Computational Metadata Generation Methods for Biological Specimen Image Collections

Kevin Karnani (kevinkarnani@gmail.com)
Drexel University https://orcid.org/0000-0002-3108-7941
Joel Pepper
Drexel University https://orcid.org/0000-0002-1601-8729
Yasin Bakis
Tulane University
Xiaojun Wang
Tulane University
Henry Bart
Tulane University
David Breen
Drexel University
Jane Greenberg
Drexel University

Research Article

Keywords: bioinformatics, metadata, image analysis, applied machine learning, contrast enhancement

Posted Date: April 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1506561/v1

Computational Metadata Generation Methods for Biological Specimen Image Collections

Kevin Karnani[1], Joel Pepper[1], Yasin Bakiş[2], Xiaojun Wang[2], Henry Bart Jr.[2], David E. Breen[1] and Jane Greenberg[3]

[1]Computer Science Department, Drexel University, Philadelphia, PA, USA.
[2]Biodiversity Research Institute, Tulane University, New Orleans, LA, USA.
[3]Information Science Department, Drexel University, Philadelphia, PA, USA.

Contributing authors: kevinkarnani@gmail.com; jcp353@drexel.edu; ybakis@tulane.edu; xwang48@tulane.edu; hbartjr@tulane.edu; david@cs.drexel.edu; jg3243@drexel.edu;

Abstract

Metadata is a key data source for researchers seeking to apply machine learning (ML) to the vast collections of digitized biological specimens that can be found online. Unfortunately, the available metadata is often sparse and, at times, erroneous. This paper extends previous research with the Illinois Natural History Survey (INHS) collection (7,244 specimen images) using computational approaches to analyze image quality, and then automatically generate 22 metadata properties representing the image quality and morphological features of the specimens. In the research reported here, we demonstrate the extension of our initial work to University of Wisconsin Zoological Museum (UWZM) collection (4,155 specimen images). Further, we enhance our computational methods in four ways: 1) augmenting the training set, 2) applying contrast enhancement, 3) upscaling small objects, and 4) refining of our processing logic. Together these new methods improved our overall error rates from 4.6% to 1.1%. These enhancements also allowed us to compute an additional set of 17 image-based metadata properties. The new metadata properties provide supplemental features and information that may also be used to analyze and classify the fish specimens. Examples of these new features include convex area, eccentricity, perimeter, skew, etc. The newly refined process further outperforms humans in terms of time and labor cost, as well as accuracy, providing a novel solution for leveraging digitized specimens with ML. This research demonstrates the ability of computational methods to enhance the digital library services associated with the tens of thousands of digitized specimens stored in open-access repositories world-wide by generating accurate and valuable metadata for those repositories.

Keywords: bioinformatics, metadata, image analysis, applied machine learning, contrast enhancement

1 Introduction

Advances in computing, imaging, and cyberinfrastructure, along with the growth digital libraries and repositories, have allowed many natural history institutions to digitize their image specimen collections [1]. The National Science Foundation's Advancing Digitization of Biodiversity Collections (ADBC) program is one exemplary program supporting the digitization and curation of hundreds of thousands of biological specimens [2]. Digital collections provide researchers, educators, students, and the general public with the capacity to study biological specimens on a scale that was previously unattainable. In addition, the availability of digitized specimen images allows for the application of machine learning (ML), which should lead to new scientific discoveries.

Although there is increased interest in applying ML to digitized specimen images, researchers have found that the potential scientific advances are, unfortunately, hindered by poor image quality [3]. Poor quality images (e.g. with low contrast, inadequate lighting, out-of-focus or cluttered visual arrangements) are unsuitable for automated image analysis by ML algorithms and lead to inferior computational results. Image quality problems associated with digitized specimens are further compounded by poor quality metadata or even the lack of pertinent metadata. Many natural history collections use the Darwin Core (DwC) metadata standard, which includes a core set of 19 descriptive properties [4]. Metadata for digital images is frequently created manually by technical staff or students and subject to human error. Additionally, although richer metadata extensions exist and curators may provide more extensive morphological metadata, it is too costly to acquire manually.

Despite existing image quality and metadata limitations, the extensive availability of digitized specimen collections still offers new opportunities for scientific study. These challenges have motivated members of Drexel University's Metadata Research Center together with Tulane University's Biodiversity Research Institute to explore computational approaches for analyzing fish image quality and extracting specimen metadata. A key impetus has been engagement of both teams in the NSF Biology Guided Neural Networks (BGNN) project, which is developing a novel class of artificial neural networks that aims to exploit machine readable, predictive knowledge associated with specimen images, phylogenies, and anatomical ontologies. Initial research successfully demonstrated computational approaches for creating image quality metadata [5]; and, further, that by combining ML and image informatics, researchers automatically determine image quality and metadata, such as fish quantity, location and orientation, and image scaling based on ruler identification and measurement [6].

The research reported on in this paper extends the methods reported on in [6], with the aim to increase the accuracy and scope of the generated metadata. Another key goal is to demonstrate that our approach is extensible to other specimen image collections, beyond the Illinois Natural History Survey (INHS) collection analyzed in our first study. Previously, object detection was performed with five detection classes (fish, fish eyes, rulers, and the twos and threes found on rulers) from 7,244 INHS images. We have augmented this dataset to include 4,155 images from the University of Wisconsin Zoological Museum Collection (UWZM) [7]. Additionally, we have trimmed the INHS dataset to 7,013 after adding some images to the training set as well as excluding certain images, resulting in a test set of 11,168 images. Figure 1 presents a typical image used in our study from each collection.

The enhancement of our computational methods has produced improved automatic metadata generation results. These enhancements include augmentation of the training set, applying contrast enhancement, upscaling small objects and refinement of our processing logic. Together these new methods improved our overall error rates from 4.6% to 1.1%. Procedures for computing additional image-based metadata properties have also been implemented. These new properties provide supplemental features and information that may also be used to analyze and classify the fish specimens. Examples of these new features include `convex_area`, `eccentricity`, `perimeter`, `skew`, etc.

The rest of the paper is organized as the following. Section 2 describes relevant previous work in metadata for image specimen collections, metadata generation and fish image analysis. Section 3 outlines the goals and objectives of our research. Section 4 describes in detail our computational methods. Section 5 includes the results of our computational experiments on two fish image collections, with Section 6 providing a discussion of our results. Section 7 concludes with comments on possible future extensions for the current work.

2 Related Work

2.1 Metadata Standards and Approaches for Natural History Digital Image Specimen Collections

Metadata used in digital image specimen collections supports resource description access. The Darwin Core (DwC) [8] and the Audubon Core [9] are two

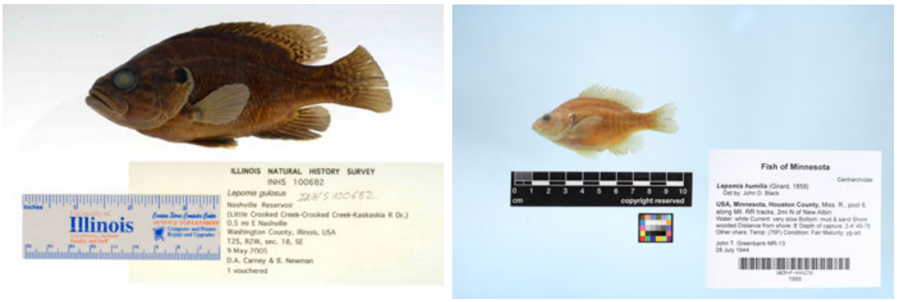


Fig. 1 A typical INHS image (left) and a typical UWZM image (right).

of the most popular metadata standards applied to digital specimen images. Curators often use content value standards, such as taxonomies, geographic codes, and other ontologies, when working with the descriptive metadata standard. Although these standards are digitally accessible, the metadata creation task is still primarily a manual, labor-intensive task and prone to human error. Moreover, image quality metadata is generally absent. These limitations have become increasingly prevalent as researchers seek to automatically leverage metadata and digitize specimen images for scientific research, which is an aim of the BGNN initiative. The biodiversity community has acknowledged this challenge and advocated for data fitness standards [10]. This point is also emphasized by Wieczorek et al. [11] in their report on the variety of DwC metadata extensions needed to meet growing community concerns and requirements, including data quality and fitness. This point is addressed in detail by Leipzig et al. [5], drawing from Tulane University's manual curation of 22 metadata properties that characterize digitized specimen image quality, and further motivates the research reported in this paper.

2.2 Automatic Metadata Generation

Automatic metadata generation advances covering both descriptive and technical metadata are relevant to the research presented in this paper. Automatic metadata generation of descriptive bibliographic data has been a research focus for close to twenty years [12–15]. Researchers have applied support vector machine (SVM) approaches [16] and associated networks to address sparse and incomplete metadata [17], and various successes are integrated into day-to-day workflows. Heidorn, et al. [18] demonstrated the use of optical character recognition (OCR) to extract specimen information from the original typed and often hand-annotated labels that are digitized along with herbarium collection holdings. The extracted information was encoded in the DwC metadata associated with the specimen's digitized rendering. There has also been some success with extracting descriptive cartographic information from maps [19]. While descriptive metadata covers taxon, geographic location, and other important aspects, and may even record the image format; extensive use of automatic processes are still limited. More significantly, descriptive metadata does not sufficiently addressed data quality.

2.3 Fish Image Analysis

Image analysis has been utilized to examine and process images of fish for well over two decades [20, 21]. It is an important application of technology for marine science, in the study of aquatic species, habitats and ecosystems, and for the seafood industry, in the development of automated fish sorting and grading systems, as well as fisheries management. Many of these computational analyses focus on the recognition and classification of the fish present in an image. The computational methods employed for fish image analysis have followed the general trends in the AI field. Hu et al. [22] presented a method of classifying species of fish based on color and texture features and a multi-class support vector machine (MSVM) [23]. Li and Hong [24] computed eleven shape and color features from fish images and derived a linear model that could discriminate between four different fishes. Rodrigues et al. [25] explored several combinations of feature extractions, input classifiers and clustering algorithms to produce a method that could distinguish between 10 different types of fish with 92% accuracy. Hernández-Serna and Jiménez-Segura [26] perform image preprocessing and extract geometric features which are fed into an artificial neural network (ANN) to predict the species of fish and other biological specimens with an accuracy around 90%.

Salman et al. [27] employed a deep Convolution Neural Networks (CNN) [28] together with classification based on K-Nearest Neighbor and Support Vector Machines trained on the features extracted by the CNN. They achieved 90% accuracy when identifying 15 different fish species in challenging underwater digital images. Utilizing texture, anchor points, and statistical measurements, Alsmadi et al. [29] implemented fish classification through a meta-heuristic algorithm known as the Memetic Algorithm. They were able to classify 24 fish families with 90% accuracy. Iqbal et al. [30] used a modified AlexNet [31] model to classify six different fish species with 90% accuracy. Yu et al. [32] segment fish images and measure fish morphological features using Mask R-CNN. Petrellis [33] employs image processing and deep learning to calculate a small set of geometric features from in-the-wild images of fish. Hao et al. [34] provide an excellent review of fish measurement efforts that utilize machine vision.

While our work is motivated by and builds upon previous work, our work stands apart in that it generates a wider range of geometric and image features than previous methods, in order to enrich the metadata for fish image collections. Most previous work examined and analyzed images of fish in the wild. Working with the structured images of museum collections allows us to compute a wider variety of metadata with higher accuracy.

3 Goals and Objectives

Digitized specimens accessible in open-access repositories provide a rich, extensive data source for ML and scientific discovery. These resources, however, remain largely untapped due to image quality issues and metadata limitations. Initially, the overall goal of our work addressed this need through the development of a computational alternative to the current manual metadata generation process, which is prohibitively costly both in terms of labor and time [6]. The specific aims of our extended objective are to:

- Demonstrate the generality of our previous work. This is accomplished through the augmentation of the training and testing sets from solely INHS images to a combination of images from the INHS and UWZM specimen image repositories.
- Optimize the previous property calculations to shorten computation times.
- Apply various techniques to improve accuracy and reduce error rates.
- Compute additional geometric properties from the specimens in order expand the features available for downstream analysis.

4 Methods

Our initial process for metadata generation can be divided into three steps: object detection with Facebook's Detectron2 [35] ML library (referred to as `detectron`), image processing at the pixel level, and calculations on the results of the previous steps to determine higher level metadata properties. We have extended the computational process with optimizations in metadata generation by replacing self-implemented code with more library calls, as well as further modularization that supports GPU parallelization. Along with additional geometric computations and error reduction techniques, our current metadata generation process has been expanded to:

- Apply contrast enhancement and equalization on the training and test sets.
- Train a model and perform object detection with Facebook's Detectron2 ML library (referred to as `detectron`).
- Select fish of highest confidence in case of multiple fish detection.
- Upscale and rerun if fish is detected without an eye.
- Adjust fish mask with pixel-level image processing.
- Compute specific geometric and statistical computations with `skimage` and `scipy`.
- Calculations on the results of the previous steps to determine higher level metadata properties.

4.1 Refined Fish Collection Criteria

Our automated metadata generation methods were developed for a specific subset of images from both the INHS and UWZM Fish Collections. Our algorithms are based on assumptions about the content and structure of the specimen images. Criteria were specified that define the properties of acceptable images for analysis. The images used in our study were evaluated to ensure that they meet these requirements. The criteria used to select the study images are:

- Must contain a fish (no eels, seashells, butterflies, seahorses, snakes, etc).
- Must contain only one of each class (except eyes).
- Specimen body must lie in-plane in a side view.
- Ruler must be consistent and one of two types.
- Fish must not be obscured by another object.
- Whole body of fish must be present (no heads, tails, or standalone features).
- Fish body must not be folded or have extreme curvature.

Applying these criteria, 216 images from the previous subset of 7,244 INHS images were removed from the testing set. Additionally, 15 images were moved from the test set and added to the training set, resulting in a testing subset of 7,013 images. Also to note, while the original UWZM collection contains 4,602 images, after removing 79 images for the training set, 368 images were filtered out based on the criteria, yielding a testing subset of 4,155 images.

4.2 Detectron

A prerequisite task to performing metadata property generation is finding the specimens (and other relevant objects) within the collection images. Object detection has been a broadly active field of study in recent years, and has resulted in a number of well-tested, purpose-built architectures. We elected to use Facebook AI Research's (FAIR) `detectron` tool [35] (specifically its implementation of the Mask R-CNN architecture) for object detection in our work, given its many flexible and robust capabilities. Most importantly, following a review of the literature and available tools, we determined that there were no other machine learning packages that returned pixel-by-pixel masks over detected objects in a comparable fashion.

`detectron` is built on `pytorch` [36] and provides a relatively straightforward method for training on `COCO` [37] format datasets. It is able to handle any number of object classes, and can classify an arbitrary number of objects within a given image. We chose `detectron` for its relative ease of use compared to lower level libraries, and its implementation of powerful architectures developed by FAIR. We use it to identify five object classes: fish, fish eyes, rulers, and the numbers two and three on rulers, as shown in Figure 2. Objects with a 30% confidence score or higher are maintained for analysis.

Table 1 lists the number of instances for each class used in our aggregate training dataset. Table 2 and Table 3 show the training set contributions from the INHS and UWZM datasets respectively. 500 rulers used in our previous study were removed from the INHS dataset due to an oversight. These 500 rulers were originally a part of the J.F. Bell Museum of Natural History (JFBM) [38] dataset, and hence, were not relevant to our current objective.

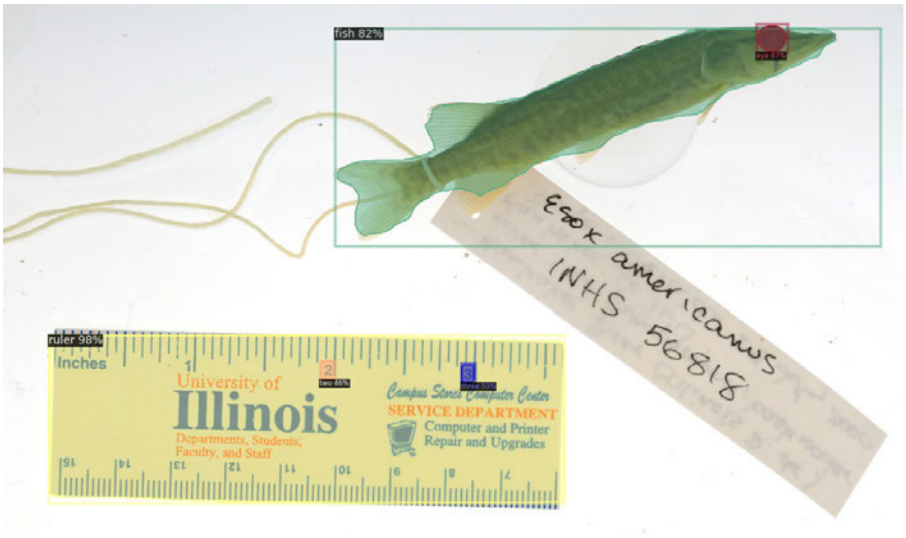


Fig. 2 Initial object detection on a specimen image using Detectron2 [35].

Table 1 Training Dataset

Class	Number of Instances
Fish	391
Ruler	1095
Eye	550
Two	194
Three	194

Table 2 INHS Training Dataset

Class	Number of Instances
Fish	312
Ruler	1016
Eye	471
Two	115
Three	115

Table 3 UWZM Training Dataset

Class	Number of Instances
Fish	79
Ruler	79
Eye	79
Two	79
Three	79

All of the training data was labeled by hand using `makesense.ai` [39] on images from the INHS [40] and UWZM [7] Fish Collections. Using `detectron`'s default training scheme, the model was trained for 15,000 epochs. Testing has shown that this default number of epochs provides optimal object detection results. All instance types were included in a single object detection model, in other words, the model is akin to a one-vs-all detector, with all five classes being detected by the same model.

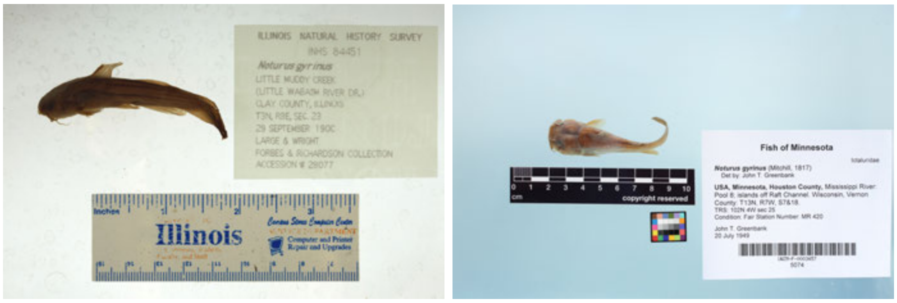


Fig. 3 A catfish where the eye is not easily detected (left) and a catfish with an eye that does not look like a normal eye (right).

4.3 Error Reduction Techniques

Four enhancements were implemented and applied to the combined datasets in order to reduce the error rates that we experienced in our initial study. These enhancements include augmenting the training set, applying contrast enhancement, selection of the highest confidence fish, and image (up)scaling.

4.3.1 Augmented Training Set

Initially, we had 64 examples of each class from the UWZM collection in the training set. One issue that we encountered was the lack of catfish (notorus genus) in the training set, which led to a high count of undetected eyes in the testing set. Visually it is difficult even for humans to determine the location of catfish eyes given that they are either very close to the color of the skin or do not look like normal fish eyes. Thus, 15 catfish images from each image dataset were added to the training set. Figure 3 presents examples of images in which the catfish eyes are difficult to detect.

4.3.2 Contrast Enhancement

It is apparent that there are differences in lighting, saturation, and contrast within and across specimen image collections. This causes the detection model to miss the ruler, scale, fish, or eye in images that are either too washed out or too dark, errors which have been seen in other object detection applications [41]. After investigating various image processing techniques to equalize the color, we found that current research in our area utilizes Contrast Limited Adaptive Histogram Equalization (CLAHE) [42, 43]. We applied this technique to all our images using the Python image processing library `OpenCV` [44]. CLAHE is frequently used in applications like underwater photography, traffic control, astronomy, and medical imaging [45, 46].

The drawback of standard Histogram Equalization is that the equalization of the contrast is performed on a global level, which is not ideal given possible varying contrast ranges in an image. Adaptive Histogram Equalization addresses this issue by computing several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values

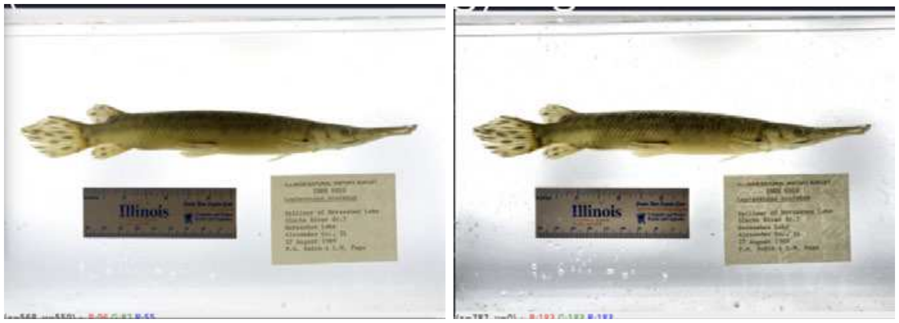


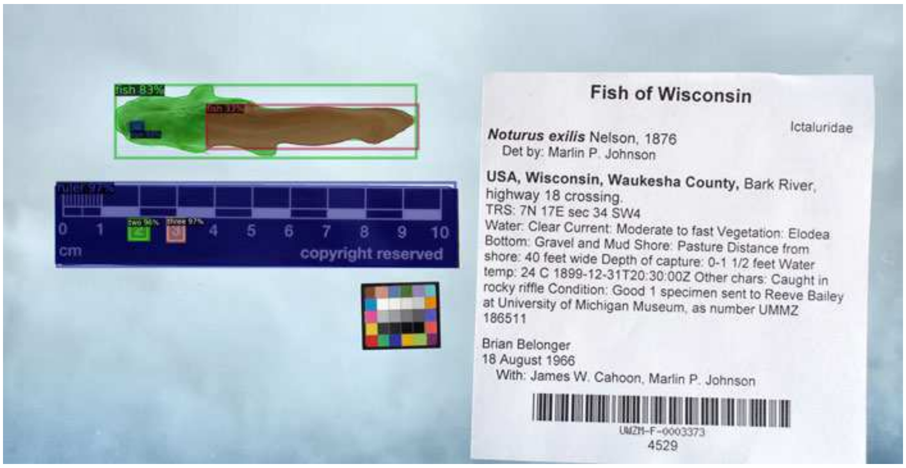
Fig. 4 A fish image before contrast enhancement (left) and after enhancement (right).

of the image. This also, however, has issues in that it may oversharpen contrast values that are already high, as well as yield noise in relatively homogeneous regions of an image. CLAHE, though, does not sharpen values higher than a given contrast threshold, thereby eliminating the issues of oversharpening and noise [42].

CLAHE should not be applied to RGB (red, blue, green) images. Applying CLAHE in color spaces like RGB and CMYK (cyan, magenta, yellow, key) will yield a different color distribution for each color channel. Instead of applying CLAHE separately to the R, G, B channels of a color image, a better approach applies the algorithm only to the luminance channel of a color image, which also prevents unwanted hue and saturation change. This, however, requires the source image to be converted to a different color space, e.g. HSV (hue, saturation, value) or CIELab (lightness, red/green, blue/yellow) first. Contrast enhancement in 3-D color spaces that makes use of luminance does not produce noisier images, unlike when processing in more common color spaces, ensuring color uniformity [47, 48]. We have utilized the CIELab space, since visual testing showed that the fish, rulers, and eyes were further pronounced than in HSV space. Detectron also yielded slightly better eye detection rates through contrast enhancement in CIELab space than in HSV space. Figure 4 shows a pre-enhanced image and the result after contrast enhancement.

4.3.3 Picking Fish of Highest Confidence

One of the issues with the previous metadata results was the detection of multiple fish in a single image, which was categorized as an error in our prior work. This "error" could occur with overlapping detection boxes over the same fish or through the erroneous detection of another random object in the image. We inspected the cases where multiple fish were detected. In all cases, the fish bounding box with the highest confidence value was the one that best covered the fish specimen present in the image. Additionally, there were never instances in which the fish bounding box of highest confidence was not actually a fish [6]. Since our study image collection only contains images with single specimen, when `detectron` returns more than one detected fish, we select the fish of highest confidence value, thus eliminating the previous multiple fish



Fish of Wisconsin

Noturus exilis Nelson, 1876 Ictaluridae
Det by: Marlin P. Johnson

USA, Wisconsin, Waukesha County, Bark River, highway 18 crossing.
TRS: 7N 17E sec 34 SW4
Water: Clear Current: Moderate to fast Vegetation: Elodea Bottom: Gravel and Mud Shore: Pasture Distance from shore: 40 feet wide Depth of capture: 0-1 1/2 feet Water temp: 24 C 1899-12-31T20:30:00Z Other chars: Caught in rocky riffle Condition: Good 1 specimen sent to Reeve Bailey at University of Michigan Museum, as number UMMZ 186511

Brian Belonger
18 August 1966
With: James W. Cahoon, Marlin P. Johnson

UWZM-F-0003373
4529

Fig. 5 An image in which the same fish was detected in two separate instances. The bounding box of highest confidence (green, 83%) provides the expected result.

error. Figure 5 shows an example in which fish were detected multiple times. The bounding box with the highest confidence score provides the expected result.

4.3.4 Image Scaling

The images in which an eye was not detected made up the majority of the erroneous cases. This led to a decision to rerun the model on images where a fish was detected, but the eye was not. It was hypothesized that the eyes were too small to be detected in these cases. To address these errors, the fish bounding box was cropped into a separate image, which was then upscaled by a factor of 4x, and the model was rerun on the scaled image. This helped to detect more eyes once they were scaled to a larger size.

If the eye is not detected even in the scaled image, it is counted as a missed eye. In the event an eye is detected on the scaled fish, however, the eye coordinate within the unscaled image needs to be returned. This requires taking the top left corner of the bounding box and adding the scaled down eye coordinate, as described in Equation 1:

$$\begin{cases} x_{eye\_original} = x_{fish\_top\_left} + \lfloor \frac{x_{eye\_scaled}}{4} \rfloor \\ y_{eye\_original} = y_{fish\_top\_left} + \lfloor \frac{y_{eye\_scaled}}{4} \rfloor \end{cases} \tag{1}$$

The most basic method for pixel interpolation during upscaling is the Nearest Neighbor algorithm, where the output pixel value is set to the nearest pixel's value [49]. Linear Interpolation estimates the appropriate intensity pixel values by finding the distance-weighted average of the four nearest pixels around the output pixel [50]. Bicubic Interpolation determines the pixel value from the weighted average of the 16 closest neighboring pixels utilizing a third-degree interpolant function [51].

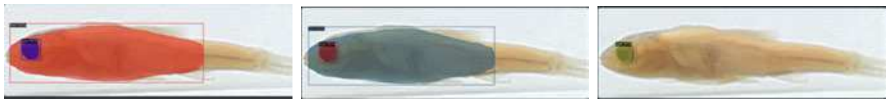


Fig. 6 Lanczos4 (left), Bicubic (middle), and Linear (right) interpolations.

Nearest Neighbor Interpolation was initially attempted with little effect on finding missing eyes. Linear Interpolation yielded significantly better results, while Bicubic Interpolation yielded the best results. Further research uncovered that there are more complex methods like Lanczos4 [52] Interpolation and even Deep Learning models like EDSR (Enhanced Deep Super-Resolution Networks) [53]. Testing demonstrated that Bicubic Interpolation yields slightly better accuracy than Lanczos4 and EDSR, although Lanczos4 and EDSR provided slightly better masking. Figure 6 presents the different scaling procedures on an image in which the fish was detected, but the eye initially was not.

4.4 Pixel Analysis

The masks and bounding boxes produced by `detectron` are generally quite good, although they almost never completely or tightly enclose the detected objects. The mask may include additional background as part of the fish, or the bounding box may clip away part(s) of the specimen. To solve these shortcomings, we utilize pixel analysis methods commonly found in image informatics to produce more accurate object masks and bounding boxes [6].

4.4.1 Threshold Adjustment

The first calculation in the pixel analysis process determines the cutoff intensity between what constitutes the foreground (i.e. the fish) and background of the image. Initially, the calculation is based on the bounding box and mask generated by `detectron`. Specimen images are read in as gray scale, and pixels in the image are treated as unsigned integers between 0 and 255. Otsu thresholding [54], a technique that maximizes the variance between the foreground and background intensities, is used to compute an initial cutoff value between foreground and background. While the Otsu value occasionally generates an accurate mask as is, usually the contrast between foreground and background is low and much of the lighter parts of the fish (such as its tail fin) are marked as background.

To overcome this improper segmentation, the threshold value should be either adjusted up or down, depending on whether the background is lighter or darker than the fish. For our current dataset, the background is always lighter (i.e. closer to 255), so the threshold value needs to be scaled up to include more of the foreground image. For optimal results the scaling should be dependent on the contrast between the background and foreground, which can be affected by the level of pigmentation of the fish.

We found that an improved threshold value can be computed as the halfway point between the Otsu threshold value and the mean of the background intensities. This adjusted threshold value usually produced an acceptable balance between capturing most of the fish's fins, without also masking parts of the background [6].

4.4.2 Consolidating the Foreground

While thresholding has the potential to generate better masks than a neural network (when provided an initial approximate bounding box), it also introduces considerable noise. Single or small groups of errant pixels can be marked as foreground depending on the consistency of the background, and interior pixels of the fish (especially around the fins) can be marked as background. To be useful for generating an accurate bounding box and for subsequent computational analysis, the mask must consist of one single "blob" over the fish, i.e. containing no holes, and no other pixels disconnected from this blob can be marked as foreground.

To accomplish this, we apply an iterative process of flood filling from all the foreground pixels in the image until a blob is generated that is large enough to constitute the fish. This leads to another metaparameter, but using greater than 10% of the current bounding box has masked the specimen in all observed cases. Once the fish's blob is found, internal noise then needs to be removed. This is done by flood filling from each of the corners of the bounding box, where the specimen is not present (all four corners in the overwhelming majority of cases), then taking the inverse of the result. The fish mask is excluded from these corner flood fills, so this process removes all noise from both the background and foreground of the image, leaving only a single mask over the fish itself [6].

4.4.3 Adjusting the Bounding Box

With a more accurate mask generated, it is then necessary to check whether the bounding box needs to be expanded or shrunk along any of its edges. Expansion is done first, by checking whether any edge intersects with any of the foreground mask pixels. If one does, the edge is expanded out by 1 pixel. If any edges are expanded, the whole process of masking and expansion is repeated to account for any changes in average intensities. Once no edges contain foreground pixels, the bounding box is then shrunk. Each edge is contracted by one pixel until it contains one or more foreground pixels. Once the shrinkage step is accomplished, the final mask and bounding box have been generated [6].

4.4.4 Fallback

The pixel analysis process occasionally fails, e.g. when flood-filling does not produce a large enough blob or the bounding box adjustment does not terminate. This can occur if certain flood fill operations behave unexpectedly, or if the image is too washed out or otherwise atypical for the thresholding process to work correctly. In the event this happens, the original mask and bounding box generated by `detectron` is used for metadata generation [6].

4.5 Metadata Generation

Table 4 lists the metadata properties that were generated in our previous work. These include properties that are produced by `detectron` and the methods described above: `has_fish`, `fish_count`, `has_ruler`, `ruler_bbox`, `background.{mean,std}`, `foreground.{mean,std}`, `bbox`, `score`, and `has_eye`. The methods to compute derived metadata properties are described below. The new properties that we are now able to generate are listed in Table 5.

Table 4 Original Metadata Properties (* indicates derived properties)

Property	Association	Type	Explanation
has_fish	Overall Image	Boolean	Whether a fish was found in the image.
fish_count	Overall Image	Integer	The quantity of fish present.
has_ruler	Overall Image	Boolean	Whether a ruler was found in the image.
ruler_bbox	Overall Image	4 Tuple	The bounding box of the ruler (if found).
scale*	Overall Image	Float	The scale of the image in $\frac{\text{pixels}}{\text{cm}}$.
bbox	Per Fish	4 Tuple	The top left and bottom right coordinates of the bounding box for a fish.
background.mean	Per Fish	Float	The mean intensity of the background within a given fish's bounding box.
background.std	Per Fish	Float	The standard deviation of the background within a given fish's bounding box.
foreground.mean	Per Fish	Float	The mean intensity of the foreground within a given fish's bounding box.
foreground.std	Per Fish	Float	The standard deviation of the foreground within a given fish's bounding box.
contrast*	Per Fish	Float	The contrast between foreground and background intensities within a given fish's bounding box.
centroid	Per Fish	4 Tuple	The centroid of a given fish's bitmask.
primary_axis*	Per Fish	2D Vector	The unit length primary axis (eigenvector) for the bitmask of a given fish.
clock_value*	Per Fish	Integer	Fish's primary axis converted into an integer "clock value" between 1 and 12.
oriented_length*	Per Fish	Float	The length of the fish bounding box in centimeters.
mask	Per Fish	2D Matrix	The bitmask of a fish in 0's and 1's.
pixel_analysis_failed	Per Fish	Boolean	Whether the pixel analysis process failed for a given fish. If *true*, *detectron*'s mask and bounding box were used for metadata generation.
score	Per Fish	Float	The percent confidence score output by *detectron* for a given fish.
has_eye	Per Fish	Boolean	Whether an eye was found for a given fish.
eye_center	Per Fish	2 Tuple	The centroid of a fish's eye.
side*	Per Fish	String	The side (i.e. *'left'* or *'right'*) of the fish that is facing the camera (dependent on finding its eye).

Table 5 Additional Metadata Properties

Property	Association	Type	Explanation
area	Per Fish	Float	Area of fish in cm^2.
cont_length	Per Fish	Float	The longest contiguous length of the fish in centimeters.
cont_width	Per Fish	Float	The longest contiguous width of the fish in centimeters.
convex_area	Per Fish	Float	Area of convex hull image (smallest convex polygon that encloses the fish) in cm^2.
eccentricity	Per Fish	Float	Ratio of the focal distance over the major axis length of the ellipse that has the same second-moments as the fish.
extent	Per Fish	Float	Ratio of pixels of fish to pixels in the total bounding box. Computed as $\frac{\text{area}}{\text{rows}*\text{cols}}$
feret_diameter_max	Per Fish	Float	The longest distance between points around the fish's convex hull contour.
major_axis_length	Per Fish	Float	The length of the major axis of the ellipse that has the same normalized second central moments as the fish.
mask.encoding	Per Fish	String	The 8-way Freeman Encoding of the outline of the fish.
mask.start_coord	Per Fish	2D Vector	The starting coordinate of the Freeman encoded mask.
minor_axis_length	Per Fish	Float	The length of the minor axis of the ellipse that has the same normalized second central moments as the fish.
oriented_width	Per Fish	Float	The width of the fish bounding box in centimeters.
perimeter	Per Fish	Float	The approximation of the contour in centimeters as a line through the centers of border pixels using 8-connectivity.
solidity	Per Fish	Float	The ratio of pixels in the fish to pixels of the convex hull image.
stddev	Per Fish	Float	The standard deviation of the mask pixel coordinate distribution.
skew	Per Fish	2D Vector	The measure of asymmetry of the frequency-distribution curve of mask pixel coordinates.
kurtosis	Per Fish	2D Vector	The sharpness of the peaks of the frequency-distribution curve of mask pixel coordinates.

4.5.1 Contrast

The contrast between the intensities of the foreground and background pixels is computed as `background.mean - foreground.mean` [6].

4.5.2 Centroid and Eye_center

Centroids are provided for the masks and bounding boxes that are generated by `detectron`, and since we do not recalculate the mask of fish eyes we can use that value directly for `eye_center`. Since we recalculate the mask of the fish, its centroid must be recalculated as well. This can be done through Equation 2:

$$(\bar{x}, \bar{y}) = (\text{round}(\frac{M_{10}}{M_{00}}), \text{round}(\frac{M_{01}}{M_{00}})), \tag{2}$$

where M_{00} is the pixel area of the fish's blob, M_{10} is the sum of all the x values of blob pixels, and M_{01} is the sum of all the y values of blob pixels [6].

4.5.3 Side

Determining which `side` of the fish is visible is predicated on finding its eye. If an eye is found, the sign of the x component of the vector from the centroid of the fish to the centroid of the eye specifies which side is up: negative for left and positive for right. This assumes the fish was photographed vertically (i.e. dorsal fin on top), which is essentially always the case for all image collections our group has worked on [6].

4.5.4 Primary_axis and Clock_value

The `primary_axis` of a fish can be calculated by taking the covariance of its blob in x and y, which yields its principle eigenvector. The eigenvector can be directly assigned to the property. If an eye is present, we ensure that `primary_axis` points in the direction of the eye relative to the fish's centroid.

Our team encoded this information as a "clock value" between 1 and 12 when manually recording it. To convert `principal_axis` to `clock_value`, the sign of x and y on the principal axis are used to determine which Cartesian quadrant the fish angles into relative to its centroid. Depending on the quadrant, we dot product the principal axis with either $[-1, 0]$, $[0, -1]$, $[1, 0]$ or $[0, 1]$, which correspond to 9, 6, 3 and "0" o'clock respectively. The resulting radian value is then converted to a polar displacement in clock value space, and added to the comparative clock value used in the dot product. This gives the fish's clock value from 0 to $11.\overline{9}$. Before recording `clock_value` in the output, the value is rounded to the nearest integer, with a 0 final result replaced with 12 [6].

4.5.5 Scale and Length

The fish length of INHS images, measured in $\frac{\text{pixels}}{\text{inch}}$, can be calculated by measuring the distance in pixels between the digits 2 and 3 (a 1 inch separation) found on the ruler by `detectron`. Converting this to $\frac{\text{pixels}}{\text{cm}}$ gives the `scale` metadata property as reported in the output. The UWZM images, in contrast, use a metric ruler in centimeters, and as such, the distance between the digits "2" and "3" is directly measured in $\frac{\text{pixels}}{\text{cm}}$.

For the fish `length` property, it is necessary to determine the furthest points from the centroid of the fish in each direction along its major axis. Since fish are normally measured in a straight line from their snout down the middle of their trunk, every pixel of the fish blob is projected onto the fish's major axis (as a line through its centroid). The projection is done through `numpy` [55] by performing PCA, or Principle Component Analysis. The first step of this process includes: finding all mask pixels, computing the covariance matrix, computing the eigenvalues and eigenvectors of the matrix, and then computing the angle of rotation from the X axis. The second part includes applying the negative rotation to the mask pixel coordinates, which aligns the fish's major axis with the X axis, and then computing the difference between the highest and lowest x values. Dividing this distance by `scale` gives the fish `length` in centimeters [6]. A similar process is done for the fish `width` as well, where the difference between highest and lowest y values are computed from the transformed pixels.

4.5.6 Contiguous Distances

Two additional computed properties are `cont_length` and `cont_width`. These are computed by using the transformed mask pixels from above, but with slight modifications. Through `numpy`, we examine the counts of the x and y values of the pixels. The indices of the x and y values with the highest counts are identified. This process identifies the x and y values with the longest contiguous strips parallel to the major and minor axes. The length and width calculations are computed as the difference between the max and min of the x and y values respectively within these bins.

4.5.7 Region Properties

One of the goals of the updated metadata generation process is to provide additional geometric properties based on the morphology of the fish. Features like `perimeter`, `area`, and `eccentricity` were immediately deemed most useful to the BGNN project use case, whereas further research is needed to determine other meaningful geometric properties. The Python machine learning library `skimage` [56] contains the `measure` package, which computes various geometric properties of the image. We made use of one of the provided functions, `regionprops`, which provides the aforementioned properties as well as: `feret_diameter_max`, `major_axis_length`, `minor_axis_length`, `solidity`, and `convex_area`. Other properties, like `euler_number` and `perimeter_crofton` are offered in this function, but were deemed unnecessary for our work.

4.5.8 Statistical Properties of the Mask Coordinates

The statistical distribution properties of the mask pixel coordinates can be calculated through Python statistical computing library `scipy` [57]. The `stddev`, `skew`, and `kurtosis` were calculated on the coordinate distribution and recorded in the metadata. These values can be used as distinguishing features of a fish's shape.

4.5.9 Mask Encoding

Another feature which may be useful for studying the morphology of fish is the outline of the specimen. A concise and efficient method for capturing the outlining boundary of an object is Freeman Chain Encoding [58]. In general, a chain code is a lossless compression algorithm for monochrome images that separately encodes the boundary of each connected component—or "blob"—in an image. For each such region, a point on the boundary is selected and its coordinates are noted. The encoder then moves along the boundary of the region and, at each step stores a symbol representing the direction of the movement. This procedure continues until the encoder returns to the starting position, at which point the blob has been completely encircled. By storing the encoding and the start coordinate, it is easy to recreate the mask by reverse encoding the sequence, then flood filling the mask. The encoding of the outline should serve as a signature that supports morphological comparisons between fish specimens, as well as providing a compressed representation of the specimen's mask.

5 Results

The computational enhancements described in subsection 4.3 produced a reduction in the overall error rate from 4.6% to 1.1%. Here, an error is defined as the inability to detect a fish, a fish eye, ruler or the numbers '2' and '3', which are used to compute image scale, within a specimen image. Whereas the INHS metadata generation process took 3.5 hours to run on 7,244 images, our GPU optimizations have helped reduce this to 2 hours on 7,013 images. The metadata generation process on the UWZM set of 4,013 images takes 2.5 hours. The difference in computation time is reasonable, since the UWZM images have higher resolution by an order of magnitude compared to the INHS images.

To demonstrate the effects of our error reduction techniques, we have run the original INHS-only model on the refined INHS dataset and compared the results of various enhancement combinations in Table 7. Additionally, we have computed results for the various enhancement combinations included in the newly trained INHS + UWZM model, which is applied to the combined INHS and UWZM testing set, as well as on the individual INHS and UWZM testing sets. The results from these studies are presented in Table 8 through Table 10.

5.1 Fish Detection

As prescribed by our collection criteria, images in the INHS and UWZM testing datasets contain exactly one fish. In the case of no enhancements, except training set augmentation, 11,125 out of 11,168 images had exactly one fish detected, a 99.6% correct rate. In 42 of the images, multiple fish were found. The one fish that was not detected was an extremely small fish from the INHS collection. This type of specimen is currently missing from the training set. In the multiple fish cases, 1 of the 42 contained tags that overlapped the fish and were themselves labeled as a second fish. Of the remaining 41, `detectron` erroneously labeled the fish as two or more separate fish objects, or labeled a subsection of the fish multiple times. Fish that were "over-detected" were generally quite large and/or dissimilar from the fish found in the training set. Applying the rule that selects the fish object of highest confidence produces no

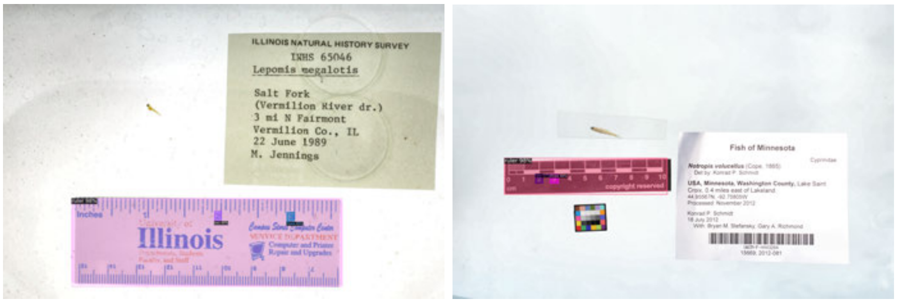


Fig. 7 The INHS image where the fish is never detected (left) and the UWZM image where the fish is not detected after contrast enhancement (right).

images with multiple fish. Examining the 42 multiple-fish cases showed that this approach always produced the correct result, with 11,167 out of 11,168 images having exactly one fish detected, a 99.9% correct rate. After applying contrast enhancement a second small fish, in the UWZM collection, was not detected, with 11,166 out of 11,168 images having exactly one fish detected, a 99.9% correct rate. Figure 7 shows the two images in which fish were not detected. These results are summarized in Table 8, Columns 3 and 5.

5.2 Ruler Detection

When no enhancements are employed `detectron` is able to detect rulers in all of the test images. Applying contrast enhancement produces one image where the ruler is not found. Still this produces a 99.9% correct rate. When contrast enhancement was not employed and the ruler was found, there were 28 cases where the numbers “2” and/or “3” on the ruler were not detected. Therefore, a scale calculation could not be performed, producing a 99.7% success rate for the `scale` computation. Applying contrast enhancement improves the calculation, with 17 images for which the numbers were not detected and the scale could not be computed. As seen in Tables 9 and 10, Columns 6 and 7, these errors all come from the INHS dataset. This is understandable, since the images in the UWZM dataset are extremely consistent, whereas the overall INHS image quality can vary significantly.

Images where one of these objects were not detected generally had some form of coloration issue. They were either washed out, very dark or yellow in hue. Some of the rulers for which “2” and/or “3” were not detected were particularly scratched and damaged. Many of the rulers where both numerals were missed were particularly small within the image, which again may be solvable through expanding the training dataset to more collections than just INHS and UWZM and/or adding more 2’s and 3’s to the training data.

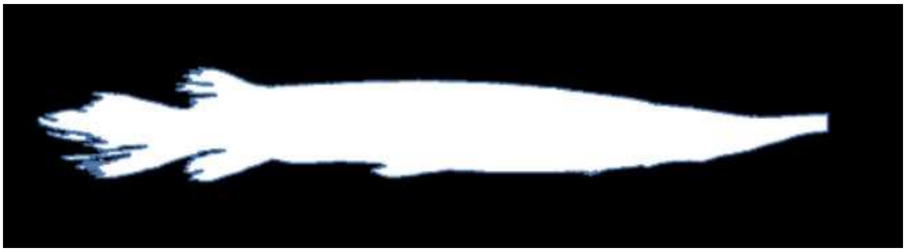


Fig. 8 The mask and outline of a fish, which are generated through pixel analysis and Freeman Encoding respectively.

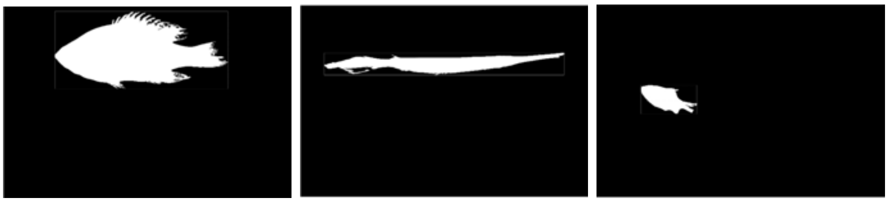


Fig. 9 The masks generated for INHS_FISH_9552 (left), INHS_FISH_80975 (middle), and INHS_FISH_026311 (right).

5.3 Eye Detection

In the case of no enhancements (besides training set augmentation), `detectron` was unable to find a fish eye in 273 of the images, a 2.4% error rate. Applying upscaling to the fish images produced a significant improvement with 206 images having no detected eyes, a 1.8% error rate. Applying contrast enhancement showed yet another major improvement with 112 missing eyes, giving an error rate of 1.0%. Contrast enhancement was meant to help all categories of errors, but ended up helping missing eyes the most. Upon investigation, the undetected eyes were generally extremely dark, small, or looked nothing like those found in the training set. These cases usually included catfish and extremely small fish, along with various fish where the eyes are effectively unrecognizable.

5.4 Mask Generation and Encoding

Fish bounding boxes were calculated for all images in which a fish was found. Manual investigation demonstrated that due to the thresholding process in grayscale space, in many cases, near translucent and light hue fins and tails were excluded. Masks and bounding boxes contain the head and trunk of the fish in nearly all cases, but further refinement of our algorithms will be needed to ensure that light fins and tails are masked consistently and accurately. The masks were then encoded and stored in the metadata along with the starting coordinate used in the encoding process. Figure 8 presents the mask (white pixels), as well as the encoded outline of the fish (blue pixels) from Figure 4.

5.5 Scale and Length

Image scale and fish lengths were calculated for $11,148$ of the images. For the remaining 20 images, either the fish, the "2" and/or the "3" on the ruler were not detected. Image scale ($\frac{\text{pixels}}{\text{cm}}$) and fish length were measured, using ImageJ [59]. Scale calculations using the "2" and "3" method are nearly identical to those calculated by hand between the tick marks on the ruler. When the tail of the fish is accurately masked and the specimen is fairly straight, the length calculation is highly accurate as well. Thus, the primary means of lowering the error of the length calculation is to improve tail masking accuracy.

5.6 Region and Statistical Properties

Region and statistical properties were computed from the masks for the three specimen images in Figure 9. The property values are listed in Table 6 and demonstrate that they provide distinguishing features based on the shape of the fish specimens.

Table 6 Metadata Property Comparisons

Property	**INHS_FISH_9552**	**INHS_FISH_80975**	**INHS_FISH_026311**
area	23.5	10.3	4.03
cont_length	9.60	10.7	2.14
cont_width	4.08	1.06	1.88
convex_area	27.5	15.5	5.05
eccentricity	0.90	0.99	0.91
extent	0.55	0.47	0.51
feret_diameter_max	9.65	15.2	4.03
perimeter	58.7	40.1	15.0
solidity	0.85	0.66	0.79
stddev	[165.6, 69.6]	[247.9, 17.9]	[52.9, 29.2]
skew	[0.19, -0.15]	[0.13, 0.07]	[0.22, -0.03]
kurtosis	[-0.66, -0.72]	[-0.94, -0.82]	[-0.68, -1.10]

Table 7 INHS Model Applied to the INHS Dataset, 100k epochs

Enhancements	Total	No Fish	No Eye	Multiple Fish	No Ruler	No Scale
C,U,S	175 (2.5%)	14 (0.19%)	153 (2.2%)	0 (0%)	0 (0%)	25 (0.35%)
U,S	266 (3.8%)	7 (0.09%)	212 (3.0%)	0 (0%)	2 (0.02%)	57 (0.81%)
None	320 (4.6%)	7 (0.09%)	240 (3.4%)	26 (0.37%)	2 (0.02%)	57 (0.81%)

n = 7,013

C - Contrast Enhancement

U - Upscaling

S - Fish Selection

Table 8 INHS + UWZM Model Applied to the INHS and UWZM Datasets, 15k epochs

Enhancements	Total	No Fish	No Eye	Multiple Fish	No Ruler	No Scale
A,C,U,S	124 (1.1%)	2 (0.02%)	112 (1.0%)	0 (0%)	1 (0.01%)	17 (0.15%)
A,C,S	198 (1.8%)	2 (0.02%)	182 (1.6%)	0 (0%)	1 (0.01%)	17 (0.15%)
A,U,S	229 (2.1%)	0 (0.01%)	205 (1.8%)	0 (0%)	0 (0%)	28 (0.25%)
A,S	300 (2.7%)	0 (0.01%)	273 (2.4%)	0 (0%)	0 (0%)	28 (0.25%)
A	338 (3.0%)	0 (0.01%)	273 (2.4%)	42 (0.37%)	0 (0%)	28 (0.25%)

n = 11,168

A - Augmenting Training Set

Table 9 INHS + UWZM Model Applied to the INHS Dataset, 15k epochs

Enhancements	Total	No Fish	No Eye	Multiple Fish	No Ruler	No Scale
A,C,U,S	96 (1.4%)	1 (0.01%)	84 (1.2%)	0 (0%)	1 (0.01%)	17 (0.24%)
A,U,S	198 (2.8%)	1 (0.01%)	174 (2.5%)	0 (0%)	0 (0%)	28 (0.39%)
A	240 (3.4%)	1 (0.01%)	205 (2.9%)	11 (0.15%)	0 (0%)	28 (0.39%)

n = 7,013

Table 10 INHS + UWZM Model Applied to the UWZM Dataset, 15k epochs

Enhancements	Total	No Fish	No Eye	Multiple Fish	No Ruler	No Scale
A,C,U,S	28 (0.67%)	1 (0.02%)	28 (0.67%)	0 (0%)	0 (0%)	0 (0%)
A,U,S	31 (0.74%)	0 (0%)	31 (0.74%)	0 (0%)	0 (0%)	0 (0%)
A	98 (2.4%)	0 (0%)	68 (1.6%)	31 (0.74%)	0 (0%)	0 (0%)

n = 4,155

6 Discussion

Where our previous work presented a proof of concept in advanced image analysis for automatic metadata generation, our current results demonstrate the extensibility of our approaches. By extending our work to a different dataset, we demonstrate that our model can be generalized as long as various image and specimen quality criteria are adhered to. By doing so, our methods have performed with high accuracy with minimal additions of unseen data to the training set. There has also been great success with applying various error reduction techniques that include image scaling, an augmented training set, selection of fish with the highest confidence score, and contrast enhancement. By augmenting the training set and performing contrast enhancement, the error rates on each class generally decreased. By performing image scaling on the cropped fish where an eye was initially missing, the number of missing eyes dropped significantly. By performing fish selection, given the nature of the images where multiple fish were detected, the best fish mask was always selected, eliminating the multiple fish error.

While we have seen some individual error rates increase after applying contrast enhancement, the overall effect is a lower error rate for the aggregate of all errors. Surprisingly, applying image scaling to the rulers had no effect on improving the number of missing "two"s and "three"s , indicating a need for more training data. Additional training epochs will not improve these errors, since we found that training beyond 15,000 epochs yielded worse results. This is known as the **exploding gradient problem**, a common problem in deep learning that has been evident since the advent of gradient-based parameter learning [60]. Our current results are more than acceptable and demonstrate an augmented proof of concept that offers a path forward for using object detection technology, enhanced by image informatics techniques, to improve and enrich the metadata needed for advanced specimen image analysis. Overall, our work should advance scientific discovery that is based on analysis of biological specimen image collections.

Our investigation has thus far focused on fish as the specimen of study. Fish are vertebrate animals (phylum Chordata), with over $34,000$ known unique species [61], with many more likely undiscovered. Species names are merely labels, and the discovery of species variation depends on both genotype and phenotype information. The ability to computationally analyze thousands of images of a single fish species, from different habitats and time periods, can lead to new discoveries that are impossible to pursue with manual methods. Digital library researchers have been concerned with computationally extracting image features, using content-based image retrieval methods. The work by Toress [62], while over 15 years old, demonstrates the challenges and opportunities to automatically generating useful metadata. Efforts to integrate such automatic metadata generation methods into digital library workflows and architectures still seem limited. This is likely due to the diversity of image shapes, sizes and the inconsistent configurations of specimens, labels, rulers, etc. within them. Object detection as explored in our research, working with an established architecture, is applicable to the larger world of biodiversity, well beyond fish, to include other fauna and flora, art and artefacts, and other digitized objects made accessible for scientific and scholarly research. Following object detection, one can apply pixel analysis and informatics methods to compute many more higher order properties from the initial segmentations.

7 Conclusion

In this paper we extended a previously described automatic metadata generation approach. Using ML and image informatics algorithms, along with a number of image processing methods, our approach is able to locate, mask and analyze specimens (currently limited to fish) in collection images with a high degree of accuracy. Additional geometric measurements on the specimens are now computed, while also improving the overall error rates, as well as the runtime through GPU parallelization. Testing this approach on 7,013 images from the INHS dataset and 4,155 images from the UWZM dataset, we see major success with only 1.1% of the 11,186 images yielding at least one error. Through further refinement and generalization beyond the INHS and UWZM images, as well as more species than just fish, we aim to create a tool that can be distributed to specimen image collection curators to correct the metadata sparsity that motivated this work.

7.1 Future Work

The most pressing next step is to refine the pixel analysis thresholding process in order to improve the accuracy of the specimen masks. The current process performs thresholding on a single color channel (intensity). Some of the lightest tails appear yellow in hue to the human eye and easily distinguishable, but when compressed to a single intensity value they are almost identical in value to the grayish background. Given what we have learned with contrast equalization, using CIELab space could be ideal for mask adjustment. Another possible approach to solving this problem is to threshold and mask on subsets of the bounding box, as to ensure that very dark trunk pixels do not affect the thresholding of lighter regions.

Our longer term goal is to create a generalized process that works on classes of specimen images. For the BGNN project we are beginning with fish images, but we are designing the metadata generation system so that it can eventually operate on other species, if appropriately trained. The first step, which has been accomplished, was to achieve greater generality by augmenting the training set from INHS to UWZM. Another requirement will be to generalize the ruler reading process beyond the reading of digits on the ruler, which will likely involve an automated method of reading ruler ticks instead of digits. Lastly, the model training setup should be modified from using the default parameters to one that is further optimized. As a result of the aforementioned factors, training beyond 15,000 epochs has yielded exploding gradients, thus producing poorer results. A suggested improvement is to experiment with the learning rate, with more complex solutions involving the use of a learning rate scheduler or optimization algorithms like RMSProp and ADAM.

Overall, the research reported in this paper will improve our BGNN workflow, and at the same time demonstrates an innovative approach that should greatly enhance digital library services for the tens of thousands of digitized specimens in a spectrum of image collections.

Acknowledgments. We thank the full BGNN team for support,Chris A. Taylor, Curator of Fishes and Crustaceans at the Illinois Natural History Survey (INHS), and John Lyons, Curator of Fish at the University of Wisconsin Zoological Museum (UWZM). INHS and UWZM are two of six fish collections participating in the Great Lakes Invasives Network (GLIN).

Funding. Research supported by NSF OAC Office of Advanced Cyberinfrastructure (OAC) #1940233 and #1940322.

Competing Interests. The authors have declared that no competing interests exist.

Code and Data Availability. Raw data for INHS is available here. Raw data for UWZM is available here. Reproducible code is available here.

References

[1] Beaman, R.S., Cellinese, N.: Mass digitization of scientific collections: New opportunities to transform the use of biological specimens and underwrite biodiversity science. ZooKeys (209), 7 (2012)

[2] Page, L.M., MacFadden, B.J., Fortes, J.A., Soltis, P.S., Riccardi, G.: Digitization of biodiversity collections reveals biggest data on biodiversity. BioScience **65**(9), 841–842 (2015)

[3] Tibbetts, J.H.: The frontiers of artificial intelligence. BioScience **68**(1), 5–10 (2018)

[4] Darwin Core Quick Reference Guide. https://dwc.tdwg.org/terms/

[5] Leipzig, J., Bakis, Y., Wang, X., Elhamod, M., Diamond, K., Dahdul, W., Karpatne, A., Maga, M., Mabee, P., Bart, H.L., Greenberg, J.: Biodiversity image quality metadata augments convolutional neural network classification of fish species. In: Garoufallou, E., Ovalle-Perandones, M.-A. (eds.) Metadata and Semantic Research, pp. 3–12. Springer, Cham (2021)

[6] Pepper, J., Greenberg, J., Bakiş, Y., Wang, X., Bart, H., Breen, D.: Automatic metadata generation for fish specimen image collections. In: 2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 31–40 (2021). https://doi.org/10.1109/JCDL52503.2021.00015

[7] University of Wisconsin Zoological Museum: Fishes Collection. https://uwzm.integrativebiology.wisc.edu/fishes-collection/ (2022)

[8] Darwin Core Maintenance Group: List of Darwin Core terms. http://rs.tdwg.org/dwc/doc/list/ (2020)

[9] GBIF/TDWG Multimedia Resources Task Group: Audubon Core Multimedia Resources Metadata Schema. http://www.tdwg.org/standards/638 (2013)

[10] Chapman, A., Belbin, L., Zermoglio, P., Wieczorek, J., Morris, P., Nicholls, M., Rees, E.R., Veiga, A., Thompson, A., Saraiva, A., *et al.*: Developing standards for improved data quality and for selecting fit for use biodiversity data. Biodiversity Information Science and Standards **4**, 50889 (2020)

[11] Wieczorek, J., Bloom, D., Guralnick, R., Blum, S., Döring, M., Giovanni, R., Robertson, T., Vieglais, D.: Darwin core: an evolving community-developed biodiversity data standard. PloS one **7**(1), 29715 (2012)

[12] Liddy, E.D., Allen, E., Harwell, S., Corieri, S., Yilmazel, O., Ozgencil, N.E., Diekema, A., McCracken, N., Silverstein, J., Sutton, S.: Automatic metadata generation & evaluation. In: Proc. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 401–402 (2002)

[13] Greenberg, J.: Metadata extraction and harvesting: A comparison of two automatic metadata generation applications. Journal of Internet Cataloging **6**(4), 59–82 (2004)

[14] Cardinaels, K., Meire, M., Duval, E.: Automating metadata generation: the simple indexing interface. In: Proc. International Conference on World Wide Web, pp. 548–556 (2005)

[15] Paynter, G.W.: Developing practical automatic metadata assignment and evaluation tools for internet resources. In: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05), pp. 291–300 (2005). IEEE

[16] Han, H., Giles, C.L., Manavoglu, E., Zha, H., Zhang, Z., Fox, E.A.: Automatic document metadata extraction using support vector machines. In: Proc. Joint Conference on Digital Libraries, pp. 37–48 (2003). IEEE

[17] Rodriguez, M.A., Bollen, J., Sompel, H.V.D.: Automatic metadata generation using associative networks. ACM Transactions on Information Systems **27**(2), 1–20 (2009)

[18] Heidorn, P.B., Wei, Q.: Automatic metadata extraction from museum specimen labels. In: International Conference on Dublin Core and Metadata Applications, pp. 57–68 (2008)

[19] Manso, M., Nogueras-Iso, J., Bernabe, M., Zarazaga-Soria, F.: Automatic metadata extraction from geographic information. In: 7th Conference on Geographic Information Science (AGILE 2004), Heraklion, Greece, pp.

379–385 (2004)

[20] Zion, B., Shklyar, A., Karplus, I.: In-vivo fish sorting by computer vision. Aquacultural Engineering **22**, 165–179 (2000)

[21] Saberioon, M., Gholizadeh, A., Císař, P., Pautsina, A., Urban, J.: Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues. Reviews in Aquaculture **9**, 369–387 (2017)

[22] Hu, J., Li, D., Duan, Q., Han, Y., Chen, G., Si, X.: Fish species classification by color, texture and multi-class support vector machine using computer vision. Computers and Electronics in Agriculture **88**, 133–140 (2012)

[23] Vapnik, V.N.: An overview of statistical learning theory. IEEE Transactions on Neural Networks **10**(5), 988–999 (1999)

[24] Li, L., Hong, J.: Identification of fish species based on image processing and statistical analysis research. In: Proc. IEEE International Conference on Mechatronics and Automation, pp. 1155–1160 (2014)

[25] Rodrigues, M.T.A., Freitas, M.H.G., Pádua, F.L.C., Gomes, R.M., Carrano, E.G.: Evaluating cluster detection algorithms and feature extraction techniques in automatic classification of fish species. Pattern Analysis and Applications **18**(4), 783–797 (2015)

[26] Hernández-Serna, A., Jiménez-Segura, L.F.: Automatic identification of species with neural networks. PeerJ **2:e563** (2014)

[27] Salman, A., Jalal, A., Shafait, F., Mian, A., Shortis, M., Seager, J.W., Harvey, E.: Fish species classification in unconstrained underwater environments based on deep learning. Limnology and Oceanography-Methods **14**, 570–585 (2016)

[28] LeCun, Y., F.J. Huang, Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 104–2 (2004)

[29] Alsmadi, M., Tayfour, M., Alkhasawneh, R., Badawi, U., Almarashdeh, I., Haddad, F.: Robust features extraction for general fish classification. International Journal of Electrical and Computer Engineering **9**, 5192 (2019)

[30] Iqbal, M.A., Wang, Z., Ali, Z., Riaz, S.: Automatic fish species classification using deep convolutional neural networks. Wireless Personal Communications **116**, 1043–1053 (2021)

[31] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. 25th International Conference on Neural Information Processing Systems - Volume 1, pp. 1097–1105 (2012)

[32] Yu, C., Fan, X., Hu, Z., Xia, X., Zhao, Y., Li, R., Bai, Y.: Segmentation and measurement scheme for fish morphological features based on mask r-cnn. Information Processing in Agriculture **7**(4), 523–534 (2020)

[33] Petrellis, N.: Measurement of fish morphological features through image processing and deep learning techniques. Applied Sciences **11**, 4416 (2021)

[34] Hao, M., Yu, H., Li, D.: The measurement of fish size by machine vision - a review. In: Proc. 9th International Conference on Computer and Computing Technologies in Agriculture, pp. 15–32 (2015)

[35] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)

[36] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc., ??? (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[37] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, pp. 740–755. Springer, Cham (2014). https://doi.org/10.48550/arXiv.1405.0312

[38] J. F. Bell Museum of Natural History: Fishes Collection. https://www.bellmuseum.umn.edu/fishes/ (2022)

[39] Skalski, P.: Make Sense. https://github.com/SkalskiP/make-sense/ (2019)

[40] Illinois Natural History Survey: INHS Fish Collection. https://fish.inhs.illinois.edu/ (2022)

[41] Cai, T., Zhu, F., Hao, Y., Fan, X.: Performance evaluation of image

enhancement methods for objects detection and recognition. In: Proceedings of the SPIE: Image Processing and Analysis, vol. 9675. SPIE, ??? (2015)

[42] Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J.B., Zuiderveld, K.: Adaptive histogram equalization and its variations. Computer Vision, Graphics, and Image Processing **39**(3), 355–368 (1987). https://doi.org/10.1016/S0734-189X(87)80186-X

[43] Manju, R.A., Koshy, G., Simon, P.: Improved method for enhancing dark images based on clahe and morphological reconstruction. Procedia Computer Science **165**, 391–398 (2019). https://doi.org/10.1016/j.procs.2020.01.033. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019

[44] Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)

[45] Singh, R., Biswas, M.: Adaptive histogram equalization based fusion technique for hazy underwater image enhancement. In: 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1–5 (2016). https://doi.org/10.1109/ICCIC.2016.7919711

[46] Lucas, J., Calef, B., Knox, K.: Image enhancement for astronomical scenes. In: Tescher, A.G. (ed.) Applications of Digital Image Processing XXXVI, vol. 8856, pp. 12–19. SPIE, ??? (2013). https://doi.org/10.1117/12.2025191. International Society for Optics and Photonics. https://doi.org/10.1117/12.2025191

[47] Yu, H., Inoue, K., Hara, K., Urahama, K.: Saturation improvement in hue-preserving color image enhancement without gamut problem. ICT Express **4**(3), 134–137 (2018). https://doi.org/10.1016/j.icte.2017.07.003

[48] Trahanias, P.E., Venetsanopoulos, A.N.: Color image enhancement through 3-d histogram equalization. In: Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,, pp. 545–548 (1992). https://doi.org/10.1109/ICPR.1992.202045

[49] Reddy, K.S., Reddy, D.K.R.L.: Enlargement of image based upon interpolation techniques. International Journal of Advanced Research in Computer and Communication Engineering **2**(12), 4631 (2013)

[50] Vidya, M.S., Shastry, A.H., Mallya, Y.: 4 - automated detection of intracranial hemorrhage in noncontrast head computed tomography.

In: Koundal, D., Gupta, S. (eds.) Advances in Computational Techniques for Biomedical Image Analysis, pp. 71–98. Academic Press, ??? (2020). https://doi.org/10.1016/B978-0-12-820024-7.00004-9. https://www.sciencedirect.com/science/article/pii/B9780128200247000049

[51] Keys, R.: Cubic convolution interpolation for digital image processing. IEEE Transactions on Acoustics, Speech, and Signal Processing **29**(6), 1153–1160 (1981). https://doi.org/10.1109/TASSP.1981.1163711

[52] Turkowski, K.: Filters for common resampling tasks. Graphics gems, 147–165 (1990). https://doi.org/10.1.1.116.7898

[53] Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)

[54] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics **9**(1), 62–66 (1979)

[55] Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2

[56] van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., the scikit-image contributors: scikit-image: image processing in Python. PeerJ **2**, 453 (2014). https://doi.org/10.7717/peerj.453

[57] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2

[58] Freeman, H.: On the encoding of arbitrary geometric configurations. IRE Transactions on Electronic Computers **EC-10**(2), 260–268 (1961). https://doi.org/10.1109/TEC.1961.5219197

[59] Schneider, C.A., Rasband, W.S., Eliceiri, K.W.: NIH Image to ImageJ:

25 years of image analysis. Nature Methods **9(7)**, 671–675 (2012)

[60] Hochreiter, S.: Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Universität München **91**(1) (1991)

[61] Team, F.: FishBase. https://www.fishbase.de/search.php (Last update: 2/2020)

[62] Torres, R.d.S., Medeiros, C.B., Gonçcalves, M.A., Fox, E.A.: A digital library framework for biodiversity information systems. International Journal on Digital Libraries **6**(1), 3–17 (2006)

```
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020/W32TeX)
(preloaded format=pdflatex 2020.5.12)  30 MAR 2022 13:12
entering extended mode
 restricted \write18 enabled.
 %&-line parsing enabled.
**sn-sample-bib.tex
(./sn-sample-bib.tex
LaTeX2e <2020-02-02> patch level 5
L3 programming layer <2020-05-05>

! LaTeX Error: Environment thebibliography undefined.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.1 \begin{thebibliography}
                           {9}
Your command was ignored.
Type  I <command> <return>  to replace it with another command,
or  <return>  to continue without it.


! LaTeX Error: Missing \begin{document}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.1 \begin{thebibliography}{9
                             }
You're in trouble here.  Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

Missing character: There is no 9 in font nullfont!

Overfull \hbox (20.0pt too wide) in paragraph at lines 1--2
[]
 []


! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.3 \bibitem{bib1}
                  I.~Podlubny, Fractional Differential Equations,
Academie ...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.
```

```
! Undefined control sequence.
<argument> \@listctr

l.3 \bibitem{bib1}
                   I.~Podlubny, Fractional Differential Equations,
Academie ...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.3 \bibitem{bib1}
                   I.~Podlubny, Fractional Differential Equations,
Academie ...
I'm forgetting what you said and using zero instead.

\bibcite{bib1}{0}
Missing character: There is no I in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no F in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no D in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no E in font nullfont!
Missing character: There is no q in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no N in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no Y in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no k in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.5 \bibitem{bib2}
                   R. Hilfer, Application of Fractional Calculus in
Physics,...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

! Undefined control sequence.
<argument> \@listctr
```

```
l.5 \bibitem{bib2}
                  R. Hilfer, Application of Fractional Calculus in
Physics,...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.5 \bibitem{bib2}
                  R. Hilfer, Application of Fractional Calculus in
Physics,...
I'm forgetting what you said and using zero instead.

\bibcite{bib2}{0}
Missing character: There is no R in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no H in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no F in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no C in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no W in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no S in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no S in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.
```

```
See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.7 \bibitem{bib3}
                  A. Nagih, G. Plateau, Fractional problems: overview of
ap...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

! Undefined control sequence.
<argument> \@listctr

l.7 \bibitem{bib3}
                  A. Nagih, G. Plateau, Fractional problems: overview of
ap...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.7 \bibitem{bib3}
                  A. Nagih, G. Plateau, Fractional problems: overview of
ap...
I'm forgetting what you said and using zero instead.

\bibcite{bib3}{0}
Missing character: There is no A in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no N in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no G in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no F in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no : in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no R in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no R in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no O in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no O in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no N in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no Y in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no k in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
```

```
Type  H <return>  for immediate help.
 ...

l.9 \bibitem{bib4}
                  J. P. Richard, Time-delay system: an overview of some
rec...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

! Undefined control sequence.
<argument> \@listctr

l.9 \bibitem{bib4}
                  J. P. Richard, Time-delay system: an overview of some
rec...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.9 \bibitem{bib4}
                  J. P. Richard, Time-delay system: an overview of some
rec...
I'm forgetting what you said and using zero instead.

\bibcite{bib4}{0}
Missing character: There is no J in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no R in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no T in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no : in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 3 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no ( in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no ) in font nullfont!
Missing character: There is no ( in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 3 in font nullfont!
Missing character: There is no ) in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 6 in font nullfont!
Missing character: There is no 6 in font nullfont!
Missing character: There is no 7 in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 6 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no 4 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.11 \bibitem{bib5}
                    H. Ye, J. Gao, Y. Ding, A generalized Gronwall
inequalit...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

! Undefined control sequence.
<argument> \@listctr

l.11 \bibitem{bib5}
                    H. Ye, J. Gao, Y. Ding, A generalized Gronwall
inequalit...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
```

```
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.11 \bibitem{bib5}
                   H. Ye, J. Gao, Y. Ding, A generalized Gronwall
inequalit...
I'm forgetting what you said and using zero instead.

\bibcite{bib5}{0}
Missing character: There is no H in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no Y in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no J in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no G in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no Y in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no D in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no z in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no G in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no q in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no q in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no J in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no M in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 3 in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 8 in font nullfont!
Missing character: There is no ( in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 7 in font nullfont!
Missing character: There is no ) in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 7 in font nullfont!
Missing character: There is no 5 in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 8 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.13 \bibitem{bib6}
                   M. Lazarevic, Stability and stabilization of
fractional ...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.
```

```
! Undefined control sequence.
<argument> \@listctr

l.13 \bibitem{bib6}
                    M. Lazarevic, Stability and stabilization of
fractional ...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.13 \bibitem{bib6}
                    M. Lazarevic, Stability and stabilization of
fractional ...
I'm forgetting what you said and using zero instead.

\bibcite{bib6}{0}
Missing character: There is no M in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no L in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no z in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no S in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no z in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no S in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no T in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no R in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no w in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 6 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no ( in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no ) in font nullfont!
Missing character: There is no 3 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no 4 in font nullfont!
Missing character: There is no 5 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: Lonely \item--perhaps a missing list environment.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.15 \bibitem{bib7}
                    Y. Li, Y. Q. Chen, I. Podlubny, Stability of
fractional-...

Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.

! Undefined control sequence.
<argument> \@listctr

l.15 \bibitem{bib7}
                    Y. Li, Y. Q. Chen, I. Podlubny, Stability of
fractional-...
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

! You can't use `\relax' after \the.
<recently read> \c@

l.15 \bibitem{bib7}
```

```
                    Y. Li, Y. Q. Chen, I. Podlubny, Stability of
fractional-...
I'm forgetting what you said and using zero instead.

\bibcite{bib7}{0}
Missing character: There is no Y in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no L in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no Y in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no Q in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no C in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no I in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no P in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no S in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no : in font nullfont!
Missing character: There is no L in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no u in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no v in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no c in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no n in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no z in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no d in font nullfont!
Missing character: There is no M in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no g in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no L in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no f in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no e in font nullfont!
Missing character: There is no r in font nullfont!
Missing character: There is no s in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no b in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no i in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no y in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no C in font nullfont!
Missing character: There is no o in font nullfont!
Missing character: There is no m in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no M in font nullfont!
Missing character: There is no a in font nullfont!
Missing character: There is no t in font nullfont!
Missing character: There is no h in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no A in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no p in font nullfont!
Missing character: There is no l in font nullfont!
Missing character: There is no . in font nullfont!
Missing character: There is no , in font nullfont!
Missing character: There is no 5 in font nullfont!
Missing character: There is no 9 in font nullfont!
Missing character: There is no ( in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no ) in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 8 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 0 in font nullfont!
Missing character: There is no - in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no 8 in font nullfont!
Missing character: There is no 2 in font nullfont!
Missing character: There is no 1 in font nullfont!
Missing character: There is no . in font nullfont!

! LaTeX Error: \begin{document} ended by \end{thebibliography}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

l.18 \end{thebibliography}

Your command was ignored.
Type  I <command> <return>  to replace it with another command,
or  <return>  to continue without it.

)
! Emergency stop.
<*> sn-sample-bib.tex

*** (job aborted, no legal \end found)


Here is how much of TeX's memory you used:
 18 strings out of 480681
 433 string characters out of 5908536
 237875 words of memory out of 5000000
 15941 multiletter control sequences out of 15000+600000
 532338 words of font info for 24 fonts, out of 8000000 for 9000
 1141 hyphenation exceptions out of 8191
 12i,1n,15p,231b,38s stack positions out of
5000i,500n,10000p,200000b,80000s
!  ==> Fatal error occurred, no output PDF file produced!
```