integrating the feature detectors into a task dependent network (such as the rivet detector), we hope to be able to construct a method for quickly learning to perform complicated tasks with great confidence that we are keying off of important features. This, as applied to the ANDI project is our current line of research.

4.0 ACKNOWLEDGEMENTS

This research was funded in part by the Federal Aviation Administration (FAA) under federal grant #G0319014 to Carnegie Mellon University and in part by a National Science Foundation Graduate Research Fellowship to Ian Davis. This research is part of a joint effort between the Robotics Institute and the Carnegie Mellon Research Institute (CMRI). We would like to thank all of the members of the ANDI team, and especially Chris Alberts, Chris Carroll, Alan Guisewite, John Hudak, William Kaufman, and Court Wolfe for their technical contributions and editorial assistance. We would also like to thank Donna Anderson of the United States Bureau of Mines for her participation, and Pat Walter at Sandia National Labs for providing the simulated aircraft panel upon which tests were performed.

5.0 REFERENCES

1. *Mars Rover/Sample Return (MRSR) Rover Mobility and Surface Rendezvous Studies, Final Report*, Martin Marietta, JPL Contract No. 958073, October 1988.

2. I.L. Davis and M.W. Siegel, “NonDestructive Inspector of Aging Aircraft,” International Symposium on Measurement Technology and Intelligent Instruments, Huazhong University of Science and Technology, Wuhan, Hubei Province, People’s Republic of China, October 1993.

3. W.H. Press and B. P. Flannery and S.A. Teukolsky and W.T. Vatterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1990.

4. J. Canny, “Finding Edges and Lines in Images,” MIT AI Laboratory Technical Report 720, June, 1983.

5. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley and Sons, 1973

6. D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982

7. D.E. Rumelhart and J.L. McClelland and the PDP Research Group (ed), *Parallel Distributed Processing*, Cambridge, MA, 1986.

8. R.C. Smith and P. Cheeseman, “On the Representation of Spatial Uncertainty,” *The International Journal of Robotics Research* Vol. 5, No. 4, Cambridge, MA, 1986.