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

Rudi Chen, Peter Van Beek
Cheriton School of Computer Science
University of Waterloo, Waterloo, Canada
vanbeek@uwaterloo.ca

**Abstract**

The passive autofocus mechanism is an essential feature of modern digital cameras and needs to be highly accurate to obtain quality pictures. 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 to finding such positions which 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 then used to induct heuristics from the data and integrated into a control algorithm. Our experimental evaluation shows improved accuracy from 91.5% to 98.5% in regular settings and 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 pictures, 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 which 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 which

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 in 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 [16] 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 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, for example, [8] [20]). In our work, we make use of an effective focus measure called the squared gradient [18]. 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

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

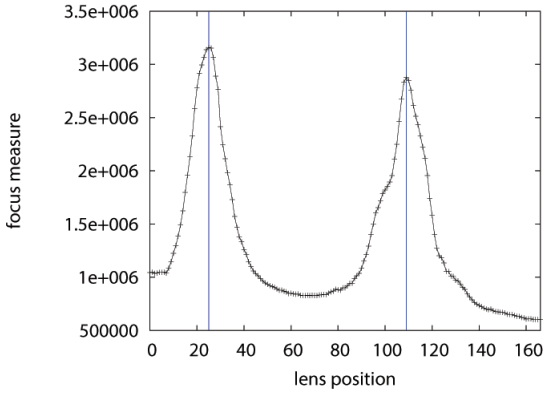
Another focus measure that has been proposed uses a convolution with the first derivative of a Gaussian, which has excellent noise reduction properties [7].

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

Here, $G_x(x, y, \sigma), G_y(x, y, \sigma)$ represent the first-order derivative of the Gaussian, $\sigma$ is the scale of the filter (higher values cut off more of the high frequencies of the image) and $f_x, f_y$ are the image derivatives at scale $\sigma$. This focus measure is used less often due to increased computational costs. However, we will later show that in low-light situations, where the signal-to-noise ratio is low [4],

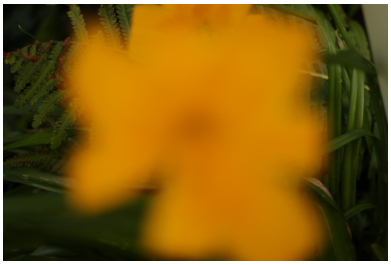
using the Gaussian focus measure can drastically improve the success rate of autofocus algorithms.

Following [12], 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 (e.g. using face or object recognition [13] [17]). Our proposals are easily adapted to the case where the ROI is an arbitrary sub-area of an image. Generally, smaller ROIs are an easier problem, as there are fewer peaks. Figure 1 shows the focus measures acquired at all possible lens positions (Canon 50mm lens) in a sample scene.




(a)

(b)



(c)

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

### 2.2 Search algorithms

A contrast-based autofocus algorithm searches for the lens position with the sharpest image as reported by the focus measure by iteratively moving the lens. 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 [12], [15]. 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 $\{a, a+1, ..., b\}$, the autofocus algorithm can solve one of multiple 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 pictures are taken consecutively. In that case, it is preferable to start the search from the current position rather than using an approach which requires moving the lens back to the first lens position. With lenses with a large number of positions, such as in the case of DSLRs, this 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 end user. This paper addresses the problem of finding a nearby peak.

## 3 Related Work

Kehtarnavaz and Oh [12] 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 [6] 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 [11] 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 [14] uses a similar method, using medium steps instead of large steps. Left unclear is the specific conditions under which the direction is reversed, and how to handle cases where the initial prediction is incorrect. In this paper, we find that the choice of such conditions can significantly affect the success rate of the autofocus algorithm.

Recent work has also focused on the idea of *predicting* the location of a peak based on a few initial measurements. This approach aims at computing the focus measure at fewer lens positions, thereby lowering the energy consumption. Chen, Hong and Chuang [2] samples the focus measures at four initial lens positions, then fits an equation to predict the location of a *nearby* 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 predicting the location of a peak have also been proposed. Chen, Hwang and Chen [1] 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, Kim, Lee and Ko [10] 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 (low $f$-number) where there is a large depth of field, we find this to not be the case with larger lenses such as the ones found on a DSLR.

Finally, the issue of low-light photography has also been addressed in the literature. Choi, Lee and Ko [3] use a frequency selective weighted median filter to reduce noise. Shen and Chen [19] use a focus measure involving the coefficients of the image in discrete cosine transform domain to achieve higher discrimination ratio between out-of-focus and in-focus parts of the image. Gamadia, Kehtarnavaz and Roberts-Hoffman [4] propose a series of preprocessing steps including median filtering, binomial smoothing and contrast enhancement, later improved [5] via a digital band-pass filterbank. In our work, we find that changing the focus measure to a first-order Gaussian derivative is sufficient for our heuristics to achieve high performance.

# 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).

## 4.1 Control algorithm

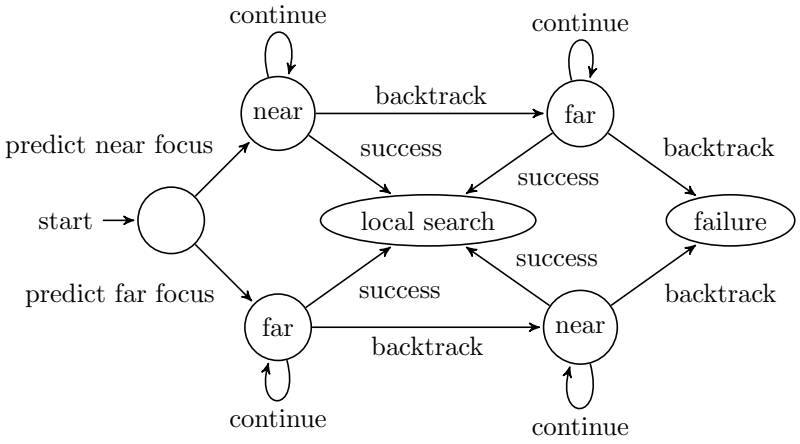



Figure 2: State diagram illustrating the control algorithm. In the initial state, a prediction is made about the direction of a peak. The lens will take steps in that direction (possibly backtracking once) until a peak is found or the search fails.

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 $T_\alpha$ which will predict the direction 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 ($i$th 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 $T_\beta$ with one of three labels at each leaf : "continue" or "backtrack", "success". Each label represents a possible state transition, as described in Table 1.

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 hillclimbing algorithm where the lens moves in fine steps until the focus value stops increasing (Algorithm 1). 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.

Table 1: Description of the state transitions in the control algorithm, one of which occuring after each lens step, as seen in Figure 2

| 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 that there is unlikely to be a peak in that direction. Reverse direction and 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. |

**Algorithm 1:** Local search algorithm

```
input  : Function φ(p), position p, direction d ∈ {−1, 1}
output: Final position
p' ← p;
done ← false;
while not atLastPosition(p', d) and not done do
    prevFocus ← φ(p');
    p' ← p' + d;
    if φ(p) < prevFocus then
        p' ← p' − d;
        done ← true;
return p';
```

## 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 $F_\alpha$ 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. The features in this set are boolean-valued and consist mainly of a comparison between those three focus values with varying levels of granularity (Table 4.2).

The second set of features $F_\beta$ is used to train a heuristic to decide between the state transitions "continue", "backtrack" and "success". These features have numerical values and describe at a higher level the focus values recorded during

Table 2: The set of features created for learning heuristics for predicting 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.

$$I = \{1.64, 1.32, ..., 1.01, 1, 1/1.01\}, J = \{-0.64, -0.32, ..., -0.01, 0, 0.01, ..., 0.64\}$$

$$\begin{aligned}
\text{ratio}(k) &= \left(\frac{f_3}{f_1} > k\right), k \in I \\
\text{diffMax}(k) &= \left(\frac{f_3 - f_1}{\max(f_1, f_3)} > k\right), k \in J \\
\text{diffMin}(k) &= \left(\frac{f_3 - f_1}{\min(f_1, f_3)} > k\right), k \in J \\
\text{diffAvg}(k) &= \left(\frac{2(f_3 - f_1)}{f_1 + f_3} > k\right), k \in J \\
\text{curving}(k) &= \left(\frac{f_1 + f_3 - 2f_2}{f_2} > k\right), k \in J \\
\text{curvingRatio}(k) &= \left(\frac{f_1 - f_2}{f_2 - f_3} > k\right), k \in J \\
\text{downtrend} &= f_1 >= f_2 >= f_3 \\
\text{uptrend} &= f_3 >= f_2 >= f_1
\end{aligned}$$

the search procedure. As the features take a list of focus values of variable length, their definition are more complex - the reader may refer to them in the appendix section. 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 value themselves. In the latter case, training would be complicated by the varying number of values available at any given point.

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 the same. 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 [10]. 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 $F$ which involves the slope between two points $(x, f_x), (y, f_y)$

will be calculated as

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

## 4.3 Data Collection

A second 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 commonday scenes. For example, there are books, buildings, scenery, plants and more. The scenes are chosen such that the lens positions corresponding to the distance to the subject (peaks in the focus value curve) cover the entire range of lens positions. The distribution of peaks can be seen in Figure 3. Among these scenes, 10 exhibit more than one peak, for a total of 53 peaks.

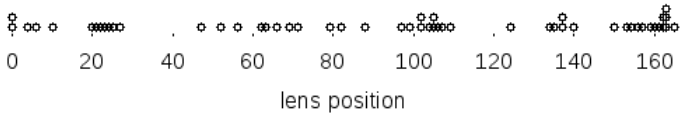



Figure 3: Distribution of the 53 peaks among the 32 benchmark sets over 167 lens positions.

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. The JPEG images are taken directly from the live-view in-between lens movement from one position to the next. 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 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 $[f_{x-2}, f_{x-1}, f_x], x \in [2, 167]$ in each scene where 167 is the total number of lens positions, an instance is created by evaluating the features in $F_\alpha$ 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 "coarse". Alternative classification rules involving the height of a peak were considered, but did not lead to better results. This gives a total of 9218 instances.

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, 1) the focus values at the current lens position and number of steps taken so far are recorded, 2) the features in $F_\beta$ for a new instance are evaluated with the recorded focus values and 3) the instance is then labelled 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 4 coarse steps have been taken and there are no more peaks in the current direction, "continue" otherwise. This gives a total of 22691 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 is balanced such that instances with labels "backtrack", "success" and "continue" appear in a 1:1:3 ratio in the final training set. This introduces 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 which are unlikely to occur is 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 reduces the likelihood that the instance associated with that state will appear in the training data by a factor $\mu$ when the correct label at that step is "continue" and $\gamma$ otherwise. That is, the likelihood of selecting an instance is $p = \mu^x \gamma^{n-x}$ where $x$ is the number of instances labelled "continue" and $n$ is the total number of steps taken during the search up to this point. The instances are then randomly sampled by $p$. This improves 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, is added to the focus value measurements.

### 4.4 Classifier Selection

Using the data collected, the final step is to learn the classifiers. This was done with Weka's [9] J48 implementation of Quinlan's C4.5 decision tree algorithm [16]. 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 2 and Algorithm 3.

Notice that the classifier for determining the initial of the direction 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.

**Algorithm 2:** Decision tree $T_\alpha$ 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**
  $initialDirection \leftarrow far$

## 5 Experimental Evaluation

We perform an empirical evaluation of the effectiveness of our machine-learning based (ml-based) heuristics 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 exhibit 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 simulation 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 noticable, even with 1 megapixel images.

We compare, using the information we could obtain, against the standard hill-climbing method used in [11] [14] where coarse or large steps are taken until a peak is found, after which the direction is reversed and fine steps are taken to

**Algorithm 3:** Decision tree $T_\beta$ 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}, ..., {n, f_n}}
if distance_to_max ≤ 0.03 then
    if ratio_to_max ≤ 0.673 then
        if ratio_min_to_max ≤ 0.378 then
            if current_slope_large ≤ −0.465 then
                result ← success
            if current_slope_large > 0.465 then
                result ← continue
        if ratio_min_to_max > 0.378 then
            result ← continue
    if ratio_to_max > 0.673 then
        result ← continue
if distance_to_max > 0.03 then
    if ratio_min_to_max ≤ 0.352 then
        if downslope_1st_half ≤ 0.833 then
            result ← success
        if downslope_1st_half > 0.833 then
            if ratio_min_to_max ≤ 0.126 then
                result ← success
            if ratio_min_to_max > 0.126 then
                result ← backtrack
    if ratio_min_to_max > 0.352 then
        result ← backtrack
```

narrow down on the peak and the conditions for reversing direction are manually created. Similarly to our heuristics, their goal is to find a nearby peak given any starting lens position.

In the standard hill-climbing method, 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 [11], this happens after the focus value decreases. The precise condition is not specified in [14]. In our comparison, we reverse direction when the focus value drops below 90% of the maximum focus value seen so far, which gives the 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. The standard hill-climbing method was run directly on each of the 32 set, 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 167 starting lens position. 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%.

As shown in Table 3, 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 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 (98.9% vs 86.5%). The limited backtracking support of the standard hill-climbing algorithm also cause 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.

# 6 Low-Light Performance

To assess the low-light performance of our method, we acquired 11 more set of images, taken in dark rooms. For every image in every set, the squared gradient focus measure was again computed. These image sets are not included in our training set. Initially, this led to a very low success rate of 17% for both the standard hill-climbing approach and the ml-based approach (Table 4), which is 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 the example shown in Figure 4, with the squared gradient focus measure, the smallest focus measure is 80% of the value of the largest. We find that if the smallest focus measure needs to be no more than 50-60% of the largest for our algorithm to succeed. The ratio between the focus value at the 1st decile and the largest focus value is shown in Table 4. A decile is used to discount outliers at low focus values, but is unnecessary for large focus values as they correspond to the peak.

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

Table 3: Results of simulating the standard hill-climbing approach and the ml-based approach using leave-one-out cross-validation. Simulations include backlash and noise simulation. Each benchmark was run over all 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.

| | standard hill-climbing | | ml-based | |
|---|---|---|---|---|
| benchmark | accuracy | steps | accuracy | steps |
| 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 |

Table 4: Result of simulating the standard hill-climbing and ml-based approaches on low-light benchmarks, using the squared gradient and gaussian derivative focus measure. Simulation include noise and backtracking. Benchmarks marked with (*) have more than one peak. The column $r$ shows the ratio between small and large focus values.

| | Squared Gradient | | | | | Gaussian Derivative, $\sigma = 2$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | standard | | ml-based | | | standard | | ml-based | |
| benchmark | $r$ | % | steps | % | steps | $r$ | % | steps | % | steps |
| 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 |

(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 conditions, 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, a first-order Gaussian derivative filter is effective as it acts as a low-pass filter which removes the contribution of the noise to the focus measure. Replacing the squared gradient focus measure with the Gaussian focus measure with $\sigma = 1$ helps alleviates this issue, and $\sigma = 2$ solves it completely. The ml-based approach performs particularly well (94% vs 70.3%) compared to the standard hill-climbing approach.

It is worth emphasizing that the effect is noise is adding a *constant* to the focus measures. It does not add significant *variation* to the focus measure curve, which remains fairly smooth, as seen in Figure 4. Almost all autofocus algorithms (e.g., [10] [11] [12]), including ours, require taking the ratio between focus measures in some way, and the ratio becomes closer to 1 as discrimination ratio decreases. Using a Gaussian filter is an effective solution, sufficient for our purposes and simpler to implement than previously proposed solutions for increasing the discrimination ratio [4] [5] [19] .

# 7 Conclusion

The standard technique for finding nearby peaks involve a hill-climbing search where the conditions for state transition (backtrack, reverse direction) are handcrafted. We show that supervised machine learning techniques can be used to construct heuristics to determine those transitions and out-performs previous work on accuracy (98.5% vs 91.5%). In particular, we show that our algorithm

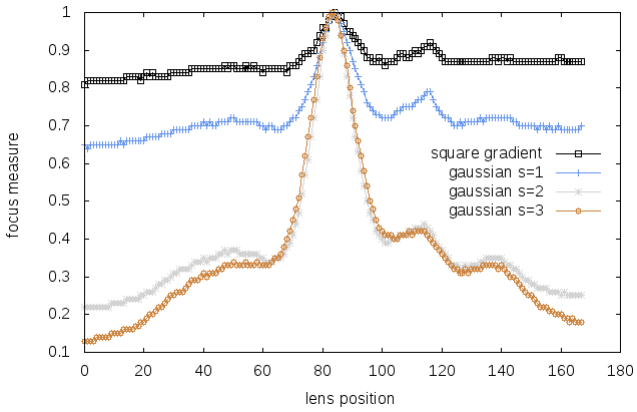



Figure 4: Low light focus measure curve for "backboard2", computed with the squared gradient focus measure and the Gaussian derivative focus measure with $\sigma = 1, 2, 3$. Each curve has been normalized by the maximum value in that curve for easier comparison (but *not* the minimum).

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.

## Acknowledgements

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

# References

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

[2] C.M. Chen, C.M. Hong, and H.C. Chuang. Efficient auto-focus algorithm utilizing discrete difference equation prediction model for digital still cameras. *IEEE Trans. Consum. Electron.*, 52:1135–1143, 2006.

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

[4] M. Gamadia and N. Kehtarnavaz. Low-light auto-focus enhancement for digital and cell-phone camera image pipelines. *IEEE Trans. Consum. Electron.*, 53:249–257, 2007.

[5] M. Gamadia and N. Kehtarnavaz. Enhanced low-light auto-focus system model in digital still and cell-phone cameras. In *IEEE International Conference on Image Processing*, pages 2677–2680, 2009.

[6] M. Gamadia and N. Kehtarnavaz. Real-time implementation of single-shot passive auto focus on dm350 digital camera processor. In *Real-Time Image and Video Processing*, page [REVIEW ME], 2009.

[7] J.-M. Geusebroek, F. Cornelissen, A. W. M. Smeulders, and H. Geerts. Robust autofocusing in microscopy. *Cytometry*, 39:1–9, 2000.

[8] F.C.A. Groen, I.T. Young, and G. Ligthart. A comparison of different focus functions for use in autofocus algorithms. *Cytometry*, 6:81–91, 1985.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[10] J.-W. Han, J.-H. Kim, H.-T. Lee, and S.-J. Ko. A novel training based auto-focus for mobile-phone cameras. *IEEE Trans. Consum. Electron.*, 57:232–238, 2011.

[11] J. He, R. Zhou, and Z. Hong. Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera. *IEEE Trans. Consum. Electron.*, 49:257–262, 2003.

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

[13] S. Y. Lee, Y. Kumar, J. M. Cho, S. W. Lee, and S. W. Kim. Enhanced autofocus algorithm using robust focus measure and fuzzy reasoning. *IEEE Trans. Circuits Syst. Video Tech.*, 18:1237–1246, 2008.

[14] J. Li. Autofocus searching algorithm considering human visual system limitations. *Optical Engineering*, 44:113201–113201–4, 2005.

[15] D. Morgan-Mar and M. R. Arnison. Focus finding using scale invariant patterns. In *Proc. SPIE 8660, Digital Photography IX*, 2013.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[17] M. Rahman and N. Kehtarnavaz. Real-time face-priority auto focus for digital and cell-phone cameras. *IEEE Trans. Consum. Electron.*, 54:1506–1513, 2008.

[18] A. Santos, C. Ortiz de Solórzano, J. J. Vaquero, J. M. Peña, N. Malpica, and F. del Pozo. Evaluation of autofocus functions in molecular cytogenetic analysis. *Journal of Microscopy*, 188:264–272, 1997.

[19] C.H. Shen and H.H. Chen. Robust focus measure for low-contrast images. In *Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on*, pages 69–70, Jan 2006.

[20] M. Subbarao and J.-K. Tyan. Selecting the optimal focus measure for autofocusing and depth-from-focus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:864–870, 1998.

# A Features for state transition heuristics

Input :

$$T_p = \text{total number of positions on the lens}$$

$$s_l = \text{size of large step} = 8$$

$$\text{Records}\{\{0, f_0\}, \{1, f_1\}, \{2, f_2\}, ..., \{n, f_n\}\}$$

$$\bar{i} = \frac{1}{n+1}\sum_{i=0}^{n} i \qquad \bar{f} = \frac{1}{n+1}\sum_{i=0}^{n} f_i \qquad \text{avg}(x, y) = \frac{x+y}{2}$$

Features :

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

$$\text{monotonicity} = \frac{\sum_{i=0}^{n}(i-\bar{i})(f_i-\bar{f})}{\sqrt{\sum_{i=0}^{n}(i-\bar{i})^2 \sum_{i=0}^{n}(f_i-\bar{f})}} \qquad \text{(Spearman's coefficient)}$$

$$\text{absMonotonicity} = |\text{monotonicity}|$$

$$\text{alternationRatio} = |\{(f_i, f_{i+1}, f_{i+2}) \mid (f_1 < f_2) \text{ XOR } (f2 < f3)\}|$$

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

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

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

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

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

$$\text{simpleSlopeUp} = \begin{cases} 0 & \text{simpleSlope} < 0 \\ 1 & \textit{otherwise} \end{cases}$$

$$\text{regressionSlope} = \frac{\bar{i}}{\bar{f}} \cdot \frac{\sum_i (i-\bar{i})(f_i-\bar{f})}{\sum (i-\bar{i})^2} \qquad \text{(Linear least squares, normalized)}$$

$$\text{regressionSlopeUp} = \begin{cases} 0 & \text{regressionSlope} < 0 \\ 1 & \textit{otherwise} \end{cases}$$

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

$$\text{currentSlopeUp} = \begin{cases} 0 & \text{currentSlope} < 0 \\ 1 & \textit{otherwise} \end{cases}$$

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

$$\text{currentSlopeLargeUp} = \begin{cases} 0 & \text{currentSlopeLarge} < 0 \\ 1 & \textit{otherwise} \end{cases}$$

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

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