Ruler Detection for Automatic Scaling of Fish Images

D. A. Konovalov
Rapid Assessment Unit
ARC Hub for Advanced Prawn Breeding,
James Cook University, Townsville, Queensland, Australia
dmitry.konovalov@jcu.edu.au

J. A. Domingos
ARC Hub for Advanced Prawn Breeding,
James Cook University, Townsville, Queensland, Australia
jose.domingos@jcu.edu.au

C. Bajema
ARC Hub for Advanced Prawn Breeding,
James Cook University, Townsville, Queensland, Australia
casey.bajema1@jcu.edu.au

R. D. White
Rapid Assessment Unit
ARC Hub for Advanced Prawn Breeding,
James Cook University, Townsville, Queensland, Australia
ronald.white@jcu.edu.au

D. R. Jerry
ARC Hub for Advanced Prawn Breeding,
James Cook University, Townsville, Queensland, Australia
dean.jerry@jcu.edu.au

ABSTRACT

Fast and low-cost image collection and processing is often required in aquaculture farms for quality/size attributes and breeding programs. For example, the absolute physical dimensions of fish (in millimeters or inches) could be estimated from electronic images. The absolute scale of the photographed fish is often unknown or requires additional hardware, data-collection and/or management overheads. One cost and time effective solution is to capture the absolute scale (in pixels-per-millimeter or dots-per-inch) by including a measuring ruler in the photographed scene. To assist that type of workflow, this paper presents a relatively simple image-processing algorithm that automatically located a sufficiently large section of the ruler in a given image. The algorithm utilized the Fast Fourier Transform and was designed to be free from adjustable parameters and therefore did not require training. The algorithm was tested on 445 images of Barramundi (Asian sea bass, *Lates calcarifer*), where a millimeter-graded ruler was included in each image. The algorithm achieved precision of 98% (on the original, 10, 20, 70, 80 90 degree rotated images) and 95-96% on 40, 50, 60 degree rotated images. The test Barramundi images were released to public domain (on this publication) via https://github.com/dmitryako/BarraRulerDataset445.

CCS Concepts

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

ICAIP 2017, August 25–27, 2017, Bangkok, Thailand

ACM ISBN 978-1-4503-5295-6/17/08…$15.00
DOI: https://doi.org/10.1145/3133264.3133271

Keywords

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

1. INTRODUCTION

Monitoring fish stock at all stages of production is essential in aquaculture farms. One approach to achieving better production efficiency and product quality is by applying computer assisted monitoring [1-3]. Machine (or synonymously *computer*) vision systems could fulfill the required monitoring role in many circumstances [4]. Computer vision technologies became very sophisticated and they could be utilized more widely in aquaculture industries [5]. In order to achieve better uptake of such vision systems, the associated installation and processing costs must be acceptable to the industry.

One of the most direct applications of the computer vision is estimation of fish morphological features such as length and width [2,6] by automatic image-processing algorithms. The fish shape could then be used to estimate its mass [7]. However, a typical image-processing algorithm could only output the dimensions in pixels, which then need to be converted to actual physical units of length (millimeters or inches). There are essentially only two ways of determining such a scale without additional potentially costly hardware. First is when the geometrical setup is known, for example the distance from the camera to the photographed object. The second approach is to have markers in the scene with known distances between them. Placing a measuring ruler next to the photographed fish could provide such markers automatically with minimal additional hardware expense. Furthermore even if the image scale was deemed to be known, having the ruler present could serve as an additional quality assurance and cross-verification mechanism.

Extracting the image scale by automatically detecting a measuring ruler was the focus of this study. Two conceptually separate algorithm types were required [8]. The first type should be able to locate all, or parts of the ruler image automatically, i.e. without asking the user to select the graduation marks in the image [8]. The second type should be able to extract the ruler interval distance in fractional number of pixels per ruler's graduation. Here

we reported an algorithm of the first type, designed to automatically locate sections of the ruler.

Figure 1. Example of an original color image (top) and its gray equivalent (bottom), where the square outline (bottom) is enlarged in the top-left of Figure 2.

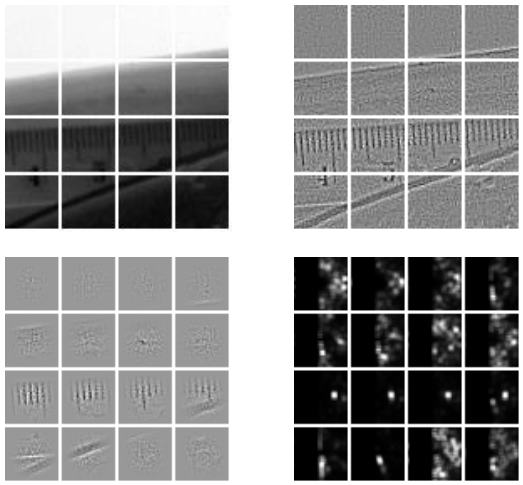

Figure 2. Four processing stages of the outlined section from Figure 1: original gray-scale blocks I_{ij} (top-left) with block-size L=32; high-pass filtered Y_{ij} (top-right); windowing applied W[Y_{ij}] (bottom-left); and low-pass filtered DFT2 power density P_{ij} (bottom-right), where $log(P_{ij}+1)$ is displayed.

1.1 Related Work

A large number of algorithms exist to estimate fish length from digital images, see for example [2] and references therein. On the other hand, the automatic scaling of the digital images has received disproportionally little attention from the research community. The most recent and related algorithms were as follows. Ueda *et al.* [9] applied Digital Fourier Transform (DFT) to estimate scale intervals on a ruler. The main limitations of the algorithm were an inability to (i) achieve sub-pixel accuracy, and (ii) treat noisy or blurry images, which are common in the aquacultural context. 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]. Bhalerao and Reynolds [8] used DFT to locate the ruler similar to the two preceding algorithms of [9] and [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 assess if it was suitable for aquacultural images.

2. METHOD

2.1 Block-Size Selection

The algorithm's steps that follow required a square block-size L as an input parameter, where L had to be an even number and was measured in the number of pixels. Once L was given, the algorithm proceeded by dividing the given gray image I into square blocks of size L. Essentially, the block-size L determined the scale at which the processing took place. Different values of L could and often did yield different estimated ruler interval distances, $d(L)$.

In this first step of the algorithm, a simple heuristic model was developed to select a set of $\{L_1, L_2, ..., L_K\}$ values automatically for any input image, where K was the total number of the required L-values. A given color image C was defined to have a number of rows (N_1) and columns (N_2), and represented as a $N_1 \times N_2 \times 3$ array of numbers. Color image C was converted to a gray-scale intensity image I, where the Matlab/Octave function *rgb2gray* was used. The resulting gray-image I was a $N_1 \times N_2$ matrix of real numbers, which was linearly scaled to have minimum value of zero and maximum value of one. Examples of C and I are shown in Figure 1. An input image I was allowed to have the range of N_1 height and N_2 width values between 100 and 10000. The largest possible square block-size L (L_{max}) and the smallest (L_{min}) were set as fractions of the smallest image dimension (rounded to even numbers),

$$L_{max} = L_K = \min(N_1, N_2)\,/N_B, \quad (1)$$

$$L_{min} = L_1 = \max(8, L_{max}/K), \quad (2)$$

where N_B was the number of blocks per smallest image dimension. The remaining L_t values were uniformly distributed in logarithmic scale,

$$L_t = (L_{max}/L_{min})^{(t-1)/(K-1)} L_{min}. \quad (3)$$

The smallest meaningful (for this algorithm) section of image processing was assumed to be the 8×8 square block. Hence the smallest possible L_{min} was set to 8. Finally, for each selected L_t value, both N_1 and N_2 were rounded down to be multiples of L_t, where the $N_1 \times N_2$ centred section of the original image was retained for further processing.

2.2 Discrete Fourier Transform

In this second step, and for each L from $\{L_1, L_2, \ldots, L_K\}$, the algorithm proceeded to automatically select blocks containing parts of the ruler image. 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, e.g. the ruler in Figure 1 had 301 tick lines (markers, or graduations) that were one millimeter apart. The distance d was measured in pixels-per-millimeter and fractional values were allowed.

The two-dimensional (2-D) Discrete Fourier Transform (DFT2) was ideally suited for detecting any repeating patterns in 2-D images and was an essential component of the algorithms in [8,10]. More specifically and following [8], the DFT2 was used via its Power Spectral Density (PSD) defined as

$$P = |\mathbf{F}[X]|^2, \tag{4}$$

where X was a 2-D matrix of real numbers, **F** was the DFT2 operator, and where P was the resulting PSD matrix with the same dimensions as X.

Bhalerao and Reynolds [8] suggested to apply high-pass filtering to the gray-scale image before DFT2 was done, while [10] did not use high-pass filtering. After extensive computational experiments in the context of aquacultural images, it appeared that the high-pass filtering provided a conceptually and practically simpler way of identifying ruler-related DFT2 signal. If the high-pass filtering was not done, then the resulting DFT2's power density P was dominated by multiple low frequency peaks due to semi-periodical textures in fish images. In general, the peak amplitudes diminished with increase in horizontal and/or vertical Fourier frequencies. Without prior detection and removal of the fish contour, it was not possible to identify which of the progressively diminishing peaks corresponded to the ruler 2-D image signal.

This study attempted to design an algorithm that had as few adjustable parameters as possible. Therefore, the conceptually simplest possible high-pass filter **H** was selected,

$$Y = \mathbf{H}[I] = I - \mathbf{L}[I], \tag{5}$$

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

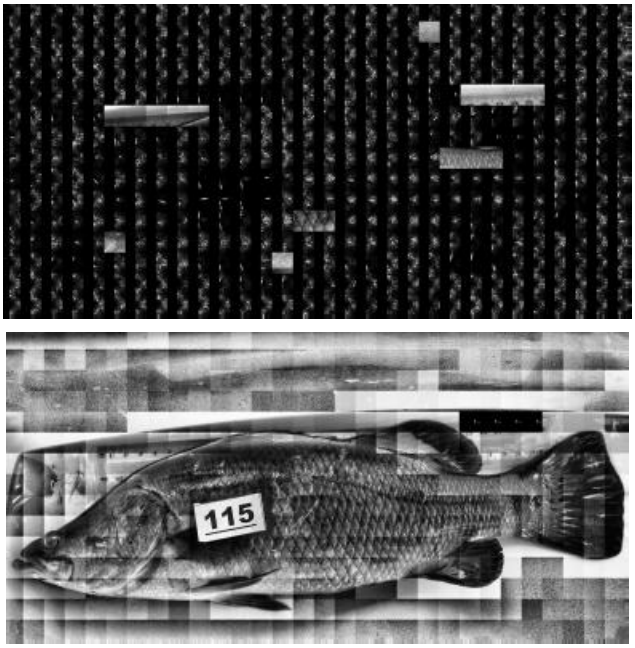

Figure 3. (Top) Candidate blobs displayed as gray-scale blocks G_{ij} and the rest of blocks displayed as P_{ij}. (Bottom) The longest blob displayed as P_{ij}, while the rest as G_{ij}, where the blob had the same flow vector in all its blocks. The block-size was L=52. P_{ij} and G_{ij} were scaled to the [0,1] range.

High-passed image Y (Eq. 5), was divided into square blocks yielding a $M_1 \times M_2$ matrix of sub-images Y_{ij}, which had $M_1 = N_1/L$ rows and $M_2 = N_2/L$ columns. To illustrate this step, Figure 2 (top-left) shows an enlarged section of the original gray image from the square outline in Figure 1. Figure 2 (top-right) shows the corresponding high-passed Y_{ij} results, where 16 blocks are shown for $L = 32$ pixels. Following [8], in order to prevent spectral leakage [11], each resulting block Y_{ij} was multiplied by a windowing mask. While many standard windowing options exist [11], the *cos* mask was selected as a reasonable choice, which did not attempt to optimize any particular aspect of this algorithm. In general, the $M \times N$ windowing matrix $W_{MN}(m,n)$ was defined as

$$W_{MN}(m,n) = w_M(m)\, w_N(n), \tag{6}$$

$$w_N(n) = sin\big((n-1)\pi/N\big), \tag{7}$$

$$1 \le m \le M, \qquad 1 \le n \le N. \tag{8}$$

Note that the above one-dimensional (1-D) window $w_N(n) = sin$ was equivalent to the cos window from [11], where the latter was defined in terms of frequencies centered at the origin. A 2-D windowing square $L \times L$ matrix was denoted by W_L and used before the DFT2 was applied via

$$F_{ij} = \mathbf{F}\left[\mathbf{W}_L[Y_{ij}]\right], \tag{9}$$

where **F** was the DFT2 operator, $\mathbf{W}_L$ was the windowing operator defined as element-wise multiplication of $L \times L$ matrices W_L and Y_{ij}. Figure 2 (bottom-left) shows the result of the chosen windowing function, $\mathbf{W}_L[Y_{ij}]$. Furthermore, each DFT2 power density matrix P_{ij} was 3×3 neighborhood averaged using the low-pass filtering operation **L** from Eq. (5),

$$P_{ij} = \mathbf{L}\left[|F_{ij}|^2\right]. \tag{10}$$

The averaging was done to make the algorithm more robust in the presence of noise and/or small ruler distortions. The corresponding example is shown in Figure 2 (bottom-right).

2.3 Ruler Blob Selection

From the preceding DFT2 step, the algorithm obtained the $M_1 \times M_2$ matrix of P_{ij} blocks, where each P_{ij} block is a $L \times L$ matrix of numbers. Next, the maximum element in each P_{ij} block was located and its row and column index values were stored as a 2-D vector $\mathbf{v}_{ij} \equiv \{v_{ij}(1), v_{ij}(2)\}$, where $v_{ij}(1)$ stored the row and $v_{ij}(2)$ stored the column indices of the maximum element. The vector $\mathbf{v}_{ij}$ was converted to the directional flow vector $\mathbf{q}_{ij}$,

$$\mathbf{q}_{ij} = (\mathbf{v}_{ij} - \mathbf{v}_{0ij})/L, \tag{11}$$

$$q_{ij} = \|\mathbf{q}_{ij}\| = \sqrt{q_{ij}^2(1) + q_{ij}^2(2)}, \tag{12}$$

where $v_{0ij}(1)$ was the row index of the zero vertical Fourier frequency, and $v_{0ij}(2)$ was the column index of the zero horizontal frequency. The vector $\mathbf{q}_{ij}$ determined the direction in which there was dominant (for each block P_{ij}) periodical flow with frequency q_{ij}. Next, the algorithm selected a ruler blob as a subset of P_{ij} using the following four checks. The terms *flow* and *blob* were adopted from [12].

2.3.1 Ruler blob check 1

A candidate ruler *blob* was assumed to be a connected set of blocks with the same dominant flow vector $\mathbf{q}_b$. Two blocks

P_{ij} and $P_{i'j'}$ were considered to have the same dominant flow if their corresponding, $\mathbf{q}_b = \mathbf{q}_{ij} = \mathbf{q}_{i'j'}$.

2.3.2 *Ruler blob check 2*

Blocks were accepted into a candidate blob only if the same vertical or horizontal flows were present in two neighboring blocks. More specifically for vertical flow,

$$q_{ij}(1) = q_{i'j}(1), \qquad |i - i'| = 1, \tag{13}$$

and for horizontal flow, see example in Figure 3,

$$q_{ij}(2) = q_{ij'}(2), \qquad |j - j'| = 1. \tag{14}$$

Furthermore, if $q_{ij}(2) \geq q_{ij}(1)$, then the flow was considered to be horizontal and the block was removed from consideration if it did not have a block with the same horizontal frequency $q_{ij'}(2)$ immediately to its left or right. Analogously, if $q_{ij}(2) < q_{ij}(1)$, the block was retained only if it had the same vertical frequency $q_{i'j}(1)$ blocks above or below.

2.3.3 *Ruler blob check 3*

All candidate blobs with the same largest number (l_{blob}) of block rows or columns (Figure 3, bottom) were retained. The minimum acceptable blob length l_{blob} was controlled by B_{min},

$$l_{blob} \geq B_{min}, \qquad l_{blob} = max(|i - i'|, |j - j'|), \tag{15}$$

$$(i,j) \in blob, \qquad (i',j') \in blob. \tag{16}$$

From numerical experiments, it was found that $B_{min} = 3$ was the smallest number that was sufficient to remove most of the dyads, which were selected as blob candidates purely due to noise. Naturally, a larger B_{min} would reduce the number of false-positives, when fish or scene environment objects were mistaken for a ruler. However, at the same time, a larger B_{min} would also increase the number of false-negatives, when the ruler was not detected due to image distortions and/or non-uniform lighting.

2.3.4 *Ruler blob check 4*

If two or more candidate blobs had the same maximum sizes l_{blob}, then P_{ij} were added pixel-wise,

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

from which its dominant flow was determined, $\mathbf{q}_B$. Then, the blocks with identical $\mathbf{q}_B = \mathbf{q}_{ij}$ was retained and the other blocks were discarded.

Note that if $\mathbf{q}_B$ did not match any of the candidate blobs' $\mathbf{q}_{ij}$, then all candidate blobs were deemed not to have strong enough signal of dominant flow and they were discarded as noise.

This last check was particularly important for blurry images, where the semi-periodical textures in fish images often yielded equivalent (in maximum length) candidate blobs to the candidate blob of the actual ruler. Furthermore, due to a weak original ruler 2-D signal, the DFT2 ruler peaks were comparable or smaller than the DFT2 peaks from the fish features. Therefore it was not possible to automatically identify which of the detected flows was associated with the ruler flow without this last check.

The outcome of the preceding ruler blob selection checks was a set of index (i,j) pairs

$$blob = \{(i,j)\}, \tag{18}$$

which was used as $(i,j) \in blob$ to specify that the block in ith row and jth column (of the original $M_1 \times M_2$ matrix of blocks) belonged to the ruler blob. Also, by construction, each of the P_{ij} blocks had the same flow vector $\mathbf{q}_{ij} = \mathbf{q}_b$. Figure 3 (bottom) shows an example of the longest blob of P_{ij}, $(i,j) \in blob$, where for illustration purposes the not-selected blocks were displayed as the gray-image blocks, $G_{i'j'}$, $(i',j') \notin blob$.

If however, even a single candidate blob could not be found, then $\mathbf{q}_b = (0,0)$ and $blob = \{\}$ were assigned to a zero vector and empty set, respectively. The algorithm was then terminated for the given image I and the current block-size L.

2.4 Distance estimation

The focus of this algorithm was to locate a section of the ruler in each image, which could be used to extract the interval distance. While an accurate estimation [8] of the interval distance from the rule blocks was outside the scope of this work, we converted the dominant flow vectors to distances via [10]

$$d(L) = 1/q_B. \tag{19}$$

This direct conversion was deemed to be insufficiently accurate [8] for this application. More specifically, the following two issues were identified. The first issue was that the accuracy of the direct conversion was limited by the block-size L. When extracted from the DFT2 frequency q_B via Eq. (19), the error in distance Δd was

$$\Delta d \sim \Delta q/q^2, \qquad \Delta q \approx 1/L, \tag{20}$$

$$1/L \leq q \leq \sqrt{0.5}, \qquad 2/L \leq \Delta d \leq L. \tag{21}$$

For example, if $L = 50$ and actual $d = 5$, then $q = 0.2$, $\Delta q = 0.02$, and $\Delta d = 0.5$ (or $\approx 10\%$ error). The second issue was that for very sharp images, the dominant frequency q_B often doubled to become $q_B \approx 2/d$ and yielded half of the actual distance $d(L) \approx d/2$. In the testing dataset of 445 images it was found that the dominant frequency doubling occurred in about 10% of the images.

2.5 Block Overlapping

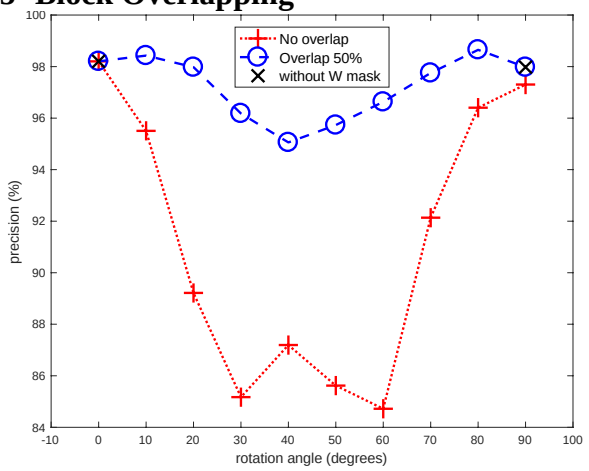

Figure 4. Algorithm precision (Eq. 22) on the test dataset of 445 Barramundi images as a function of image rotation angle, where blue circles denote 50%-overlapped blocks, and red crosses denote non-overlapping blocks.

The presented algorithm identified ruler blocks from the non-overlapping blocks Y_{ij} that were high-pass filtered. Following [8] who used the image blocks overlapped by 50%, such overlapping

blocks were also implemented and referred to as Z_{ij}. To allow for direct comparison between the overlap and no-overlap versions of the algorithm, each Z_{ij} block of $2L \times 2L$ pixels was centered on the corresponding Y_{ij}, so that the inner central $L \times L$ pixels in each Z_{ij} were identical to Y_{ij}, where $2 \le i \le M_1$ and $2 \le j \le M_2$. The rest of the algorithm remained the same.

3. RESULTS

The presented algorithm was tested on $n = 445$ images of Barramundi (also known as Asian sea bass or *Lates calcarifer*), which was released (on this publication) to public domain via https://github.com/dmitryako/BarraRulerDataset445. The measured graduation interval distance of the *i*th image was denoted as y_i. The image was manually loaded into *ImageJ* [13] software and the ruler was measured in the number of pixels. For example, if both ends of the ruler were visible then a line was drawn between the very first and very last ticks, where the first tick was at the 0 mm mark and the last tick was at 300 mm mark. The software (*ImageJ*) calculated the length of the drawn line in pixels L_{px}, which was then divided by $L_{mm} = 300$ to obtain the actual interval distance $y_i = L_{px}/L_{mm}$ in pixels-per-millimeter. If one or both ends of the ruler were not visible, the total visible ruler length L_{mm} was adjusted accordingly. The mean (μ) and standard deviation (σ) of test image heights and widths were $(\mu_h = 1402,\ \sigma_h = 637)$ and $(\mu_w = 1869,\ \sigma_w = 850)$, respectively.

Since every test image had a full or partially visible ruler, the algorithm was assessed by the standard two-class classifier precision metric [14],

$$precision = 100 \times TP/n, \tag{22}$$

where *TP* was the number of true-positives, and $n = 445$ was the number of test images. The number of 50%-overlapped boxes per shortest image dimension was set to 10 (see Section 2.5), which corresponded to $N_B = 12$ in Eq. (1). Total number of different (per image) trial block-sizes was set to $K = 4$ (Eq. 2).

For the purposes of this work, a reliable automatic (rather than manual-per-image) indicator was needed to confirm that a located candidate ruler blob was correct. That is, the blob contained a sufficient section of the ruler, which could be further processed to estimate the distance more accurately, e.g. via method of [8]. The robust indicator of correctly located ruler section (for a given L) was identified as the pair $\{d(L), 2d(L)\}$, where either $d(L)$ or $2 \times d(L)$ were within 10% of the manually measured distance y_i, and where $d(L)$ was calculated via Eq. (19). Throughout the testing this *correct location indicator* (CLI) was true only when the candidate blob contained a section of the ruler, i.e. the indicator had the false-positive rate of 0%. In other words, the probability of the algorithm returning y_i or $y_i/2$ without a section of the ruler was negligibly low for this particular dataset of test images. Note that the CLI was not part of the algorithm, but only a testing tool to assist in automated checking if the algorithm worked correctly for any given test image. For example, a different indicator may be required to automate testing on a different dataset. No further work was done towards making the CLI more general since the forthcoming accurate distance extraction algorithm could fulfill that role naturally [8].

For the *i*th image and for each $L \in \{L_1, L_2, L_3, L_{K=4}\}$, the algorithm returned the interval distance $d_i(L)$, which was CLI corrected to $2d(L)$ via

$$CLI[d_i] = \begin{bmatrix} 2d_i\,, & |2d_i - y_i|/y_i \le 0.1 \\ d_i\,, & \text{otherwise} \end{bmatrix}. \tag{23}$$

A set of L values was specific for each image. It was expected that some of the $d_i(L)$ values could be completely wrong due to the semi-periodical fish textures. Specifically within the aquacultural context, a fish image often displayed features, which were easily confused with ruler-markers if the processing block-size L was comparable with the features, e.g. strongly periodic lines of fins or fish-scales. Therefore, more statistically robust [15] median (rather than mean) of non-zero $d_i(L)$ values were used as the final algorithm result,

$$x_i = median\big(d_i(L_1), \dots, d_i(L_K)\big), \qquad d_i(L_t) \neq 0, \tag{24}$$

where zero $d_i(L_t)$ values were used as the indicator that the ruler selection checks (Section 2.3) could not locate a ruler blob for the particular block-size L_t. If the Absolute Relative Error (ARE) of x_i,

$$\text{ARE}_i = 100 \times |x_i - y_i|/y_i, \tag{25}$$

was larger than 10%, then that x_i was reported as false-positive. The total number of such false-positives was denoted *FP*, and used in Eq. (22) via $TP = n - FP$, e.g. Figure 5 reported $FP = 8$ outliers. The Mean Absolute Relative Error (MARE) [7] was used to evaluate the algorithm performance on true-positives,

$$MARE = \textstyle\sum_{i=1} \text{ARE}_i\,/TP, \qquad \text{ARE}_i \le 10\%. \tag{26}$$

In order to test how well the algorithm worked on arbitrary positioned rulers, the original set of images was augmented by rotated each image by $10, 20, \dots, 90$ degrees obtaining the total of 4450 test images, where the Matlab/Octave function *imrotate(…,'crop', 'bilinear')* was used to rotate each image. Furthermore, each original gray image was multiplied (before rotation) by the windowing mask $W_{N_1 N_2}$ (see Eq. 6) so that the pixels near the edges of the images were reduced to zero. Because of the W mask, the difference in algorithm's precision for different rotation angles (Figure 4) was due to the algorithm rather than to possible edge effects of the rotated image. Note that the window mask W was not part of the algorithm and was not required for running the algorithm in production. Figure 4 (black '×' symbols at 0 and 90 degrees) demonstrates that the algorithm precision was not affected by removing the W mask, where the 90-degree rotation could be considered equivalent to the original images since no between-pixel interpolation was required.

Figure 4 clearly demonstrates that the 50%-overlapped blocks [8] performed significantly better than the non-overlapped blocks. Interestingly, for nearly horizontal or vertical rulers (zero and 90 degree points in Figure 4), the block overlapping yielded no advantage. The algorithm with overlapped blocks required 2-4 times longer to process each image. Therefore if the rulers were known to be horizontal, noticeably faster processing could be achieved with non-overlapping blocks without sacrificing the precision. Figure 5 (MARE = 1.5%) confirms the suggestion of [8] that the direct conversion of DFT2 peaks to distance (Eq. 19) could be improved by more accurate distance extraction algorithm, where [8] obtained MARE < 0.1%.

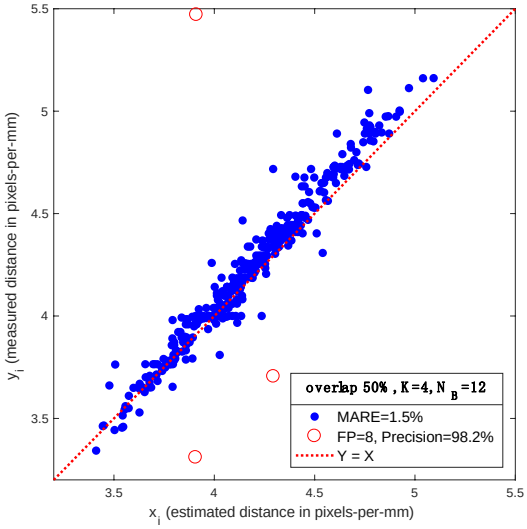

Figure 5. Actual $n = 445$ test image distances $\{y_i\}_{i=1}^{n}$ and estimated interval distances $\{x_i\}_{i=1}^{n}$ from Eq. (24), where $d_i(L_t)$ were calculated by Eq. (19) and CLI corrected by Eq. (23). Original test images were converted to gray-scale and used here without the windowing mask and without rotation. MARE was from Eq. (26).

4. CONCLUSIONS AND FUTURE WORK

Note that the test images were not used to *calibrate* the algorithm in any way since the algorithm was designed not to have any adjustable parameters. Furthermore, for those few algorithm's steps which could be viewed as adjustable, they were not optimized for the test images. Therefore, the reported precision could be viewed as the estimation of the algorithm's predictive performance [15], rather than how well the algorithm fitted the test images.

In conclusion, this study presented a relatively simple algorithm based on the Fourier transform, which could automatically locate a section of a ruler in a given image. The algorithm achieved true-positive rate of 95-98% on a very challenging set of 445 test images, where 10-20% of images were either blurry or the ruler was partially obscured by Barramundi fish. The direct conversion of the dominant Fourier frequency to the ruler graduation interval distance yielded 1-2% accuracy, which was comparable with false-positive rate (2-5%) of the algorithm. If such level of accuracy is sufficient, only a very simple distance extraction method is required, where the double-frequency check (Eq. 23) is done as part of the algorithm and without the ground-truth y_i values. Then Eq. (19) should deliver 1-2% accurate interval distance.

5. ACKNOWLEDGMENTS

D.A.K. acknowledges the computational hardware supported by JCU DTES 2016 RIBG.

6. REFERENCES

[1] Hong, H., Yang, X., You, Z., and 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. and 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., Shklyar, A., and 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.

[4] Saberioon, M., Gholizadeh, A., Cisar, P., Pautsina, A., and Urban, J. 2016. Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues. *Rev. Aquacult.* 0, 1–19. DOI= http://dx.doi.org/10.1111/raq.12143.

[5] 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.

[6] Costa, C., Antonucci, F., Boglione, C., Menesatti, P., Vandeputte, M., and 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., and 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] Bhalerao, A. and Reynolds, G. 2014. Ruler detection for autoscaling forensic images. *Int. J. Digit. Crime Forensics*, 6, 9–27. DOI= http://dx.doi.org/10.4018/ijdcf.2014010102.

[9] Ueda, K., Baba, T., Nakagawa, Y., and Amano, K. 2005. Detection of scale intervals in digital images, *21st Int. Conf. Data Engineering Workshops*. ICDEW'05. Tokyo, Japan, 1232–1232. DOI= http://dx.doi.org/10.1109/ICDE.2005.211.

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

[11] Harris, F. J. 1978. On the use of windows for harmonic analysis with the discrete Fourier transform. *P. IEEE* 66, 51–83. DOI= http://dx.doi.org/10.1109/PROC.1978.10837.

[12] Carson, C., Belongie, S., Greenspan, H. and Malik, J. 2002. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE T. Pattern Anal.* 24, 1026–1038. DOI= http://dx.doi.org/10.1109/TPAMI.2002.1023800.

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

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

[15] Konovalov, D. A. Llewellyn, L.E., Vander Heyden, Y., and Coomans, D. 2008. Robust cross-validation of linear regression QSAR models. *J. Chem. Inf. Model.* 48, 2081–2094. DOI= http://dx.doi.org/10.1021/ci800209k.