

An autofocus algorithm for digital cameras based on supervised machine learning

Hashim Mir, Peter Xu, and Peter van Beek
Cheriton School of Computer Science
University of Waterloo
vanbeek@uwaterloo.ca

Abstract

Digital cameras are equipped with passive autofocus mechanisms where a lens is focused using only the camera’s optical system and an algorithm for controlling the lens. The speed and accuracy of the autofocus algorithm are crucial to user satisfaction. In this paper, we show that supervised machine learning techniques can be used to construct a passive autofocus algorithm that out-performs an existing state-of-the-art hand-crafted algorithm. In our approach, training and test data were produced using an offline simulation on a suite of 25 benchmarks and correctly labeled in a semi-automated manner. A decision tree learning algorithm was then used to induce an autofocus algorithm from the data. The automatically constructed machine-learning-based (ml-based) algorithm was compared against the best previously proposed hand-crafted algorithm for autofocusing. In our experiments, the ml-based algorithm had improved speed—reducing the number of steps needed to focus by 56.3% in the best case and by 32.7% on average—as well as improved accuracy.

1 Introduction

Modern digital cameras are equipped with one or more passive autofocus mechanisms. In passive autofocus mechanisms, a lens is focused using only the camera’s optical system and an algorithm for controlling the lens. Passive autofocus mechanisms come in two basic kinds: contrast-detection and phase-detection. Contrast-detection autofocus is the most common—being standard in a wide range of cameras from mobile phones cameras, to point-and-shoots, to high-end DSLRs—whereas currently only highend DSLRs also come equipped with phase-detection autofocus. Phase-detection is faster and better able to track subject movement, whereas contrast-detection is less costly and can be more accurate [3]. Our concern here is with contrast-detection autofocus.

In this paper, we show that machine learning can be used to semi-automate the construction of heuristics for focusing in digital cameras. Our approach uses supervised learning. In supervised learning, one learns from training examples

that are labeled with the correct answers. More precisely, each training example consists of a vector of feature values and the correct classification or correct answer for that example. In our approach, training and test data were produced using an offline simulation on a suite of 25 benchmarks, and correctly labeled in a semi-automated manner. Once the data was gathered, a decision tree learning algorithm [15] was used to induce a heuristic from the data. In a decision tree the internal nodes of the tree are labeled with features, the edges to the children of a node are labeled with the possible values of the feature, and the leaves of the tree are labeled with a classification. To classify a new example, one starts at the root and repeatedly tests the feature at a node and follows the appropriate branch until a leaf is reached. The label of the leaf is the predicted classification of the new example.

Once learned, the resulting decision tree heuristic was incorporated into a sweep focusing algorithm and compared against the best previously proposed, hand-crafted algorithm on a benchmark suite of images. On these benchmark suites, the automatically constructed decision tree heuristic had improved speed—reducing the number of steps needed to focus by 56.3% in the best case and by 32.7% on average—as well as improved accuracy.

2 Background

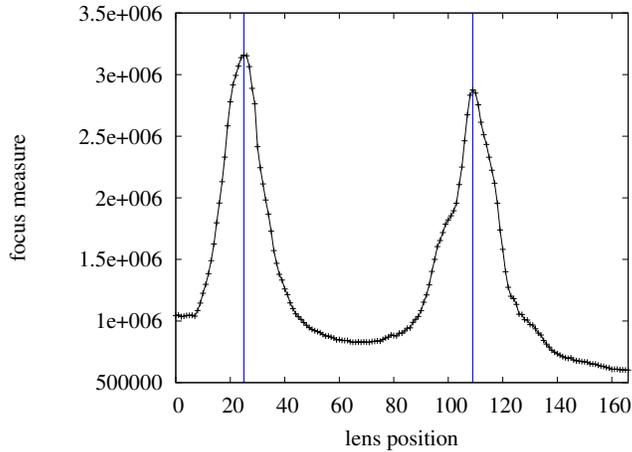
In this section, we review the necessary background in contrast-detection autofocus, focus measures, and focus search algorithms.

2.1 Focus measures

Contrast-detection autofocus makes use of a focus measure that maps an image to a value that represents the degree of focus of the image. Many focus measures have been proposed and evaluated in the literature (see, e.g., [4, 18]). In our work, we make use of an effective focus measure called the squared gradient [17]. Let $f(x, y)$ be the luminance or grayscale at pixel (x, y) in an image of size $M \times N$ pixels. The value $\phi(p)$ of the squared gradient focus measure for an image acquired when the lens is at position p is then given by,

$$\phi(p) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-2} (f(x, y+1) - f(x, y))^2.$$

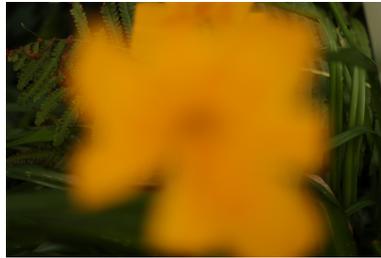
In our work we assume that the region of interest (ROI) is the entire image. In practice, a user can either (i) specify the ROI by moving a rectangle over the desired part of the image when the camera is in live preview mode, or (ii) have the camera automatically determine the object or region of interest to bring into focus (e.g., using face or object recognition [12, 16]). Our proposals are easily adapted to the case where the ROI is an arbitrary sub-area of an image.



(a)



(b)



(c)

Figure 1: (a) Focus measures of images at each of the 167 lens positions (Canon 50 mm lens) for an example scene using the squared gradient focus measure. The two (blue) vertical bars refer to the two images that have objects that are in maximal focus: (b) flower in focus, and (c) fern and grasses in focus.

2.2 Search algorithms

A contrast-detection autofocus algorithm iteratively moves the lens searching for an image with the maximal focus according to the focus measure. Lenses are moved by step motors that can be positioned at discrete positions. The images are acquired from the same stream of images that is displayed in the camera’s live preview mode. (In live preview mode, the camera displays images streamed from the sensor at video frame rates; e.g., 24 frames per second on many Canon cameras.) Figure 1 shows the focus measures of the images acquired at all of the possible lens positions (Canon 50 mm lens) for an example scene.

The search problem that we address in this paper is to find the lens position that corresponds to the image with the *maximum* or *highest peak* of the focus measure $\phi(p)$ over some set of lens positions $p \in \{a, a + 1, \dots, b\}$. At each

iteration, step motors can be moved a single step or larger steps. For example, in the Canon DSLR cameras, the largest step size corresponds to eight single steps. Each step, small or large, is followed by a latency that can be hundreds of milliseconds. As well, step motors can suffer from backlash where the lens (or the software controlling the lens) loses track of the lens position when changing the direction of the lens movement [10, 14]. Thus, two desirable features of a search algorithm are that it (i) takes as large of steps as possible and (ii) minimizes changes in direction. The goal is to focus as quickly as possible without sacrificing accuracy.

Determining the maximum of a function over an interval that can be evaluated at discrete points can be solved using two generic search algorithms that are often used as points of comparison: Global search and Fibonacci search (we review autofocus-specific search algorithms in the next section). The *Global* search algorithm simply steps through all possible lens positions. Although impractically slow, the algorithm is guaranteed to find the maximum value of the focus measure. The *Fibonacci* search algorithm [11], in a manner similar to binary search, narrows at each iteration the interval in which the peak can lie. Fibonacci search is guaranteed to find the maximum in the *fewest* number of steps if the function is unimodal; i.e., has a single peak over the interval. Unfortunately, the assumption of unimodality usually does not hold in our setting and even when it does hold, the back-and-forth movement of the algorithm is undesirable because of backlash.

3 Related Work

The problem of designing search algorithms for passive contrast-based autofocus has been quite well-studied in the literature. The algorithms are designed to either find the *first* peak or the *highest* peak. Of course, an algorithm that performs a full sweep to find the highest peak can easily be modified to stop early. Most of the algorithms designed to find the *first* peak appear to depend on the unimodal assumption and it is unclear how they would perform in cases where this assumption does not hold.

He, Zhou, and Hong [9] propose a coarse-to-fine search, where initially the search algorithm takes coarse or large steps until a *first* peak is found and then reverses direction and takes fine steps to determine the peak. Although the algorithm depends on the unimodal assumption, the idea of a coarse-to-fine search has been influential.

Kehtarnavaz and Oh [10] develop a rule-based autofocus algorithm to find the *highest* peak over an interval by performing a full sweep (see Algorithm 1). The hand-crafted rules predict whether to move the lens a coarse, medium, or fine step at each iteration as it sweeps the lens from near focus to far focus (Lines 1–16 in Algorithm 1). The goal of the heuristic is to move the lens larger steps but without missing any peaks in the focus measure. In our approach we use machine learning to devise a heuristic to determine the step size to move the lens. To the best of our knowledge, the rule-based algorithm remains the state-

Algorithm 1: Kehtarnavaz and Oh [10] rule-based search algorithm for finding a maximum of a focus measure $\phi(p)$ over a set of focus positions $p \in \{a, a + 1, \dots, b\}$.

input : Function $\phi(p)$ and interval $[a, b]$
output: Return maximum of $\phi(p)$ over $p \in \{a, a + 1, \dots, b\}$
 $k \leftarrow 0$; $down \leftarrow 0$;
 $F_{Current} \leftarrow 0$; $F_{Max} \leftarrow 0$;
 $p \leftarrow a$;
while $p \leq b$ **do**

	$F_{Previous} \leftarrow F_{Current}$;
	$F_{Current} \leftarrow \phi(p)$;
1	if $k \leq 5$ then
2	$stepSize \leftarrow Initial$;
3	else
4	if $F_{Current} \leq 0.25 \cdot F_{Max}$ then
5	$stepSize \leftarrow Coarse$; $down \leftarrow 0$;
6	else
7	$D_F \leftarrow F_{Current} - F_{Previous}$;
8	if $D_F > 0.25 \cdot F_{Previous}$ then
9	$stepSize \leftarrow Fine$; $down \leftarrow 0$;
10	else if $stepSize = Fine$ and $D_F > 0$ then
11	$down \leftarrow 0$;
12	else if $D_F < 0$ then
13	if $stepSize = Fine$ then $down \leftarrow down + 1$;
14	if $down = 3$ then $stepSize \leftarrow Mid$; $down \leftarrow 0$;
15	else
16	$stepSize \leftarrow Mid$; $down \leftarrow 0$;

if $F_{Current} > F_{Max}$ **then** $F_{Max} \leftarrow F_{Current}$;
 $k \leftarrow k + 1$;
 $p \leftarrow p + stepSize$;

of-the-art and we perform a detailed experimental comparison of the rule-based algorithm to our machine-learning-based algorithm in Section 5.

Li [13] proposes an algorithm that trades off accuracy for speed by only using a medium-sized search step to find the *first* peak. The algorithm is suited for mobile phone and compact cameras with a fast aperture (low f -number), where there is a large depth-of-field and the lens is relatively easy to focus as there are several indistinguishable (to the human eye) focus positions. In our work, we focus on speed without giving up accuracy.

Chen, Hong, and Chuang [2] propose an algorithm that iteratively samples the focus measures at various lens positions, fits an equation to predict the location of the *first* peak, takes coarse steps to be near the predicted peak, and

finally takes fine steps within a bisection search algorithm to find the peak. Only a limited evaluation of the proposed algorithm is performed: three test images, all close to the camera, and there is no comparison to previously proposed algorithms.

Chen, Hwang, and Chen [1] propose an algorithm that uses a self-organizing neural network to predict the location of the *first* peak based on sampling the focus measure at three places. Only a limited evaluation of the proposed algorithm is performed: ten test prints, all close to the camera, and the only comparison is to the Global search algorithm. However, the approach is one of the first to be based on supervised machine learning techniques.

Han et al. [7] also use a supervised machine learning technique, a variation of 1-nearest neighbor, to predict the location of the *first* peak based on sampling the focus measure at three places. Their method is shown to be faster, but slightly less accurate, than Kehtarnavaz and Oh’s [10] rule-based algorithm on a testbed where all objects were close to the camera (within 0.10m to 1.20m). Not addressed in this work is how to generalize to arbitrary subject distances.

4 Learning to Focus

In this section, we describe the methodology we followed to automatically construct a search heuristic for autofocusing by applying techniques from supervised machine learning. We explain the construction of the initial set of features (Section 4.1), the collection of the data (Section 4.2), the use of the data to filter and rank the features to find the most important features (Section 4.3), and the use of the data and the important features to learn a simple heuristic for use within an iterative focusing algorithm (Section 4.4).

4.1 Feature construction

A critical factor in the success of a supervised learning approach is whether the features recorded in each example are adequate to distinguish all of the different cases. We began with sixty features that we felt would be promising. The features are all functions of the current value of the focus measure and previous values of the focus measure. The sixty features we began with were a mixture of generalizations of previously proposed features and novel features. For previously proposed features, we generalized some of the features used in Kehtarnavaz and Oh’s [10] hand-crafted heuristic.

We also created many novel features for autofocusing. A more accurate classifier can often be achieved by synthesizing new features from existing basic features. We constructed novel features by applying simple functions or combinations of simple functions to basic features. Examples of simple functions include comparison of two features, the ratio of two features, and the log of the difference of two features. Table 1 shows the full set of features that we considered. All of the features are Boolean valued except for the feature *downhillCount*, which is four-valued.

Table 1: The full set of sixty features considered in the machine learning approach. $F_{Current}$, $F_{Previous}$, and $F_{Previous2}$ are the values of the focus measure of the image at the current lens position, and at earlier lens positions, respectively; *down* is the number of consecutive decreasing steps including the most recent step; and *up* is the number of consecutive increasing steps.

$$\begin{aligned}
ratio(x, y) &= \left(\frac{F_{Current}}{F_{Previous}} > \frac{x}{y} \right), \quad x = 1, \dots, 15, \quad y = 8 \\
ratioI(x, y) &= \left(\frac{F_{Previous}}{F_{Current}} > \frac{x}{y} \right), \quad x = 2, \dots, 12, \quad y = 8 \\
downSlope(x, y) &= \left(\frac{F_{Previous2}/F_{Previous}}{F_{Previous}/F_{Current}} > \frac{x}{y} \right), \quad x = 2, \dots, 12, \quad y = 4 \\
upSlope(x, y) &= \left(\frac{F_{Current}/F_{Previous}}{F_{Previous}/F_{Previous2}} > \frac{x}{y} \right), \quad x = 2, \dots, 10, \quad y = 4 \\
downTrend &= (F_{Current} \leq F_{Previous} \text{ and } F_{Previous} \leq F_{Previous2}) \\
upTrend &= (F_{Current} \geq F_{Previous} \text{ and } F_{Previous} \geq F_{Previous2}) \\
downOrFlatTrend &= (F_{Current} \leq (1.005 * F_{Previous}) \text{ and } \\
&\quad F_{Previous} \leq (1.005 * F_{Previous2})) \\
upOrFlatTrend &= (F_{Current} \geq (1.005 * F_{Previous}) \text{ and } \\
&\quad F_{Previous} \geq (1.005 * F_{Previous2})) \\
downhillCount &= \begin{cases} a & \text{if } down \leq 1 \\ b & \text{if } down = 2 \\ c & \text{if } down = 3 \\ d & \text{if } down \geq 4 \end{cases} \\
uphillCount &= (up \geq 1) \\
logDiff(x, y) &= \left(\frac{\log(F_{Current} - F_{Previous})}{\log(F_{Previous})} > \frac{x}{y} \right) \quad x = 1, \dots, 8, \quad y = 8
\end{aligned}$$

4.2 Data collection

In addition to the choice of distinguishing features (see Section 4.1 above), a second critical factor in the success of a supervised learning approach is whether the data is representative of what will be seen in practice. The experimental methodology for gathering representative data consisted of three stages. All implementations were in C++ running under Windows 7¹.

In the first stage, we implemented a camera remote control application whereby a camera is tethered to a computer via a USB cable and controlled by

¹The software and data are available at: <https://cs.uwaterloo.ca/~vanbeek/research>.

the software running on the computer. Our remote control application makes use of the Canon SDK (Version 2.11) and can control and replicate the basic functionality of the camera such as setting the aperture, displaying the live preview stream, and controlling the focus position of the lens. Using the remote control software, we gathered 25 sets of benchmark images that covered a range of common photography settings including landscapes, closeups, interiors, still lifes, and so on. Each of the sets of benchmark images contains either 167 (when using a 50 mm lens) or 231 (when using a 200 mm lens) jpeg images, one for each focus position of the lens. The jpeg images are captured from the live preview stream once the lens is moved from one position to the next. The camera used in our experiments was a Canon EOS 550D/Rebel T2i.

In the second stage, we obtained the focus measurements and the correct labels for each focus position within each benchmark. In supervised learning, each instance in the data is a vector of feature values and the correct classification or label for that instance. In our approach, the class label is either “Fine”, a single step of the lens, or “Coarse”, a large step of the lens corresponding to eight single steps. Given a benchmark set of images, the squared gradient focus measure was applied to each jpeg image in the benchmark (see Figure 2 for an example). Each focus measurement was then labeled with either Coarse or Fine, depending on what was judged to be the best stepsize at that point if one were performing a sweep from near focus to far focus. For example, for the example focus measurement graph shown in Figure 2, if the current lens position is anywhere from position 30–86, the best stepsize to take would be a Coarse step (in order to step quickly through the lens positions), and from positions 87–113 the best stepsize would be a Fine step (in order not to miss the peak). The focus measurements were labeled in a semi-automated manner where an automatically generated labeling was then refined by hand to remove any anomalies due to noise.

In the third and final stage, we generated the actual machine learning data using a forward simulation of a sweep autofocus algorithm. The input to the simulation was a set of labeled focus measurements for a benchmark. The algorithm starts at near focus and at each lens position until it reaches far focus it does the following: (i) outputs the values of the full set of sixty features (see Table 1) using the current values of $F_{Current}$, $F_{Previous}$, $F_{Previous2}$, $down$, and up ; (ii) outputs the class label associated with that lens position; and (iii) steps forward the distance specified by the class label.

Several key techniques allowed us to greatly improve the quality of our data and the resulting efficiency and accuracy of our approach.

1. We separated the problem of learning a single heuristic for predicting the next stepsize into the problem of learning two heuristics: a heuristic for when the autofocus algorithm should transition from taking Fine steps to taking Coarse steps (i.e., predicting the next stepsize when the last step taken was a Fine step), and a heuristic for when to transition from taking Coarse steps to taking Fine steps (i.e., predicting the next stepsize when the last step taken was a Coarse step).

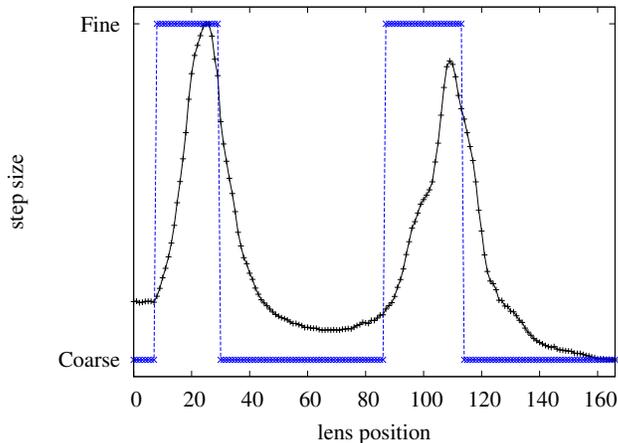


Figure 2: Focus measures of images at each of the 167 lens positions for an example scene labeled with the best next step to take (a Fine step, or a Coarse step) when sweeping the lens from near to far (left to right in graph).

2. We added noise to the forward simulation. With some small probability p (in our experiments we used $p = 0.10$), the simulation will make a mistake either by taking a Coarse step, when the best stepsize would have been a Fine step, or vice-versa. The simulation is then repeated 10 times for each benchmark. We found that without the added noise, the heuristics that were learned were brittle: once a mistake was made, they did not recover. With the added noise, the heuristics robustly recover from mistakes.
3. We balanced the data sets. A data set is balanced if the frequency of the classes is approximately equal. Imbalanced data sets can lead to poor predictive accuracy. In our case, the initial data was quite imbalanced. Consider the case where the forward simulation of the autofocus algorithm is taking Coarse steps and we are learning when to transition from taking Coarse steps to taking Fine steps. Most of the time the best next step is to continue to take Coarse steps, and only quite rarely to transition to Fine steps. To balance the data sets, we used the simple but effective technique of duplicating the instances from the minority class [19].

We obtained a total of 22,794 and 36,580 instances for learning a heuristic when the last step taken was a Fine step and a Coarse step, respectively.

4.3 Feature selection

Once the data was collected but prior to actually learning the heuristics, the next step that we performed was feature selection. The goal of feature selection

is to select only the most important features for constructing good heuristics. The selected features are then retained in the data and subsequently passed to the learning algorithm and the features identified as irrelevant or redundant are deleted. There are two significant motivations for performing this preprocessing step: the efficiency of the learning process can be improved and the quality of the heuristic that is learned can be improved (many learning methods, decision tree learning included, do poorly in the presence of redundant or irrelevant features [20, pp. 231-232]).

Table 2: The features selected for learning a heuristic when the last step taken was a Fine step and when a Coarse step.

last step	features			
Fine	<i>ratio</i> (8, 8)			
Coarse	<i>ratio</i> (10, 8)	<i>ratio</i> (11, 8)	<i>ratioI</i> (9, 8)	
	<i>downSlope</i> (9, 8)	<i>upSlope</i> (8, 4)	<i>upTrend</i>	<i>logDiff</i> (6, 8)

Many feature selection methods have been developed (see, for example, [5] and the references therein). To perform feature selection, we used the Weka (Version 3.6.9) open source machine learning software [6]. In particular, we used Weka’s best first search with the default parameters, a greedy hillclimbing method augmented with limited backtracking that searches forward starting from the empty set of features and adds features as long as the feature evaluator indicates improvement. The feature evaluator we used was the classifier subset evaluator, which evaluates a subset of features by constructing a decision tree classifier using the subset of features, using 10-fold cross-validation (see [8, pp. 241-249]) to estimate the accuracy of the decision tree classifier, and finally using the accuracy estimate as a figure of merit for the subset of features. For the construction of the decision tree classifier, we used the default settings with the exception that we set the minimum number of instances per leaf to 256 since, as noted in Section 4.2, lots of data was available. Table 2 shows the features that remained after selection. All of these features appeared in all 10 of the cross validation tests, an indication of their robustness.

4.4 Classifier selection

The next step is to actually learn the heuristics using all of the available data, where the data contains only the features that passed the selection step (see Table 2). To learn a classifier, we used Weka’s [6] J48 implementation of Quinlan’s C4.5 decision tree algorithm [15]. We chose decision tree classifiers for learning the heuristics over other possible machine learning techniques because of their excellent fit with our goals of accurate and efficient to evaluate. The software was run with the default parameter settings, with the exception that we again (as in feature selection) set the minimum number of instances per leaf to 256 since lots of data was available and it resulted in much simpler trees.

Algorithm 2: Machine-learning-based search algorithm for finding a maximum of a focus measure $\phi(p)$ over a set of focus positions $p \in \{a, a + 1, \dots, b\}$.

```

input : Function  $\phi(p)$  and interval  $[a, b]$ 
output: Return maximum of  $\phi(p)$  over  $p \in \{a, a + 1, \dots, b\}$ 
 $lastStep \leftarrow Coarse$ ;
 $F_{Current} \leftarrow \phi(a)$ ;  $F_{Max} \leftarrow F_{Current}$ ;
 $F_{Previous2} \leftarrow F_{Current}$ ;  $F_{Previous} \leftarrow F_{Current}$ ;
 $p \leftarrow a + 1$ ;
while  $p \leq b$  do
     $F_{Previous2} \leftarrow F_{Previous}$ ;
     $F_{Previous} \leftarrow F_{Current}$ ;
     $F_{Current} \leftarrow \phi(p)$ ;
    if  $lastStep = Fine$  then
1      if  $ratio(8, 8) = 0$  then  $stepSize \leftarrow Coarse$ ;
2      if  $ratio(8, 8) = 1$  then  $stepSize \leftarrow Fine$ ;
    if  $lastStep = Coarse$  then
3      if  $ratio(10, 8) = 0$  then
4          if  $downSlope(9, 8) = 0$  then  $stepSize \leftarrow Coarse$ ;
5          if  $downSlope(9, 8) = 1$  then
6              if  $ratioI(9, 8) = 0$  then
7                  if  $logDiff(6, 8) = 0$  then
8                      if  $upSlope(8, 4) = 0$  then  $stepSize \leftarrow Coarse$ ;
9                      if  $upSlope(8, 4) = 1$  then  $stepSize \leftarrow Fine$ ;
10                     if  $logDiff(6, 8) = 1$  then
11                         if  $upTrend = 0$  then  $stepSize \leftarrow Fine$ ;
12                         if  $upTrend = 1$  then  $stepSize \leftarrow Coarse$ ;
13                     if  $ratioI(9, 8) = 1$  then  $stepSize \leftarrow Coarse$ ;
14             if  $ratio(10, 8) = 1$  then
15                 if  $downSlope(9, 8) = 0$  then
16                     if  $ratio(11, 8) = 0$  then  $stepSize \leftarrow Coarse$ ;
17                     if  $ratio(11, 8) = 1$  then  $stepSize \leftarrow Fine$ ;
18                 if  $downSlope(9, 8) = 1$  then  $stepSize \leftarrow Fine$ ;
    if  $F_{Current} > F_{Max}$  then  $F_{Max} \leftarrow F_{Current}$ ;
     $lastStep \leftarrow stepSize$ ;
     $p \leftarrow p + stepSize$ ;

```

As noted, the final decision trees were constructed using all of the available data. Algorithm 2 shows the final decision trees in algorithmic form, as would be incorporated into the autofocus routine of a camera. Lines 1–2 correspond to the heuristic learned when the last step taken was a Fine step, and Lines 3–18 correspond to the heuristic learned when the last step was a Coarse step. The autofocus algorithm accepts an interval $[a, a + 1, \dots, b]$ over which to search, where a is the starting lens position, and returns the position of the maximum focus measure over the interval.

5 Experimental Evaluation

In this section, we empirically evaluate the effectiveness of our machine-learning-based (ml-based) heuristics. We compare against the Fibonacci search algorithm and Kehtarnavaz and Oh’s rule-based heuristic [10]. Both Kehtarnavaz and Oh’s rule-based heuristic and our ml-based heuristics were incorporated into a sweep focusing algorithm for finding the highest peak over the full range of lens positions for a lens (see Algorithm 1 and Algorithm 2, respectively). Following Kehtarnavaz and Oh [10], in Algorithm 1 we used the squared gradient focus measure and we used 1, 3, and 10 steps for the *Fine*, *Mid*, and *Coarse* stepsizes, respectively. In our Algorithm 2 we used the squared gradient focus measure and we used 1 and 8 for the *Fine* and *Coarse* stepsizes, respectively, as these correspond to the stepsizes for Canon cameras.

As a test suite for comparing the algorithms, we used the same 25 sets of benchmark images that we used in data collection (see Section 4.2). There are a total of 4,303 images (23 of the benchmarks have 167 images; two of the benchmarks have 231 images). We compare the algorithms on three criteria: (i) the number of lens movements taken by an algorithm to sweep from near focus to far focus, (ii) whether the algorithm found the lens position corresponding to the highest peak in the focus measure, and (iii) whether the algorithm found all of the peaks in the focus measure. An algorithm found a peak if the lens was at that position during the course of the algorithm; an algorithm did not find a peak if the lens was stepped over that position by taking a larger stepsize that began before the peak and ended after the peak.

The Fibonacci and rule-based algorithms were run directly on the benchmark suite (see Table 3). To assess the ml-based heuristics, we used a variation of leave-one-out cross-validation (cross validation methods are widely used for assessing performance in machine learning; see [8, pp. 241-249]). For *each* of the 25 benchmarks in turn, we performed the following: (i) set aside the current benchmark, call it the *test* benchmark; (ii) collect the machine learning data (as in Section 4.2), but using only the remaining 24 benchmarks, omitting the test benchmark; (iii) perform feature selection (as in Section 4.3), but using only the data that omits the test benchmark; (iv) perform classifier selection (as in Section 4.4), but again using only the data and features selected using the machine learning data that omits the test benchmark; and (v) incorporate the learned heuristics into an algorithm and evaluate on the test benchmark.

Table 3: Steps (lens movements) taken and peaks found by Fibonacci, rule-based, and machine-learning-based focusing algorithms using the squared-gradient focus measure. Peaks found is in the form x/y , where x is the number of peaks found and y is the total number of peaks. A red (dark) entry indicates that an algorithm did not find the lens position for which the focus measure was at a maximum. A yellow (light) entry highlights that an algorithm did not find all peaks in the focus measure. Global search takes 167 steps on each benchmark, except for the benchmarks Cat and Moon, where it takes 231 steps. The last column shows the speedup in steps of the proposed machine-learning-based algorithm over the previous rule-based algorithm.

test benchmark	Fibonacci		rule-based		ml-based		
	steps	peak	steps	peak	steps	peak	speedup
Backyard	12	1/1	75	1/1	54	1/1	28.0%
Bench	12	1/1	70	1/1	38	1/1	45.7%
Book	12	0/1	41	1/1	35	1/1	14.6%
Bridge	12	1/1	75	1/1	46	1/1	38.7%
Building1	12	1/1	67	1/1	46	1/1	31.3%
Building2	12	1/1	66	1/1	52	1/1	21.2%
Building3	12	1/1	67	1/1	30	1/1	55.2%
Cat	13	1/2	71	2/2	43	2/2	39.4%
Cup1	12	1/1	62	1/1	36	1/1	41.9%
Cup2	12	1/2	59	1/2	32	1/2	45.8%
Cup3	12	1/2	64	1/2	34	1/2	46.9%
Cup4	12	1/4	67	3/3	36	3/3	46.3%
Fabric	12	1/1	57	1/1	34	1/1	40.4%
Flower	12	1/2	66	2/2	41	2/2	37.9%
Interior1	12	1/1	74	1/1	62	1/1	16.2%
Interior2	12	1/4	77	3/4	46	3/4	40.3%
Lamp	12	1/4	72	3/4	84	4/4	-16.7%
Landscape1	12	1/1	78	1/1	52	1/1	33.3%
Landscape2	12	1/1	75	1/1	46	1/1	38.7%
Landscape3	12	1/1	71	1/1	41	1/1	42.3%
Moon	13	0/1	87	1/1	38	1/1	56.3%
Screen	12	1/2	43	1/2	62	2/2	-44.2%
Snails	12	1/1	57	1/1	37	1/1	35.1%
StillLife	12	0/1	64	1/1	37	1/1	42.2%
Vase	12	1/1	52	1/1	31	1/1	40.4%

Table 3 summarizes the results of the empirical evaluation. On these benchmarks, the Fibonacci algorithm [11] was fast but inaccurate, often missing the highest peak and sometimes not finding any peak (due to small amounts of noise in the focus measurements). The rule-based algorithm [10] always found the highest peak and missed a lesser peak on five of the benchmarks. Our ml-based algorithm also always found the highest peak and missed a lesser peak on only three of the benchmarks. Thus, our ml-based heuristic is somewhat more accurate than the rule-based heuristic. However, the main advantage of our ml-based heuristic is in speed: it is faster on 23 of the benchmarks and in the two cases where it is slower, it is more accurate. The ml-based heuristic is 32.7% faster on average over all 25 benchmarks and 38.2% faster on average if one excludes the two benchmarks where it is more accurate.

6 Conclusion

The speed and accuracy of a digital camera’s contrast-based autofocus algorithm are crucial to user satisfaction. Previous work has proposed a hand-crafted rule-based autofocus heuristic to find the peak focus [10]. We showed that supervised machine learning techniques can be used to construct heuristics that out-perform the state-of-the-art hand-crafted heuristic. We gathered an extensive set of benchmark images that covered a range of common photography settings. Offline simulation was then used to construct the machine learning data and decision tree heuristics were induced from the data. In our experiments, the machine-learning-based algorithm was significantly faster—reducing the number of steps needed to focus by 56.3% in the best case and by 32.7% on average—as well as more accurate.

References

- [1] C.-Y. Chen, R.-C. Hwang, and Y.-J. Chen. A passive auto-focus camera control system. *Applied Soft Computing*, 10:296–303, 2010.
- [2] C.M. Chen, C.M. Hong, and H.C. Chuang. Efficient auto-focus algorithm utilizing discrete difference equation prediction model for digital still cameras. *IEEE Trans. Consum. Electron.*, 52:1135–1143, 2006.
- [3] R. Cicala. Autofocus reality part 1: Center-point, single-shot accuracy, 2012. Retrieved October 29, 2013, from <http://www.lensrentals.com/blog/2012/07/autofocus-reality-part-1-center-point-single-shot-accuracy>.
- [4] F.C.A. Groen, I.T. Young, and G. Ligthart. A comparison of different focus functions for use in autofocus algorithms. *Cytometry*, 6:81–91, 1985.
- [5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. of Machine Learning Research*, 3:1157–1182, 2003.

- [6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.
- [7] J.-W. Han, J.-H. Kim, H.-T. Lee, and S.-J. Ko. A novel training based auto-focus for mobile-phone cameras. *IEEE Trans. Consum. Electron.*, 57:232–238, 2011.
- [8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction (2nd ed.)*. Springer, 2009.
- [9] J. He, R. Zhou, and Z. Hong. Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera. *IEEE Trans. Consum. Electron.*, 49:257–262, 2003.
- [10] N. Kehtarnavaz and H.-J. Oh. Development and real-time implementation of a rule-based auto-focus algorithm. *Real-Time Imaging*, 9:197–203, 2003.
- [11] J. Kiefer. Sequential minimax search for a maximum. *Proceedings American Mathematical Society*, 1953. See also Donald E. Knuth, *The Art of Computer Programming (2nd ed.)*, vol. 3, p. 418.
- [12] S. Y. Lee, Y. Kumar, J. M. Cho, S. W. Lee, and S. W. Kim. Enhanced autofocus algorithm using robust focus measure and fuzzy reasoning. *IEEE Trans. Circuits Syst. Video Technol.*, 18:1237–1246, 2008.
- [13] J. Li. Autofocus searching algorithm considering human visual system limitations. *Optical Engineering*, 44:113201–113201–4, 2005.
- [14] D. Morgan-Mar and M. R. Arnison. Focus finding using scale invariant patterns. In *Proc. SPIE 8660, Digital Photography IX*, 2013.
- [15] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [16] M. Rahman and N. Kehtarnavaz. Real-time face-priority auto focus for digital and cell-phone cameras. *IEEE Trans. Consum. Electron.*, 54:1506–1513, 2008.
- [17] A. Santos, C. Ortiz de Solórzano, J. J. Vaquero, J. M. Peña, N. Malpica, and F. del Pozo. Evaluation of autofocus functions in molecular cytogenetic analysis. *Journal of Microscopy*, 188:264–272, 1997.
- [18] M. Subbarao and J.-K. Tyan. Selecting the optimal focus measure for auto-focusing and depth-from-focus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:864–870, 1998.
- [19] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning*, pages 935–942, 2007.
- [20] I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann, 2000.