

An autofocus heuristic for digital cameras based on supervised machine learning

Hashim Mir · Peter Xu · Rudi Chen ·
Peter van Beek

Received: date / Accepted: date

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 address the problems of identifying the global optimum and significant local optima (or peaks) when focusing an image. We show that supervised machine learning techniques can be used to construct a passive autofocus heuristic for these problems that out-performs an existing hand-crafted heuristic and other baseline methods. 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 heuristic from the data. The automatically constructed machine-learning-based (ml-based) heuristic was compared against a previously proposed hand-crafted heuristic for autofocusing and other baseline methods. In our experiments, the ml-based heuristic had improved speed—reducing the number of iterations needed to focus by 37.9% on average in common photography settings and 22.9% on average in a more difficult focus stacking setting—while maintaining accuracy.

Keywords Autofocus · Live preview · Digital camera · Machine learning

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

Hashim Mir, Peter Xu, Rudi Chen, and Peter van Beek
Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
Corresponding author: Peter van Beek
Tel: 519-888-4567, x35344
Fax: 519-885-1208
E-mail: vanbeek@uwaterloo.ca

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 high-end DSLRs also come equipped with phase-detection autofocus. Phase-detection is faster and better able to track subject movement, whereas contrast-detection is less expensive in monetary terms and can be more accurate (Cicala, 2012). Our concern here is with contrast-detection autofocus.

In this paper, we address the problems of identifying the global optimum and significant local optima (or peaks) when focusing an image. The problems are important in focus stacking (see, for example, (Vaquero et al, 2011)), where a set of images is acquired and then merged in post processing in order to achieve a final image that is all-in-focus. We show that machine learning can be used to semi-automate the construction of heuristics for these problems. 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 (Quinlan, 1993) 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.

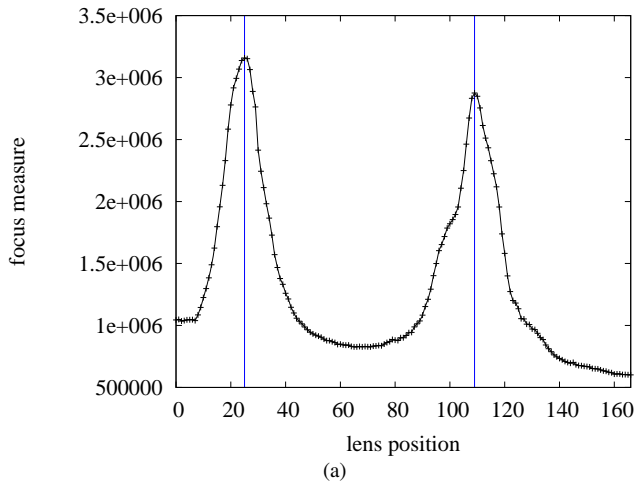
Once learned, the decision tree heuristic was compared against a previously proposed hand-crafted heuristic and other baseline methods by incorporating the heuristics into a sweep focusing algorithm and applying the algorithm on benchmark suites of images. On these benchmark suites, the automatically constructed decision tree heuristic had improved speed—reducing the number of iterations needed to focus by 37.9% on average in common photography settings and 22.9% on average in a more difficult focus stacking setting—while maintaining 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., (Groen et al, 1985; Subbarao



(b)



(c)

Fig. 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 identify peaks in the focus measure and correspond to the two images: (b) flower in focus, and (c) fern and grasses in focus.

and Tyan, 1998)). In our work, we make use of an effective focus measure called the squared gradient (Mir et al, 2014; Santos et al, 1997). 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,

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

Kehtarnavaz and Oh (2003) used the squared gradient in developing a hand-crafted heuristic for focusing and to have the fairest comparison by holding the focus measure constant we chose the same focus measure. However, other effective focus measures, such as those based on the second derivative, could also have been chosen.

Following Kehtarnavaz and Oh (2003), 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 (Lee et al, 2008; Rahman and Kehtarnavaz, 2008)). Our proposals are easily adapted to the case where the ROI is an arbitrary sub-area of an image.

2.2 Search algorithms

A contrast-detection autofocus algorithm iteratively moves the lens searching for a lens position that brings the image into best 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.

Given a set of lens positions $\{a, a + 1, \dots, b\}$ from near focus to far focus, three autofocus search problems can be defined. The first search problem is to find the lens position that corresponds to the image with the *nearest* peak in the focus measure, where nearest can be either defined relative to the camera (i.e., nearest to a) or to the current position of the lens. The second search problem is to find the lens position that corresponds to the image with the *maximum* or *highest* peak in the focus measure. A third search problem, related to finding the highest peak, is to find *all* lens positions that correspond to a peak in the focus measure. In general, autofocus based on nearest peak may be preferred over that based on highest peak, as finding the nearest peak can be less costly than finding the highest peak and foreground elements are the subject in many common photography settings. However, a deficiency of autofocus based on nearest peak is that it can settle on bringing inconsequential objects into focus, whereas that based on highest peak can be more likely to bring the dominant object into focus or to bring most of the region of interest into focus. As well, finding the highest peak can be useful in situations where the focus measure is noisy and thus has false peaks (for example, in low light situations). Finally, the problem of finding all peaks is important in focus stacking (Vaquero et al, 2011), where a set of images is acquired and then merged in post processing in order to achieve a final image that is all-in-focus.

In this paper, we present a heuristic algorithm that identifies all lens positions $p \in \{a, a + 1, \dots, b\}$ such that the focus measure $\phi(p)$ is a peak (a significant local optima) and also finds the highest peak (the global optimum). 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 tens 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 (Kehtarnavaz and Oh, 2003; Morgan-Mar and Arnison, 2013). 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 all peaks and to find the maximum value of the focus measure. The *Fibonacci* search algorithm (Kiefer, 1953), in a manner similar to binary search, narrows at each iteration the interval in which the peak can lie. Fibonacci search is a special case of Golden Section search (Kiefer, 1953), where the function is restricted to being evaluated over only integer positions. Fibonacci search is guaranteed to find the maximum in the *fewest* number of iterations if the function is unimodal; i.e., has a single peak over the interval. Unfortunately, the assumption of unimodality often does not always hold in our setting and even when it does hold, the back-and-forth movement of the algorithm is undesirable because of backlash and power consumption.

3 Related Work

The problem of designing search algorithms for passive contrast-based autofocusing has been quite well-studied in the literature. However, the majority of the work has been on finding the nearest peak and there has been little work on finding the highest peak or on identifying all peaks.

3.1 Finding nearest peak

He, Zhou, and Hong (2003) propose a coarse-to-fine search, where initially the search algorithm takes coarse or large steps until a *nearest* peak is found (the value of the focus measure is decreasing) and then reverses direction and takes fine steps to narrow down the location of the peak. The idea of a coarse-to-fine search has been influential in subsequent work, including the present work. We perform an experimental comparison against He et al.'s search algorithm in Section 5.

Li (2005) proposes an algorithm that trades off accuracy for speed by only using a medium-sized search step to find the *nearest* 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.

Chen, Hong, and Chuang (2006) propose an algorithm that iteratively samples the focus measures at various lens positions, fits an equation to predict the location of the *nearest* peak, takes coarse steps to be near the predicted peak, and finally takes fine steps within a bisection search algorithm to find the peak.

Supervised machine learning approaches to finding the *nearest* peak have also been proposed. Chen, Hwang, and Chen (2010) propose an algorithm that uses a self-organizing neural network to predict the location of the nearest peak based on sampling the focus measure at three places. Their approach is one of the first to be

based on supervised machine learning techniques. Han, Kim, Lee, and Ko (2011) also use a supervised machine learning technique, a variation of 1-nearest neighbor, to predict the location of the nearest peak based on sampling the focus measure at three places. While these approaches are specific to the problem of finding the nearest peak, we adapt some of their features in our supervised learning approach to the problems of finding the highest peak and all peaks.

Zografos, Lenz, and Felsberg (2013) present a framework for low-level image processing based on processing images with simple difference-based filters and fitting a 2-parameter Weibull distribution to the filtered output. Important tasks, including auto focusing, can then be solved by formulating an appropriate cost function and solving a constrained optimization problem over the Weibull manifold.

3.2 Finding highest peak and all peaks

Kehtarnavaz and Oh (2003) develop a rule-based autofocus algorithm to find the *highest* peak and *all* peaks 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. We perform a detailed experimental comparison of the rule-based algorithm to our machine-learning-based algorithm in Section 5. As well, as noted in Section 2.2, the Global search algorithm can be used to find all peaks and the Fibonacci search algorithm can be used to find the highest peak—provided the function is unimodal over the lens positions—and we also compare against these baseline methods in Section 5.

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 were felt to 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 were a mixture of generalizations of previously proposed

Algorithm 1: Kehtarnavaz and Oh (2003) 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
   $F_{Previous} \leftarrow F_{Current}$ ;
  A  $F_{Current} \leftarrow \phi(p)$ ;
  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  $F_{Max} \leftarrow F_{Current}$ ;
   $k \leftarrow k + 1$ ;
  B  $p \leftarrow p + stepSize$ ; move the lens  $stepSize$  forward;

```

features and novel features. For previously proposed features, we generalized some of the features used in Kehtarnavaz and Oh's (2003) hand-crafted heuristic for choosing the next step size and He, Zhou, and Hong's (2003) ratio feature for deciding when to reverse the search direction.

We also created many novel features for autofocusing. A more accurate classifier can often be achieved by synthesizing new features from existing basic features, where here the basic features are the raw values of the focus measures. 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.

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

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.

$$\begin{aligned}
ratio(x,y) &= \left(\frac{F_{Current}}{F_{Previous}} > \frac{x}{y} \right), \quad x = 1, \dots, 15, \quad y = 8 \\
ratioI(x,y) &= \left(\frac{F_{Previous}}{F_{Current}} > \frac{x}{y} \right), \quad x = 2, \dots, 12, \quad y = 8 \\
downSlope(x,y) &= \left(\frac{F_{Previous2}/F_{Previous}}{F_{Previous}/F_{Current}} > \frac{x}{y} \right), \quad x = 2, \dots, 12, \quad y = 4 \\
upSlope(x,y) &= \left(\frac{F_{Current}/F_{Previous}}{F_{Previous}/F_{Previous2}} > \frac{x}{y} \right), \quad x = 2, \dots, 10, \quad y = 4 \\
downTrend &= (F_{Current} \leq F_{Previous} \text{ and } F_{Previous} \leq F_{Previous2}) \\
upTrend &= (F_{Current} \geq F_{Previous} \text{ and } F_{Previous} \geq F_{Previous2}) \\
downOrFlatTrend &= (F_{Current} \leq (1.005 * F_{Previous}) \text{ and } \\
&\quad F_{Previous} \leq (1.005 * F_{Previous2})) \\
upOrFlatTrend &= (F_{Current} \geq (1.005 * F_{Previous}) \text{ and } \\
&\quad F_{Previous} \geq (1.005 * F_{Previous2})) \\
downhillCount &= \begin{cases} a & \text{if } down \leq 1 \\ b & \text{if } down = 2 \\ c & \text{if } down = 3 \\ d & \text{if } down \geq 4 \end{cases} \\
uphillCount &= (up \geq 1) \\
logDiff(x,y) &= \left(\frac{diff}{\log(F_{Previous})} > \frac{x}{y} \right) \quad x = 1, \dots, 8, \quad y = 8 \\
\text{where } diff &= \begin{cases} 0 & \text{if } F_{Current} \leq F_{Previous} \\ \log(F_{Current} - F_{Previous}) & \text{if } F_{Current} > F_{Previous} \end{cases}
\end{aligned}$$

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 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, and still lifes, as well as various lighting conditions including daylight, shade, cloudy, and lowlight. Each of the sets of benchmark images contains either 167 (when using a 50 mm lens) or 231 (when using a 200 mm lens)

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

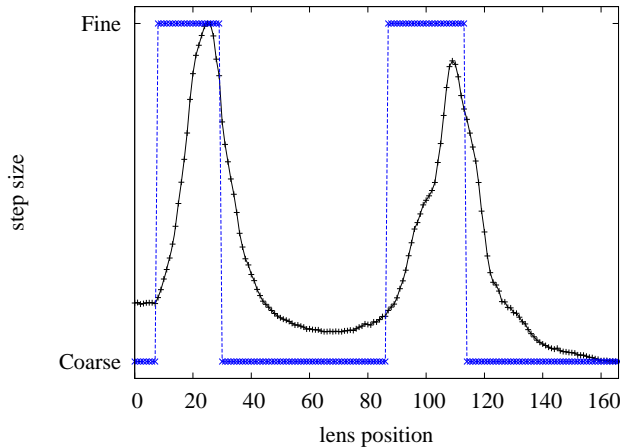


Fig. 2 Focus measures of images at each of the 167 lens positions for an example scene labeled with the best next step to take (a Fine step, or a Coarse step) when sweeping the lens from near to far (left to right in graph).

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. We restrict the class labels to just two step sizes as these were the only two step sizes available on both the lenses used in our experimental setup. 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

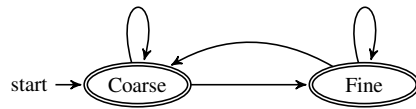


Fig. 3 State diagram illustrating the control algorithm. In the initial state, the lens is in the near focus position and the autofocus algorithm takes a Coarse step. The autofocus algorithm continues to take Coarse steps or Fine steps, and transitions between Coarse and Fine Steps until the lens reaches the far focus position.

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). The control algorithm is illustrated in Figure 3. In essence, we have replaced a single heuristic for predicting Fine step vs Coarse step with two heuristics: the first for predicting Coarse step vs not Coarse step and the second for predicting Fine step vs not Fine step. Separating into two heuristics allows each hypothesis (heuristic) to cover the structure of and to be specialists for each problem independently, resulting in more accurate classification overall (see Alpaydin, 2010, pp. 32-34)
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 (Van Hulse et al, 2007).

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 (see Witten and Frank, 2000, pp. 231-232)).

Many feature selection methods have been developed (see, for example, Guyon and Elisseeff (2003) and the references therein). To perform feature selection, we used the Weka (Version 3.6.9) open source machine learning software (Hall et al, 2009). In particular, we used Weka's best first search with the default parameters, a greedy hill-climbing 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 evaluates a set of features by (i) constructing a decision tree classifier using the subset of features, (ii) using 10-fold cross-validation (see Hastie et al, 2009, pp. 241-249) to estimate the accuracy of the decision tree classifier, and (iii) 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 training instances at a leaf to 256 since, as noted in Section 4.2, lots of data was available.

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

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

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. For the heuristic that predicts Fine step vs not Fine step, a single feature based on upward slope is enough of a signal to indicate the impending presence of a peak. For the heuristic that predicts Coarse step vs not Coarse step, a more elaborate set of features are needed to identify the cases where (i) there is a long gentle upward or downward trend in the focus measure and (ii) a peak has just been passed and there is a rapid descent.

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 $p \in \{a, a+1, \dots, b\}$
 $lastStep \leftarrow Coarse$;
A $F_{Current} \leftarrow \phi(a)$;
 $F_{Max} \leftarrow F_{Current}$; $F_{Previous2} \leftarrow F_{Current}$; $F_{Previous} \leftarrow F_{Current}$;
B $p \leftarrow a + Coarse$; move the lens $Coarse$ steps forward;
while $p \leq b$ **do**
 $F_{Previous2} \leftarrow F_{Previous}$;
 $F_{Previous} \leftarrow F_{Current}$;
A $F_{Current} \leftarrow \phi(p)$;
 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** $F_{Max} \leftarrow F_{Current}$;
 $lastStep \leftarrow stepSize$;
 B $p \leftarrow p + stepSize$; move the lens $stepSize$ forward;

4.4 Classifier selection

The next step is to actually learn the heuristics from the data, where the data contains only the features that passed the selection step (see Table 2). To learn a classifier, we used Weka's (Hall et al, 2009) J48 implementation of Quinlan's C4.5 decision tree algorithm (Quinlan, 1993). We chose decision tree classifiers for learning the heuristics over other possible machine learning techniques because they are accurate, easy to understand, 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 at a leaf to 256 since lots of data was available and it resulted in much simpler trees. Algorithm 2 shows the final decision trees in algorithmic form, as would be incorporated into a sweep autofocus routine of a camera. Lines 1–2 corre-

spond 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 He et al.’s (2003) hill-climbing algorithm, the Fibonacci search algorithm, the Global search algorithm, and Kehtarnavaz and Oh’s (2003) rule-based heuristic. Recall that the Global search algorithm can be used to find all peaks and the hill-climbing and Fibonacci search algorithms can be used to find the highest peak—provided the function is unimodal over the lens positions. 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 Algorithms 1 & 2, respectively). Following Kehtarnavaz and Oh (2003), 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.

We compare the algorithms on speed and accuracy using the following performance measures.

- *Iterations.* The number of focusing iterations is important as it is directly related to focusing time (Kehtarnavaz and Oh, 2003). Each iteration of an autofocus algorithm involves: (i) computing the focus measure for an image acquired at the current lens position (Lines A in Algorithms 1 & 2), (ii) determining which direction and how far to move the lens (the heuristics shown as Lines 1–16 in Algorithm 1 and Lines 1–18 in Algorithm 2), and (iii) sending a control signal to the step motor to move the lens (Lines B in Algorithms 1 & 2). Sending a control signal to the step motor causes a starting and ending delay and a waiting time to stabilize and synchronize with the focus control hardware (Kehtarnavaz and Oh, 2003). The computation time for the heuristics (measured in tens of microseconds) is dwarfed in comparison with the total time needed to compute the focus measure of an image and especially by the total time needed to move and stabilize the lens (measured in tens of milliseconds).
- *Steps.* The number of steps that the step motor moves the lens is important as it is directly related to power consumption (Kehtarnavaz and Oh, 2003). As an example, each time the step motor moves the lens a *Coarse* step in Algorithm 2, 8 steps are added to the total number of steps.
- *Peaks found.* Recall that our goal is to identify both the highest peak and all significant peaks in the focus measure across the lens positions. Thus, we compare the algorithms on (i) whether the algorithm found the lens position corresponding to the highest peak in the focus measure, and (ii) whether the algorithm found all

of the significant peaks in the focus measure. The hill-climbing and Fibonacci algorithms return a single lens location, so there is no ambiguity over whether these algorithms found the highest peak or a significant peak. For Kehtarnavaz and Oh’s rule-based and our ml-based heuristics, an algorithm was only considered to have found a peak if the lens arrived at that position during the course of the algorithm while taking a *Fine* step.

We test the algorithms on two benchmark suites.

1. The original benchmark suite that corresponds to common photography settings and was used in learning the heuristics. In this benchmark suite there are a total of 4,303 images (23 of the benchmarks have 167 images; two of the benchmarks have 231 images).
2. A holdout benchmark suite from a challenging focus stacking setting, where the images were chosen to give rise to more difficult multi-modal focus measures. In this benchmark suite there are a total of 1,456 images (6 of the benchmarks have 167 images; two of the benchmarks have 227 images). These benchmarks were not used in learning the heuristics.

As the original benchmark suite was used in learning the heuristics, we use the following adaptation of simple resampling to obtain reliable estimates of the performance of the heuristics on common photography settings when incorporated into Algorithm 2 (see Japkowicz and Shah, 2011, pp. 161-205). 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 again using only the data that omits the test benchmark; (iv) learn the heuristics (as in Section 4.4), but again using only the data and features selected using the machine learning data that omits the test benchmark; (v) incorporate the learned heuristics into the sweep autofocusing algorithm (i.e., replace Lines 1–2 and Lines 3–18 in Algorithm 2 with the newly learned heuristics); and (vi) run Algorithm 2 with the newly learned heuristics on the test benchmark and record the performance measures (see Table 3). It is important to note that each time we perform the above, we are learning the heuristics from one set of benchmarks, fixing the heuristics, incorporating the heuristics into the autofocusing algorithm, and evaluating the autofocusing algorithm on a different benchmark not used in learning the heuristics. Note as well that, following best practice, we perform a nested feature selection using only the data that excludes the test benchmark (see Hastie et al, 2009, pp. 241-249).

As the focus stacking benchmark suite was not used in learning, we run Algorithm 2 directly on each test benchmark with no modifications and record the performance measures.

Table 3 summarizes the results of the empirical evaluation on the benchmark suite for common photography settings. On these benchmarks, the Fibonacci (Kiefer, 1953) and hill-climbing (He et al, 2003) algorithms were both fast but inaccurate, often missing the highest peak and one time not finding any peak (due to small amounts of noise in the focus measurements). The rule-based algorithm (Kehtarnavaz and Oh,

Table 3 Number of focusing iterations (iter.) and number of steps (step) taken by Fibonacci (Kiefer, 1953), hill-climbing (He et al, 2003), rule-based (Kehtarnavaz and Oh, 2003), and machine-learning-based focusing algorithms using the squared-gradient focus measure. Also shown are the peaks found 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 iterations and steps on each benchmark, except for the benchmarks cat and moon, where it takes 231 iterations and steps. The last column shows the speedup in iterations of the proposed machine-learning-based algorithm over the previous rule-based algorithm.

test benchmark	Fibonacci		hill-climb		rule-based			ml-based			speedup
	iter.	step	iter.	step	iter.	step	peak	iter.	step	peak	
backyard	12	176	26	166	75	162	1/1	54	165	1/1	28.0%
bench	12	171	28	168	70	163	1/1	31	163	1/1	55.7%
book	12	226	15	43	41	165	1/1	35	160	1/1	14.6%
bridge	12	166	26	173	75	166	1/1	46	164	1/1	38.7%
building1	12	166	26	173	67	166	1/1	39	164	1/1	41.8%
building2	12	166	27	174	66	165	1/1	24	163	1/1	63.6%
building3	12	166	28	175	67	164	1/1	30	162	1/1	55.2%
cat	13	283	16	79	71	227	2/2	43	224	2/2	39.4%
cup1	12	226	24	122	62	158	1/1	36	161	1/1	41.9%
cup2	12	224	22	120	59	159	1/2	32	164	1/2	45.8%
cup3	12	226	18	116	64	163	1/2	34	166	1/2	46.9%
cup4	12	208	22	106	67	166	3/3	36	161	3/3	46.3%
fabric	12	228	26	124	57	161	1/1	34	159	1/1	40.4%
flower	12	212	13	41	66	159	2/2	41	166	2/2	37.9%
interior1	12	178	32	172	74	166	1/1	55	159	1/1	25.7%
interior2	12	223	23	121	77	162	3/4	46	164	3/4	40.3%
lamp	12	267	20	83	72	165	3/4	42	160	4/4	41.7%
landscape1	12	166	27	174	78	165	1/1	52	163	1/1	33.3%
landscape2	12	166	26	173	75	166	1/1	46	164	1/1	38.7%
landscape3	12	165	24	171	71	166	1/1	41	166	1/1	42.3%
moon	13	288	6	27	87	228	1/1	38	226	1/1	56.3%
screen	12	226	18	46	43	164	1/2	62	166	2/2	-44.2%
snails	12	221	22	113	57	159	1/1	37	162	1/1	35.1%
stillLife	12	184	27	153	64	163	1/1	37	162	1/1	42.2%
vase	12	206	19	96	52	157	1/1	31	163	1/1	40.4%

2003) always found the highest peak and missed a lesser peak on five of the benchmarks (for two of those peaks, the algorithm was only off by 1 and 2 lens positions, respectively). 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 24 of the benchmarks and in the one case where it is slower, it is more accurate. The ml-based heuristic is 37.9% faster on average over all 25 benchmarks and 41.3% faster on average if one excludes the benchmark where it is more accurate.

Table 4 summarizes the results of the empirical evaluation on the holdout benchmark suite from the focus stacking setting. On these benchmarks, the rule-based algorithm (Kehtarnavaz and Oh, 2003) always found the highest peak and missed a total of seven lesser peaks. Our ml-based algorithm missed finding the highest peak

Table 4 Number of focusing iterations (iter.) and number of steps (step) taken by rule-based (Kehtarnavaz and Oh, 2003) and machine-learning-based focusing algorithms using the squared-gradient focus measure on difficult focus stacking benchmarks.

test benchmark	rule-based			ml-based			speedup
	iter.	step	peak	iter.	step	peak	
garden	71	162	2/2	54	165	2/2	23.9%
bars	74	166	2/2	64	161	2/2	13.5%
books+chairs	67	161	2/6	49	160	4/6	26.9%
backpack	77	164	1/2	39	164	1/2	49.4%
cans	69	158	3/3	47	165	3/3	31.9%
coins	65	222	1/2	39	220	1/2	40.0%
gerbers	42	220	2/3	46	220	3/3	-9.5%
trail	70	164	2/2	65	162	2/2	7.1%

on one benchmark and missed a total of three lesser peaks. In each case that a peak was missed, it was missed by enough lens positions to fall outside of “acceptable focus”; i.e., the region or object in the image corresponding to the peak was visibly out-of-focus when the corresponding image was inspected. Once again our ml-based heuristic has the advantage in speed with an average of 22.9% fewer iterations.

The question arises as to how the current proposal compares to autofocusing on commercially available cameras. Fortunately, the question can be partially answered albeit indirectly. Gamadia and Kehtarnavaz (2009) present a real-time implementation of a modification of the rule-based algorithm on a Texas Instruments DM350 processor, where the rule-based algorithm has been modified to stop at the nearest peak (Peddigari et al, 2005). As is the case with the camera’s autofocus algorithm running on the camera’s on-board processor, the DM350 processor then controls the lens directly with no communication delays. In Gamadia and Kehtarnavaz’s (2005) experimental evaluation, it is shown that the real-time implementation of the rule-based algorithm is able to perform nearest peak autofocusing faster than commercial cameras. Speed-ups of from 27.6% to 84.0% are reported for five common point-and-shoot commercial cameras and both wide and zoom modes (see Gamadia and Kehtarnavaz, 2009, Table 2). Thus, given that our algorithm performs between one-fifth and one-third fewer iterations on average than the rule-based algorithm, there is good reason to believe that our proposal will be faster still than these commercial cameras. Note that the time taken to calculate the next step to take (the computation of the heuristics) is negligible, taking on the order of 45 microseconds and 70 microseconds for Algorithm 1 and Algorithm 2, respectively. Thus, even though our heuristic is slightly more time consuming to compute, the time taken to compute the heuristics is dwarfed by the time taken to compute the focus measure and control the lens and it is the reduction in these calls that leads to time savings.

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 (Kehtarnavaz and Oh, 2003). We showed

that supervised machine learning techniques can be used to construct heuristics that out-perform a hand-crafted heuristic and other baseline methods. 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 iterations needed to focus by 37.9% on average in common photography settings and 22.9% on average in a more difficult focus stacking setting—while maintaining accuracy.

Acknowledgements This work was supported in part by a University of Waterloo President’s Scholarship of Distinction, an NSERC USRA award, and an NSERC Discovery Grant.

References

- Alpaydin E (2010) *Introduction to Machine Learning*, 2nd edn. The MIT Press
- Chen C, Hong C, Chuang H (2006) Efficient auto-focus algorithm utilizing discrete difference equation prediction model for digital still cameras. *IEEE Trans Consum Electron* 52:1135–1143
- Chen CY, Hwang RC, Chen YJ (2010) A passive auto-focus camera control system. *Applied Soft Computing* 10:296–303
- Cicala R (2012) Autofocus reality. <http://www.lensrentals.com/blog/2012/07>, Accessed April 5, 2014
- Gamadia M, Kehtarnavaz N (2009) Real-time implementation of single-shot passive auto focus on DM350 digital camera processor. In: *Real-Time Image and Video Processing (SPIE Vol. 7244)*
- Groen F, Young I, Ligthart G (1985) A comparison of different focus functions for use in autofocus algorithms. *Cytometry* 6:81–91
- Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. *J of Machine Learning Research* 3:1157–1182
- Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) *The WEKA data mining software: An update*. *SIGKDD Explorations* 11
- Han JW, Kim JH, Lee HT, Ko SJ (2011) A novel training based auto-focus for mobile-phone cameras. *IEEE Trans Consum Electron* 57:232–238
- Hastie T, Tibshirani R, Friedman J (2009) *The Elements of Statistical Learning: Data mining, Inference and Prediction*, 2nd edn. Springer
- He J, Zhou R, Hong Z (2003) Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera. *IEEE Trans Consum Electron* 49:257–262
- Japkowicz N, Shah M (2011) *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press
- Kehtarnavaz N, Oh HJ (2003) Development and real-time implementation of a rule-based auto-focus algorithm. *Real-Time Imaging* 9:197–203
- Kiefer J (1953) Sequential minimax search for a maximum. *Proc American Mathematical Society* Fibonacci algorithm; see also Donald E. Knuth, “The Art of Computer Programming (2nd edition)”, vol. 3, p. 418

-
- Lee SY, Kumar Y, Cho JM, Lee SW, Kim SW (2008) Enhanced autofocus algorithm using robust focus measure and fuzzy reasoning. *IEEE Trans Circuits Syst Video Tech* 18:1237–1246
- Li J (2005) Autofocus searching algorithm considering human visual system limitations. *Optical Engineering* 44:113,201–4
- Mir H, Xu P, van Beek P (2014) An extensive empirical evaluation of focus measures for digital photography. In: *Proc. SPIE 9023, Digital Photography X*
- Morgan-Mar D, Arnison MR (2013) Focus finding using scale invariant patterns. In: *Proc. SPIE 8660, Digital Photography IX*
- Peddigari V, Gamadia M, Kehtarnavaz N (2005) Real-time implementation issues in passive automatic focusing for digital still cameras. *Journal of Imaging Science and Technology* 49:114–123
- Quinlan JR (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann
- Rahman M, Kehtarnavaz N (2008) Real-time face-priority auto focus for digital and cell-phone cameras. *IEEE Trans Consum Electron* 54:1506–1513
- Santos A, Ortiz de Solórzano C, Vaquero JJ, Peña JM, Malpica N, del Pozo F (1997) Evaluation of autofocus functions in molecular cytogenetic analysis. *Journal of Microscopy* 188:264–272
- Subbarao M, Tyan JK (1998) Selecting the optimal focus measure for autofocusing and depth-from-focus. *IEEE Trans Pattern Anal Mach Intell* 20:864–870
- Van Hulse J, Khoshgoftaar TM, Napolitano A (2007) Experimental perspectives on learning from imbalanced data. In: *Proc. of the 24th International Conference on Machine Learning*, pp 935–942
- Vaquero D, Gelfand N, Tico M, Pulli K, Turk M (2011) Generalized autofocus. In: *IEEE Workshop on Applications of Computer Vision*
- Witten IH, Frank E (2000) *Data Mining*. Morgan Kaufmann
- Zografos V, Lenz R, Felsberg M (2013) The Weibull manifold in low-level image processing: An application to automatic image focusing. *Image and Vision Computing* 31:401–417