Automatic Scaling of Fish Images

D. A. Konovalov
James Cook University
Townsville, Queensland, Australia
dmitry.konovalov@jcu.edu.au

J. A. Domingos
Department of Agriculture and Fisheries
Queensland Government
Woorim, Queensland, Australia
jose.domingos@daf.qld.gov.au

R. D. White
James Cook University
Townsville, Queensland, Australia
ronald.white@jcu.edu.au

D. R. Jerry
James Cook University
Townsville, Queensland, Australia
dean.jerry@jcu.edu.au

ABSTRACT

In aquaculture breeding programs where large numbers of fish need to be rapidly phenotyped, the absolute physical dimensions of fish (in millimeters or inches) are often required to be extracted from electronic images in order to measure the size of the fish. While it is possible to infer the length of the fish in pixels, the absolute scale of the image (in pixels-per-millimeter or dots-per-inch) is largely unknown without a reference grid, or requires additional hardware, data collection and/or record-keeping management overheads. One cost and time effective solution is to capture the absolute scale by including a measuring ruler in the photographed scene and from which a computer program can automatically identify the scale of the photo and calculate fish morphometric measurements. To assist such workflow, this study developed an algorithm that automatically detects a ruler in a given image, and automatically extracts its scale as distance (in fractional number of pixels) between the ruler's graduation marks. The algorithm was applied to 445 publicly available images of barramundi or Asian seabass (*Lates calcarifer*), where a millimeter-graded ruler was included in each image. Convolutional Neural Network (CNN) was trained to segment the images into ruler, background, fish and label sections. Then the distance-extraction algorithm was applied to the ruler section of the images. The false-negative rate was less than 2%, where the ruler graduation distances could not be extracted in only 2-6 (out of 445) images even when the test images were rotated up to 90 degrees. The mean absolute relative error (MARE) of the inferred distances was 1-2%.

CCS Concepts

• Computing methodologies→Object detection • Applied computing→Imaging;

Keywords

Aquaculture; Computer vision; Barramundi; Image processing; Ruler detection

ICAIP 2018, June 16–18, 2018, Chengdu, China

ACM ISBN 978-1-4503-5295-6/17/08…$15.00
DOI: https://doi.org/

1. INTRODUCTION

With the widespread use of low-cost smart mobile phones and high-definition digital video cameras, images and videos are routinely used for record-keeping purposes. Efficiency and product quality in aquaculture could be assisted by computer vision systems [1-5]. One feasible application of computer vision is to automatically estimate fish morphological features such as length, width, weight and body area [2,6,7] on industrial scale from images or video streams for rapid high-throughput phenotyping [8]. However, typical computer-vision algorithms currently can only output the dimensions of fish in number of pixels, which then need to be converted to the actual physical units of length (millimeters or inches). A practical low-cost solution, which can simplify this 2-step process, is to have a measuring ruler present in the photographed scene. Then, to assist record keeping, a computer-vision algorithm could automatically estimate the image physical scale in fractional number of pixels per ruler's graduation. Extracting the image scale by automatically detecting a measuring ruler was the focus of this study.

Related to the automatic scaling of the digital images, Ueda *et al*. [9] applied Digital Fourier Transform (DFT) to estimate scale intervals on a ruler. The main limitation of the algorithm was an inability to achieve sub-pixel accuracy. Zambanini *et al*. [10] reported an algorithm, which also used DFT to locate the ruler in images containing ancient coins, and then to estimate the ruler scale interval. Zambanini *et al*. [10] claimed to improve upon the algorithm of Ueda *et al*. [9]. However, their proposed scale extraction method was not suitable for fish images, which contain multiple semi-periodical textures. Bhalerao and Reynolds [11] used DFT to locate the ruler similar to the algorithms in [9,10]. However, once the ruler image was located, the algorithm did not use DFT but extracted the sub-pixel scale interval by fitting the sine wave to the ruler. The algorithm was tested on rulers in forensic images. The test images and source code of the algorithm were not available, therefore it was not possible to assess if it was suitable for aquacultural images.

Konovalov *et al*. [12] developed an algorithm also based on DFT, which could automatically locate a section of a ruler in a given image, and which is referred to as RS1 (after ruler scale) hereafter. The algorithm achieved true-positive rate of 95-98% correctly locating all or parts of the ruler in 445 test images [12].

The RS1 algorithm had two main limitations. First, the algorithm could not distinguish between the DFT ruler and fish signals when the ruler image was partially obscured by fish, or when the image was too blurry. A solution based on Convolutional Neural Networks (CNNs) [13] is presented in Section 2.2. CNNs in their Deep Learning form [13] dramatically improved the state-of-the-art in computer vision object recognition and classification. The ability of CNNs to perform semantic segmentation [14-16] of images was utilized here. The second remaining limitation of RS1 was the inability to automatically identify the correct DFT frequency from a series of DFT frequencies generated by the ruler's periodical graduation marks. A solution based on autocorrelation is described in Section 2.3. In Section 3, we apply the CNNs to the test images while in Section 4 we present concluding remarks.

2. MATERIALS AND METHODS

2.1 Automatic Ruler Detection (RS1)

This study extended the RS1 ruler detection algorithm [12]. The key steps of RS1 are briefly reviewed here for completeness and to introduce notation. Let a given color image C have n_1 rows and n_2 columns of pixel values, which was represented as $n_1 \times n_2 \times 3$ tensor. The color image C was preprocessed by converting it to a gray-scale intensity image I represented by an $n_1 \times n_2$ matrix of real numbers normalized to the $[0,1]$ range. In addition, a simple high-pass filter $\mathbf{H}$ was applied to I resulting in matrix Y,

$$Y = \mathbf{H}[I] = I - \mathbf{L}[I], \tag{1}$$

where for each pixel, 3×3 enclosing matrix of gray-scale values were averaged (denoted as $\mathbf{L}$ operator), and then subtracted from the original image I.

The RS1 algorithm processed the high-passed image Y in square blocks of size l, where l was an even number and was measured in number of pixels. A set of $\{l_1, l_2, \dots, l_k\}$ block-sizes for any input image was determined by the following rule,

$$l_{\max} \equiv l_k = \min(n_1, n_2)/n_b, \tag{2}$$

$$l_{\min} \equiv l_1 = \max(8, l_{\max}/k), \tag{3}$$

where n_b was the number of blocks per smallest image dimension. The remaining $\{l_2, \dots, l_{k-1}\}$ values were uniformly distributed in logarithmic scale,

$$l_j = l_{\min} \left(\frac{l_{\max}}{l_{\min}}\right)^{\frac{j-1}{k-1}}, \tag{4}$$

where $k = 4$ and $n_b = 12$ were used. For each l_j value, both n_1 and n_2 were rounded down to be multiples of l_j, and then the $n_1 \times n_2$ centred section of the image Y was retained for further processing.

Within the RS1 algorithm, a ruler was assumed to be a large number of repeated parallel lines, where d denoted the distance between the center points of adjacent ticks (or graduation marks). The distance d was measured in fractional number of pixels per millimeter. Following [10] and [11], RS1 used the two dimensional (2-D) Discrete Fourier transform (DFT2) via its Power Spectral Density (PSD),

$$P = |\mathbf{F}[X]|^2, \tag{5}$$

where X was a 2-D matrix of real numbers, $\mathbf{F}$ was the DFT2 operation, and where P was the resulting PSD matrix with the same dimensions as X. The high-passed image Y (Eq. 1), was divided into square blocks yielding a $m_1 \times m_2$ matrix of sub-images Y_{ij}, which had m_1 rows and m_2 columns,

$$m_1 = n_1/l\,, \qquad m_2 = n_2/l. \tag{6}$$

Following [11], in order to prevent spectral leakage, each resulting block Y_{ij} was multiplied by a windowing mask. A 2-D windowing square $l \times l$ matrix was denoted by W and used before the DFT2 was applied via

$$F_{ij} = \mathbf{F}\left[\mathbf{W}[Y_{ij}]\right], \tag{7}$$

where $\mathbf{F}$ was the DFT2 operator, $\mathbf{W}$ was the windowing operator defined as element-wise multiplication of $l \times l$ matrices W and Y_{ij}. In general, the $m \times n$ windowing matrix $W(p,q)$ was defined as

$$W(p,q) = w_m(p) w_n(q), \tag{8}$$

$$w_m(p) = \sin\left(\frac{p-1}{m}\pi\right), \qquad w_n(q) = \sin\left(\frac{q-1}{n}\pi\right), \tag{9}$$

$$p = 1,2,\dots m, \qquad q = 1,2,\dots n. \tag{10}$$

The RS1 algorithm converted each Y_{ij} to the corresponding DFT2 power density P_{ij} via Eq. (7) and

$$P_{ij} = \mathbf{L}\left[|F_{ij}|^2\right], \tag{11}$$

where each P_{ij} block was a $l \times l$ matrix of numbers, and was 3×3 neighbourhood averaged using the low-pass filtering operation $\mathbf{L}$ from Eq. (1).

The maximum element in a P_{ij} block was converted to vector $\boldsymbol{q}_{ij}$, which defined the dominant periodical flow with frequency q_{ij} within the block,

$$q_{ij} = \|\boldsymbol{q}_{ij}\|. \tag{12}$$

Next, the RS1 algorithm selected a ruler blob as a subset of the P_{ij}-blocks or equivalently a set of index (i,j) pairs with identical $\boldsymbol{q}_{ij}$,

$$blob = \{(i_1, j_1), (i_2, j_2), \dots, (i_b, j_b)\}, \tag{13}$$

$$\boldsymbol{q}_b = \boldsymbol{q}_{ij}, \qquad (i,j) \in blob \tag{14}$$

where, in total, four checks were used to uniquely identify the most suitable blob, see [12] for details.

So far, the preceding description of the RS1 algorithm used the spatially non-overlapping blocks Y_{ij}. Following [11], the RS1 algorithm also implemented 50%-overlapping Z_{ij} blocks. Each Z_{ij} block of $2l \times 2l$ pixels was centered on the corresponding non-overlapping Y_{ij} block, so that the inner central $l \times l$ pixels in each Z_{ij} were identical to Y_{ij}, where $1 < i < m_1$ and $1 < j < m_2$. The overlapping blocks performed much better when a ruler was at an angle, hence only the overlapping RS1 version was used in this study. If the RS1 algorithm detected a candidate ruler blob, RS1 extracted the interval distance between two adjacent marks on the ruler (in number of pixels) via [10]

$$d(l) = 1/q_b, \qquad q_b = \|\boldsymbol{q}_b\|. \tag{15}$$

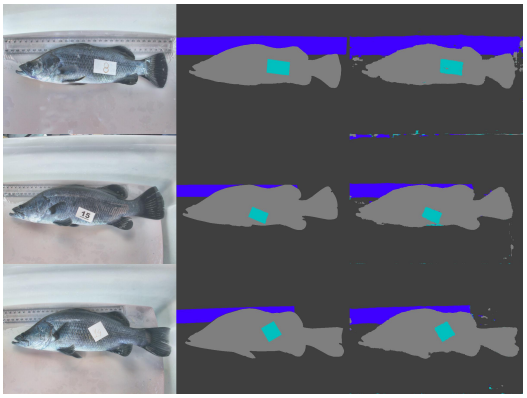

Figure 1. Validation images not used for training. Original images, segmented-by-human masks, and segmented-by-CNN masks are in first, second, and third columns, respectively.

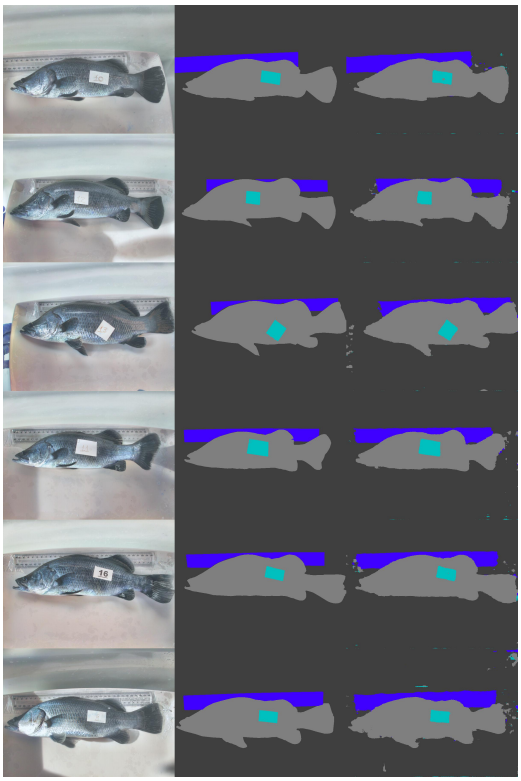

Figure 2. The same as in Figure 1 but for the images used in training.

2.2 Ruler Location via Deep Learning AI

The preceding section summarizes the RS1 algorithm, which by its construction is not able to distinguish between a rule and a fish in an image. RS1 relied on the ruler image to be sufficiently informative to generate strong DFT2 signal. However, within the aquaculture context, the semi-periodical textures of fish images could also produce strong DFT2 signals making them difficult to distinguish from the ruler's DFT2. A natural solution is to have additional image preprocessing where the image is segmented into background, ruler, and fish. Such segmentation of an image into per-pixel categories is known as semantic segmentation in the field of computer vision [14-16]. In this work, the most accurate FCN-8s model from the Fully Convolutional Networks (FCN) of [14,15] was selected as the base architecture. Itself FCN-8s is based on the VGG16 image classification network of Simonyan and Zisserman [17], which won the localization competition of ImageNet [18], Large Scale Visual Recognition Challenge 2014 (ILSVRC14) [19].

The original Caffe [20] source code for the FCN-8s model [14,15] was re-implemented in Keras [21] with TensorFlow [22] as the backend. Once the FCN-8s was rebuilt, the VGG16 convolutional layers were loaded with the ImageNet pre-trained VGG16 weights and the layers where excluded from training. The remaining convolutional layers were initialized by the uniform distribution as per [23]. The *softmax* activation function was used in the last layer. All trainable weights were regularized by a weight decay set to 1×10^{-4}. Following [15], segmentation pixel accuracy is defined as

$$PA = \frac{1}{T}\sum_i n_{ii}, \qquad T = \sum_i t_i, \qquad t_i = \sum_j n_{ij}, \tag{16}$$

where n_{ij} is the number of pixels of ith class predicted as jth class, and t_i is the total number of pixels of class i. Only nine (out of 445) images were segmented by hand using Matlab, resulting in four per-pixel binary masks: background, fish, label, and ruler pixel classes. The four mutually exclusive classes were *one-hot* encoded into $n_1\times n_2\times 4$ ground-truth tensor G, where $G(p,q,r) = 1$ if the (p,q) pixel belonged to rth class and zero otherwise. The fish tag labels were segmented out as a separate class because they overlapped with the fish-contours in this particular dataset. Three segmented images where randomly selected for cross-validation, see Figure 1. The remaining six segmented images were used for training, see Figure 2. The very small number of training images was intentional and used to confirm the minimum amount of human effort required for this algorithm to be applied to different fish species and/or different geometric setup.

The following preprocessing was done for all 445 images: (i) Each image and if available the corresponding training mask were resized to 600×800 pixels; (ii) Following [24], each RGB image component was histogram equalized using the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm [25]. The training of the FCN-8s model was done in cycles. For each cycle, the pre-processed images and the corresponding masks were randomly rotated within ±20 degree range, randomly scaled in the range of $[0.6,1.4]$, then cropped to 512×512 pixels, and randomly flipped horizontally and vertically. The standard categorical cross-entropy loss function was modified to include categorical weights, where class and category are used here as equivalent terms. The weights at each cycle were set inversely proportional to the total number of pixels of each category. Typical weights were approximately 0.05, 0.15, 2.9, 0.9 for the background, fish, label, and ruler, respectively. Keras implementation of Adam [26] with default parameters was used as the optimizer. Early stopping was used if the loss value did not improve after 2 epochs. One feed-forward and one back-propagation passes through all training cropped images was considered to be one epoch. Training was done in batches with the batch size set to one training image. The maximum number of epoch within each cycle was 16. The total number of cycles was up to 100.

Segmentation pixel accuracy (Eq. 16) was around $PA = 97\%$ and 96% for the six training and three cross-validation images, respectively. All training was done on a desktop computer, where the training took approximately one day. The same training was also performed on a GPU (Nvidia GeForce GTX1070), where the complete training took approximately one hour. Very similar

results were obtained regardless of which three images were reserved for cross-validation. On completion of training, the model was applied to the pre-processed 445 images, resulting in 600×800×4 segmentation predictions, with typical examples presented in the last column of Figure 1.

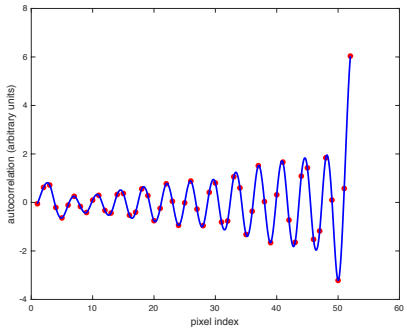

Figure 3. Interval distance extraction by autocorrelation from the ruler blob. Red dots represent S_h values from Eq. (23). Solid line represented the cubic spline interpolation of S_h for sub-pixel index values.

2.3 Distance from autocorrelation (RS2)

In approximately 10% of test images [12], the dominant DFT2 power density peak corresponded to $q_b = 2/d$ and yielded half of the actual distance $d(l) = d/2$. In this section, a conceptually simple distance-extraction procedure is described, which is based on autocorrelation and could be implemented using functions readily available in Matlab/Octave.

The RS1 algorithm from Section 2.1 was applied to a given $n_1 \times n_2 \times 3$ color image C with the following modifications. The binary 600×800 ruler mask was predicted by the FCN-8s model from the preceding section and resized to $n_1 \times n_2$ mask G. Then the ruler mask G was applied to the high-passed Y (Eq. 1). RS1 continued without any other changes except for retaining all the local maximum peaks of the total DFT2 power density of the blob,

$$P(m,n) = \sum_{(i,j)\in blob} P_{ij}(m,n). \tag{17}$$

The flow vectors $\{\boldsymbol{q}_\beta\}$ corresponding to the local maximum peaks were retained if they had approximately the same (or opposite) direction as the dominant flow $\boldsymbol{q}_b$,

$$\left|(\boldsymbol{q}_b \cdot \boldsymbol{q}_\beta)\right| > 0.9 \times q_b q_\beta, \tag{18}$$

where $\boldsymbol{q}_b$ from Eq. (14) corresponded to the absolute maximum. Given the set of candidate flows $\{\boldsymbol{q}_\beta\}$ including the dominant flow $\boldsymbol{q}_b$, the RS2 algorithm selected the most suitable frequency as follows. The ruler mask G was applied to the gray-image I, which was then divided into square 50%-overlapping blocks I_{ij} in exactly the same manner as it was done for the high-passed 50%-overlapping blocks Z_{ij}, see the end of Section 2.1. Next, the autocorrelation matrix R_{ij} was calculated for each I_{ij} from the RS1 ruler blob, $(i,j) \in blob$. All R_{ij} were added resulting in the total autocorrelation matrix of the blob R,

$$R_{ij}(k,p) = \sum_{m,n} I_{ij}(m,n) I_{ij}(m-k+l, n-p+l), \tag{19}$$

$$R(k,p) = \sum_{(i,j)\in blob} R_{ij}(k,p), \tag{20}$$

where $1 \le k,p \le (2l-1)$, and where Matlab/Octave's *xcorr2* function was used to calculate Eq. (19). The matrix R was then rotated by angle θ determined from the RS1 blob flow vector $\boldsymbol{q}_b$ as

$$\theta = \arctan\left(\frac{q_b(1)}{q_b(2)}\right), \tag{21}$$

obtaining the rotated R_θ in which the ruler was expected to be positioned horizontally. The rotated 2-D autocorrelation image R_θ was converted to a 1-D signal vector S by adding row values in each column,

$$S(n) = \sum_k R_\theta(k,n), \qquad 1 \le n \le l, \tag{22}$$

where only the first half of the 1-D signal was retained due to S being symmetric around the middle (index l). Next, S was detrended by applying a high-pass filter operator $\mathbf{H}_1$ arriving at S_h,

$$S_h = \mathbf{H}_1[S], \tag{23}$$

where $\mathbf{H}_1$ was the 1-D equivalent of the 2-D filter $\mathbf{H}$ from Eq. (1).

From the properties of autocorrelation (Eq. 19), the largest peak of $S_h(p)$ was expected to be located at $p = l$, where the signal was not shifted. The next closest peak corresponded to the shift in the ruler signal where the neighboring graduation markers overlapped. In order to extract its sub-pixel distance value, $S_h(p)$ vector was cubic spline interpolated for 0.01 index step, see typical example in Figure 3. In the majority of test images, S_h could be used to extract the ruler graduation distance. However in some test cases and especially when the ruler was initially rotated, S_h had its absolute maximum shifted from $p = l$ mainly due to error in the angle θ (Eq. 21). To overcome that issue, it was found that if S_h was further processed by autocorrelation operation, the resulted S_x was a cleaner signal for both sharp and blurry 2-D ruler images,

$$S_x(k_i) = \sum_j S_h(k_j) S_h(k_j - k_i + l), \tag{24}$$

where k_i and k_j are the fractional index coordinates. Peak locations k_i and amplitudes a_i of the S_x-curve were found (via the Matlab/Octave's *findpeaks*) denoting them as

$$a_i = S_x(k_i). \tag{25}$$

The interval distance (denoted as d_X) was then calculated as the difference between the maximum peak at $k_0 = l$ and a peak k_j closest to it by the 2-D Euclidian metric,

$$d_X = k_0 - k_j, \qquad D_j = \min D_i, \tag{26}$$

$$D_i = \left[\left(1 - \frac{a_i}{a_0}\right)^2 + \frac{(k_0 - k_i)^2}{l^2}\right]^{\frac{1}{2}}, \tag{27}$$

where this 2-D metric filtered out some spurious peaks. The RS2 algorithm concludes by selecting $\boldsymbol{q}_\beta$ (Eq. 18) using the direct conversion of Fourier frequency to the interval distance from Eq. (15), the distance $d_\beta = 1/q_\beta$ closest to the distance d_X. To allow for small noise variations, Δd_X error was permitted,

$$\left|d_\beta - d_X\right| \le \Delta d_X,$$

where $\Delta d_X = \sqrt{2}$ was used to the accommodate maximum possible error of one pixel in both horizontal and vertical directions. If such matching d_β did not exist, then the preceding RS2 steps in this Section were re-run with the grey-intensity I_{ij} blocks replaced by the high-passed Z_{ij} blocks.

The RS2 distance-extraction algorithm was run for each block-sizes $\{l_1, l_2, l_3, l_4\}$ (Eq. 4) resulting in $\{d_1, d_2, d_3, d_4\}$, where $d_j \equiv d_\beta(l_j)$. As per [12], median of non-zero dj values was used as the final result,

$$p = \text{median}(d_j), \qquad d_j \neq 0, \tag{28}$$

where zero d_j values were used as the indicator that the algorithm could not extract the distance interval for a particular l_j.

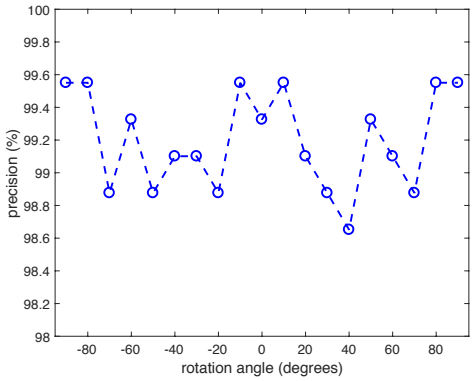

Figure 4. Algorithm precision (Eq. 31) on the test dataset of 445 Barramundi images as a function of image rotation angle.

3. RESULTS AND DISCUSSION

The presented RS2 algorithm was tested on the publicly available [12] images of *Lates calcarifer* (barramundi or Asian seabass), $n = 445$. In each test image, the ruler length L_{px} was measured manually (in pixels) using ImageJ software [27]. The physical ruler length was $L_{mm} = 300$ millimeters, thus arriving at the actual interval distance $y = L_{px}/L_{mm}$. For the ith test image, the measured-by-human and predicted-by-algorithm distances were denoted by y_i and p_i, respectively. The performance of the algorithm was measured by the absolute relative error (ARE) and the mean absolute relative error (MARE) [2,7],

$$ARE_i = 100\times |y_i - p_i|/y_i, \tag{29}$$

$$MARE = \frac{1}{TP}\sum_i ARE_i, \tag{30}$$

where n is the number of test images, and TP is the number of true-positives when RS2 extracted $p_i \neq 0$. Since every test image contained the ruler, the standard two-class classifier precision metric [28] was also used

$$precision = 100\times \frac{TP}{n}. \tag{31}$$

Figure 4 demonstrates how the RS2 algorithm is robust to the ruler orientation achieving 98.5-99.5% precision rates (Eq. 31), where the windowing mask (Eq. 8) was applied to each of the test images before rotation to reduce possible edge effects [12]. For all rotation angles, only 2-6 (out of 445) images failed to yield the distance value ($p_i = 0$) or the ARE (Eq. 29) was greater than 10%. Similarly, Figure 5 confirms the algorithm's ability to handle arbitrary ruler orientations achieving 1.5-2% MARE. Since the random rotations were included in the FCN-8s model training, the rotations had no noticeable effect on the AI segmentation results. For the original non-rotated test images, the predicted distances were fitted by the linear regression model in Figure 6 achieving the coefficient of determination $R^2 = 0.99$ and negligible bias of less than 0.01 pixel (from the model fit $p_i = 1.014y_i - 0.009$).

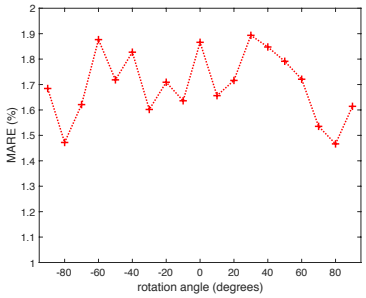

Figure 5. Same as Figure 4 but for MARE (Eq. 30).

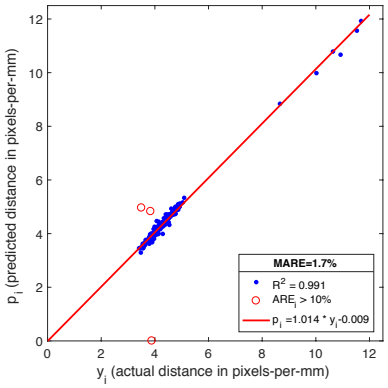

Figure 6. Predicted distances vs. the actual distances for the original not-rotated 445 test images.

4. CONCLUSIONS

The RS2 algorithm developed in this study completed the work started in the RS1 algorithm [12]. Working together, the RS2 and RS1 algorithms achieved reliable and accurate (to at least 1-2%) automated extraction of ruler scale. RS2 successfully leveraged the advantages provided by the modern convolutional neural networks (CNN) and the Digital Fourier Transform (DFT), where CNN exceled in image segmentation while DFT delivered accurate ruler graduation distance extraction. The algorithm is expected to provide additional quality assurance of the human record keeping. Already in this study a number of human measuring errors were found and corrected by RS2. The long-term goal for the algorithm is to become reliable enough in production environment and deliver cost saving alternative to the human record keeping procedures when embedded in the next-generation phenotyping platforms.

5. ACKNOWLEDGMENTS

D.A.K. acknowledges the use of Nvidia Tesla GPU from Queensland Research and Innovation Services Cloud (QRIScloud), and access to Nvidia GTX1070 GPU from JCU IoT Program.

6. REFERENCES

[1] Hong, H., Yang, X., You, Z., Cheng, F. 2014. Visual quality detection of aquatic products using machine vision. *Aquacult. Eng.* 63, 62-71. DOI= http://dx.doi.org/10.1016/j.aquaeng.2014.10.003.

[2] Miranda, J.M., Romero, M. 2017. A prototype to measure rainbow trout's length using image processing. *Aquacult. Eng.* 76, 41-49. DOI= http://dx.doi.org/10.1016/j.aquaeng.2017.01.003.

[3] Zion, B. 2012. The use of computer vision technologies in aquaculture a review. *Comput. Electron. Agr.* 88, 125-132.

DOI= http://dx.doi.org/10.1016/j.compag.2012.07.010.

[4] Zion, B., Shklyar, A., Karplus, I. 2000. In-vivo fish sorting by computer vision. *Aquacult. Eng.* 22, 165-179. DOI= http://dx.doi.org/10.1016/S0144-8609(99)00037-0.

[5] Saberioon, M., Gholizadeh, A., Cisar, P., Pautsina, A., Urban, J. 2016. Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues. *Rev. Aquacult.* 9, 369-387. DOI= http://dx.doi.org/10.1111/raq.12143.

[6] Costa, C., Antonucci, F., Boglione, C., Menesatti, P., Vandeputte, M., Chatain, B. 2013. Automated sorting for size, sex and skeletal anomalies of cultured seabass using external shape analysis. *Aquacult. Eng.* 52, 58-64. DOI= http://dx.doi.org/10.1016/j.aquaeng.2012.09.001.

[7] Viazzi, S., Van Hoestenberghe, S., Goddeeris, B.M., Berckmans, D. 2015. Automatic mass estimation of jade perch scortum barcoo by computer vision. *Aquacult. Eng.* 64, 42-48. DOI=http://dx.doi.org/10.1016/j.aquaeng.2014.11.003.

[8] Cobb, J.N., DeClerck, G., Greenberg, A., Clark, R., McCouch, S. 2013. Next-generation phenotyping: requirements and strategies for enhancing our understanding of genotype-phenotype relationships and its relevance to crop improvement. *Theor. Appl. Genet.* 126, 867-887. DOI=http://dx.doi.org/10.1007/s00122-013-2066-0.

[9] Ueda, K., Baba, T., Nakagawa, Y., Amano, K. 2005. Detection of scale intervals in digital images. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*. 1232-1232. DOI= http://dx.doi.org/10.1109/ICDE.2005.211.

[10] Zambanini, S., Herrmann, M., Kampel, M. 2013. An Automatic Method to Determine the Diameter of Historical Coins in Images. *Scientific Computing and Cultural Heritage. Contributions in Mathematical and Computational Sciences.* 3. Springer, Berlin, Heidelberg. 99-106. DOI= http://dx.doi.org/10.1007/978-3-642-28021-4_11.

[11] Bhalerao, A., Reynolds, G. 2014. Ruler detection for autoscaling forensic images. *Int. J. Digit. Crime For.* 6, 9-27. DOI=http://dx.doi.org/10.4018/ijdcf.2014010102.

[12] Konovalov, D.A., Domingos, J.A., Bajema, C., White, R.D., Jerry, D.R. 2017. Ruler detection for automatic scaling of fish images, In *Proceedings of the International Conference on Advances in Image Processing* (Bangkok, Thailand). ACM, New York, NY, 90-95. DOI=http://doi.acm.org/10.1145/3133264.3133271.

[13] LeCun, Y., Bengio, Y., Hinton, G. 2015. Deep learning. *Nature* 521, 436-444. DOI= http://dx.doi.org/10.1038/nature14539.

[14] Shelhamer, E., Long, J., Darrell, T. 2016. Fully Convolutional Models for Semantic Segmentation. *PAMI 2016*. https://arxiv.org/abs/1411.4038.

[15] Long, J., Shelhamer, E., Darrell, T. 2015. Fully Convolutional Models for Semantic Segmentation. *CVPR 2015*. https://arxiv.org/abs/1411.4038v2.

[16] Ronneberger, O., Fischer, P., Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Lecture Notes in Computer Science*, 9351. Springer, Cham, 234-241. DOI= http://dx.doi.org/10.1007/978-3-319-24574-4_28.

[17] Simonyan, K., Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*. https://arxiv.org/abs/1409.1556 .

[18] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database, In *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, 248-255. DOI= http://dx.doi.org/10.1109/CVPR.2009.5206848.

[19] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bern- stein, M., Berg, A.C., Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis*. 115, 211-252. DOI= http://dx.doi.org/10.1007/s11263-015-0816-y.

[20] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia* (Orlando, Florida, USA) ACM, New York, NY. 675-678. DOI=http://dx.doi.org/10.1145/2647868.2654889.

[21] Chollet, F., et al. 2015. Keras. https://github.com/fchollet/keras.

[22] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Lev- enberg, J., Man´e, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Vi´egas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X. 2015. TensorFlow: Large- scale machine learning on heterogeneous systems. http://tensorflow.org.

[23] Glorot, X., Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Sardinia, Italy). *PMLR* 9, 249-256. http://proceedings.mlr.press/v9/glorot10a.html.

[24] Badrinarayanan, V., Kendall, A., Cipolla, R. 2017. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 2481-2495. DOI= http://dx.doi.org/10.1109/TPAMI.2016.2644615.

[25] Zuiderveld, K. 1994. Contrast Limited Adaptive Histogram Equalization. *Graphics Gems IV*. Academic Press Professional Inc., San Diego, CA. 474-485.

[26] Kingma, D.P., Ba, J. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, http://arxiv.org/abs/1412.6980.

[27] Schneider, C.A., Rasband, W.S., Eliceiri, K.W. 2012. NIH Image to Imagej: 25 years of image analysis. *Nat. Meth.* 9, 671-675. DOI=http://dx.doi.org/10.1038/nmeth.2089.

[28] Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recog. Lett.* 27, 861-874. DOI= http://doi.org/10.1016/j.patrec.2005.10.010.