# Towards a Machine Learning Based Algorithm for Performing Low Brightness Photometry using Star Shape

Alec Lessing

June 12, 2021

## 1 Introduction

Photometry is the practice in astronomical data processing of measuring the radiation flux of astronomical sources (stars, galaxies, asteroids, ect.) from telescope images. In astronomy, the flux of a source is the amount of energy incoming from that source per unit time passing through a surface of unit area perpendicular to the incoming light. Essentially, it is a quantification of how bright a source appears from Earth. Modern astronomy uses CCD detectors, arrays of pixels that count how many photons hit each pixel in a certain period of time. There is a complicated science behind calibrating photometric measurements from CCDs, factoring in the different distributions of freqeuncies of light coming from different sources as well as the sensitivities of detectors to different frequencies and numerous factors having to do with the electronics of the detectors. However, at the end of the day, for a given detector, the flux of a source will be proportional to the *true instrumental flux* $f_t$, the expected value of the total number of photon counts on the detector from the source in question (in a certain *exposure time*). Astronomers often express fluxes $f$ (of any sort, true, measured, instrumental, whatever) in terms of magnitude $m$, defined as

$$m = -2.5 \log_{10} f \tag{1}$$

Thus photometry is essentially the practice of inferring $f_t$ (usually expressed in terms of its associated instrumental magnitude $m_t$) from the image of a source on a detector. There is a variety of methods currently used in photometry (which will be reviewed below), all of which essentially amount to measuring the ratio of the number of photon counts of a source to some reference value. This is because, in expectation, the shape of the figure in an image is exactly the same regardless of how bright it is, so the only thing that changes is how "tall" that shape is (how many photon counts make up that shape). However, these methods completely ignore the fact that low flux signals are noisier and so the shape of a dimmer source in the image differs more from the average, expected shape. The goal of this work will be to infer the true magnitude of sources based solely on their shape in detector images. Such measurements will be restricted to point sources (or "stars," in astronomy lingo since stars are so far away that they appear as infinitesimally small points of light to our detectors) since point sources' shapes are all identical for a detector (unlike, say, a galaxy, whose shape depends both on telescope characteristics as well as the actual shape of the galaxy).

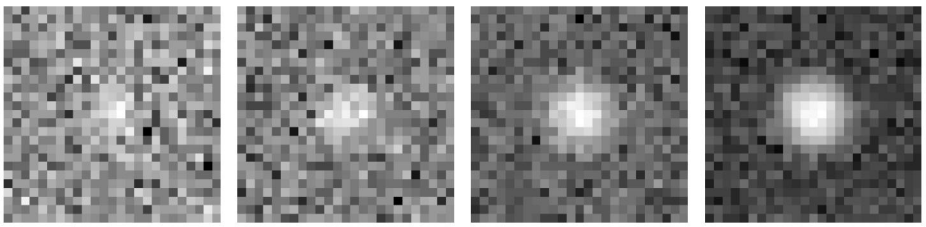

Figure 1: A series of star images simulated using the algorithm explained in section 3, with star brightness increasing from left to right by a factor of 2.5 in each successive image. Observe that the shape of the stars stays the same but gets less noisy as the stars get brighter.

## 2 Current Photometric Methods

To discuss current photometric methods, it is necessary to introduce the concept of the *point spread function* (PSF), which is essentially the expected shape that a point source is expected to deposit on a detector. Formally, the PSF for a point source centered at coordinates $x_s, y_S$ is defined such that the integral of the PSF over a region of the detector is the expected fraction of the light from the source that will fall in that region. Thus,

$$\tilde{S}_{xy} = \mathbb{E}\big[S_{xy} \mid f_t, x_s, y_s\big] = f_t \iint_{C_{xy}} PSF(\bar{x} - x_s, \bar{y} - y_s) d\bar{x} d\bar{y} \tag{2}$$

where $S_{xy}$ is the signal (number of photons) counted in pixel $x, y$ from a point source with true flux $f_t$ at point $(x_s, y_s)$, and $C_{xy}$ is the set of points $(\bar{x}, \bar{y})$ that lie inside pixel $x, y$. (Note that $x, y$ must be integers because they label pixels while $x_s, y_s$ do not because in general a source's location does not correspond exactly to the camera grid and $\bar{x}, \bar{y}$ also are not integers because they parameterize all the locations within a pixel, not just pixels themselves.)

### 2.1 Aperture Photometry

Aperture photometry is the most basic (and most common) method in photometry, particularly for low brightness stars, where other methods tend to fail. Aperture photometry essentially entails counting all the photons in a certain "aperture" (usually a circle) centered on the star. Consider a star centered at location $x_s, y_s$, the measured flux of the star is

$$f_m = \frac{\sum_{xy} a_{xy} I_{xy} \ - As}{\sum_{xy} a_{xy} \iint_{C_{xy}} PSF(\bar{x} - x_s, \bar{y} - y_s) d\bar{x} d\bar{y}}$$

where $I_{xy}$ is the value of the image at pixel $x, y$, $a_{xy}$ is the fraction of pixel $x, y$ that lies inside the aperture, $A$ is the area of the aperture, and $s$ is the average value of the sky background. Essentially, the first term in the numerator sums the number of counts within the aperture, the second term in the numerator subtracts off the number of counts expected to be within the aperture from the sky background, and the value of the denominator is the fraction of the total light from the star expected to fall inside the aperture (so dividing by this quantity accounts for the fact that you are not counting for some photons that hit the detector outside the aperture). Aperture photometry is a very simple method dating back to the earliest days of digital astronomy, yet it is very effective for many purposes. Indeed, much of the software currently used in the field is essentially a python wrapper around the original, decades-old software for aperture photometry, and aperture photometry is a primary workhorse of the new *photutils* project that is rewriting many of the most common photometric methods as part of one modern-style python-native package and is increasingly becoming the new standard. There are still new state-of-the-art (for various bells and whistles and statistical methods built around the core algorithm) aperture photometry packages being produced, such as the two year old A-PHOT package (Merlin et al., 2019).

### 2.2 PSF-Fitting Photometry

PSF-fitting photometry is commonly used method for very precise photometry of moderately-bright to bright stars or in images crowded with many stars that nearly blend together. It entails iteratively approximating a least-squares fit of the locations and fluxes of a set of $N$ (possibly one) stars, where the objective is to minimize the square sum of the residuals between the actual image and the image expected given the parameters. This essentially amounts to a Gaussian Mixture Models algorithm, except rather than learning parameters from data and later using the resulting generative model to classify test data, an algorithm similar to the GMM learning algorithm is used on each data point. (Note that the other difference is that the covariance of the distribution is known and fixed rather than fitted, and often the PSFs used for fitting are not Gaussians but more detailed distributions specific to the telescope and detector). The form of this algorithm most commonly used is a version laid out in a seminal 1987 paper describing the DAOPHOT algorithm (Stetson, 1987). The new (increasingly standard) *photutils* package mentioned earlier implements the DAOPHOT algorithm as one of the main workhorses other than aperture photometry (Bradley et al., 2021).

The current state of the art is the *ePSF* method, first published in a 2000 paper applying the method to the Hubble Space Telescope, where the detector quality and relative constancy of conditions in space allows for extremely precise empirical measurement of the PSF. The *ePSF* method is essentially an implementation of the 1987 DAOPHOT PSF-fitting algorithm, where the fact that empirical measurements of the PSF are made in images where the PSF is broken up into pixels is directly taken into account when fitting a PSF to a star when running the algorithm, rather than estimating an approximate pixel-less PSF from the data and then converting it back into pixel format when comparing the PSF-based prediction of the image to the actual image (Anderson & King, 2000). An updated version of the *ePSF* method was published in 2016 (Anderson, 2016).

As of 2017 state-of-the-art photometric surveys using the Hubble Telescope were performed using PSF-fitting for bright stars or stars in dense fields, a variant of aperture photometry for very faint stars, and a hybrid method for slightly less faint stars (Bellini, Anderson, et al., 2017; Bellini, Milone, et al., 2017).

## 3 The dataset: simulated stars

To create a labelled dataset, 31x31 images of stars with Gaussian PSFs were simulated. To very good approximation, detector counts are independent of eachother (a given photon doesn't interfere with other incoming photons, although this doesn't hold up perfectly when it comes to artifacts of the detector electronics). Thus, detector images are accurately sampled using poisson statistics. If $I_{xy}$ is the image at pixel $x, y$ (including electronics, noise, background sky glow, and the signal from the star), then we can randomly sample images according to $I_{xy} \sim Poi(T_{xy})$ where $T$ is the "true" image (defined as the expected value of the image, given the electronics, sky, and source). To good approximation, noise from electronics creates a

homogeneous noise background, and so can be included as part of sky noise, a constant $s$, which is the average background number of counts. Then we can simulate the "true" image of a star with flux $f_t$ by adding expected signal to the sky

$$T_{xy} = s + \tilde{S}_{xy} = s + f_t \iint_{C_{xy}} PSF(\bar{x} - x_s, \bar{y} - y_s) d\bar{x} d\bar{y} \tag{3}$$

(see equation (2) for why the source signal takes this form) and then create simulated images by sampling each pixel with poisson noise. The realization of images from the Gaussian PSF and the application of Poisson noise were computed using the *photutils* python package (Bradley et al., 2021). To ensure that the machine learning algorithm only was given information about the shape (and not the number of counts in the image), the average sky was subtracted out and the images were divided by total measured signal in the image to get the datapoints for the machine learning algorithm.

$$\mathcal{I}_{xy} = \frac{I_{xy} - s}{S_{total}} \qquad \text{where} \quad S_{total} = \sum_{xy} I_{xy} - s \tag{4}$$

The signal-normalized images $\mathcal{I}_{xy}$ are the datapoints for the machine learning algorithm and the true instrumental fluxes $f_t$ are the labels for the algorithm. This method of signal-normalized star image simulation was used both to create training data as well as to test the efficacy of the algorithm. See Figure 1 above for examples of simulated stars created by this algorithm.

# 4 Methods

## 4.1 Feature Extraction

The goal of this project is to infer the true magnitude of stars in images based on the shapes of the stars in the images, and, in particular, the way those shapes deviate from an ideal PSF. Thus, the features are strange quantities that intuitively seem like they should correspond to the "noisyness" of the star's shape. Features were picked based on this intuition and based on plotting the values of these features vs. the true star flux to see if there were any patterns that a machine learning algorithm would be able to use.

***Mean Square Radius*** The mean square radius of the image (the image weighted average square distance from the center of the star) was taken as a feature. Intuitively, the PSF should have a certain mean square radius and so the degree of deviation from this average mean square radius should be an indication of how "noisy" the shape of the star is.

***Image centroids*** The centroid is the image-weighted average of the pixel coordinates. For a perfect PSF, the centroid should lie exactly in the center of the image, so the degree of deviation of the centroid from the center is an indication of the noisiness of the stars shape. The image centroid and the centroid of the image divided by the square root of the PSF

***Absolute Residual Based Features*** Other features depended on the absolute residual image (another image found by taking the absolute value of the deviation of the image from the PSF). Intuitively, this image should correspond most directly to the noisiness of the star's shape since the image is simply an image of that noise. Several features were made based off of this image. One feature summed all the values in the image. Another was simply the value of this image at the center of the star. Others included the sum of this image divided by the PSF (and another which was the same using the square root of the PSF) and the centroids of of this image multiplied or divided by various powers of the PSF.

## 4.2 Learning Algorithms

Two learning algorithms were employed: kernel-based ridge regression and a three-hidden layer neural net.

*Ridge regression* is simply least squares linear regression (fitting parameters to minimize the mean square error between predicted fluxes and true fluxes) with an additional penalty of the sum of the l2 norm of the coefficients of the linear regression (the algorithm minimizes the sum of the mean square error and the l2 norm, called the training objective). The l2 norm term makes the algorithm prefer smaller values of $\theta$ to avoid making tweaks to very large $\theta$ that do not really correspond to any real behavior because they are so tiny relative to the magnitude of $\theta$ (and so instead correspond to better matching artifacts of noise in the training set).

$$J(\theta) = \lambda ||\theta||_2^2 + \frac{1}{n} \sum_{i=1}^{n} SE^{(i)} \quad \text{where} \;\; SE^{(i)} = (f_t^{(i)} - f_p^{(i)})^2 \tag{5}$$

where $J$ is the training objective, $\theta$ is the vector of linear regression coefficients (such that $f_p^{(i)} = \theta \cdot x^{(i)}$, where $x^{(i)}$ is the vector of features described in the last subsection), and $f_t^{(i)}$ and $f_p^{(i)}$ are the true and predicted flux of the $i$th example star, $n$ is the total number of stars, and $\lambda$ is some number controlling the importance of the l2 regularization. The gradient of $J$ with respect to $\theta$ represents the direction that you have to change $\theta$ to increase $J$ (and the rate it changes). Therefore, to

minimize $J$, you should change $\theta$ in the opposite direction. To make best use of computing resources, we do this with one training example at a time and update $\theta$ according to

$$\theta = \theta - \alpha \nabla_\theta \left( \frac{\lambda}{n} ||\theta||_2^2 + SE^{(i)} \right) \tag{6}$$

iterating using a different $i$ each time (until all $i$ are looped through: then they are repeated), for some positive constant $\alpha$ caled the learning rate. A linear model is very limited. Therefore, in order to be able represent more complicated functions, we can replace $x^{(i)}$ with a much higher dimensional vector $\phi(x^{(i)})$ for some nonlinear function $\phi$, so that ridge regression now can tweak arbitrary linear combinations of a bunch of nonlinear functions of $x^{(i)}$, giving much more flexibility in the set of mappings $x^{(i)} \mapsto f_p^{(i)}$ that ridge regression considers and optimizes over. As it turns out, the only way in which $\phi$ is ever used in the training process and prediction process when this substitution is made is taking the inner product of $\phi$ of two $x$s. Therefore, we can replace this inner product with any nonlinear function of two $x$s that we like, called the kernel, and everything works the same. In this project, kernel-based ridge regression was computed using *scikit-learn* (Pedregosa et al., 2011). The *scalar* input vector normalization from *scikit-learn* was also used to apply a linear tansformation to the input vectors such that they transformed set had mean zero and variance one. Without this transformation, the values of $x^{(i)}$ tended to be such that the variation between them avoided the part of the domain of the kernel that expressed nonlinear behavior, making learning work poorly. Once the transformation was applied, the default learning rate for kernel-based ridge regression worked well, because the data was normalized such that the gradients that would appear in training were of the same order of magnitude expected for the default parameters of the learning algorithm.

The second method used was a *neural net*. A neural net with three hidden layers was used (35 tanh nodes, 25 relu nodes, 8 relu nodes, and then a linear output layer). This means that $m_p$, the predicted magnitude (it turned out by trial and error that doing the regression in terms of magnitude worked better, because many of the features varied more linearly with magntiude than with flux) was computed from the features $x$ described in the last subsection via the relationship

$$\begin{aligned}
{}_1h_{j_1} &= \sum_{k=1}^{d} {}_1w_{j_1 k} x_k + {}_1b_k \\
{}_2h_{j_2} &= \sum_{j_1=1}^{35} {}_2w_{j_2 j_1} \tanh({}_1h_{j_1}) + {}_2b_{j_1} \\
{}_3h &= \sum_{j_2=1}^{25} {}_3w_{j_2} Relu({}_2h_{j_2}) + {}_3b_{j_2} \\
{}_4h &= \sum_{j_3=1}^{8} {}_4w_{j_3} Relu({}_3h_{j_3}) + {}_3b_{j_3} \\
{}_{mp} &= {}_4w\ ({}_4h) + {}_4b
\end{aligned} \tag{7}$$

where $Relu(x) = 0$ for $x \leq 0$ and $Relu(x) = x$ otherwise. The idea is that by tweaking the $w$s and the $b$s, a very large set of nonlinear mappings $x_k \mapsto m_p$ can be represented. The parameters $w$s and $b$s are just like the $\theta$ from ridge regression, and training can proceed the same way, by stochastic gradient descent with a mean square error training objective (plus a l2 regularization term for each layers' $w$s). We tweak the $w$s and $b$s in each iteration by an amount proportional to the gradient of the training objective (for one training example at a time), just as with stochastic gradient descent for ridge regression. This algorithm was executed computationally using Tensorflow (Martın Abadi et al., 2015). Tweaking of the layers was done by seeing what the minimal networked needed before mean square error accuracy declined was. Regularization parameters were set by observing their effect on a curve comparing validation error to training error for each epoch (they were made as small as possible before the behavior of these two curves started to diverge).

# 5 Results

The two machine learning solutions were tested for Gaussian-shaped stars with standard deviations of 3 pixels on a sky background of 50 counts on average. Testing datasets of tens of thousands of training examples were used for both the neural net and the kernal-based ridge regression, and the efficacy of their fits was tested with a separate testing dataset with 2000 datapoints. To examine the efficacy of the algorithms, the magnitude error (absolute value of actual minus predicted) magnitudes were plotted against the true instrumental magnitude, to see how accurate each algorithm was at each magnitude. In addition, the machine learning based accuracies were compared to the accuracy for the standard aperture photometry based magnitude. Note that the PSF photometry method was not used because PSF photometry either fails or produces poor results for dim stars that are studied here.

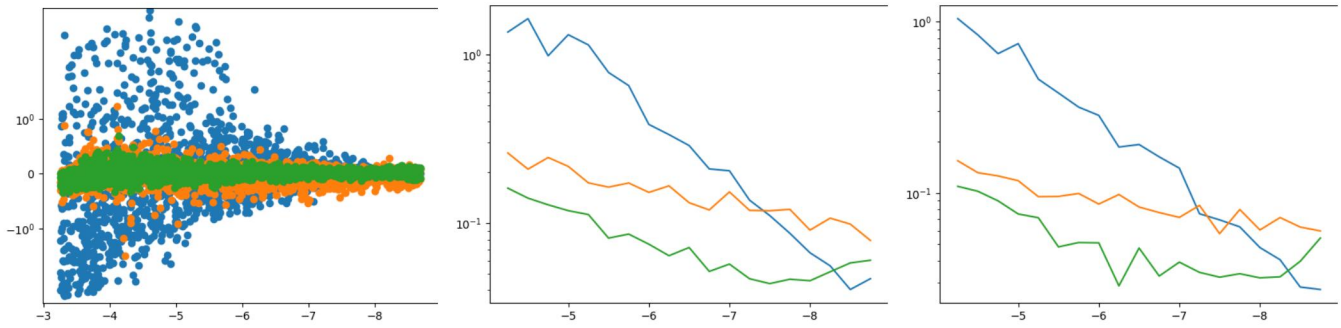

Figure 2: In each plot the x axis is the true instrumental mangitude of each testing datapoint and the y axis is the magnitude error (how far from the correct magnitude the predicted magnitude is). The blue points were computed with the aperture photometry method, the orange with kernel-based ridge regression, and the green with the neural net. The leftmost plot shows each datapoint, the middle and rightmost plots show the rms and median error at each instrumental magnitude.

It is clear from the plots above that the machine learning based methods hold a significant advantage over the traditional aperture photometry measurement for dim stars (higher magnitudes). As the stars get brighter, their relative advantage compare to aperture photometry decreases. This makes a lot of sense, because the features that the ML algorithms are using are designed to correspond to the deviations of the star shape from an ideal PSF. Since brighter stars have more idealized shapes, these features become relatively less predictive as the stars get brighter. Indeed, if anything the ML algorithms start to perform slightly worse for bright stars because all the stars start to look quite regular, so it becomes hard to distinguish between them based on the features given to the algorithm. Also note that the neural net always performs better than the kernel-based solution. The likely reason for this is that the neural net draws from a larger set of models that it can chose from, whereas the ridge regression is somewhat limited by the characteristics of its kernel function (the laplacian kernel used here worked by far better than any other kernel, so this is approximately the best it gets).

# 6 Future Work and Implementations

If future work shows that this method continues to have robust performance in multiple situations and on data simulated accounting for all the specific details of a real-life telescope, it could prove extremely useful for studying low-signal transients and variable stars. It is common in astronomy that the presence of a dim time-varying star is known, because it shows up clearly when summing the images taken at multiple different times, but that it is too dim in an image from any given time to measure its brightness at that specific time. This severely limits the number of time-varying stars that can be studied, because many are too dim. These new methods may allow for studying time-varying stars that are several times dimmer, greatly increasing the effectiveness of synoptic astronomical surveys (telescope surveys that take a bunch of pictures over time to see how things vary). The most important such survey currently is the up and coming Large Synoptic Sky Survey (LSST), which promises to be miles better than any previous synoptic survey and will come online in a year or two. The PhoSim package provides state-of-the art simulation of LSST images by statistically sampling photons and the path the photons can take through the telescope. Such physics-based computations are so time consuming that it was not feasible to simulate enough training data in the timeframe allotted for this work, but it is an obvious future step. The PhoSim package allows users to define physics parameters for any telescope they like, meaning that it can theoretically simulate any telescope (and already simulates a few telescopes other than LSST) (Peterson et al., 2015). The author is interested in developing a general-purpose low-brightness photometry python software that could potentially become a new state-of-the-art tool for super faint photometry, particularly for synoptic surveys. Part of this process would likely include testing a ton of new data features and new machine learning algorithms (such as deep learning from raw images rather than features based on raw images).

# 7 Acknowledgements

The author would like to thank the CS 229 course staff for a very interesting quarter learning about machine learning.
This research made use of Photutils, an Astropy package for detection and photometry of astronomical sources (Bradley et al., 2021).

# References


Anderson, J. (2016). *Empirical models for the wfc3/ir psf* (tech. rep.). https://www.stsci.edu/files/live/sites/www/files/home/hst/instrumentation/wfc3/documentation/instrument-science-reports-isrs/_documents/2016/WFC3-2016-12.pdf

Anderson, J., & King, I. R. (2000). Toward high-precision astrometry with WFPC2. i. deriving an accurate point-spread function. *Publications of the Astronomical Society of the Pacific*, *112*(776), 1360–1382. https://doi.org/10.1086/316632

Bellini, A., Anderson, J., Bedin, L. R., King, I. R., van der Marel, R. P., Piotto, G., & Cool, A. (2017). The state-of-the-artHSTAstro-photometric analysis of the core ofωcentauri. i. the catalog. *The Astrophysical Journal*, *842*(1), 6. https://doi.org/10.3847/1538-4357/aa7059

Bellini, A., Milone, A. P., Anderson, J., Marino, A. F., Piotto, G., van der Marel, R. P., Bedin, L. R., & King, I. R. (2017). The state-of-the-artHSTAstro-photometric analysis of the core ofωcentauri. III. the main sequence's multiple populations galore. *The Astrophysical Journal*, *844*(2), 164. https://doi.org/10.3847/1538-4357/aa7b7e

Bradley, L., Sipőcz, B., Robitaille, T., Tollerud, E., Vinícius, Z., Deil, C., Barbary, K., Wilson, T. J., Busko, I., Donath, A., Günther, H. M., Cara, M., Conseil, S., Bostroem, A., Droettboom, M., Bray, E. M., krachyon, Lim, P. L., Bratholm, L. A., ... Souchereau, H. (2021). *Astropy/photutils: 1.1.0* (Version 1.1.0). Zenodo. https://doi.org/10.5281/zenodo.4624996

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Merlin, E., Pilo, S., Fontana, A., Castellano, M., Paris, D., Roscani, V., Santini, P., & Torelli, M. (2019). A-PHOT: A new, versatile code for precision aperture photometry. *Astronomy & Astrophysics*, *622*, A169. https://doi.org/10.1051/0004-6361/201833991

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peterson, J. R., Jernigan, J. G., Kahn, S. M., Rasmussen, A. P., Peng, E., Ahmad, Z., Bankert, J., Chang, C., Claver, C., Gilmore, D. K., Grace, E., Hannel, M., Hodge, M., Lorenz, S., Lupu, A., Meert, A., Nagarajan, S., Todd, N., Winans, A., & Young, M. (2015). SIMULATION OF ASTRONOMICAL IMAGES FROM OPTICAL SURVEY TELESCOPES USING a COMPREHENSIVE PHOTON MONTE CARLO APPROACH. *The Astrophysical Journal Supplement Series*, *218*(1), 14. https://doi.org/10.1088/0067-0049/218/1/14

Stetson, P. B. (1987). DAOPHOT - a computer program for crowded-field stellar photometry. *Publications of the Astronomical Society of the Pacific*, *99*, 191. https://doi.org/10.1086/131977