Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY LETTERS

Mon. Not. R. Astron. Soc. **403,** L1–L5 (2010) doi:10.1111/j.1745-3933.2009.00797.x

# General two-dimensional least-squares image subtraction

J. L. Quinn,[1⋆] A. Clocchiatti[1] and M. Hamuy[2]

[1]*Departamento de Astronomía y Astrofísica, Pontificia Universidad Católica de Chile, Vicuña Mackenna 4860, Santiago, Chile*
[2]*Departamento de Astronomía, Universidad de Chile, Camino El Observatorio 1515, Santiago, Chile*

Accepted 2009 December 2. Received 2009 November 24; in original form 2009 October 5

**ABSTRACT**

A very general two-dimensional solution to the image convolution problem is presented extending the least-squares approach of Bramich. Image convolution is of wide applicability in astronomical data reduction including, but not limited to, light-curve analysis, microlensing and planet transits. The authors present the solution in a manner of interest to someone wishing to code a spatially varying implementation of the least-squares method.

**Key words:** methods: data analysis – methods: numerical – techniques: image processing – techniques: photometric.

## 1 INTRODUCTION

Differential photometry via image subtraction is one of the most important tools for the professional astronomer. There have been many different approaches to measuring the change in intensity of astronomical sources and, as computers continue to get faster, they open up new ways that were previously too time consuming. There are two main modern avenues for accomplishing high-precision image subtraction of CCD images. A current widespread technique, primarily due to Alard and Lupton, involves the use of basis functions in a least-squares fit of the image that may also allow a space-varying kernel (Alard & Lupton 1998; Alard 2000). Bramich (2008) takes the more direct but computationally intensive route of solving for the best kernel by least-squares solutions of equations related to minimizing the value of chi-square. This approach is compelling as it omits the choice of basis functions from the user input; however, the original presentation handles the case of a spatially varying kernel only via interpolation of solutions on subimages and it does not discuss a variable background except in passing in Section 2.4. This Letter attempts to generalize the least-squares approach to the most general case of both a spatially varying kernel and background. Higher order background variation is immediately applicable using today's computers as well as the first-order and second-order kernel variation for small kernels.

The next section presents a useful mathematical formulation of the problem which is followed by an alternative formulation in Section 3 that is extremely concise and conceptually simple. Section 4 discusses the algorithmic properties of the least-squares approach, and Section 5 mentions some results in practice from the PUCDIA software before the concluding remarks in Section 6.

⋆E-mail: jquinn@astro.puc.cl

## 2 THE GENERAL 2D SOLUTION

Image subtraction is based on the idea that a reference image, $R$, can be convolved with a suitable kernel, $K$, to produce a convolved image, $M$, that when subtracted from the target image, $I$, produces a difference image, $D$, with small residuals except for variable sources. The possibility of a *differential* background, $B$, is also allowed (as opposed to an *absolute* background). We have then the relations

$$M \equiv R * K + B, \tag{1}$$

$$D \equiv I - M, \tag{2}$$

where the symbol '$*$' represents convolution. The goal is to find the best possible $K$ and $B$ such that the subtraction is the cleanest, where the residuals are primarily due to Poisson and readout noise. The relative quality of given kernel and background solutions can be compared with a chi-square statistic given by the squared differences of the image and convolved image.

$$\chi^2 = \sum_{i,j} \left( \frac{D_{ij}}{\sigma_{ij}} \right)^2 = \sum_{i,j} \left( \frac{I_{ij} - M_{ij}}{\sigma_{ij}} \right)^2 \tag{3}$$

Here, we have implemented weighting by specifying an error associated to each pixel, $\sigma_{ij}$. In practice, this is a combination of read noise and photon noise and depends upon the flat-field image, $F$. (Assigning pixel sigmas, $\sigma_{ij}$, is non-trivial. The treatment given by Bramich (2008) follows without modification to the present case and is not repeated here.) Any proper image-subtraction technique will be some method of trying to minimize this $\chi^2$. Astronomical images frequently have a megapixel or more which means our $\chi^2$ function has many terms and depending on the choice of kernel may be a function of hundreds or even thousands of variables.

The $i$- and $j$-sums in equation (3) ostensibly range from 1 to $X$ and 1 to $Y$ (the image dimensions) but technically the sums are only over the pairs $(i, j) \in G$, the set of 'problemless' pixels. Problemless means that the pixel is good in the image at that location $I_{ij}$ and

*all* the pixels included in the convolution of $R_{ij}$ are good in the reference image $R$. 'Good' means that the pixel of a CCD does not lie in a bad column or row, does not include a cosmic-ray strike, is not above some saturation or non-linearity level, below some unphysical count-level, or is tainted by any other issue. The pixels containing flux from variable sources should also be rejected.

In the Bramich approach, the kernel shape is completely arbitrary and each kernel element, $K_n$, may be labelled using a subscript, $n$, ranging from 1 to $N_\mathrm{k}$, the total number of kernel elements. An invertible mapping, $n \leftrightarrow (l_n, m_n)$, that gives the kernel coordinates for each pixel is also required. These coordinates are best interpreted as offsets from the centre of the kernel, yet it is worth noting that the (0, 0) pixel need not be included as an active pixel in the kernel. Using this notation, convolution by a spatially varying kernel $K$ with $N_\mathrm{k}$ elements over an image $R$ is given by

$$M_{ij} = \sum_{n=1}^{N_\mathrm{k}} K_n(i, j) R_{(i+l_n)(j+m_n)} + B(i, j), \tag{4}$$

where $B(i, j)$ is a differential background. Following the approach of Alard (2000) to incorporate spatial variance, $K$ and $B$ may be Taylor expanded as

$$\begin{aligned} K_n(x, y) = {} & K_n^0 \\ & + K_n^{1,x} x + K_n^{1,y} y \\ & + K_n^{2,x^2} x^2 + K_n^{2,xy} xy + K_n^{2,y^2} y^2 + \cdots \end{aligned} \tag{5a}$$

$$\begin{aligned} B(x, y) = {} & B^0 \\ & + B^{1,x} x + B^{1,y} y \\ & + B^{2,x^2} x^2 + B^{2,xy} xy + B^{2,y^2} y^2 + \cdots, \end{aligned} \tag{5b}$$

where we have used $(x, y)$ instead of $(i, j)$ to indicate the full spatial domain of $K$ and $B$. The expansion shown here is about zero merely to keep the equations compact. Expanding about the centre of the image may be more beneficial for many data sets. A perusal of this Pascal's triangle-like chart given in Table 1 helps demonstrate that these formulae may be expressed compactly as

$$\begin{aligned} & K_n(x, y; x_0^k, y_0^k) \\ & = \sum_{d_\mathrm{k}=0}^{\infty} \sum_{t=0}^{d_\mathrm{k}} K_n^{d_\mathrm{k},(x-x_0^k)^{d_\mathrm{k}-t}(y-y_0^k)^t} (x - x_0^k)^{d_\mathrm{k}-t} (y - y_0^k)^t, \end{aligned} \tag{6a}$$

$$\begin{aligned} & B(x, y; x_0^b, y_0^b) \\ & = \sum_{d_\mathrm{b}=0}^{\infty} \sum_{t=0}^{d_\mathrm{b}} B^{d_\mathrm{b},(x-x_0^b)^{d_\mathrm{b}-t}(y-y_0^b)^t} (x - x_0^b)^{d_\mathrm{b}-t} (y - y_0^b)^t, \end{aligned} \tag{6b}$$

where the full Taylor expansion about $(x_0^k, y_0^k)$ and $(x_0^b, y_0^b)$ is now explicitly shown. Here, the superscripts on the $K$s and $B$s on the right-hand side are *not* exponents but labels associating each (as yet unknown) parameter to a basis function. The $d_\mathrm{k}$ and $d_\mathrm{b}$ denote the current degree of the kernel and background during the expansion. The optimal parameters will be shown to be solutions to equation (12). Once the solutions are known, the convolution may be achieved immediately via equation (4).

**Table 1.** Pascal's triangle-like chart.

| Polynomial degree | Basis functions |
|---|---|
| 0 | $1$ |
| 1 | $x \quad y$ |
| 2 | $x^2 \quad xy \quad y^2$ |
| 3 | $x^3 \quad x^2y \quad xy^2 \quad y^3$ |
| 4 | $x^4 \quad x^3y \quad x^2y^2 \quad xy^3 \quad y^4$ |

For numerical implementations, practical considerations limit the total number of parameters $N$ used to match an image $R$ to another image $I$. One must always truncate the kernel and background expansion to finite degrees which we will call $D_\mathrm{k}$ and $D_\mathrm{b}$, respectively. Let $N_\mathrm{k}$ be the kernel size in pixels. If we recall the Gauss sum rule $\sum_{i=1}^{n} i = (n + 1)n/2$ and define $T(d) = (d + 2)(d + 1)/2$, it allows us to write $T_\mathrm{k}$ and $T_\mathrm{b}$ for the number of terms in the Taylor expansion of $K$ and $B$, respectively, and

$$N = N_\mathrm{k} \frac{(D_\mathrm{k} + 2)(D_\mathrm{k} + 1)}{2} + \frac{(D_\mathrm{b} + 2)(D_\mathrm{b} + 1)}{2} = N_\mathrm{k} T_\mathrm{k} + T_\mathrm{b} \tag{7}$$

for the total number of parameters in our fit. Once we have chosen our expansions, that is, $D_\mathrm{k}$ and $D_\mathrm{b}$ (and kernel size $N_\mathrm{k}$), we can proceed to try to calculate our best-fitting parameters that minimize the chi-square statistic. Every optimization technique has its own advantages and disadvantages: some are deterministic and others probabilistic, some are less computationally demanding than others, some are numerically unstable, all have to worry about global versus local extrema and so on. In general, it is possible to limit the parameter space searched to put constraints on the parameters, and it may happen that the global maximum or minimum that one is seeking is on the boundary of the allowed parameter region. The subset of optimization dealing with quadratic systems is known as quadratic optimization or quadratic programming. It will be shown that this quadratic problem (that is, minimizing equation 3) is actually equivalent to a linear optimization or linear programming problem.

If $\boldsymbol{a} \equiv (a^1, a^2, \ldots, a^N)$ represents an $N$-dimensional parameter vector, the extrema of a function in those variables lie where the gradient is equal to zero, that is, $\nabla_{\boldsymbol{a}} \chi^2 = \mathbf{0}$. If the chi-square function is shown to have a standard quadratic form, it is well known that there is a unique point where the gradient is zero, a minimum in this case. Once additional notation is developed, equation (3) will be shown to have a standard quadratic form. Assume it is true for now. The gradient condition ultimately allows the kernel-background solution to be part of a matrix equation, which greatly aids in visualization. Setting the derivative of our chi-square function equal to zero yields the set of $N$ formulae

$$\sum_{i,j} \frac{1}{\sigma_{ij}^2} (I_{ij} - M_{ij}) \frac{\partial M_{ij}}{\partial a^p} = 0, \tag{8}$$

which can be rearranged to

$$\sum_{i,j} \frac{M_{ij}}{\sigma_{ij}^2} \frac{\partial M_{ij}}{\partial a^p} = \sum_{i,j} \frac{I_{ij}}{\sigma_{ij}^2} \frac{\partial M_{ij}}{\partial a^p}.$$

It is helpful at this point to define

$$e_p^{ij} \equiv \frac{\partial M_{ij}}{\partial a^p} \tag{9}$$

so that we can write

$$\sum_{i,j} \frac{M_{ij}}{\sigma_{ij}^2} e_p^{ij} = \sum_{i,j} \frac{I_{ij}}{\sigma_{ij}^2} e_p^{ij}.$$

Lastly, if we note that $M_{ij} = \sum_{q=1}^{N} a^q e_q^{ij}$, the equation may be expressed in the form

$$\sum_{q=1}^{N} a^q \sum_{i,j} \frac{e_p^{ij} e_q^{ij}}{\sigma_{ij}^2} = \sum_{i,j} \frac{I_{ij}}{\sigma_{ij}^2} e_p^{ij}. \tag{10}$$

Equation (10) suggests a matrix $\boldsymbol{U}$ and vector $\boldsymbol{b}$ be defined via components as

$$U_{pq} \equiv \sum_{i,j} \frac{e_p^{ij} e_q^{ij}}{\sigma_{ij}^2} \quad \text{and} \quad b_p \equiv \sum_{i,j} \frac{I_{ij}}{\sigma_{ij}^2} e_p^{ij} \tag{11}$$

so that the system may be re-expressed as an ordinary matrix equation,

$$\boldsymbol{U}\boldsymbol{a} = \boldsymbol{b}, \tag{12}$$

and one may solve this system for $\boldsymbol{a} = \boldsymbol{a}_{\rm best}$ using any standard technique including the computationally inefficient $\boldsymbol{a}_{\rm best} = \boldsymbol{U}^{-1}\boldsymbol{b}$. The $\boldsymbol{U}$ matrix is symmetric and positive definite so Choleski decomposition is the most practical way to solve the system. It is worth explicitly noting that $\boldsymbol{U}$ and $\boldsymbol{b}$ are just arrays of numbers determined by the input data once desired kernel and background expansion degrees, expansion centres and basis function ordering are decided.

Now that we have some economic notation, it is insightful to reduce equation (3) to the standard quadratic form whereby it follows that the gradient condition returns a unique global minimum for unconstrained parameter space when the function is bounded from below. A residual function is clearly bounded by zero. Define $\boldsymbol{e}^{ij} \equiv \nabla_{\boldsymbol{a}} M_{ij}$ in accordance with equation (9) so that $M_{ij} = \boldsymbol{a} \cdot \nabla_{\boldsymbol{a}} M_{ij} = \boldsymbol{a} \cdot \boldsymbol{e}^{ij}$. Equation (3) can be expressed as

$$\chi^2(\boldsymbol{a}) = \sum_{i,j} \left[ \frac{(\boldsymbol{a} \cdot \boldsymbol{e}^{ij})(\boldsymbol{a} \cdot \boldsymbol{e}^{ij}) - 2I_{ij}(\boldsymbol{a} \cdot \boldsymbol{e}^{ij}) + I_{ij}^2}{\sigma_{ij}^2} \right],$$

which suggests we define new quantities

$$\boldsymbol{Q}_{ij} \equiv \boldsymbol{e}^{ij}(\boldsymbol{e}^{ij})^{\dagger} \quad \text{and} \quad \boldsymbol{c}_{ij} \equiv (\boldsymbol{e}^{ij})^{\dagger}, \tag{13}$$

where '†' is the matrix adjoint, so that we can write

$$\chi^2(\boldsymbol{a}) = \sum_{i,j} \left[ \frac{\boldsymbol{a}^{\dagger}\boldsymbol{Q}_{ij}\boldsymbol{a} - 2I_{ij}(\boldsymbol{c}_{ij}\boldsymbol{a}) + I_{ij}^2}{\sigma_{ij}^2} \right].$$

Since the parameters, $\boldsymbol{a}$, are constants over these sums, we may define

$$\mathcal{A} \equiv \sum_{i,j} \frac{2\boldsymbol{Q}_{ij}}{\sigma_{ij}^2} \quad \text{and} \quad \mathcal{B} \equiv \sum_{i,j} \frac{2I_{ij}\boldsymbol{c}_{ij}}{\sigma_{ij}^2} \quad \text{and} \quad \mathcal{C} \equiv \sum_{i,j} \frac{I_{ij}^2}{\sigma_{ij}^2}. \tag{14}$$

The chi-square function may now be written as

$$\chi^2(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^{\dagger}\mathcal{A}\boldsymbol{a} + \mathcal{B}\boldsymbol{a} + \mathcal{C}, \tag{15}$$

which has the canonical form of a quadratic system when $\mathcal{A}$, the Hessian matrix, is symmetric. In this case, it is easy to see from the definition that $\mathcal{A}$ is symmetric; in fact, it is positive semidefinite, which is an another test for a quadratic function to have a unique global minimizer if it has a lower bound.

There corresponds a map $p \leftrightarrow (d, t)$ that gives the degree and 'term number' labelling the $d$th degree components for each parameter index $p$. If $p$ refers to a kernel component, there is an additional map $p \rightarrow n$ that determines the kernel offset via $n \leftrightarrow (l, m)$. The ordering of the parameters in $\boldsymbol{a}$ is arbitrary and there are several reasonable choices. If one chooses to always put all the kernel parameters first, followed by the background parameters, it allows for an explicit representation of $e_p^{ij}$ as

$$e_p^{ij} = \begin{cases} x^{d-t} y^t R_{(i+l)(j+m)} & \text{if } p \leq N_{\rm k} T_{\rm k} \\ x^{d-t} y^t & \text{if } p > N_{\rm k} T_{\rm k} \end{cases}. \tag{16}$$

It is still necessary, however, to choose an ordering for the kernel elements. The kernel parameters could be ordered by expanding each specific kernel pixel in turn (equation 17) or they could be ordered according to the Taylor series for each pixel (equation 18):

$$\begin{pmatrix} K_1^0, & K_2^0, & K_3^0, & \ldots, & K_{N_{\rm k}}^0, \\ K_1^{1,x}, & K_2^{1,x}, & K_3^{1,x}, & \ldots, & K_{N_{\rm k}}^{1,x}, \\ & & \vdots & & \\ K_1^{D_{\rm k},y^{D_{\rm k}}}, & K_2^{D_{\rm k},y^{D_{\rm k}}}, & K_3^{D_{\rm k},y^{D_{\rm k}}}, & \ldots, & K_{N_{\rm k}}^{D_{\rm k},y^{D_{\rm k}}}, \\ B^0, & B^{1,x}, & B^{1,y}, & \ldots, & B^{D_{\rm b},y^{D_{\rm b}}} \end{pmatrix} \tag{17}$$

$$\begin{pmatrix} K_1^0, & K_1^{1,x}, & K_1^{1,y}, & \ldots, & K_1^{D_{\rm k},y^{D_{\rm k}}}, \\ K_2^0, & K_2^{1,x}, & K_2^{1,y}, & \ldots, & K_2^{D_{\rm k},y^{D_{\rm k}}}, \\ & & \vdots & & \\ K_{N_{\rm k}}^0, & K_{N_{\rm k}}^{1,x}, & K_{N_{\rm k}}^{1,y}, & \ldots, & K_{N_{\rm k}}^{D_{\rm k},y^{D_{\rm k}}}, \\ B^0, & B^{1,x}, & B^{1,y}, & \ldots, & B^{D_{\rm b},y^{D_{\rm b}}} \end{pmatrix} \tag{18}$$

Choosing a parameter ordering is equivalent to imposing a vector space decomposition on our parameter space, $V_N$, such that the contents of equation (17) are contained in the statement

$$V_N = \sum_{d_{\rm k}=0}^{D_{\rm k}} \sum_{t=0}^{d_{\rm k}} \oplus V_{N_{\rm k}}^{d_{\rm k},x^{d_{\rm k}-t}y^t} \oplus \sum_{d_{\rm b}=0}^{D_{\rm b}} \sum_{t=0}^{d_{\rm b}} \oplus V_1^{d_{\rm b},x^{d_{\rm b}-t}y^t}. \tag{19}$$

Here, '$\oplus$' is the usual direct sum of vector spaces.

The parameter ordering influences two things: how you code an implementation and a structural interpretation of the $\boldsymbol{U}$ matrix and $\boldsymbol{b}$ vector. It is not suggested that equations (17) or (18) are somehow the 'best' choices. They are merely examples showing two ways it could be done.

When the parameter vector $\boldsymbol{a}$ is ordered as in equation (17), the $N \times N$ $\boldsymbol{U}$ matrix has a block-diagonal structure, where each term in the kernel expansion has a corresponding $N_{\rm k} \times N_{\rm k}$ submatrix on the diagonal and each background term has a $1 \times 1$ submatrix there too. This is shown in Fig. 1 for $D_{\rm k} = 1$ and $D_{\rm b} = 1$. The off-diagonal blocks of $\boldsymbol{U}$ consist of entries dealing with correlations between the variables. In this picture, the matrix $\boldsymbol{U}$ contains $(T_{\rm k} + T_{\rm b})^2$ logically separate parts.

The parameter space $V_N$ may be decomposed in many different ways and this gives the programmer much freedom to think in the way they do best.

## 3 ALTERNATIVE FORMULATION

An alternative formulation for finding $\boldsymbol{a}_{\rm best}$ is to express the problem as an overdetermined linear system:

$$I_{ij} = \boldsymbol{e}^{ij} \cdot \boldsymbol{a} \quad \text{or} \quad \boldsymbol{I} = \boldsymbol{e} \cdot \boldsymbol{a}. \tag{20}$$

This system contains as many equations as good pixels, (that is, $|G|$) and $\boldsymbol{I}$ is a $|G|$-dimensional vector in 'image space' while $\boldsymbol{e}$ is

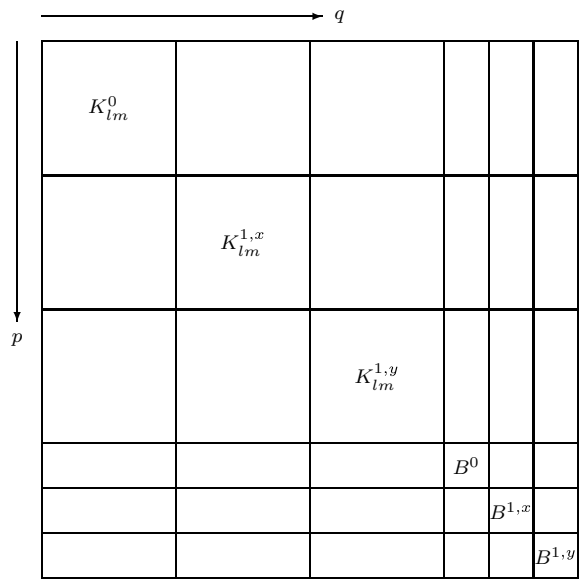


**Figure 1.** Schematic illustration for $\boldsymbol{U}$ when $D_\mathrm{k} = 1$ and $D_\mathrm{b} = 1$ when the parameter order of $\boldsymbol{a}$ is given by equation (17).

an $|G| \times N$ matrix. Introduce the $|G| \times |G|$ weight matrix, $\boldsymbol{w}$. The solution to a weighted overdetermined system requires solving the following system for $\boldsymbol{a}$,

$$\left(\boldsymbol{e}^\dagger \boldsymbol{w} \boldsymbol{e}\right) \boldsymbol{a} = \boldsymbol{e}^\dagger \boldsymbol{w} \boldsymbol{I}. \tag{21}$$

In theory, the weight matrix can incorporate the effects of pixel correlation. This would, however, require modelling the sources themselves, which would be difficult, and the enormous size of the full weight matrix makes it impractical anyway. Since the present techniques give nearly Poisson noise after subtraction, correlation is experimentally minor under normal circumstances. For now, the pixel errors are treated as uncorrelated and the weight matrix consists solely of the $\sigma^2_{ij}$ elements along the main diagonal and zeros off the diagonal so that we obtain our original system, $\boldsymbol{U}\boldsymbol{a} = \boldsymbol{b}$ from equation (12) if

$$\boldsymbol{U} = \boldsymbol{e}^\dagger \boldsymbol{w} \boldsymbol{e} \quad \text{and} \quad \boldsymbol{b} = \boldsymbol{e}^\dagger \boldsymbol{w} \boldsymbol{I}. \tag{22}$$

Perhaps some sparse matrix technique could partially include effects of pixel correlation in a realistic manner in the future.

This formulation is conceptually simpler and easier to code, but a straightforward implementation requires more memory which limits the number of parameters and therefore the size of the kernel.

## 4 COMPUTATIONAL ORDER

This general approach to a large extent answers the question, 'What is the best kernel to match two images?' when the kernel order, background order, kernel shape and the maximum number of acceptable iterations are specified. It is natural to inquire about the feasibility of any implementation. The execution time, $t_\mathrm{exec}$, is proportional to the number of iterations, $i_\mathrm{max}$. There are two bottlenecks in each iteration: the calculation of $U_{pq}$ and solving the linear system $\boldsymbol{U}\boldsymbol{b} = \boldsymbol{a}$. For $U_{pq}$, doubling the number of pixels in, for example, the $x$-direction doubles the execution time. The same holds for the $y$-direction. It follows that the $t_{\mathrm{calc},U}$ is directly proportional to the area, $A$ (in square pixels), of the images being matched. $U_{pq}$ has $N^2$ elements. Therefore, $t_{\mathrm{calc},U} \sim \mathcal{O}(AN^2)$. Solving an $N \times N$ linear system via simple algorithms is an $\mathcal{O}(N^3)$ calculation, so $t_\mathrm{solve} \sim \mathcal{O}(N^3)$. In an algorithmic sense, $t_\mathrm{exec} \approx t_{\mathrm{calc},U} + t_\mathrm{solve} \sim \mathcal{O}(i_\mathrm{max} N^3)$. The computational order only depends on the number of parameters and is agnostic to the kernel versus background parameters. In practice, the $t_{\mathrm{calc},U}$ term is currently the dominant term because the value of $A$ is typically large. The best Taylor expansion points of the kernel and background could in theory be found numerically but is too much time-consuming, and they must be treated as input parameters (the centre of the image is the best default choice).

It is extremely unfortunate that the $\sigma_{ij}$ elements must appear in the definition of $\boldsymbol{U}$ (equation 11) because it limits the $U_{pq}$ matrix to each reference-image pair. Much of the computational time for each iteration is spent calculating $\boldsymbol{U}$. There exists an image-independent object that characterizes the reference for a specific convolution expansion, but it is a four index object $U^{ij}_{pq}$ or $\boldsymbol{U}^{ij}$. Nevertheless, for some applications, such as time-series analysis, it may be desirable to use this large object to shorten the execution time when system memory permits. It should be noted that $U^{ij}_{pq}$ is well defined for some $i$, $j$ pairs that should not be included in the sums that yield $U_{pq}$. On the other hand, if one is willing to sacrifice some accuracy by treating $\sigma_{ij}$ as a constant, this condition can be exploited to dramatically shorten execution time for convolution matching an entire time series because the errors cancel altogether.

**Table 2.** Execution time (55 total parameters and three iterations).

| CPU | RAM (GB) | Test 1[a] | Test 2[b] | Test 3[c] |
|---|---|---|---|---|
| i7 2.6 GHz | 8 | $6^\mathrm{m}12^\mathrm{s}$ | $5^\mathrm{m}23^\mathrm{s}$ | $1^\mathrm{m}30^\mathrm{s}$ |
| Core 2 Duo 2.2 GHz | 3 | $13^\mathrm{m}25^\mathrm{s}$ | $11^\mathrm{m}47^\mathrm{s}$ | $3^\mathrm{m}16^\mathrm{s}$ |
| Pentium D 3.4 GHz | 2 | $38^\mathrm{m}26^\mathrm{s}$ | $15^\mathrm{m}07^\mathrm{s}$ | $4^\mathrm{m}26^\mathrm{s}$ |
| Pentium 4 3.0 GHz | 2 | | $24^\mathrm{m}06^\mathrm{s}$ | |
| Atom 1.5 GHz | 1 | OOM | OOM | $12^\mathrm{m}27^\mathrm{s}$ |

[a] $2041 \times 2046$ images
[b] $1792 \times 1897$ images
[c] $982 \times 982$ images.

## 5 THE PUCDIA SOFTWARE

The author has implemented the techniques presented in this Letter in a software package called PUCDIA, which will be released to the community in the future.

Table 2 shows some of PUCDIA's execution times on different machines for various size images while keeping the number of parameters the same. The times shown are for a serial (as opposed to parallel) implementation. Early parallel versions have shown that the speed increases up to 30 per cent. In each case, 55 total parameters were fit (49 for a circular kernel of radius 7 and six for a second-degree background) and three iterations were used. Test 1 requires nearly 2 GB of RAM, Test 2 requires roughly 1.5 GB and Test 3 requires about 0.5 GB. Tests that cannot succeed due to insufficient memory are marked with 'OOM'.

## 6 CONCLUSIONS

A comprehensive formalism for two-dimensional image subtraction has been presented. It reduces to the technique presented by Bramich (2008) in the case where the kernel and backgrounds are treated with constant order. Well-written implementations can generalize the Bramich approach to higher order such that they should run at roughly the same speed (that is, there is only minor overhead) as a constant order implementation for problems with the same number of parameters.

**ACKNOWLEDGMENTS**

The authors acknowledge support from the Millennium Center for Supernova Science through grant P06-045-F funded by 'Programa Iniciativa Científica Milenio de MIDEPLAN'. AC and MH also acknowledge funding from grants Basal CATA PFB 06/09 and FONDAP No. 15010003.

**REFERENCES**

Alard C., 2000, A&AS, 144, 363
Alard C., Lupton R. H., 1998, ApJ, 503, 325
Bramich D. M., 2008, MNRAS, 386, L77

This paper has been typeset from a TEX/LATEX file prepared by the author.