Pattern Recognition Letters
journal homepage: www.elsevier.com

Improving the accuracy and low-light performance of contrast-based autofocus using supervised machine learning

Rudi Chen, Peter van Beek**

University of Waterloo, 200 University Ave W, Waterloo, ON, N2L 3G1, Canada

ABSTRACT

The passive autofocus mechanism is an essential feature of modern digital cameras and needs to be highly accurate to obtain quality images. In this paper, we address the problem of finding a lens position where the image is in focus. We show that supervised machine learning techniques can be used to construct heuristics for a hill-climbing approach for finding such positions that out-performs previously proposed approaches in accuracy and robustly handles scenes with multiple objects at different focus distances and low-light situations. We gather a suite of 32 benchmarks representative of common photography situations and label them in an automated manner. A decision tree learning algorithm is used to induce heuristics from the data and the heuristics are then integrated into a control algorithm. Our experimental evaluation shows improved accuracy over previous work from 91.5% to 98.5% in regular settings and from 70.3% to 94.0% in low-light.

1. Introduction

The passive autofocus mechanism is an essential feature of modern digital cameras. To maximize the likelihood of obtaining high quality images, the mechanism must be quick and accurate. The two main forms of passive autofocus are phase-detection and contrast-detection. Phase-detection is faster and more efficient at tracking subject movement, but typically requires hardware found on higher-end cameras. Contrast-detection can be more accurate and uses image processing techniques to obtain a measure of the sharpness of an image that can be used on a wide range of devices, including point-and-shoot cameras, mobile phones and DSLRs. This paper focuses on contrast-detection.

We present a novel algorithm for finding an in-focus lens position through a local search, in contrast to sweeping through the entire range of lens positions. We build a control algorithm supported by supervised machine learning that is used to train a classifier to transition between the states of the algorithm. In supervised learning, classifiers are built using training examples (instances) consisting of a vector of feature values and labeled with the correct answer. We obtain training and test data using an offline simulation on a suite of 32 benchmarks with each instance labeled in an automated manner. From the gathered data, a decision tree learning algorithm (Quinlan, 1993) was used to induce multiple heuristics. 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 instance.

The final result is compared with previous work by performing an extensive evaluation over a range of real-life photography situations. Our approach is shown to be more accurate, including in low-light scenarios and situations where the focus measure is not unimodal.

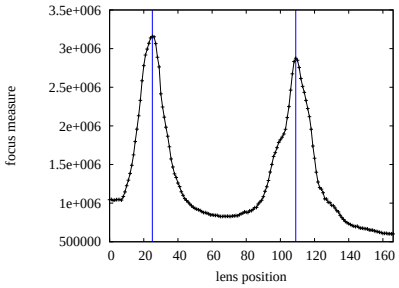
2. Background

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

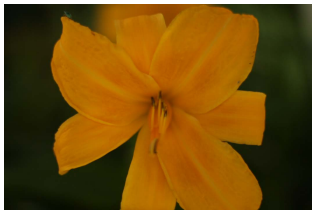
2.1. Focus measures

Contrast-detection autofocus (AF) 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, for example, Groen et al., 1985; Mir et al., 2014). In our work, we make use of two focus

**Corresponding author: Tel.: +1-519-888-4567, x35344; fax: +1-519-885-1208;
e-mail: vanbeek@cs.uwaterloo.ca (Peter van Beek)



(a)



(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 refer to the two images that have objects that are in maximal focus: (b) flower in focus, and (c) fern and grasses in focus.

measures: (i) the squared gradient focus measure (Santos et al., 1997) and (ii) a focus measure based on the convolution of the image with the derivatives of the Gaussian (Geusebroek et al., 2000; Gamadia and Kehtarnavaz, 2009b). Let $f(x, y)$ be the luminance or grayscale at pixel (x, y) in an image of size $M \times N$. 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,

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

The value $\phi(p)$ of the focus measure based on convolution with the first derivatives of the Gaussian is given by,

$$\phi(p) = \sum_{x} \sum_{y} [(f * G_x^\sigma)(x, y)]^2 + [(f * G_y^\sigma)(x, y)]^2,$$

where f is the image, G_x^σ and G_y^σ are the first-order derivatives of the Gaussian in the vertical and horizontal direction, σ is the scale of the filter (higher values cut off more of the high frequencies of the image), and * is the 2D convolution operator. Both of these focus measures have been shown to be highly effective across many types of images (Mir et al., 2014) and the Gaussian focus measure has excellent noise reduction properties for low-light situations (Gamadia and Kehtarnavaz, 2009b).

Following Kehtarnavaz and Oh (2003), 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 previous move, or (ii) have the camera automatically determine the object or region of interest to bring into focus, say by using face or object recognition (Rahman and Kehtarnavaz, 2008). Our proposals are easily adapted to the case where the ROI is an arbitrary sub-area of an image. Figure 1 shows the focus measures acquired at all possible lens positions (Canon 50mm lens) in a sample scene.

2.2. Focus search algorithms

A contrast-based AF algorithm searches for the lens position with the sharpest image as reported by the focus measure by iteratively moving the lens. Lenses are moved in discrete steps using a step motor. The images are streamed from the sensor at video frame rates (e.g., 24 frames per second on many Canon cameras) and also shown on the camera's live preview mode.

At each iteration, step motors can be moved in a single step or larger steps. In our case, the largest step is equivalent to eight small steps. Each step, small or large, is followed by a latency of hundreds of milliseconds. As well, step motors can suffer from backlash when the lens movement changes direction, making any absolute measure of lens position unreliable (Kehtarnavaz and Oh, 2003; Morgan-Mar and Arnison, 2013). In our work, we assume that the only information available to the camera in regards to lens position is whether the lens has reached the first or last lens position. An efficient AF algorithm will therefore take as large steps as possible and be able to handle losses in accuracy due to backlash.

Given a set of lens positions $\{a, a+1, \ldots, b\}$, an autofocus algorithm can solve one of several search problems. The first is to find the lens position that corresponds to the *maximum* or *highest* peak. This often involves finding *all* peaks. Another problem is to find a *nearby* peak. During real usage, the lens could be resting at any lens position prior to autofocus activation. In fact, the lens is likely to be near a peak already when images are taken consecutively. In that case, it is preferable to start the search from the current position rather than using an approach that requires moving the lens back to the first lens position. For lenses with a large number of positions, such as those used with DSLRs, moving the lens back to the first lens position would also incur a significant visual artifact in the live preview where the image goes significantly out of focus before coming back to focus again. This would not be desirable for the user. In this paper, we address the problem of finding a nearby peak without moving the lens back to the first lens position.

3. Related Work

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. 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. The goal of the heuristic is to move the lens larger steps but without missing any peaks in the focus measure. Gamadia and Kehtarnavaz (2009a) later use this sweeping algorithm to focus more quickly by terminating the search at the first peak. While the sweeping approach has been shown to be efficient, it requires the lens to be brought to the first lens position, which is undesirable when the lens is already close to a nearby peak.

He, Zhou and Hong (2003) propose a coarse-to-fine search where initially the search algorithm predicts whether to move towards near focus or far focus, then takes coarse or large steps until a first peak is found and finally, reverses direction and takes fine steps to determine the peak. Li (2005) uses a similar method, using medium steps instead of fine steps. Left unclear is the specific conditions under which the direction is reversed, and how to handle cases where the initial prediction is incorrect. We found that the choice of such conditions can significantly affect the success rate of the autofocus algorithm. Our approach can be viewed as showing how to improve He et al.'s search algorithm to make it more robust and accurate by using a more elaborate control algorithm and using machine learning to automatically learn the heuristics for the control algorithm. We perform an extensive experimental comparison against He et al.'s search algorithm in Section 5.

Recent work has also focused on the idea of *predicting* the location of a peak based on a few initial measurements. Chen, Hong and Chuang (2006) sample the focus measures at four initial lens positions, then fit an equation to predict the location of a *nearby* peak, take coarse steps to be near the predicted peak, and finally take fine steps within a bisection search algorithm to find the peak. Supervised machine learning approaches to predicting the location of a 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 a peak based on sampling the focus measure at the first three lens positions. Their approach is one of the first to be based on supervised machine learning techniques. Han et al. (2011) also use a machine learning approach, a variation of 1-nearest neighbor, to predict the location of a peak by sampling the focus measure at the first three lens positions. Both approaches require the camera to be moved to the first lens position and in general, prediction algorithms rely on the ratio of consecutive lens positions to be a good predictor of the location of a peak. While this is often true with mobile phones and compact cameras where there is a large depth of field, we find this to not be the case with longer lenses such as the ones found on a DSLR (see Section 5).

The issue of low-light photography has also been addressed in the literature. Choi, Lee and Ko (1999) use a frequency selective weighted median filter to reduce noise. Shen and Chen (2006) use a focus measure involving the coefficients of the image in the discrete cosine transform domain to achieve a higher discrimination ratio between out-of-focus and in-focus images. Gamadia et al. (2007; 2009b) show that a focus measure based on a Gaussian smoothing step, a contrast enhancement step, and a derivative step, is very effective in low-light situations. In our low-light experiments, we adopt a simplified, but still very competitive, version of Gamadia et al.'s proposal, where we omit the contrast enhancement step and perform Gaussian smoothing and taking the derivative all in one step.

4. Our Solution

Our proposed solution consists of a hand-crafted control algorithm (Section 4.1), represented as states in Figure 2, which uses machine learning trained heuristics to transition between states while searching for the best-in-focus lens position. The construction of heuristics begins with the creation of a set of features (Section 4.2), followed by the collection of training data (Section 4.3) and the training of two decision tree classifier (Section 4.4).

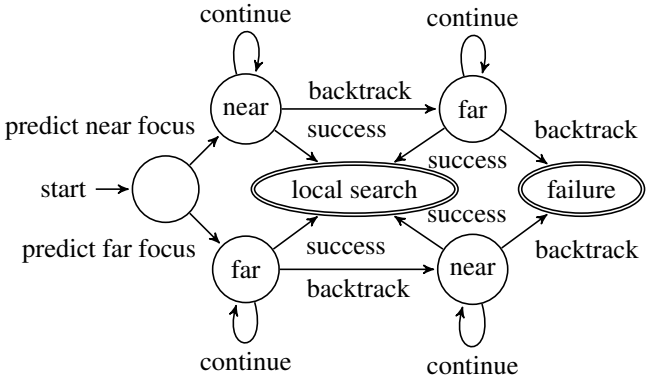


Fig. 2. State diagram illustrating the control algorithm. In the initial state, a prediction is made about the direction of a peak. The lens will take steps in that direction (possibly backtracking once) until a peak is found or the search fails.

4.1. Control algorithm

The states of the control algorithm are illustrated in Figure 2. Initially, the lens is at rest at an arbitrary position and the algorithm needs to determine whether to look for a peak towards near focus or far focus. The focus value is measured at three consecutive lens positions by taking two fine steps towards far focus. These three measurements are used by a decision tree T_α that will predict the direction, near or far, in which a peak is more likely to be found.

The algorithm will then begin a search by moving the lens in coarse steps in the chosen direction to find a peak. At each step, a tuple (i, f_i) is recorded where i is the number of steps taken during the search (ith step) and f_i is the corresponding focus value measured after the step was taken. The tuple $(0, f_0)$ corresponds to the focus value at the initial lens position. The recorded tuples are then used by a second decision tree T_β with one of three labels at each leaf : "continue" or "backtrack", "success". Each label represents a possible state transition:

label	description
continue	Continue the search by taking another step in the same direction and repeat the process.
backtrack	The initial direction chosen is predicted to have been incorrect and there is unlikely to be a peak in that direction. Return the lens to the position where it started and repeat the search process in the opposite direction, reinitializing the list of measured focus values. If this label is obtained a second time, the search is considered a failure and the algorithm falls back to a full sweep.
success	A peak is predicted to be close to one of the visited lens positions. Reverse direction and return the lens to the position where the focus value is highest among visited lens positions.

When the search is considered a success and the lens has been returned to the visited position with the highest focus value, it is expected that the lens is either at a peak or close to one. A local search is then performed, which consists of a simple hill-climbing algorithm where the lens moves in fine steps until the focus value stops increasing. The local search maximizes the focus value and increases tolerance to sources of error such as backlash or small changes in the peak location during autofocus due to camera or subject movement. Since we only perform local search when the primary ML-based search outputs "success", the lens is already close enough to a peak that the local search will not be affected by local extrema.

4.2. Features

In supervised machine learning, the construction of a set of features represented in each instance is an important factor in the success of the approach. Features should enable effective discrimination between the different classifications of each set of instances.

Two sets of features were manually designed. The first set of features, denoted F_α, is used to train a heuristic to determine whether to move the lens towards near focus or far focus after taking the initial three focus measurements (see Table 1 for the definitions of these features). The features in this set are Boolean-valued and consist mainly of a comparison between those three focus values with varying levels of granularity.

The second set of features, denoted F_β, is used to train a heuristic to decide between the state transitions "continue", "backtrack" and "success" (see Appendix A for the definitions of these features). These features have numerical values and describe at a higher level the focus values recorded during the search procedure. Examples of features include the number of steps taken, the slope between the two most recent focus values and the ratio between the current focus value and the largest focus value encountered so far. Focus values represented as descriptive features are easier to handle for a machine learning scheme than the focus values themselves.

A challenge in constructing features is that each camera and lens (if considering a camera with interchangeable lenses) will have a different number of total lens positions T_p. This will affect the distribution of values of instances of each feature. For example, if a camera has fewer lens positions (coarser granularity), it is clear that the slope of the focus values would increase more quickly with each step, all else being equal. Another challenge is that the scale on which focus values are measured is arbitrary and influenced by the amount of detail in the scene as well as lighting conditions (Han et al., 2011). Therefore, to ensure generality, lens position counts are normalized to $[0, 1]$ whenever they are used in the calculation of a feature and focus values are also normalized when necessary. For example, a feature F that involves the slope between two points $(x, f_x), (y, f_y)$ will be calculated as,

$$F((x, f_x), (y, f_y)) = \frac{(f_y - f_x) \overbrace{/ \frac{(f_y + f_x)}{2}}^{\text{normalize by focus value}}}{(y - x) \underbrace{/ T_p}_{\text{normalize by total lens positions}}}.$$

Table 1. Set of features F_α for learning the heuristic to predict the direction in which a peak is most likely to be found. The values f_1, f_2, f_3 are the first three focus measurements taken by moving two fine steps.

$$
\begin{aligned}
I &= \{1.64, 1.32, \ldots, 1.01, 1, 1/1.01\} \\
J &= \{-0.64, -0.32, \ldots, -0.01, 0, 0.01, \ldots, 0.64\} \\
\text{ratio}(k) &= \left(\tfrac{f_3}{f_1} > k\right), k \in I \\
\text{diffMax}(k) &= \left(\tfrac{f_3 - f_1}{\max(f_1, f_3)} > k\right), k \in J \\
\text{diffMin}(k) &= \left(\tfrac{f_3 - f_1}{\min(f_1, f_3)} > k\right), k \in J \\
\text{curving}(k) &= \left(\tfrac{f_1 + f_3 - 2f_2}{f_2} > k\right), k \in J \\
\text{curvingRatio}(k) &= \left(\tfrac{f_1 - f_2}{f_2 - f_3} > k\right), k \in J \\
\text{downtrend} &= f_1 \geq f_2 \geq f_3 \\
\text{uptrend} &= f_3 \geq f_2 \geq f_1
\end{aligned}
$$

4.3. Data Collection

Another important factor in the success of a supervised machine learning approach in this context is to have data representative of photography situations seen in practice. During data collection, a set of benchmark images was taken for each of 32 everyday scenes that covered a range of common photography settings including landscapes, closeups, interiors, still lifes, and so on. Among these scenes, 10 exhibit more than one peak, for a total of 53 peaks.

To gather the benchmark scenes, we use a remote camera application to control a Canon EOS 550D/Rebel T2i camera connected to the computer via a USB cable. The remote camera application can display the camera's live view video stream and move the lens in small or coarse steps using the Canon SDK (v2.13). Using this setup, for each scene, a JPEG image is captured at each of the 167 different lens focus positions with a two second delay between each capture to ensure the lens stabilizes. The JPEG images are taken directly from the live-view at each possible lens position. The square gradient focus measure is applied to each image to calculate the focus value.

Once the images are collected, the machine learning training data is generated. The data is a set of instances, where each instance is a vector of feature values and a label representing the correct classification for that instance. The heuristics used by the control algorithm discussed in Section 4.1 require two training sets. The first training set is used to determine whether to start the search towards the left side (near focus) or right side (far focus). The data is gathered as follows: for each three consecutive focus values $[f_{x-2}, f_{x-1}, f_x], x \in [2, 166]$ in each scene where the lens positions are numbered $0, \ldots, 166$, an instance is created by evaluating the features in F_α with those focus values as parameters. These correspond to the initial three measurements. The class label for that instance is generated automatically using a simple rule: if the closest peak at position x in the focus measure curve is on the left of x, corresponding to near focus, then the label is "near"; otherwise, the label is "far". Alternative classification rules involving the height of a peak were

considered, but did not lead to better results. This gives a total of 9,218 training instances across all scenes.

Algorithm 1: Decision tree T_α for selecting the direction of the first sweep.

input: Focus measures f_1, f_2, f_3
if $ratio(1) = 0$ **then**
  $initialDirection \leftarrow near$
if $ratio(1) = 1$ **then**
  $initialDirection \leftarrow far$

The second training set is used to decide whether to continue searching, backtrack, or end the search with a success. The instances are generated by simulating a peak search starting at an arbitrary lens position. The simulation has full information of the focus curve. It is run once for every lens position, as it is assumed that the lens could be at rest at any position prior to autofocus, and in both the near focus and far focus directions. At every step during the search, (i) the focus values at the current lens position and number of steps taken so far are recorded, (ii) the features in F_β for a new instance are evaluated with the recorded focus values and (iii) the instance is then labeled with the correct action to take at this point in the search. This is repeated until the first or last lens position is reached. The label is assigned to be "success" if a peak has been passed two or more steps ago, "backtrack" if at least four coarse steps have been taken and there are no more peaks in the current direction, "continue" otherwise. This gives a total of 22,691 training instances across all scenes.

A few techniques improved the quality of the data and the resulting efficiency and accuracy of our approach.

1. The dataset was balanced such that instances with labels "backtrack", "success" and "continue" appear in a 1:1:3 ratio in the final training set. This introduced a bias during classifier training in favor of taking more steps, which is preferable to terminating the search prematurely.
2. The frequency of instances associated with states that are unlikely to occur was reduced. For example, it is unlikely in practice for a search to pass over two or more peaks without terminating, whereas in the simulation, the search will continue until the lens reaches the very last position. To do so, each lens step taken during the simulation of a given search reduced the likelihood that the instance associated with that state will appear in the training data by a factor μ when the correct label at that step is "continue" and γ otherwise. That is, the likelihood of selecting an instance is $p = \mu^x \gamma^{n-x}$ where x is the number of instances labeled "continue" and n is the total number of steps taken during the search up to this point. The instances were then randomly sampled by p. This improved the robustness of the learning scheme. In our setting where a full sweep is 19 coarse steps, we use $\mu = 0.99, \gamma = 0.93$ such that a full sweep encountering no peaks will have $p = 0.99^{19} \approx 90\%$ and a full sweep encountering a peak in the beginning will have $p = 0.93^{19} \approx 25\%$, which we found to work well.
3. Some noise, up to 10% of the lowest focus value, was added to the focus value measurements.

Algorithm 2: Decision tree T_β for selecting the transition between states after n steps were taken and n+1 focus values are obtained.

input: Records $\{\{0, f_0\}, \{1, f_1\}, \{2, f_2\}, \ldots, \{n, f_n\}\}$
if distanceToMax ≤ 0.03 **then**
  if ratioToMax ≤ 0.673 **then**
    if ratioMinToMax ≤ 0.378 **then**
      if currentSlopeLarge ≤ -0.465 **then**
        $result \leftarrow success$
      if currentSlopeLarge > -0.465 **then**
        $result \leftarrow continue$
    if ratioMinToMax > 0.378 **then**
      $result \leftarrow continue$
  if ratioToMax > 0.673 **then**
    $result \leftarrow continue$
if distanceToMax > 0.03 **then**
  if ratioMinToMax ≤ 0.352 **then**
    if downslope1stHalf ≤ 0.833 **then**
      $result \leftarrow success$
    if downslope1stHalf > 0.833 **then**
      if ratioMinToMax ≤ 0.126 **then**
        $result \leftarrow success$
      if ratioMinToMax > 0.126 **then**
        $result \leftarrow backtrack$
  if ratioMinToMax > 0.352 **then**
    $result \leftarrow backtrack$

4.4. Classifier Selection

Using the data collected, the final step is to learn the classifiers. This was done with Weka's (Witten et al., 2011) J48 implementation of Quinlan's (1993) C4.5 decision tree learning algorithm. Among the classification-based machine learning algorithms that were tested, decision trees performed consistently well. Decision trees also have the advantage of being efficient to evaluate and produce human-understandable output. The software was run using default settings, with the exception of the minimum number of instances per leaf, which was set to 512. Increasing this value is a standard technique to obtain simpler trees and reduce overfitting when lots of data is available. The decision trees generated are shown in Algorithm 1 and Algorithm 2. Notice that the classifier for determining the initial direction (Algorithm 1) is a simple decision stump, comparing whether the near focus or far focus position has the largest focus value. However, despite being the best classifier, only 87% of instances are correctly classified on our dataset, justifying the need for the ability to backtrack in cases where the initial direction chosen is incorrect.

For Algorithm 2, the features in the training set for the decision tree learning algorithm (J48) are continuous (numeric) features. For a feature f, J48 automatically chooses a binary split point c that gives the highest information gain and branches on that split point; i.e., $f \leq c$ is one branch and $f > c$ is the other branch. Thus, each of the values in Algorithm 2 is automatically chosen by J48 as part of the learning process. Notice that,

while many possible features where made available to the decision tree learning algorithm (see Table 1 and Appendix A), only subsets of these features were actually chosen by J48 for inclusion in the final decision trees that are shown in Algorithm 1 and Algorithm 2.

5. Experimental Evaluation

We perform an empirical evaluation of the effectiveness of our machine-learning based (ml-based) approach on the criteria of speed and accuracy. Speed is defined as the number of steps used to autofocus. We found that the Canon Rebel T2i camera used in these experiments exhibits the same latency between fine steps and coarse steps, which suggests a strong correlation between the number of steps and the total time taken to focus. Accuracy is defined as the proportion of simulations of the autofocus algorithm where the final lens position is within one lens position of a peak (i.e., a peak was found). We find that a difference of more than one lens position from the peak is noticeable, even with 1 megapixel images.

We compare, using the information we could obtain, against the hill-climbing method proposed in (He et al., 2003), where coarse or large steps are taken until a peak is found, after which the direction is reversed and fine steps are taken to narrow down on the peak and the conditions for reversing direction are manually created. (The proposal in (Li, 2005) differs mainly in the step size used and in relaxing the accuracy goal, but uses the same underlying idea.) Similarly to our approach, their goal is to find a nearby peak given any starting lens position.

In He et al. (2003), the first step taken to determine the initial direction is a coarse step, and the direction is determined by the largest of two focus values. Backtracking will occur if and only if the first step after deciding on the initial direction causes the focus value to decrease. The other decision that needs to be made is when to stop the search, reverse direction and switch to fine steps. In (He et al., 2003), this happens after the focus value decreases. (The precise condition is not specified in (Li, 2005)). In our comparison, we reverse direction when the focus value drops below 90% of the maximum focus value seen so far, which gives their hill-climbing method higher accuracy.

The test suite consists of the same 32 sets of benchmark images used during data collection in Section 4.3. Each set consists of 167 images, one for every lens position. He et al.'s (2003) hill-climbing method was run directly on each of the 32 sets, with one test for each starting lens position.

To evaluate the ml-based heuristics, we use a variation of leave-one-out cross-validation. For each of the 32 benchmarks to be tested, the training data is collected as in Section 4.3 using only the remaining 31 benchmarks. In other words, the test benchmark is left out from the training data. Classifier selection is then performed on the training data and the results are incorporated into heuristics as decision trees. The search algorithm using those heuristics is then evaluated against the test benchmark on each of the 165 starting lens position (lens positions 2, . . . , 166). This procedure assesses the generalization performance of the heuristics. By excluding the test benchmark, the heuristics are evaluated on their ability to handle new data. In all cases, the simulated camera model will generate some noise at each focus measurement, up to 5% of the lowest focus measure. We found that this approximately corresponds to moderate amounts of camera shake. We also simulate backlash—when the direction changes, the step taken will be thrown off by up to 30%.

Table 2. Results of comparing He et al.'s (2003) hill-climbing approach and our machine-learning approach (ml-based) using leave-one-out cross-validation. Simulations include backlash and noise simulation. Each benchmark was run over the 165 starting lens positions. The accuracy is the percentage of such simulations where a peak was found and the number of steps are averaged over those simulations. Benchmarks marked with (*) have more than one peak.

	He et al. (2003)		ml-based	
benchmark	accuracy	avg step	accuracy	avg step
backyard	99.4	20.2	100.0	19.3
bench	75.5	17.2	76.4	18.6
book	85.5	18.6	98.2	24.0
books1	96.9	20.6	100.0	24.0
books2	97.5	20.3	100.0	21.8
books3	99.4	20.1	100.0	23.8
books4	98.7	18.5	100.0	25.8
bridge	98.7	16.2	100.0	22.3
building1	89.9	15.2	99.4	22.3
building2	83.0	15.0	99.4	22.5
building3	98.7	18.4	100.0	21.8
cup1	88.7	16.8	100.0	17.9
cup2*	94.3	14.8	100.0	17.7
cup3*	62.9	13.3	100.0	19.8
cup4*	89.3	15.4	98.8	17.3
fabric	98.7	16.7	99.4	17.6
flower*	100.0	16.0	99.4	16.1
gametree*	84.9	15.1	100.0	21.6
gorillapod*	99.4	19.2	92.1	21.5
granola*	54.1	12.7	98.8	19.4
interior1	99.4	19.9	100.0	20.1
interior2*	94.3	15.9	100.0	16.9
lamp*	90.6	14.3	100.0	15.7
landscape1	93.7	16.9	100.0	22.6
landscape2	96.9	16.0	100.0	21.9
landscape3	96.9	14.0	100.0	19.3
screen*	95.0	16.7	100.0	15.5
snails	100.0	17.1	98.8	17.8
stillLife	86.8	17.4	98.8	19.0
timbuk	92.5	13.9	100.0	21.8
ubuntu	84.9	16.6	96.4	21.7
vase	100.0	17.9	97.6	17.2
average	91.5	16.8	98.5	20.1

As shown in Table 2, our method takes 20% more steps, but is significantly more accurate in finding peaks (98.5% vs 91.5%). The only benchmark in which it struggles is "bench", which is an outdoor scene where the lighting conditions changed considerably while the set of images was acquired (clouds covered the sun in some images, but not others). Our method performs considerably better on benchmarks with more than one peak

Table 3. Result of simulating He et al.'s (2003) hill-climbing and our ml-based approaches on low-light benchmarks, using the squared gradient and Gaussian derivative focus measure. Simulations include noise and backtracking. Each benchmark was run over the 165 starting lens positions. Benchmarks marked with (*) have more than one peak. The column r shows the ratio between the focus value at the 1st decile and the largest focus value.

	Squared Gradient					Gaussian first-derivative, $\sigma = 2$				
		He et al. (2003)		ml-based			He et al. (2003)		ml-based	
benchmark	r	accuracy	avg step	accuracy	avg step	r	accuracy	avg step	accuracy	avg step
blackboard1*	0.6	43.0	13.6	20.5	28.4	0.1	62.5	14.0	95.2	18.5
blackboard2*	0.8	0.6	11.3	1.1	25.2	0.2	85.0	17.5	80.7	20.2
pillow1	0.8	9.9	12.2	5.1	28.6	0.1	86.0	17.5	100	19.4
pillow2	0.9	0.0	14.6	0.0	26.1	0.3	75.9	16.6	98.2	19.4
pillow3	0.8	0.6	11.8	10.7	30.5	0.3	42.0	13.7	86.9	19.5
projector1	0.3	78.5	16.5	86.0	18.8	0.0	100	19.4	100	19.1
screws1	0.5	57.4	15.2	67.3	29.2	0.1	88.3	18.1	98.8	21.7
screws2*	0.9	2.4	13.0	0.0	24.3	0.2	61.7	13.3	96.5	22.5
whiteboard1	0.9	0.0	14.8	0.0	24.9	0.1	40.6	15.1	92.8	20.0
whiteboard2	0.9	0.0	14.8	0.0	24.7	0.1	43.8	13.8	84.9	20.6
whiteboard3	0.8	1.2	14.5	0.6	25.8	0.0	87.5	19.2	100	21.1
average	0.7	17.6	13.8	17.4	26.0	0.1	70.3	16.2	94.0	20.2

(98.9% vs 86.5%). The limited backtracking support of He et al.'s (2003) hill-climbing algorithm also causes it to fail in some scenes where the lens starts in an out-of-focus region where the focus value curve is mostly flat and noise. When the lens starts the search in the wrong direction, it may never recover. As our classifier was trained to be conservative about backtracking, fluctuant focus curves did not in general cause a reduction in performance.

Recall that Han et al. (2011) also propose a machine learning approach (see Section 3). Han et al. (2011) bring the camera to the front, then take two steps to obtain three focus values, which are then used to calculate two relative focus measures. Using 1-nearest-neighbor, these two features are used to predict the location of the next peak. They show that the technique works well in mobile phones and compact cameras where there is a large depth of field. However, in our DSLR setting, our benchmark scenes often contain focus curves with long flat segments. As shown in Figure 3, the features used by Han et al. are unable to discriminate between mid-distance subjects and landscape images with very distance objects in focus. Other prediction-based techniques such as Chen et al. (2010) use similar features, leaving us doubtful that these techniques can be adapted to a DSLR setting.

To assess the low-light performance of our method, we acquired 11 more set of images, taken in darkened rooms. For every image in every set, the squared gradient focus measure was again computed. These image sets were not included in our training set. Initially, this led to a very low success rate of approximately 17% for both He et al.'s hill-climbing approach and our ml-based approach (Table 3), which is also approximately the same success rate as the Canon Rebel T2i's default autofocus for the same scenes. A closer examination reveals that the root cause is the lack of discrimination power between the focus measure of out-of-focus images and in-focus images. In our benchmarks using the squared gradient focus measure, the smallest focus measure often reaches 80% of the value of the largest. We find that the smallest focus measure needs to be

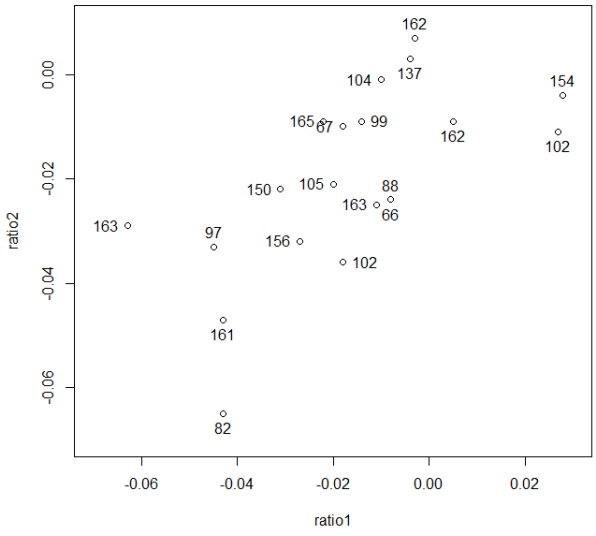


Fig. 3. Plot of the location of the lens position (0-166) of the nearest peak, indicated by labels for a subset of our benchmark scenes. The x and y axis are the relative focus measure difference between the first and second focus value measurement and second and third value measurement, respectively.

no more than 50-60% of the largest for our algorithm to succeed. The ratio r between the focus value at the 1st decile and the largest focus value is shown in Table 3. A decile is used to discount outliers at low focus values, but is unnecessary for large focus values as they correspond to the peak. A value close to 1.0 indicates that the peak in the focus measure differs little from the non-peak values, which makes focusing difficult; a value close to 0.0 indicates the peak has high amplitude, which makes focusing potentially easier for a focusing algorithm.

The reduced difference between out-of-focus and in-focus images in low-light situations is due to a lower signal-per-noise ratio. The features of the image (signal) have a smaller range on the RGB scale whereas the value added to the focus measure by noise remains constant or increases in low-light condi-

tions, often due to increased ISO. This reduces the discrimination power. However, low-light noise will typically be found at higher frequencies relative to the features of the image. Thus, we want a focus measure that acts as a low-pass filter to remove the contribution of the noise to the focus measure effectively. As it has the desired properties, we chose a first-derivative Gaussian filter, which is a simplified, but still very competitive, version of Gamadia et al.'s proposal (see Section 3). Replacing the squared gradient focus measure with the Gaussian focus measure with $\sigma = 1$ helps alleviate this issue, and $\sigma = 2$ solves it completely. The ml-based approach performs particularly well (94% vs 70.3%) compared to He et al.'s hill-climbing approach. We also note that the measure of the number of steps is only over the number of times that an algorithm succeeds. Thus, while He et al. can perform fewer steps on average than our method, this is only because it fails more often.

Our goal was to emphasize our algorithm's high performance versus previous work rather than advocate for particular focus measures. However, it should be noted that, while implementing the Gaussian filter (with $\sigma = 2$ where the tail of the Gaussian is cutoff at 3σ) on a smartphone controlling an onboard camera or tethered to a DSLR is feasible, implementing the filter onboard a camera may require the addition of a floating point co-processor and/or additional memory (in general, camera processors are proprietary and information is sparse). One plus is that Gaussian filters do have the nice property that they are separable: a two dimensional filter can be implemented using two independent one-dimensional calculations.

6. Conclusion

Previous proposals for finding nearby peaks in autofocusing involve a hill-climbing search where the conditions for state transition (backtrack, reverse direction) are hand-crafted. We show that supervised machine learning techniques can be used to construct heuristics to determine those transitions and outperforms previous work on accuracy (98.5% vs 91.5%). In particular, we show that our algorithm is more robust, even in difficult cases where multiple peaks in the focus measure are present and, by using the first-order Gaussian derivative focus measure, also more robust in low-light conditions.

Acknowledgments

This work was supported in part by an NSERC USRA award and an NSERC Discovery Grant.

References

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, C.Y., Hwang, R.C., Chen, Y.J., 2010. A passive auto-focus camera control system. Applied Soft Computing 10, 296–303.

Choi, K.S., Lee, J.S., Ko, S.J., 1999. New autofocusing technique using the frequency selective weighted median filter for video cameras. IEEE Trans. Consum. Electron. 45, 820–827.

Gamadia, M., Kehtarnavaz, N., 2009a. Enhanced low-light auto-focus system model in digital still and cell-phone cameras, in: IEEE International Conference on Image Processing, pp. 2677–2680.

Gamadia, M., Kehtarnavaz, N., 2009b. Real-time implementation of single-shot passive auto focus on DM350 digital camera processor, in: Real-Time Image and Video Processing (SPIE Vol. 7244).

Gamadia, M., Kehtarnavaz, N., Roberts-Hoffman, K., 2007. Low-light auto-focus enhancement for digital and cell-phone camera image pipelines. IEEE Trans. Consum. Electron. 53, 249–257.

Geusebroek, J.M., Cornelissen, F., Smeulders, A.W.M., Geerts, H., 2000. Robust autofocusing in microscopy. Cytometry 39, 1–9.

Groen, F., Young, I., Ligthart, G., 1985. A comparison of different focus functions for use in autofocus algorithms. Cytometry 6, 81–91.

Han, J.W., Kim, J.H., Lee, H.T., Ko, S.J., 2011. A novel training based autofocus for mobile-phone cameras. IEEE Trans. Consum. Electron. 57, 232–238.

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.

Kehtarnavaz, N., Oh, H.J., 2003. Development and real-time implementation of a rule-based auto-focus algorithm. Real-Time Imaging 9, 197–203.

Li, J., 2005. Autofocus searching algorithm considering human visual system limitations. Optical Engineering 44, 113201–113201–4.

Mir, H., Xu, P., van Beek, P., 2014. An extensive empirical evaluation of focus measures for digital photography, in: Proc. SPIE 9023, Digital Photog. X.

Morgan-Mar, D., Arnison, M.R., 2013. Focus finding using scale invariant patterns, in: Proc. SPIE 8660, Digital Photography IX.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. MK.

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, J.J., Peña, J.M., Malpica, N., del Pozo, F., 1997. Evaluation of autofocus functions in molecular cytogenetic analysis. Journal of Microscopy 188, 264–272.

Shen, C., Chen, H., 2006. Robust focus measure for low-contrast images, in: Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on, pp. 69–70.

Witten, I.H., Frank, E., Hall, M.A., 2011. Data Mining. 3rd ed., MK.

Appendix A. Features for state transition heuristics

Set of features F_β for learning the state transition heuristic, where T_p is the total number of positions on the lens, s_l is the size of a large step (8 in our experiments), and the records $\{\{0, f_0\}, \{1, f_1\}, \{2, f_2\}, \ldots, \{n, f_n\}\}$ are the focus measures obtained so far during the search in the current direction. In the definitions of the features, $\text{avg}(x, y) = \frac{x+y}{2}$.

$$\text{distanceSwept} = \frac{n}{T_p}$$

$$\text{ratioToMax} = \frac{f_n}{\max_i f_i}$$

$$\text{ratioToRange} = \frac{f_n}{\max_i f_i - \min_j f_j}$$

$$\text{ratioMinToMax} = \frac{\min_i f_i}{\max_j f_j}$$

$$\text{distanceToMax} = \frac{s_l(n - \arg\max_i f_i)}{T_p}$$

$$\text{simpleSlope} = \frac{(f_n - f_0)/\text{avg}(f_n, f_0)}{n/T_p}$$

$$\text{currentSlope} = \frac{(f_n - f_{n-1})/\text{avg}(f_n, f_{n-1})}{n/T_p}$$

$$\text{currentSlopeLarge} = \frac{(f_n - f_{n-2})/\text{avg}(f_n, f_{n-2})}{n/T_p}$$

$$\text{downslope1stHalf} = |\{(f_i, f_{i+1}) \ : \ f_i > f_{i+1}, i < \frac{n}{2}\}|/n$$

$$\text{downslope2ndHalf} = |\{(f_i, f_{i+1}) \ : \ f_i > f_{i+1}, i \geq \frac{n}{2}\}|/n$$