Computer Assisted Photometry Using Simulated Annealing

G. Groisman and J.R. Parker

Department of Computer Science, University of Calgary, Calgary, AB, T2N 1N4, Canada
parker@cpsc.ucalgary.ca [Internet]

ABSTRACT

The computational technique known as simulated annealing, which originally was used to solve problems in combinatiorial optimization, can be quite effectively used to fit functions of many parameters to a set of data. This paper discusses the application of simulated annealing to the fitting of Moffat functions to crowded field star images. Overlapping star profiles can be fit simultaneously and missed stars can be located based on the χ^2 value of the fit, and the resulting functional model yields excellent photometric results. This method has been integrated into a photometry system which allows the user to select the accuracy of the results at the expense of time needed to perform the calculations.

Key words: Image processing- crowded field photometry- simulated annealing

1. Introduction

Computers have been applied for some time to crowded field photometry, and computer programs for this purpose are in common use [Stetson,Mighell,..]. While many diferent approaches may be taken, measuring the brightness of a star is equivalent to fitting a function to the data and integrating that function over the area of the star. This is not as easy as it sounds given a crowded star field. Where the frames contain so many stars that the their images overlap many stars have to be fitted simultaneously. Optimizing this fit requires of a method able to explore the parameter space involved and not get caught in local minima, since the best result is that corresponding to the global minimum. Because of this the normal downhill methods may not yield the best results.

It is proposed here to apply the technique of *simulated annealing,* [refs] a biased random walk method, to the optimization of this fit. In order to properly demonstrate the usefulness of this technique a computer program for crowded field photometry called *saphot* has been devised. Following Section 2 of this paper, in which simulated annealing is described in more detail, Section 3 will discuss saphot and some of its internal workings. Section 4 will present results generated by saphot and compare these to other existing results. Section 5 will summarize and outline future directions for this work.

2. Simulated Annealing

Many methods have been devised to optimize the fit of a function, in one of more parameters, to a set of data. These methods usually employ a figure of merit that determines how good the fit is, then proceed to optimize this value by changing the parameters. The most straight forward approach is to choose new values of the parameters by changing them in the direction that reduces the value of the figure of merit. Though this would work fine for functions with a single minimum, it has the well-known and unpleasant side effect of somethines trapping the optimization process in local minima. To improve the chances of locating the one global minimum these methods are usually run several times from several different starting points and the best result obtained is taken as the global minimum. Notice that the problem arises from the down hill behavior of the algorithm. It always proceeds in the direction of *steepest descent* even when going up and above a small local hump may eventually find a better minimum.

In 1953 a paper was published by Metropolis [Metropolis53] that proposed a different approach to the optimization problem. The idea was to observe how nature optimizes the placement of atoms in a lattice when a metal sample is slowly cooled, a method known as *annealing.* When this happens the atoms find the lowest energy configuration possible, yielding a very regular structure. On the other hand, if the sample is cooled quickly (quenching) then the result is a very irregular structure far from the optimum energy state possible.

To understand the role the temperature plays it is necessary to think of the process at play, and this is described by quantum mechanics. At any given temperature T the probability of a system of having energy E is given by

$$Prob(E) = e^{\frac{-E}{kT}}$$

where k is the Boltzmann constant. This is the so-called Boltzmann probability distribution and shows that the probability of the system being in a state other than the optimum is finite. This is what allows a system to climb out of a local minimum; the annealing method sometimes allows the system to go uphill, unlike the steepest descent methods. But as the temperature gets smaller the chance of an uphill climb diminishes too. Slow cooling allows the system to explore some uphill climbs and go into other areas, whereas quenching will make the chance of leaving a local minimum too small too fast.

This process can be simulated easily with a computer algorithm. We first need a way to express the probability P of the system changing energy by an amount ΔE defined as $E_2 - E_1$, the difference between the new energy value and the old one. This change is negative for a downhill change (E_2, the new energy is smaller that the old one) and positive otherwise. We now define the probability of accepting the change as

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E \le 0 \\ e^{-\Delta E / kT} & \text{if } \Delta E > 0 \end{cases}$$

This is because we always want to take downhill steps ($\Delta E \le 0$) but the uphill ones only based on the Boltzmann probability distribution. This is known as the Metropolis criterion and is illustrated in Figure 1. The process of deciding based on this probability can be simulated in a computer [Bohachevsky86] by comparing the value of a random number between 0 and 1 with the value of $P(\Delta E)$.

$$Random \le P(\Delta E) \rightarrow accept$$

$$Random > P(\Delta E) \rightarrow reject$$

A decision has to be made as to how the temperature will be lowered: this is known as an *annealing schedule* [Basu90,Kirkpatrick83,Kirkpatrick84]. It can be as simple as reducing it by a constant fraction every so many parameter adjustments or using a more complicated method.

The figure of merit usually employed to determine the goodness of the fit is the χ^2 value, which gets smaller as the fit improves. What remains to be decided are the values of k and T. As we shall see later these are related to the function being optimized and can easily be decided based on information available during the fitting process.

To illustrate these concepts we recreate an example introduced by Bohachevsky [Bohachevsky86]. Here we want to minimize the function

$$f(x,y) = x^2 + 2y^2 - .3cos(3\pi x) - .4cos(4\pi y) + .7$$

As can be seen in Figure 2 this function has many local minima and one well defined global minimum at (0,0). Any downhill method could get caught in one of the local valleys and never reach (0,0). To test simulated annealing we start at (1,1) and let the algorithm proceed.

To illustrate the main points of simulated annealing we use a very simple implementation. Knowing the function being fitted we notice that in order to detect the global minimum (or any minimum for that matter) we need a step size in x and y smaller that about 0.2 since this is about the size of each minimum. Our method then begins at (1,1) and takes a step in the x direction of size $Rand\ step_x$, where *Rand* is a random number between -1 and 1, and $step_x$ is the maximum step size allowed in the x direction. It then computes the value of the function at this new point and applies the Metropolis criterion to decide wether to move there or not. Then next step is taken in the y direction in the same manner, and the whole process is repeated until no more steps are accepted by the algorithm.

Figure 3 shows the path followed by the algorithm. The points indicate the steps the algorithm followed. Dead end paths (those not leading to a point, marked by dotted lines in the figure) correspond to the uphill moves the algorithm tried by decided to discard. Still some points in the path correspond to uphill climbs the algorithm chose to follow. This is how it was able to escape the valleys in (.6,.5) and (.6,0), for example, and proceed on to the global minimum. For this simple test the temperature was lowered by 0.95 every 30 steps, and the value of k was also set to 1. The plot clearly illustrates why simulated annealing is considered a biased random walk. Overall the path is biased towards lower values, and this is why it converges to the global minimum. But from one place to the next its behavior is quite random,

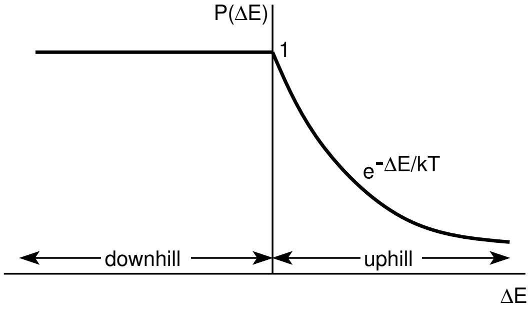

Figure 1 - Boltzmann probability distribution.

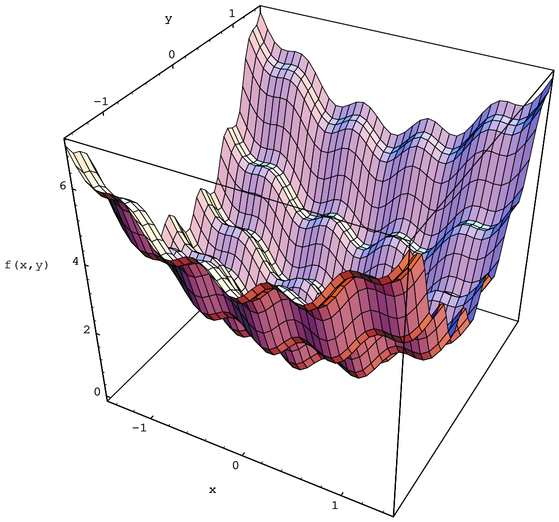

Figure 2 - SA Test function exhibiting many local minima and one good global one.

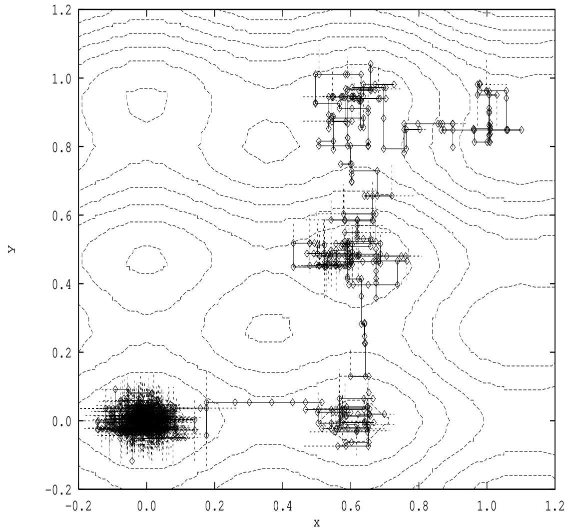

Figure 3 - Test of SA showing all values that the method tested, starting at (1,1) and locating the global minimum at (0,0).

making it almost impossible to tell wether it will go uphill or not.

An improved algorithm for performing simulated annealing optimization is outlined in Figure 4. Several steps are involved in the process. The kernel of the method consists of a loop involving a series of parameter adjustments, one for each parameter being adjusted at constantant temperature. At each adjustment the Metropolis algorithm is consulted to decide whether to accept the new value or not. To improve performance the optimum value obtained in every cycle is recorded. After a fixed number of cycles have been performed the algorithm consults how many adjustments were accepted for every parameter. The intention here is to keep the ratio of acceptance to rejections to about 1/2. To do this the step sizes are now adjusted and the whole process repeated.

Several sets of step adjustments are performed, after which the temperature is reduced. At this point the whole process is restarted, but with the parameters set to the optimum value found in the previous iterations. This as a whole is repeated until the stopping criterion is achieved.

The parameters and their step sizes will be initialized to values chosen by the calling routine. This may come from previous fits, guesses, etc. Initialization of the temperature can be done by finding the average change in energy after trying some random steps in every parameter, as stated before. The initial temperature is chosen to keep

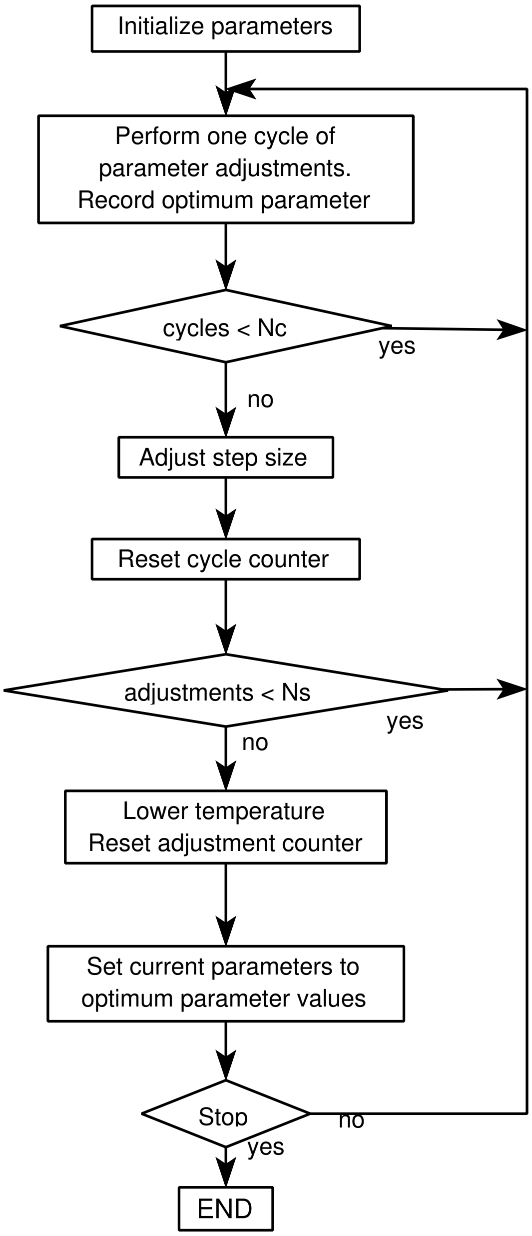

Figure 4 - Flowchart of Simulated Annealing algorithm. (After [Corona])

the ecceptance to rejection ratio to about 50%. This however is only true if the function being optimized is already close to a minimum. usually the starting guesses are in a region where the function is still changing very fast. As a consequence the average change is very large and so is the value of the initial temperature chosen. The problem is that in a few steps the process will most likely find the plateau where the minima are, but the temperature will now be too large. This will cause the parameter steps to increase and the process to wonder far from this region. This is equivalent to having chosen parameter step sizes larger than we have already. Eventually the temperature will lower to a value that allows the parameters to settle to the minima. This can be seen in Figure 5. Though this would be a very desirable behavior if not much was known about the minima location, it

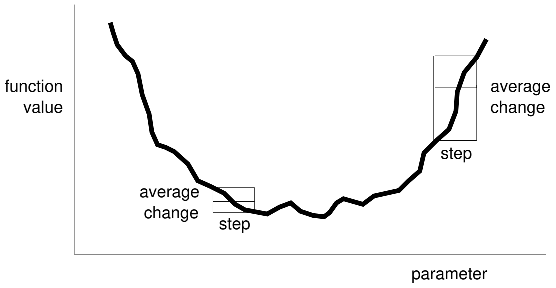

Figure 5 - Average function change for a given step size.

turns out to be a waste of computer time when the general location of the parameters of the object is clearly known. The solution is to allow the parameters to wander downhill only for a few times, until they reach the above mentioned plateau. If the average function change is now computed it will better reflect the changes the annealing process will encounter, and yield a more efficient initial temperature.

The size of the average step taken in any direction in variable space is a difficult value to determine. In the simulated annealing example shown earlier a lot was known about the function being optimized, so it was easy to determine appropriate step sizes for the variables.

Recognizing this an early modification was made to the basic algorithm based on the common method of paging used in virtual memory systems. A counter was kept for each variable that kept track of how many times a fitting step in that variable's direction was rejected. It can be assumed that too many rejections in a certain direction indicate too many large uphill climbs that suggest too large a step size. Once the counter exceeded a certain value no more changes were attempted in that direction until the step size was changed to half its current value later on. By the same token, too many acceptances mean that the step size is too small, and the step size is increased. Implementing this resulted in faster execution time.

Yet a more desirable method would adjust the step size as the process progresses in a manner proportional to how bad the fitting is doing in that direction. Too small a step in the early stages of the fitting may slow it down needlessly. By the same token, to large a step in a later stage may miss the minimum altogether. Another method uses the ratio of accepted to rejected steps as an indication of wether the step size is appropriate [Corana87]. As the algorithm progresses it keeps track of how many times each parameter was adjusted. This includes all the downhill moves (always accepted) and those uphill ones that were taken. It is desirable to keep this to about 50% of the parameter adjustment tries. To do this the new step size must be changed accordingly.

Once again it is assumed that too many acceptances mean the step is too small and must be made larger. On the other hand too few acceptances may mean a step size too large. The new value of the step size for the i^{th} parameter, S_i, is computed based on the number of adjustment tries, N_{adj} and those accepted, N_{acc} (after A. Corana [Corana87])

$$S_i = S_i\left(1 + 2\frac{N_{acc}/N_{adj} - 0.6}{0.4}\right) \quad \text{if} N_{acc} > 0.6 N_{adj}$$

$$S_i = S_i/\left(1 + 2\frac{0.4 - N_{acc}/N_{adj}}{0.4}\right) \quad \text{if} N_{acc} < 0.4 N_{adj}$$

$$S_i = S_i \qquad \textit{otherwise}$$

This can be see graphically in Figure 6. Notice that the factor 2 simply means that the step will grow as much as 3 times or shrink by the same amount. This value can be changed resulting in different speeds of convergence. It must be noticed however that too fast a contraction could make the algorithm stop too soon on a local minimum, while too slow a contraction could waste time.

It is necessary to establish a way of deciding when the process must be stopped. For real data a fit is rarely exact and hence one cannot expect the value of χ^2 to reach 1. A more reasonable way of detecting a solution is to look at how the values of χ^2 progress. When the fractional change between the previous and the current values and the fractional change between the optimum and the current value both fall below a certain value ε it can be said that the algorithm has

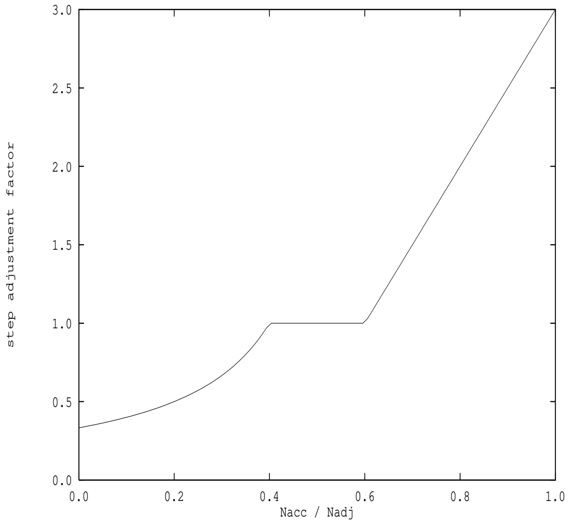

Figure 6 - Step Adjustment Function.

converged on a solution.

For those cases when the accuracy of the expected result can be known the step sizes offer an ideal method. The algorithm can be run until all step sizes have became smaller then the desired accuracy. Any further calculation will not add anything to the result, as further improvements are smaller that the uncertainty.

3. A Computerized Photometry Package

Having completed the simulated annealing module, it remained to design a new photometry package that could make use of it. Once the image is read the background has to be estimated and stars located. These stars are then grouped and fitted with a function and for this a figure of merit is needed to qualify the goodness of the fit. The optimization algorithm then proceeds to explore the different parameter values in search of that combination that gives the best fit.

To estimate the background properly some star free regions must be located. This could be done automatically, but our system allows the user to identify background regions interactively using a 'mouse'. Notice that these must not be totally black areas; Stetson [Stetson87] indicates that the background estimated will be used to remove from the photometry the type of background that would probably be there if the star was not present. This is not the darkest background but rather the combination of unresolved stars and other noise present throughout the image, so the user should be selecting areas without stars that could be measured while not intentionally avoiding nebulosities and other background noise. The system then fits a plane (2-D) to these areas, which gives the average value at any point in the image. The standard deviation (σ) of the selected data from this fit is also computed and is later used as the background noise value.

Given this simple numerical model of the background illumination it is possible to locate the positions of the stars. Any pixel whose value is more than $n\sigma$ above the local value of the sky fit will be considered a candidate. Normally many pixels within a star will result in candidates in which case a local maximum is searched in a 3 x 3 pixel array. This pixel is then taken to be the closest point to the center of a star. Now, by interpolating between the pixel values near the central pixel, it is possible to improve the star peak location to within a fraction of a pixel. We used a bilinear interpolation using a total of 9 pixels, but any interpolation method could be used.

In crowded fields it is important to locate those stars whose PSF overlap so they can be fitted together. Grouping is done from a list of all located stars and their positions. One star is selected and then those which lie close enough are located and removed from the list. The process is now recursively repeated for all these stars, until no other overlapping star can be found. At this point a new star is selected from the list and the whole process repeated until no more stars remain in the list. The resulting grouping can be seen in

Figure 7 for a typical image.

These groups of stars must now be fitted simultaneously. In that way the overlapping regions can be fitted by the appropriate combination of light from all contributing stars. To perform this fit of multiple functions to multiple stars the function parameters for position and brightness are first approximated from the interpolation. It is then necessary to compute the χ^2 value to measure the goodness of fit. In fact, we compute the *reduced* χ^2, called χ^2_υ, which is done by dividing each difference in the normal χ^2 computation by the expected uncertainty at that pixel. This gives a value that approaches 1 as the fit improves.

If the final value of the reduced χ^2_υ is much larger than 1 we can conclude that either the fit is bad or the noise was underestimated. If it is much smaller than 1 then the fit is too good to be true, so the errors were overestimated. However, if a high degree of accuracy is to be achieved then all noise sources must be identified and their values computed.

The first source of noise is the background, usually called the sky, and this has already been accounted for. Indeed, a value has been computed for the standard deviation of the background, which will be called σ_{sky}, and the pixel error will include this uncertainty. Photon noise is another source of noise. Photometry is a counting process and as such it is subject to Poisson noise. This means that there is an uncertainty associated with the number of photons detected, and the RMS of this uncertainty is equal to the square root of the count itself. Using electrons as units this can be expressed as $\sigma_{Poisson} = \sqrt{e^-}$. But we know that given a CCD with gain g there are g electrons per DN, so the uncertainty can be expressed in DN units as $\sigma_{Poisson} = \sqrt{\frac{DN}{g}}$. The uncertainty of any one pixel can be expressed as the sum of these two sources of error:

$$\sigma_{pixel}{}^2 = \sigma_{sky}{}^2 + \frac{DN}{g}$$

The photometry system, considered as a piece of software, is divided into three separate programs. Data is kept in separate files that these programs read. This gives the user flexibility to change working parameters and experiment with different settings. A data flow diagram is shown if Figure 8. The image is kept in a file called *fits*, which is stored using the FITS format. The FITS format was chosen as it is a standard in the astronomical community.

The first part of the system, named FOTO, reads the image and displays it. To do proper photometry the gain of the CCD in electrons per DN and the readout noise also in DN are needed, these values must be provided by the user when prompted. Next the program estimates the value of the sky. This is done from a series of small square patches the user selects with the mouse. Once the selection is finished a flat surface is fitted

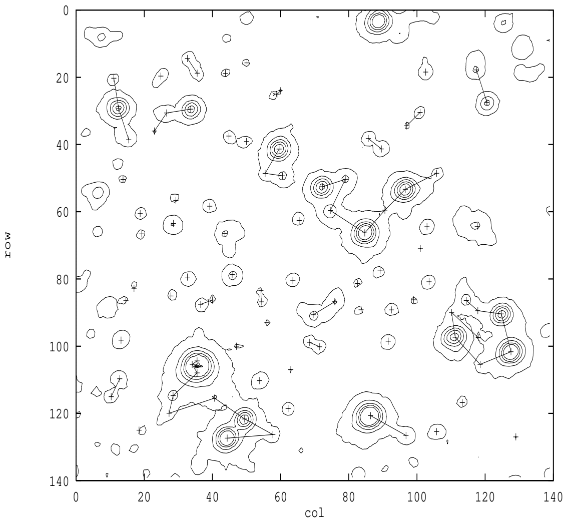

Figure 7 - Star groups computed by the program for a star image (NGC 6101)

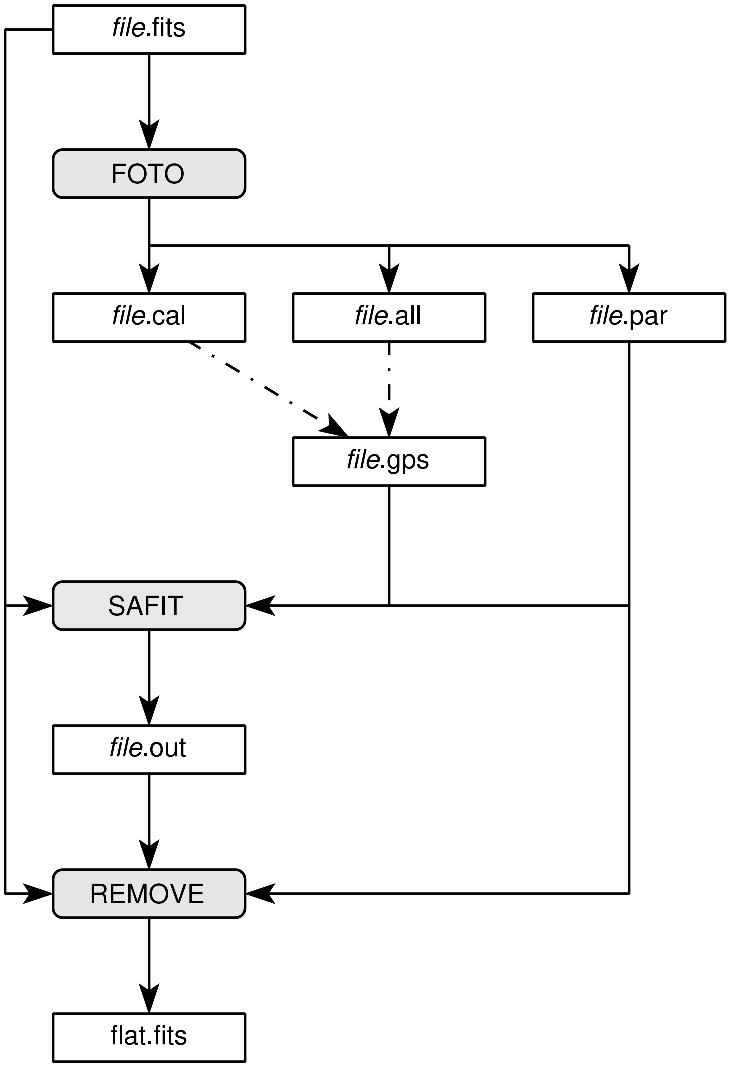

Figure 8 - Flow diagram of the photometry system.

and the result displayed. These values are written to file *par*.

Finally the user selects a set of calibration stars also using the mouse. These should be clean isolated stars the the program can later use to find the best values of ρ and β. FOTO also fits simple Gaussians to these stars to come up with the best fitting value of σ for the stars in the frame. The calibration star parameters (position and approximate brightness) are written to file *cal*.

Finally the program identifies all stars in the image and proceeds to group them so that stars that are close in the image will be reduced together. These star groups are written to file *all*.

The second program in the system, called SAFIT, uses simulated annealing to fit Moffat functions to the stars. Using the data generated by FOTO, groups of stars are fit simultaneously and the resulting parameters are written to the file *out*.

The resulting fits can be checked by running REMOVE. It reads the fits from *out* and the parameters used from *par*. It also reads the original image from *fits* and proceeds to remove all the stars by subtracting the fitted function from it. The resulting image is written to a file called file *flat.fits*. This file can be inspected to determine the quality of the fit, and whether any stars were missed in the first run. The coordinates of these stars can be used in a new iteration round of FOTO and REMOVE.

Testing on synthetic images was done to explore the ability of the algorithm to solve the different problems crowded-filed photometry imposes in a controlled environment. The main issues to test were

- The ability to compute the position of the star, regardless of its position within the pixel array (centered, off center, etc.).
- The ability to compute the brightness of each star. In the case of multiple fits, whether the results for each star were not significantly affected by the other intruding star images.
- The ability to perform under different noise conditions. Signal dependent noise gives larger uncertainty to values near the core of the star image. This should not seriously affect the outcome.
- The ability to perform with different PSF models. Some models may have parameters that are not completely independent of each other. This should not prevent the algorithm from converging.

Several sets of test data were created and different types of noise added to evaluate performance under conditions that went from ideal to realistic.

In a test image a star is represented by a two dimensional peak. A pair of *x* and *y* coordinates determine the center of the star, and a value of *I* in DN its brightness. The PSF of the image is controlled through the shape of the star function. Two types of functions were used to generate simulated stars: Gaussians with $\sigma=1$ and Moffats with $\rho=2.0$ and $\beta=3.0$.

The images were subjected to three types of noise. No noise images allowed a test under ideal image quality. Gaussian noise and Poisson noise made the images more realistic.

The results from the fits are shown in the following tables. Here the value obtained by the fit is showed in each respective column. Next to it, in parenthesis is the step size when the fit stopped as it applies to the last digit shown in the parameter value. For example 12.345(4) means the parameter has a value of 12.345 and the step adjustment was 0.004 when the fitting was stopped. A zero in the parenthesis indicates that the step size was even smaller than the indicated accuracy of the parameter value. The final value of χ^2_υ is also shown.

The first set of simulated test images consists of a single star, either Moffat or Gaussian. There are four such images having known position and intensity values:

Image	I	X	Y
1	2000	5.0	5.0
2	2000	5.3	5.0
3	2000	5.0	5.6
4	2000	5.3	5.6

Table 1 shows the fits to the Gaussian stars in the presence of three different noise backgrounds: none (ideal), Gaussian noise and Poisson noise. As expected the results are very close to their original values. However, one interesting result is seen in line 9 where the value of σ is -1.00(1), a negative value. This is because though the fitted parameter is σ the fitting function actually uses σ^2 so the sign isn't relevant. To avoid this one could fit σ^2 itself, or keep the step size in the σ direction smaller.

Table 1: Fits to single Gaussian stars.

Image	Noise	χ	σ	x	y	I
1	None	0.1148	1.001(0)	5.000(1)	5.000(1)	1991.(1)
2	None	0.1433	1.002(0)	5.299(1)	5.000(1)	1991.(1)
3	None	0.1332	1.001(0)	5.000(1)	5.600(1)	1991.(2)
4	None	0.1465	1.001(1)	5.300(0)	5.600(1)	1991.(3)
1	Gauss	0.9702	1.005(6)	5.00(1)	4.995(8)	1987.(30)
2	Gauss	0.9508	1.003(6)	5.31(3)	5.00(2)	1981.(39)
3	Gauss	0.9384	1.005(2)	4.997(8)	5.599(7)	1986.(11)
4	Gauss	0.9318	0.99(2)	5.303(9)	5.598(3)	1995.(11)
1	Poisson	0.3495	-1.00(1)	4.98(3)	5.00(3)	1988.(41)
2	Poisson	0.5558	0.994(5)	5.29(4)	4.99(4)	2006.(89)
3	Poisson	0.5820	1.002(7)	5.03(5)	5.62(2)	1949.(30)
4	Poisson	0.4645	0.99(1)	5.29(3)	5.59(3)	2030.(37)

Table 2 shows results of fitting to the Moffat stars, again in the presence of the three noise types. The results are still very close to the original values in spite of the fact that the parameters to the Moffat function are not independent of one another. The annealing routines appear to handle this situation quite well.

Table 2: Fits to single Moffat stars.

Image	Noise	χ	ρ	B	x	y	I
1	None	0.1541	2.034(1)	3.061(1)	5.000(0)	5.000(1)	1984.(1)
2	None	0.1313	2.029(1)	3.051(1)	5.300(2)	5.000(1)	1985.5(9)
3	None	0.1745	2.035(1)	3.063(1)	5.000(1)	5.600(1)	1985.1(8)
4	None	0.1776	2.028(2)	3.049(2)	5.300(1)	5.600(3)	1985.7(4)
1	Gauss	1.1877	2.09(1)	3.1(1)	5.00(1)	5.00(1)	1972.(21)
2	Gauss	0.9906	1.99(1)	2.95(2)	5.30(1)	5.00(1)	1975.(9)
3	Gauss	0.9660	1.98(2)	2.94(3)	5.006(4)	5.597(7)	1980.(13)
4	Gauss	1.0524	1.87(1)	2.75(2)	5.30(1)	5.605(8)	2015.(15)
1	Poisson	0.7599	1.98(3)	3.01(5)	5.03(2)	5.01(3)	2003.(47)
2	Poisson	0.8580	2.13(1)	3.25(3)	5.32(5)	5.02(4)	1934.(67)
3	Poisson	0.9607	2.17(4)	3.2(1)	4.99(4)	5.61(8)	1886.(85)
4	Poisson	0.9574	2.16(4)	3.35(8)	5.26(3)	5.58(3)	1992.(197)

The next set of simulated test images contain pairs of stars; the first four have stars of equal intensity, and the last four have stars of differing intensity. As before, one star is centered while the other is moved about in the field. The positions and intensities of the simulated star pairs are:

Image	I	X	Y	Image	I	X	Y
1	1000	3.0	3.0	5	2000	3.0	3.0
	1000	6.0	6.0		1000	6.0	6.0
2	1000	3.0	3.0	6	2000	3.0	3.0
	1000	6.2	6.0		1000	6.2	6.0
3	1000	3.0	3.0	7	2000	3.0	3.0
	1000	6.0	6.3		1000	6.0	6.3
4	1000	3.0	3.0	8	2000	3.0	5.0
	1000	6.1	6.3		1000	6.1	6.3

Once again the no noise images are fitted very well as can be seen from Tables 3 and 4. The noisy images also yield very good results.

Table 3: Fits to pairs of Gaussian stars.

Image	Noise	χ	σ	x	y	I
1	None	0.0980	1.001(0)	3.000(0)	3.000(0)	995.7(6)
				6.000(1)	6.000(1)	996.2(6)
2	None	0.1293	1.001(0)	3.001(1)	3.001(1)	996.0(3)
				6.200(2)	6.000(1)	996.2(6)
3	None	0.1323	1.001(0)	3.000(0)	3.000(1)	995.6(6)
				6.000(2)	6.300(2)	995.2(8)
4	None	0.1185	1.001(0)	3.000(1)	3.000(1)	995.5(5)
				6.100(1)	6.300(1)	995.3(9)
5	None	0.1034	1.001(0)	3.000(0)	3.000(1)	1991.9(7)
				5.999(1)	5.999(1)	995.3(1)
6	None	0.1521	1.002(0)	3.000(1)	3.000(0)	1991.5(2)
				6.199(1)	6.000(4)	995.3(8)
7	None	0.1557	1.001(0)	3.000(1)	3.000(0)	1991.8(8)
				6.000(1)	6.300(1)	995.5(3)
8	None	0.1538	1.001(0)	3.000(0)	3.000(1)	1992.1(6)
				6.100(0)	6.300(1)	995.9(4)
1	Gauss	0.9659	0.998(8)	3.01(4)	3.00(3)	1005.(18)
				6.01(2)	5.99(2)	1002.(18)
2	Gauss	0.9492	1.003(3)	2.99(1)	2.99(1)	996.(8)
				6.21(2)	6.00(2)	995.(7)
3	Gauss	0.9964	0.997(9)	3.00(2)	3.00(2)	1007.(7)
				6.01(2)	6.29(2)	1001.(7)
4	Gauss	0.9615	1.00(1)	3.00(2)	3.01(1)	1001.(10)
				6.10(3)	6.30(4)	996.(12)
5	Gauss	0.9434	1.000(6)	2.997(8)	2.999(9)	2001.(15)
				6.00(3)	5.99(2)	1002.(25)
6	Gauss	0.9509	1.001(4)	3.00(2)	3.004(3)	1997.(16)
				6.21(1)	5.99(3)	995.(12)
7	Gauss	1.0129	1.000(4)	2.998(6)	3.00(2)	2002.(17)
				6.00(1)	6.29(2)	992.(7)
8	Gauss	1.0450	1.003(5)	2.996(6)	2.99(1)	1989.(8)
				6.12(4)	6.29(2)	999.(8)
1	Poisson	1.2691	1.01(2)	3.01(4)	3.02(8)	990.(73)
				5.9(1)	6.0(1)	994.(121)
2	Poisson	0.9080	0.99(1)	3.0(1)	3.03(4)	998.(22)
				6.1(1)	6.00(4)	1011.(39)
3	Poisson	0.9021	1.002(7)	2.99(3)	3.02(2)	959.(62)
				5.99(2)	6.30(4)	970.(31)
4	Poisson	0.8907	0.999(9)	2.96(8)	2.99(4)	982.(29)
				6.10(4)	6.27(6)	998.(34)
5	Poisson	1.0573	0.988(7)	2.98(2)	3.00(2)	1997.(58)
				6.01(2)	5.96(2)	1045.(20)
6	Poisson	0.8605	0.99(2)	3.00(6)	3.00(4)	2035.(75)
				6.20(3)	6.00(6)	999.(47)
7	Poisson	0.9845	0.99(2)	2.98(8)	2.98(4)	2040.(93)
				5.98(6)	6.2(1)	1048.(53)
8	Poisson	0.8297	0.98(1)	2.97(2)	2.99(4)	2064.(102)
				6.08(4)	6.29(5)	1035.(51)

Table 4: Fits to pairs of Moffat stars.

Image	Noise	χ	ρ	B	x	y	I
1	None	0.1695	2.039(1)	3.071(2)	3.000(1)	3.000(1)	992.(1)
					6.000(1)	6.000(1)	991.4(6)
2	None	0.1744	2.017(1)	3.026(1)	2.999(3)	3.000(4)	993.(1)
					6.200(1)	5.999(1)	993.(1)
3	None	0.1111	2.033(1)	3.059(1)	3.000(1)	3.000(1)	992.1(4)
					6.000(1)	6.300(1)	992.3(9)
4	None	0.9861	1.969(4)	2.933(3)	3.000(8)	2.999(6)	996.(2)
					6.100(3)	6.299(3)	998.(1)
5	None	0.1363	2.031(1)	3.054(1)	3.000(0)	3.000(1)	1985.(1)
					5.999(3)	6.000(4)	992.4(6)
6	None	0.1304	2.027(1)	3.047(1)	3.000(1)	3.000(1)	1985.7(7)
					6.200(2)	5.999(1)	992.9(9)
7	None	1.1677	1.982(1)	2.954(2)	3.000(1)	3.000(1)	1992.(2)
					6.001(4)	6.300(4)	994.(1)
8	None	0.1927	2.035(1)	3.063(2)	3.000(2)	3.000(1)	1983.(1)
					6.100(1)	6.300(2)	991.(2)
1	Gauss	1.0560	2.01(1)	3.00(2)	2.98(1)	3.00(1)	990.(16)
					6.01(3)	6.00(3)	995.(13)
2	Gauss	1.1943	2.11(2)	3.20(2)	3.00(1)	2.99(3)	986.(10)
					6.21(3)	5.99(3)	986.(23)
3	Gauss	1.3040	2.16(2)	3.35(3)	2.98(4)	3.00(2)	981.(17)
					6.00(3)	6.28(2)	987.(21)
4	Gauss	1.0110	2.003(9)	3.00(3)	2.99(1)	3.00(2)	1002.(5)
					6.11(1)	6.30(1)	994.(11)
5	Gauss	1.2173	2.09(1)	3.17(1)	2.99(1)	3.00(1)	1970.(12)
					6.01(1)	6.00(2)	988.(15)
6	Gauss	1.0171	2.018(8)	3.03(1)	2.99(1)	2.99(1)	1995.(9)
					6.20(1)	5.99(2)	998.(11)
7	Gauss	1.0912	2.04(1)	3.09(2)	2.99(1)	3.00(1)	1994.(19)
					5.99(1)	6.30(1)	997.(30)
8	Gauss	1.0344	2.007(9)	3.00(1)	2.99(1)	3.00(0)	1999.(9)
					6.10(1)	6.28(2)	999.(9)
1	Poisson	1.2995	2.20(2)	3.39(9)	2.9(1)	2.98(1)	950.(95)
					6.03(7)	6.0(2)	997.(82)
2	Poisson	1.2256	2.28(4)	3.52(7)	2.98(4)	2.99(4)	962.(30)
					6.2(1)	5.96(8)	946.(74)
3	Poisson	1.1956	2.11(2)	3.34(6)	3.0(1)	2.9(1)	1041.(27)
					5.98(6)	6.29(4)	1036.(77)
4	Poisson	1.1220	2.17(2)	3.40(9)	3.04(7)	3.0(1)	1012.(20)
					6.0(1)	6.28(3)	1011.(26)
5	Poisson	1.3402	2.01(2)	3.11(7)	3.00(4)	3.01(8)	2062.(105)
					6.02(5)	5.9(1)	1022.(75)
6	Poisson	1.3922	2.04(5)	3.1(1)	2.99(7)	3.00(3)	2000.(116)
					6.2(1)	5.98(7)	1018.(57)
7	Poisson	1.3613	2.05(3)	3.2(2)	2.98(7)	3.01(4)	2065.(99)
					6.0(2)	6.2(1)	987.(36)
8	Poisson	1.4371	1.96(3)	3.00(3)	3.00(5)	3.03(3)	2104.(64)
					6.10(4)	6.27(7)	992.(129)

To test the ability of SAPHOT to resolve stars that are very close together (or blended), a sequence of simulated images was used, having separations of 3, 2.8, 2.5, 2.1, 1.8 and 1.4 pixels. The stars had different intensities and only noise of the poisson type was tried with these images. As seen before, if the method performs well under this type of noise it can be expected to do as well or better under less noisy situations. The parameters used to generate these test images are:

Image	I	X	Y	Image	I	X	Y
1	2000	3.0	3.0	4	2000	3.0	3.0
	1000	6.0	6.0		1000	5.1	5.1
2	2000	3.0	3.0	5	2000	3.0	3.0
	1000	5.8	5.8		1000	4.8	4.8
3	2000	3.0	3.0	6	2000	3.0	3.0
	1000	5.5	5.5		1000	4.4	4.4

About the hardest type of fit can be seen in Tables 5 and 6. The results show that as the blending becomes more serious the ability to fit the Moffat shape parameters diminishes. This is exactly the reason that these parameters must be estimated from fits to clean stars and then kept constant. On the other hand, even at a distance of only 1.4 pixels annealing does a good job of estimating the coordinates of both stars centers and intensities.

Table 5: Fits to pairs of Gaussian stars at varying distances + Poisson noise

Image	χ	σ	x	y	I
1	1.0573	0.988(7)	2.98(2)	3.00(2)	1997.(58)
			6.11(2)	5.96(2)	1045.(20)
2	0.9139	0.98(1)	3.00(3)	2.99(1)	2052.(63)
			5.78(5)	5.78(3)	1032.(49)
3	0.7489	0.993(7)	2.97(2)	3.00(1)	2033.(79)
			5.49(4)	5.48(4)	1039.(24)
4	0.7081	0.99(1)	2.99(5)	3.01(3)	2050.(130)
			5.11(6)	5.10(3)	990.(63)
5	0.5939	0.997(8)	2.97(2)	3.00(2)	1985.(92)
			4.79(6)	4.79(2)	1043.(43)
6	0.6297	0.994(8)	2.95(1)	2.98(2)	2004.(40)
			4.38(2)	4.35(3)	1026.(32)

Table 6: Fits to pairs of Moffat stars at varying distances + Poisson noise

Image	χ	ρ	B	x	y	I
1	1.3402	2.01(2)	3.11(7)	3.00(4)	3.01(8)	2062.(105)
				6.02(5)	5.9(1)	1022.(75)
2	1.2188	2.02(3)	3.07(3)	3.00(5)	3.0(1)	2051.(87)
				5.8(1)	5.75(6)	972.(68)
3	1.1473	2.03(1)	3.14(5)	3.00(3)	3.02(3)	2016.(53)
				5.50(8)	5.46(2)	1029.(88)
4	1.3312	2.23(2)	3.41(7)	2.9(1)	2.99(5)	1928.(49)
				5.09(7)	5.08(9)	920.(59)
5	1.3333	2.19(4)	3.38(3)	2.97(2)	3.00(7)	1978.(129)
				4.84(9)	4.7(2)	941.(89)
6	1.2627	2.13(3)	3.26(5)	2.98(4)	2.96(2)	1950.(119)
				4.33(6)	4.37(4)	1058.(47)

Tests were also performed to verify the ability to fit stars by iterating when a star is missed in a first pass, and later added to the fitting list. An image was created with three stars in it: two were large and separated enough to make sure they would be detected in the first pass, and a third, smaller star was placed between these two so it would not be possible to see it in a first pass. The parameters for this image are:

I	X	Y
2000	5.3	5.1
5000	9.1	8.5
800	7.2	6.7

Figure3StarResiduals.a shows both the image and a three dimensional representation of the data. Observation of the original image seems to indicate only two stars. When two stars are fitted the median filtered residuals look like those in Fig3StarResiduals.b. Based on this a third star is added and all three fitted again. The resulting median filtered residuals can be seen in Fig3StarResiduals.c. The fitted parameters for both the 2 and 3 star fits are:

	I	X	Y
2 stars:			
	2271.5	5.47	5.29
	5332.6	9.04	8.41
3 stars:			
	2000.3	5.29	5.11
	5003.6	9.12	8.49
	795.3	7.20	6.72

Notice that in the first attempt the lack of the third star pushes the centers of the two fitted stars closer, to compensate for the missing star in the center. Their brightness are also larger than expected to account for the extra DN. The residuals from the fit clearly show the poorly fitted data where that star should be. When a third star is added all parameters converge to proper values.

4. Results from SAPHOT

Synthetic images provide control situations were the performance of the algorithm can be tested. Still the final test is the application to real data. For this purpose several CCD images of the globular clusters NGC6397, NGC6101 and 47 Tucana were obtained and reduced.

NGC 6397: In this example a pair of B and V observations of the cluster NGC6397 was reduced by the SAPHOT software. First the frames were reduced selecting a few calibration stars, and fitted with $\varepsilon = 0.1$. This meant a relatively fast reduction, but the quality could be expected to be marginal. The resulting diagram can be seen in Figure 10a. Though the main sequence is clearly visible, the spread of the data points is too large. This is mostly due to a large value of ε which translates into the annealing process stopping too soon. A second run was performed with a value $\varepsilon = 0.05$. This increased the execution time significantly (about a factor of 4) but gave much better results as can be seen in Figure 10b. This illustrates a important feature of SAPHOT: the user may choose the accuracy level of the fit, and make the time-accuracy trade-off personally instead of having it dictated. The more strict parameters forced the fitting process to continue until a better fit (hence a lower value for χ^2_υ) was found, and the points now gather closer to the main sequence.

As it can be expected the quality of the results is not only a reflection of the code, but also of the operator. It was decided to go back to the begining and select new calibration more carefuly. These were then fitted for all parameters, shape parameters included. The resulting fits were examined and averages for ρ and β computed. New fits were performed with these new values and the smaller value of ε. The results can be seen in Figure 10c. The data points now gather very close to the main sequence.

Data of a previously published C-M diagram for NGC6397 was also obtained from papers by Alcaino [Alcaino]. This made it possible to campare the final C-M diagram with published data. Alcaino's photometry had been calibrated by observing some stars of known brightness. To make both diagrams match several stars were identified in Alcaino's work and in the present work and the correlation computed. The resulting values and fit can be seen in Figure 11. The best line fits correspond to the equations

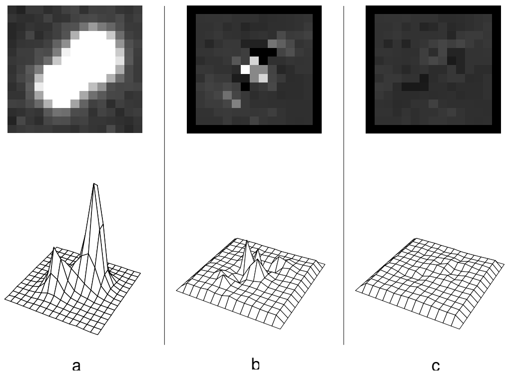

Figure 9 - Iterative process for locating three overlapping stars

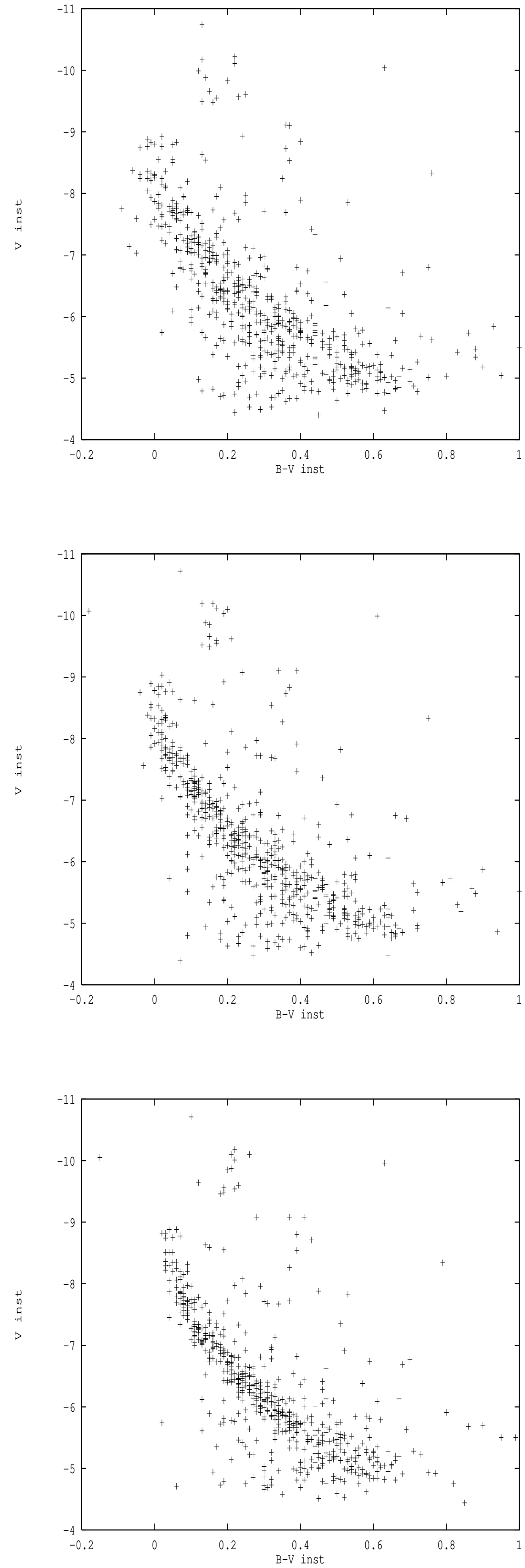

Figure 10 - Colour-Magnitude diagram of NGC 6397:(a) with ε=0.1, (b) with ε=0.05, and (c) with ε=0.05 (new calibration stars).

$$B_{pub} = 25.2126 + 0.95288\ B_{pres}$$

$$V_{pub} = 24.9123 + 0.98688\ V_{pres}$$

Where the subscript *pub* refers to the published values and *pres* to the values from the present work. Using the resulting fit the measured magnitudes were converted to the same scale Alcaino used. The two sets of Color-Magnitudes are shown in Figure 12. Notice that the published work was able to cover a much wider range of magnitudes by combining several types of observations, both photographic and with CCDs. The results from the present work on the other hand come from a single pair of images and are much more restricted in range. The overlap region shows how the agreement between the two methods is excellent.

NGC6101: A section 140 x 140 pixels in size was extracted from a CCD image of the globular cluster NGC6101. This is a moderately crowded section as can be seen from FigNGC6101. The image was chosen because accurate photometry computed by DAOPHOT was also made available by Dr. Ata. The image was visually inspected and 5 calibration stars were selected. These seemed to be isolated stars of varying brightness that properly represented the PSF of the image. Tablecalib:tab} lists the coordinates of the 5 stars. The Moffat function parameters that best fit them were computed to be ρ=3.37, β=3.52.

Next the program identified all stars in the image by using the method previously described. It later grouped them by identifying those stars the were too close together. The resulting groups can be seen in the contour plot in Figure 7 where the crosses indicate the location of each star as guesses by the star location routine, and lines join those stars that are grouped together. All stars in a group were then fitted at the same time.

Calibration stars used with the image of NGC6101:

x	y
121	28
92	99
46	79
54	110
28	64

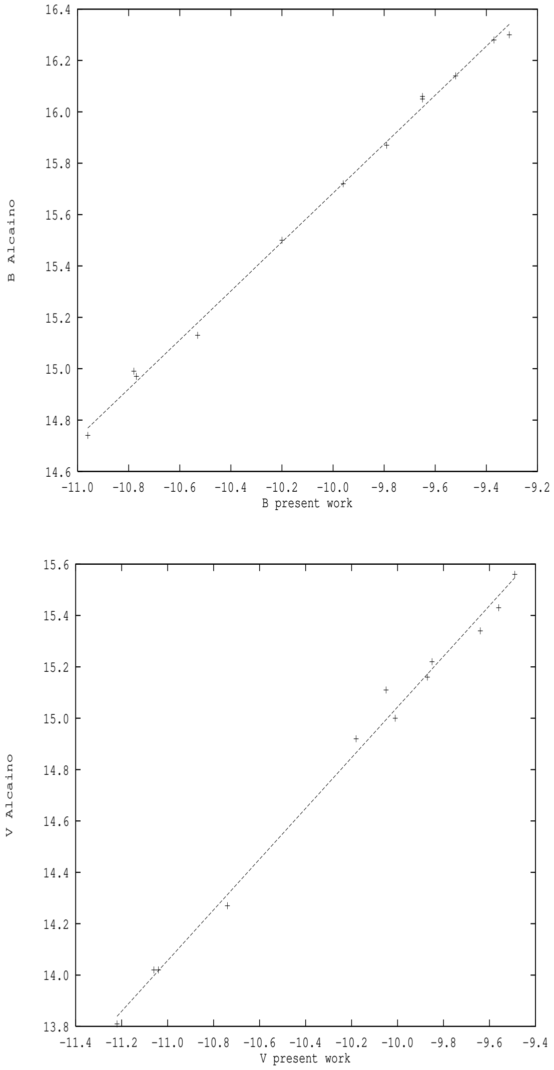

Figure 11 - Calibration of observations of NGC 6397.

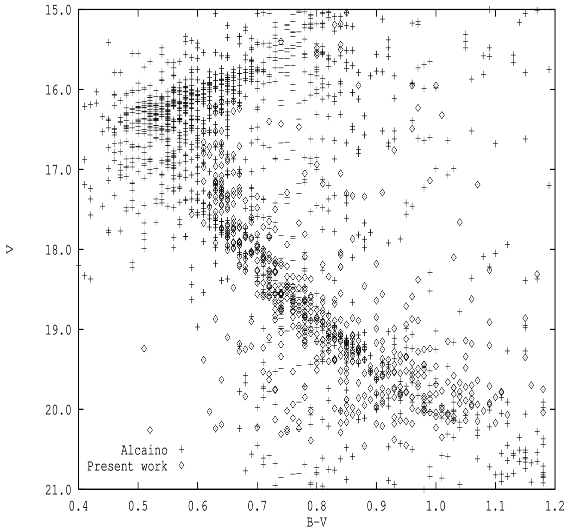

Figure 12 - CM Diagram of NGC 6397

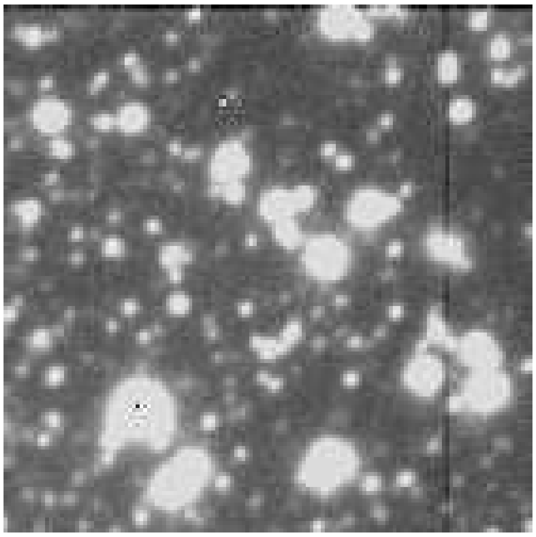

Figure 13 - Test Image, a section of the globular cluster NGC 6101.

The fit resulted in a list of intensities (in DN) and positions for 96 stars. Of these stars 7 were rejected based on poor χ^2_υ. The remaining stars were then correlated with the results previously obtained with DAOPHOT. The values of the magnitudes of the stars as computed by the two methods were plotted with the corresponding error bars and a linear fit was computed. This can be seen in Figure 14. The slope of the fit is 0.959 showing how well the magnitudes agree.

It is also important to notice how well the error bars agree for most data points though the methods for computing them were quite different. DAOPHOT computes the error from the values of the derivatives of the fitting function with respect to the fitted parameters. On the other hand the present method took as a measurement of the error the current values of the step size in intensity. This is based on the assumption that the method can not tell the exact brightness any better than the size of the step used to fit it, but ignores the effects of the uncertainties in position for the star. On the other hand the step sizes for the position tend to become very small very quickly, showing that it is very well determined and may have little effect.

47 Tuc: Finally a pair of frames of the globular cluster 47 Tucana were also reduced and a color magnitude diagram was built. Only data for the diagram from a past DAOPHOT reduction was available for comparison. Because of this individual photometry could be found, and the two diagrams could not be calibrated to a common scale. The published diagram can be seen in Figure 15. Figure 15 also shows the one based on the reduction using the method presented in this work. Once again the main sequence is clearly visible in both plots, and the spread around it seems to be of similar degree. The only significant difference is that DAOPHOT was able to locate more faint stars in the frame. This is not so much a function of the program itself, but of the way it is employed. By setting the number of standard deviations above the sky noise to a smaller value, FOTO should be able to locate all those stars too.

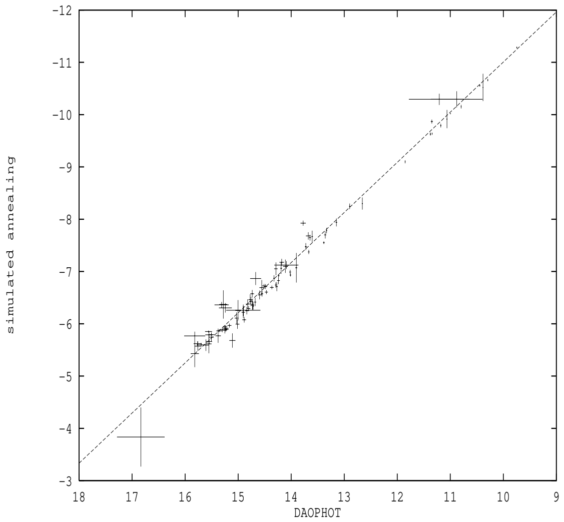

Figure 14 - Simulated Annealing VS DAOPHOT results.

5. Summary and Future Directions

It can be concluded that simulated annealing offers a viable way of optimizing function fits to data, in particular as it applies to astronomical photometry.

The main drawback remains speed. By its own nature simulated annealing is slow, randomly exploring regions of parameter space that other methods may never visit. But it is also here were its strength lies, since in doing so it is able to come up with results other methods may miss. So it is not a matter of rejecting simulated annealing based on the longer execution time but rather it should be judged on the need for better results it can provide.

Speed, however, may still be increased significantly. In particular new methods have been reported [Press91] that make use of the shape of the χ^2_υ surface to guide the random walk. This is achieved by combining simulated annealing with simplex, which has the ability of choosing to move in a way consistent with the surface but keeping

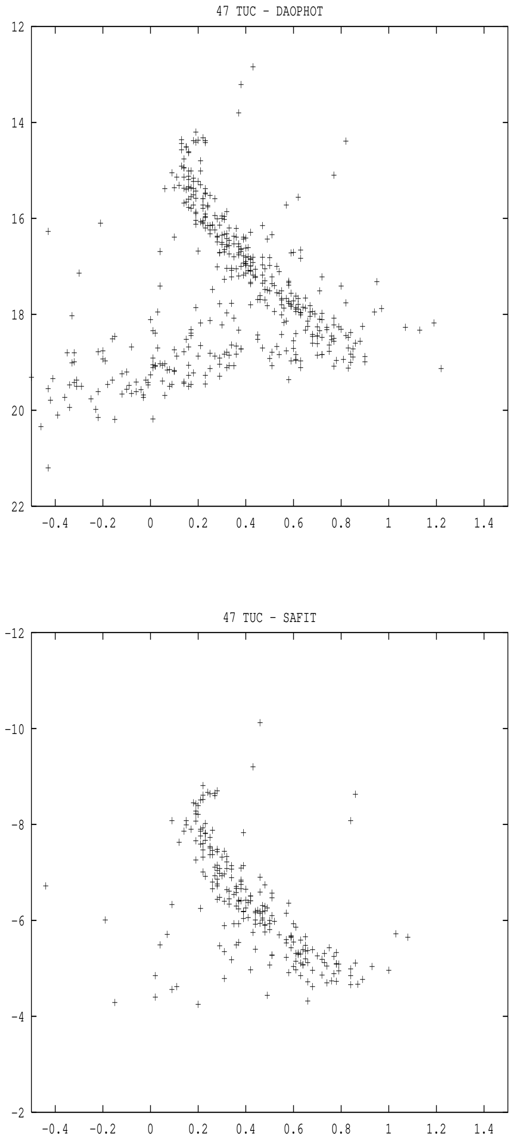

Figure 15 - CM Diagram of 47 Tuc, (a) published (b) current work.

the Metropolis criterion.

The bulk of the execution time is taken up by SAFIT as it runs through the groups of stars in the *gps* file, fitting them. Since all these groups are independent of each other it should be possible to reduce several of them at the same time, without affecting the quality of the results. To achieve this *gps* has to be divided into n smaller files each containing a part of all the star groups in *gps*. Then SAFIT can be started in n different CPUs each reading it's input from one of the different *gps* files, and all reading the parameters from the *par* file. Each of this will write its own *out* file, all of which can be combined into one single output file later. This simple process would reduce the reduction time by approximately $1/n$. This method would become more desirable as networks of workstations become more common in research institutions. Process migration code could be added to SAFIT to have it search for a CPU in the network that is not being used. Once one is located the process could migrate there. This takes advantage of unused CPU cycles.

The data in all of the tables found in this paper may be accessed in machine readable form via anonymous FTP from cpsc.ucalgary.ca. Also included in this set of data are the individual results for all stars measured in the two frames of NGC6101 as measured by both the above method and by DAOPHOT. The authors can be reached by E-mail for details on access to this data.

References