Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 000

A Reinforcement Learning Autoguider for Astrophotography

Shashank Srivastava[1], Michael Hirsch[2], Jan Peters[2], Bernhard Schölkopf[2]

June-July 2009
(Revised: October 2009)

[1] Indian Institute of Technology, Kanpur ; [2] MPI für biologische Kybernetik, AGBS.

A Reinforcement Learning Autoguider for Astrophotography

Shashank Srivastava, Michael Hirsch, Jan Peters, and Bernhard Schölkopf

Abstract. Contemporary autoguiders for guiding telescope mounts for astronomy are essentially rule-based, and depend on rule-based corrections based on object positions at different time instances. We propose a two variable Q-learning optimization algorithm is proposed for autoguiding a German Equitorial Mount for star-tracking and astrophotography. We formulate the problem as a continuous Markov Decision Process with agent 'actions' corresponding to motor movement durations, using the GPUSB controller from Shoestring astronomy. The new approach uses robust star-tracking heuristics and reinforcement learning to learn favourable guiding policies, while incorporating mount specific behaviour and current motion of the mount in deciding new movements. The method is extensively tested on artifical data using simulations, as also for real tracking. The learning algorithm is seen to converge quickly to an optimum strategy, and error rates using the proposed autoguider are seen to be lower than rule-based approaches.

---

1 Introduction

Deep space astronomy using ground based telescopes depends heavily on autoguiders to keep the field of view stationary, and prevent drifting. Autoguiders compensate for the rotation of the Earth as well as possible movement of the object in view. In astrophotography especially, long exposures are needed, or point objects can appear as elongated streaks. Contemporary autoguiders are rule-based, and depend on object position at successive time instants to issue commands to the telescope mount for tracking. In this paper, we study the scope of statistical learning in building an intelligent autoguider, that derives the best guiding policies for an observation session based on experience. This is especially significant in the context of astronomy, since an observation session depends on a number of variables such as the goodness of polar alignment, physical stability of the setup, and mount specific behaviour.

Current solutions are characterized by two types of autoguiders: general purpose/ built-in peripheral softwares such as MaximDL, MaxDSLR and GuideDog, or dedicated tracking modules such as PHDguiding by Stark Lab and the Santa Barbara Instruments Group (SBIG) ST-4. While the second category is generally considered more sensitive and accurate than the first, both categories depend entirely on preset heuristics, and successive star locations for tracking. Current methods also fail, if the intensity of the guidestar dips by a threshold fraction (50 % in case of ST-4).

In this report, we present and analyze a Q-learning based optimization algorithm to decide the behaviour of the mount. The system learns optimum guiding policies specific to the current observation session over a small training period, by evaluating different strategies for issue of commands that minimize the track error, require relatively minimal mount movement, and are stable(small variations in mount activity). The convergence of the algorithm is seen to be fast, and is seen to improve the tracking performance in extensive simulations, and actual stars.

In Section 2, we describe the preliminary approaches and efficient heuristics for removing background noise, accurately estimating star positions from photo frames, and camera alignment. Section 3 explains our approach in detail; and the learning formulation that minimizes the defined error. In Section 4, we elaborate on implementation details, and describe the setup of the experiments. In Section 5, we analyse and compare results obtained from the proposed approach on both simulated and real data. Section 6 concludes with a summary of the proposed method, shortcomings, and suggested directions for further testing and enquiry.

2 Preliminary steps

In this section, we explain the preliminary steps towards accurate identification of star coordinates using a mounted camera, that can send a steady stream of images. We use a hybrid approach towards robust star detection and star coordinate tracking.

2.1 Star Detection Heuristics:

For each image in the stream, we estimate the pixelwise mean and variance over the image by a two step process. The mean μ of all pixels is calculated, and pixels with intensity in range $\mu \pm 2\sigma$ are annotated as 'normal', with the motivation of excluding effects of stars, hot pixels (due to damage to the camera chip) and eliminating dark noise. The normal pixels are taken to be representative, and their mean μ_e and deviation σ_e are taken as the estimated order statistics for the image.

The pixels with intensity greater than $\mu_e n \sigma_e$ are taken as potentially belonging to stars. Connected components of such pixels are calculated, and large enough 'bright connected components' (BCC) are taken as stars. This heuristic is see to work well. The choice of n however remains important,and values of 3 and 4 are found to work for all tracking purposes. In the context of tracking, this provides a sufficient margin of allowance for change in intensity of the guide star, which is important due to effects of twinkling, cloud interference and other atmoshpheric phenomenon.

2.2 Star Coordinates:

Since realistic 'point spread functions' (or star convolutions) for astronomy can take upto several pixels due to atmospheric blurring, irregular refraction effects, and lens imperfections; images of stars are not represented as points. Thus it becomes imperative to accurately predict star centers from convoluted images, when the convoluting functions are unknown.

The differential amount of photon reception and corresponding intensity at different pixels can be used to locate the actual position of the star with a subpixel accuracy. We estimate the actual position of a star as the weighted centroid of the corresponding BCC. If w_{ij} is the intensity at point $x_{i,j}$ of the component, the weighted centroid is given as:

$$\bar{y} = \sum w_{ij} * x_{i,j}$$

The accurate specification of the position of a star in successive frame captures is important for the purpose of tracking. Once the guide star is chosen, in each successive frame, the displacement computation by weighted centroids is supplemented by two-dimensional cross correlation between the successive frames. The cross correlation indicates that alignment (linear translation) of the second image which has the maximum overlap with the first in terms of dot product of the two images, normalized by squared pixel values.

The supplementary cross correlation calculation is useful, since it uses *all* features in the two images for computation of displacement, and not just the local BCC. However, the method does'nt directly yield sub-pixel predictions, and is computationally expensive for whole images as indicated in Section 5. Thus, cross correlation is employed on a section of the image in the vicinity of the star, and is especially useful if a second bright feature appears in the field of view, in close proximity of the guide star.

During our simulations using the above approach, coordinates of a point object with a Gaussian filter, and moving along a sinusoidal curve at sidereal speed (earth's rotational speed) could be located with confidence within 2 arc seconds. The approach is seen to be robust to the effects of Poisson noise that can be used to model characterstic behvaiour of imaging devices at short exposures (since the reception of 'noise' photons by any pixel in a short duration occurs at approximately constant rate, and is naturally modeled as a Poisson process). The approach also works well with added random Gaussian noises that better model higher exposure rates. Thus, the approach is expected to support any guiding strategy built on it.

2.3 Alignment

The RA and Dec axes of the earth in general would not match with the X-Y axes of photographic devices, since the latter depend on the physical orientation of the camera. Before actual guiding, the relative positions of these axes needs to be calibrated to express star displacements as shifts along RA and Dec axes.

We do this by successively issuing medium duration directional commands along RA and Dec axes in isolation, and checking the displacement in the camera coordinates to determine the angle between the axes. Hereafter we can apply a rotational transformation on the axes, using an *axes rotation* matrix to convert camera coordinates of a point along the RA- Dec axes.

$$R = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix}$$

It is important to note that order of RA and Dec axes changes to Dec-RA in the normal anticlockwise convention.

3 Learning Approach

In this section, we first describe a rule-based autoguider, and then describe our learning approach in detail. We formulate the tracking problem as a Markov Decision process; next we propose a solution in terms of latent parameters, and an algorithm for the estimation.

3.1 Rule based

The star detection and corrdinate tracking heuristics described in Section 2 can be used to build a rule based autoguider, using a stepwise greedy strategy. The autoguider samples new frames at a constant rate (twice per second) through an imaging device. In each time window, the mount moves in the same direction as the previous error, and for the appropriate duration(considering the uniform speed of the mount); so as to attempt to rectify the observed error in the last frame. Since motion of the mount is ideally avoided, no correction is made if the noted displacement between the intended and actual positions is less than 0.1 pixels(In our experiments, even this setting leads to almost constant motion of the mount, especially if the mount alignment is not ideal, or when tracking a complex motion).

3.2 Learning Formulation and theory

We pose the problem as a continuous state reinforcement learning scenario, with agent 'actions' corresponding to motor movement durations in a classical agent learning scenario. For each of the RA and Declination axes independently, we can manipulate the direction of motor movement and its duration. The state of the mount at any instant is decided by the current motion of the mount, and displacement from the intended position.

Let X_t describe the state of an agent(controlling the mount) at time step t. Let the immediate expected reward that the agent earns on performing action u_t be $r(X_t, u_t)$. Let $\pi(X_t)$ be a policy that maps states to actions. Then, using an infinite horizon model with discounted rewards, the expected long term return of a policy is given by:

$$\delta(\pi) = E(\sum_{t=1}^{\infty} \gamma^{t-1} r(X_t, \pi(X_t)))$$

where γ is the discount factor. Let value function $V(X_t)$ be the long term expected return starting in state X_t at time t, while $Q(X_t, u_t)$ be the long term expected return on taking action u_t in state X_t at time t. Then,

$$V(X_t) = E(\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_t | X_t)$$

$$Q(X_t, u_t) = E(\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_t | X_t, u_t) = r(X_t, u_t) + \gamma V(X_{t+1})$$

Define u_t, the optimum action at time step t as the action which yields the maximum return in the long term, starting from a given state.

$$\begin{aligned} \mathbf{u}_t &= argmax_{u^\star} Q(X_t, u^\star) \\ &= argmax_{u^\star} r(X_t, u^\star) + \gamma V(X_{t+1}) \end{aligned}$$

The intention is to learn the optimum reward policies by iteratively inferring better estimates for reward policies and $Q(X_t, u_t)$

In our modeling of the dynamics of an autoguider, the state X_t comprises of the current displacement θ_t of the autoguider from the intended position (squared error) and the current motion (direction and duration) of the mount α_t . The one dimensional control input (action) u_t at each step corresponds to the difference in the new command from the previous command: change in duration, and possibly direction (if the control input is large enough).

This is approximately shown as a Markov Process as,

$$\begin{pmatrix} \theta_{t+1} \\ \alpha_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_t \\ \alpha_t \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t$$

$$\theta_{t+1} = \theta_t + \alpha_t$$
$$\alpha_{t+1} = \alpha_t + u_t$$

The first equation, that models the transfer function as $\theta_{t+1} = \theta_t + \alpha_t$ is a seemingly reliable guess about the process, but not a necessary assumption for what follows. The second equation reveals α as a regularizer for movement sequences. The control input doesnt directly manipulate the next action, but through α. This is intended to smoothen movements of the mount.

We model the reward function as follows:

$$r_t = -(k_1 \times {\theta_t}^2 + k_2 \times {\alpha_t}^2 + k_3 \times {u_t}^2)$$

The first term is most significant, and penalizes the error in position; the second penalizes very long commands. This is important since potentially a new action is decided at each time step, so command lengths should not exceed the sampling rate. The third term punishes large changes in mount motion. The choice of the reward function and coefficients k_1, k_2 and k_3 can critically change the working of the autoguider. While the optimum values are empirically arrived at, we expect the first coefficient to be dominating.

$Q(X_t, u_t)$ can be expressed in terms of basis functions ${\theta_t}^2$, $\theta_t \alpha_t$, ${\alpha_t}^2$, and $\theta_t u_t$, $\alpha_t u_t$ and ${u_t}^2$. Let us assume the form of the function such that the dependence is linear in the basis functions, and is given by :

$$Q(X_t, u_t) = \phi(X_t, u_t)^T w$$

where the vector ϕ consists of the six basis functions. Maximizing Q with respect to u_t yields the following relation:

$$\frac{d}{du_t} Q = \theta_t w_4 + \alpha_t w_5 + 2 u_t w_6 = 0$$
$$\Rightarrow {u_t}^* = -\frac{\theta_t w_4 + \alpha_t w_5}{w_6} = \pi^k X_t$$

Thus we can get the optimum value function as:

$$V^*(X_t) = max_{u_t} Q(X_t, u_t) = Q(X, u^*)$$
$$= \phi(X_t, \pi(X_t))^T w$$

Thus for the optimum strategy, we have:

$$Q(X_t, u_t) = r(X_t, u_t) + \gamma V^*(X_{t+1})$$
$$\Rightarrow \phi(X_t, u_t) w = r(X_t, u_t) + \gamma \phi(X_{t+1}, \pi^k(X_{t+1}))^T w$$

Bringing in the matrix vector form, we define Φ, $\hat{\Phi}$ and R as:

$$\Phi = \begin{pmatrix} \phi(X_1, u_1)^T \\ .... \\ .... \\ \phi(X_n, u_n)^T \end{pmatrix}, \hat{\Phi} = \begin{pmatrix} \phi(X_2, \pi^k(X_2))^T \\ .... \\ .... \\ \phi(X_{n+1}, \pi^k(X_{n+1}))^T \end{pmatrix}, R = \begin{pmatrix} r(X_1, u_1) \\ .... \\ .... \\ r(X_n, u_n) \end{pmatrix}$$

Using the matrix notation of the Bellman equation, we get:

$$\Phi w = R + \hat{\Phi} w$$

The matrix $\Phi - \gamma\hat{\Phi}$ is rectangular, and thus inversion is ill-defined. However, through a pseudo inversion, the solution can be seen to be:

$$w = (\Phi^T(\Phi - \gamma\hat{\Phi}))^{-1}\Phi^T R$$

Thus, we begin with π_0 as any random initial policy; and iteratively run the policy to get values for X_i, u_i and r_i. Using these observations to compute Φ, $\hat{\Phi}$ and R; we solve for the prescribed w using the formulation above. The convergent value of w after iterating through the process is the required optimum policy.

We address the 'exploration-exploitation dilemma' for the initial policy by making error corrections in the appropriate directions, while offsetting the corrections by adding a random ratio of of the intended duration.

There are also natural asymmetries inherent in the working of the autoguider, which could be encoded into the model rather than be learnt from scratch. Since the RA motor is always tracking as sidereal rate, and added motion is only superimposed on the existing velocity; the gears that perform the motion are free from any latency due to backlash. However, this is not the case along the Declination axes, and when a change in direction is involved, the motor command should be larger than calculated to account for backlash. i.e. if $sgn({u_{t-1}}^{Dec})$ is different from $sgn({u_t}^{Dec})$, the Dec motor command is modified to $u_t + \alpha$.

4 Experimental Settings

4.1 Mount details

A Mach1GTO German Equitorial mount with a GTOPC3 control box is used for the experiments in the study. The mount has identical dedicated motors for Right Ascension (RA) and Declination (Dec). Once aligned to the Pole star, the RA motor of the mount moves along at sidereal speed, to compensate for the motion of the earth. The purpose of the autoguiding is to provide additional modifications for exact tracking. Shoestring Astronomy's GPUSB guide port interface is used for the issue and deassertation of commands. Detailed descriptions of the mount and the Guide Port are freely available on the web. The following commands relevant to tracking can be made:

1. Start/stop motion in +RA direction
2. Start/stop motion in -RA direction
3. Start/stop motion in +Dec direction
4. Start/stop motion in -Dec direction

The speed of the motion can be set to 0.25, 0.5, or 1 times the sidereal speed. It should be noted that the commands from the autoguider are up and over the regular Sidereal tracking of the mount. Since the maximum possible speed of the autoguider equals the sidereal speed, if the sidereal tracking is off, the mount can't follow a star based solely on the autoguider.

4.2 Set up

Extensive indoor simulations with a range of different star convolutions were made on a LCD screen at a distance of 5.7 m from the camera for determining the optimum star detection and coordinate prediction heuristics. Variants including large Poisson noise in the background, multiple stars, and blinking stars were also experimented with. For predicting coordinates, simulations were made in which stars traversed predetermined pixel paths on the laptop screen, along both linear and sinusoidal paths(on the laptop screen). Validation was done by comparing predicted pixel location by the camera (using the lab machine at one end) with actual location on the screen(on the LCD screen at the other end of the lab). The task was made harder because of considerable glare from the otherwise black screen. Indoor learning algorithm implementation and star-tracking simulations were made by fixing a laser gun on the mount moving at sidereal speed, and tracking it through a stationary ccd camera. Simulations were

made in complete darkness (at night).

Preliminary outdoor testing was done on a wooden floor. The setup was susceptible to vibrations of the floor due to walking (see video footage). Both the rule based and the proposed learning based autoguider were evaluated with respect to average error(displacement from intended position) per frame over a time window. In case of the learning autoguider, error was calculated only for the testing phase. The preliminary exploratory training phase was made to run until convergence of the algorithm; and subsequently testing was made. In general, the training phase lasted between 3 to 10 iterations; and between 1 to 3 minutes in duration. For some runs however, the training duration greatly exceeded this and the runs were terminated. The same evaluation procedure was also done in an indoor setting for both autoguiders on a complex sinusoidal star path.

The GPUSB SDK by Shoestring Astronomy offers support through a Visual C API. However, for reasons of efficiency and easy portability, the autoguider and cameras have been implemented in MATLAB, and using dynamic libraries to connect with the Guide Port USB. Specifically, video image streaming is directly accomplished from a Firewire camera using the Matlab Image Acquisition Toolbox.

5 Results

5.1 Star coordinate determination

In the indoor simulations; using Gaussian, Poisson and Astronomy specific star convolutions on an LCD screen at a distance of $\approx$ 5.48 m were used to validate the robustness of coordinate determination approach. At this distance, the sidereal speed of the earth comes to around 0.398 mm/s, translating to roughly 2 pixels/s on the LCD screen.

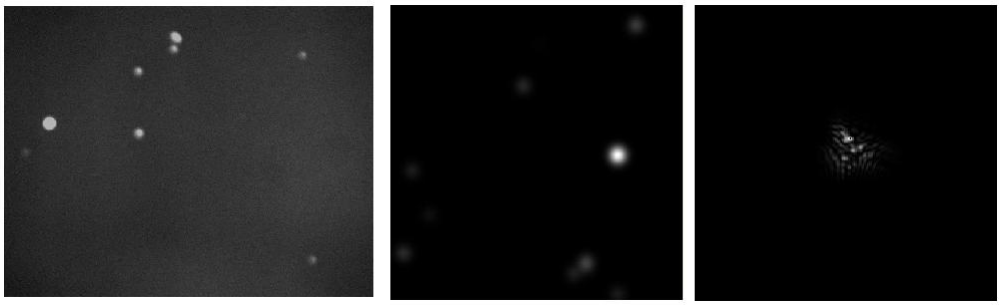

Figure 1: Representative point spread functions. From left to right: 1)Gaussian convolution with Poisson noise 2)Gaussian PSF with large spread 3)Short exposure PSF of star (domain specific)

Figure 1 shows three sample Point spread functions used for the validation. Table 1 shows average pixel errors during star coordinate tracking on predetermined trajectories using the suggested preliminary heuristics. Table 2 shows representative times for cross correlation computation on subimages of different sizes. In all subsequent experiments, the cross correlation was computed for sub-image of size 50×50 around the expected position of the star.

Convolution	Star Trajectory	Error (in pixels)
Gaussian	Sinusoidal	0.19
Gaussian with Poisson noise	Linear	0.22
Gaussian with Poisson noise	Sinusoidal	0.29
Short duration star PSF	Linear	0.34

Table 1

Image size	Time (in seconds)
50 $\times$50	0.0468
80 $\times$80	0.1404
100 $\times$100	0.2964
120 $\times$120	0.5404

5.2 Rule based guider

The rule based guider was first tuned for optimal performance before comparing with the proposed approach. The autoguider could track simulated star trajectories with high precision (even without sidereal tracking of the mount) if the star velocity was as high as 1/2 of the earth's sidereal speed. Representative results are mentioned in Table 3.

Star's speed	Star trajectory	Avg error (RA)	Avg error(Dec)
1/2 $\times sidereal$	Linear	0.2671	0.0407
1/2 $\times sidereal$	Sinusoidal	0.6285	0.2963
2/3 $\times sidereal$	Sinusoidal	1.778	1.6551

Table 3

5.3 RL autoguider

The proposed method was tested in both complex indoor simulations (following sinusoidal curvature), videos of actual stars(but played at different angular speed), and for outdoor star-tracking. In all cases, the tracking using reinforcement learning was immediately preceded by the rule-based autoguider for reasonable comparision of results. The training phase lasted about 3 minutes in most cases. However, in a few cases, convergence was not reached in a reasonable duration, and these runs were terminated. The RL autoguider is seen to do do better than the naive rule-based autoguider for both indoor and outdoor runs. Some representative results are documented in tables below.

1) Average errors for indoor sine trajectory (without sidereal tracking of mount)

	Avg error (RA)	Avg error(Dec)	Star speed
Net displacement	24.198	-28.650	-
Naive rule based	0.465	0.489	Low
RL based	0.294	0.406	Low
Naive rule based	-0.358	0.6545	High
RL based	0.1866	0.5285	High

Table 4

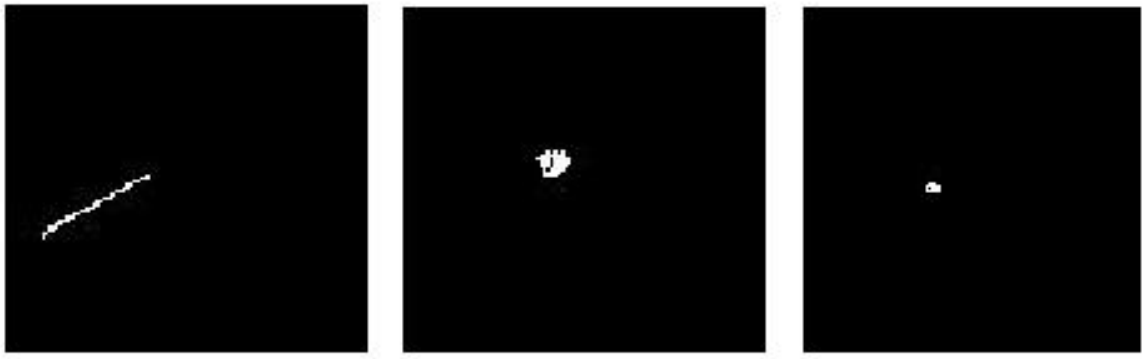

Figure 2: Tracking on segment of sinusoidal trajectory, each white dot indicates position of star on screen at different instants during tracking. From left to right: 1)No tracking(actual path) 2)Tracking with naive rule based autoguider 3)Tracking with Reinforcement learning approach

2) Average errors for outdoor autoguiding (with sidereal tracking)

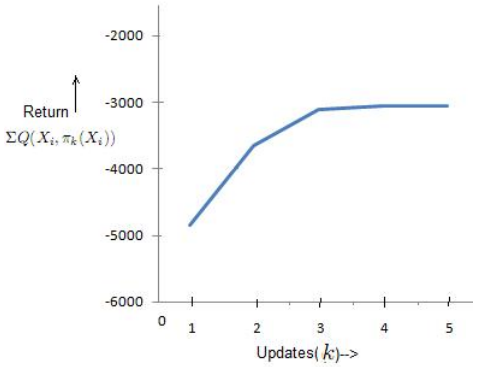

Figure 3: Expected return Vs Updates for a sample run of RL autoguider.

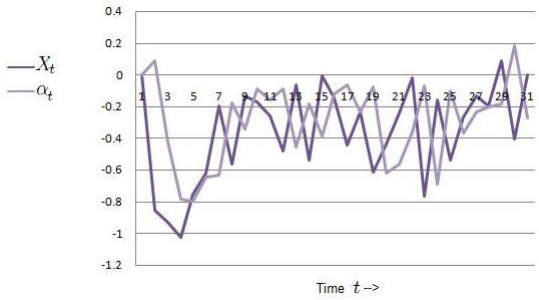

Figure 4: Mount motions and spatial errors along RA axes for a short sample run

	Avg error (RA)	Avg error(Dec)
Sidereal only	0.4103	0.2342
Naive rule based	0.0837	0.0358
RL based	0.0533	0.0378

Table 5

The optimum action is given by $\frac{w_4 X + w_5 \alpha}{2w_6}$. After convergence of learners for both RA and Dec axes, signs of w_5 and w_6 are typically the same, whereas w_4 has the reverse sign, suggesting a typical optimum strategy of form $u_1 X - u_2 \alpha$ where u_1 and u_2 are both positive. This suggests that the incremental difference in the new command should be in a direction to reduce the error and proportional to its magnitude; whereas an already large value of α tempers the new command, thus acting as a stabilizer.

6 Conclusion

The proposed approach shows a promising direction for statistical autoguiders. The suggested heuristics for star coordinate prediction is seen to be robust to noise, and real life scenarios such as cloud cover, leading to temporary disappearance. The proposed idea for a reinforcement learning based autoguider shows potential for star tracking in both indoor simulations and outdoor studies. As part of future work, comprehensive outdoor simulations are needed for testing and fine tuning since most of the experiments yet have focused on indoor studies, as also against proprietary software such as PHD guiding. Significantly, the stability of the learning approach was found to be suspect in some of the runs(which had to be terminated); and hence heuristics about initial policies that lead to

faster convergence need to be figured.

Also in the current study, the mount itself provided reasonably good guidance by its sidereal tracking alone. Differences in a rule based against a statistical approach might be more insightful for mounts where mount errors are of greater magnitude. One shortcoming in the outdoor study in this case was the instability of the mount due to a wooden platform. Hence small disturbances such as walking might have slightly affected observations. In addition to the two variable formulation proposed here; slightly more elaborate models with more complex states can be similarly formulated.

Figure 5: Setups for indoor simulations and outdoor tracking

References

1. Shoestring astronomy GPUSB. url: www.shoestringastronomy.com
2. Matlab: The language for scientific computing. url: www.mathworks.com
3. www.astrophysics.com
4. PHD Guiding. url: //www.stark-labs.com/phdguiding.html
5. Inverted autonomous helicopter flight via reinforcement learning, Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger and Eric Liang. International Symposium on Experimental Robotics, 2004
6. The Imaging Source Europe GmbH. url: www.theimagingsource.com