



Improving an Optical Flow Estimator Inspired by Insect Biology using Adaptive Genetic Algorithms

Phillip S.M. Skelton 

*Defence and Systems Institute
University of South Australia
Mawson Lakes, 5095*

South Australia, Australia


phillip.skelton@mymail.unisa.edu.au

Anthony Finn 

*Defence and Systems Institute
University of South Australia
Mawson Lakes, 5095*

South Australia, Australia

anthony.finn@unisa.edu.au

Russell S.A. Brinkworth 

*Centre for Maritime Eng., Control, and Imaging
Flinders University*

Tonsley, 5042

South Australia, Australia

russell.brinkworth@flinders.edu.au

Abstract—Computer vision algorithms that make use of optical flow are constantly increasing in complexity, especially in the context of elaborated algorithms that are heavily inspired by biology. To develop upon and utilise these algorithms for real-world tasks, their extensive parameter sets need to be tuned. Due to algorithmic complexity, and non-linearities present throughout their parameter set, this is no small task. Using an adaptive genetic algorithm, which itself is biologically-inspired, we look at the performance and behaviour of the tuning when significant changes have been made to a low speed rotational velocity optical flow estimation algorithm. We validate that previously reported changes to the optical flow estimator yielded a fitness increase of over 30% when compared to the baseline model the changes were made to, and a 15% increase once that baseline model was also tuned. This improvement would be extremely unlikely without the aid of evolutionary computation algorithms. This shows that even an extremely complex computer vision algorithm with many parameters, 36 in this case, can be tuned to an operating point, facilitating the continued development work on the vision algorithm by allowing for the validation and quantification of algorithmic changes.

Index Terms—Optical flow, genetic algorithm, computer vision, evolutionary computation, biologically inspired

I. INTRODUCTION

As computer vision algorithms increase in complexity, so too do the parameter sets required for their optimal operation [1]–[3]. While manual tuning of these parameters is sufficient for simple algorithms with few parameters, once non-linearities appear in the relationships between parameters, this quickly becomes a challenging task. While many methods exist for automating the tuning of these parameters, evolutionary computation algorithms are of particular interest due to their mimicry of biology [4], something that computer vision algorithms are also increasingly deploying [5], [6].

This paper looks at the application of an adaptive genetic algorithm for developing upon and tuning a highly-elaborated, biologically-inspired optical flow estimation algorithm. Due to

the biologically-inspired nature of the optical flow algorithm, high levels of non-linear adaptation are present. This results in the algorithm functioning adequately in a degraded state, but not optimally, even when the parameter set is far from optimal. Thus if the parameter set is not optimal it is impossible to quantify changes to the algorithm. If the baseline model is not optimised then changes to the model may simply be the result of moving the system towards a more optimised condition, rather than representing true algorithmic improvement. Conversely, if changes to the model push the system away from its optimal operating point, then the changes may erroneously be seen as detrimental. Only if the parameter sets, both before and after the proposed changes, are in a close-to-optimal state can the changes be reliably quantified.

Full details of the algorithmic changes made to the existing algorithm from literature, BIV09¹ [7], to form the updated algorithm, BIV19, can be found in [8]. The work previously presented includes both comparisons to other state-of-the-art optical flow algorithms, and input response characterisation of those algorithms. The reader is directed to [8] for that information as it is outside of the scope of this paper.

II. PREVIOUS WORK

Research into biological vision systems and their implementation as engineered functions has shown that these models often contain a large quantity of parameters that can be tuned to produce different operating behaviours depending on the situation tested [7], [9]–[12]. When searching for the optimal set of parameters, there are two fundamentally different methods for obtaining that solution: classical search methods, which encompass basic techniques such as exhaustive search (enumerative) or iterative methods; and modern heuristic methods, which find a global solution based on some fitness function that is driving the optimisation [13].

Iterating through multiple variables that have non-linear interactions with one another is a computationally expensive process to gauge how the non-linear interactions between the

Corresponding author: Phillip S.M. Skelton
Extensive computational resources were donated by the University of South Australia's High Performance Computing cluster. PSMS was supported by an Australian Government Research Training Program Scholarship.

¹BIV is an acronym of Biologically Inspired Vision.

variables affect the model output as a whole. Even a simple 3-value approach to each variable (for example, an initial guess and initial guess plus and minus an offset), would require no less than 3^n iterations, where n represents the population size of variables. This also does not guarantee that the 3 values selected for each variable are optimal, or even close to it.

Classical search methods, also known as direct search methods, have been used to optimise algorithms since the early 1960's [14]–[16]. These methods are characterised by the lack of a differential approach to the problem. That is, they are 'derivative-free', and operate without a cohesively constructed model of the output of the system [17]. The Nelder-Mead simplex method is not only one of the most widely used, but arguably the foundation direct search method [18], [19]. It attempts to minimise the non-linear, scalar-valued function of variables using only function values. That is, it does not require implicit or explicit derivative information, but the function is assumed differentiable. These techniques were largely rejected by the mathematical and scientific community in the 1970's due to slow solution times, the tendency to get stuck in local optima, and a lack of mathematical proof behind their convergences [20]. However, they regained popularity in the 1990's due to the advent of parallel/distributed computing technologies that overcame the extensive solving times. Much stronger mathematical convergence analyses also gave credibility to the algorithms [21].

Modern heuristic methods can also be classified depending on the nature of the problem that the algorithm is simulating; for example evolutionary algorithms, and swarm intelligence based algorithms [22], [23]. Evolutionary algorithms are suited to real-world problems that feature multiple, and quite often competing, optimisation objectives [24]. Genetic algorithms, a subset of evolutionary algorithms, operate on the principal of natural selection. A population of possible outcomes is spawned and the fitness of each member of that population is quantified against a user-specified fitness function, with the most suitable child going on to seed the next generation [25]. Determining the fitness of a generation to the problem at hand can be approached in a multitude of ways depending upon the desired outcome of the problem, such as maximising profit, minimising solution time, minimising travel time, etc. [26].

There exists numerous strategies for approaching both single-objective and multi-objective problems. While single-object problems are typically less complex than multi-objective, they are still the source of constant development of strategies to improve their efficacy, such as the adaptive probabilities widely used from [27] or adaptive traits specific to an application [28]. The extremely popular nondominated sorting genetic algorithm II (NSGA-II) [29], [30] has been a staple of solving multi-objective algorithms for many years, and has been the focus of attempts at reducing the run-time complexity of multi-objective tuning [31].

For these problems, there is often no single clear solution. Instead, these algorithms aim to achieve what is known as a Pareto-optimal solution [30]. Pareto-optimality infers that there are no other combinations of parameters in which the

improvement of one individual within the population (in this case, a model parameter) would not adversely affect at least one other individual, with no net benefit to the system as a whole [32]. However, it has been shown that arriving at the final Pareto-optimal solution from a large number of Pareto-optimal possibilities, even after ruling out the non-Pareto-optimal solutions, is a largely subjective process based on the scaling that each individual has been given by the user [33]; or, more specifically, by the fitness function used to evaluate the suitability of any given child [34], [35].

Although optimisation algorithms are an extremely diverse and mature field, there does not appear to be a set standard researchers can follow for selecting an optimisation algorithm for their specific problem. Instead, there exist multiple approaches towards finding optimal solutions, with key factors such as time-to-solution, solution accuracy, ability to handle multi-objective, multi-constraint, and multi-modal parameter sets key factors that drive algorithm selection.

A. Background

The visual processing algorithm tuned in this paper is a highly-elaborated biologically-inspired optical flow estimation algorithm that is heavily inspired by the visual pathways of insects such as flies, bees, and dragonflies. It has applications in autonomous vehicle localisation and navigation [8], and derivatives have applications in target detection [36]. The original baseline algorithm, known as BIV09, can be seen in Fig. 1(a). Several alterations have previously been made to BIV09 to form BIV19 (Fig. 1(b)), as reported in [8]. Both of these algorithms contain models of the photoreceptors (PR), lamina monopolar cells (LMC), elementary motion detectors (EMD), medulla-lobular interneurons (MLI), and lobula plate tangential cells (LPTC), modelled as H1-H5 horizontally-selective cells. The base algorithm, BIV09, has been shown to have a synergistic relationship. That is, the performance as a whole is greater than the sum of the individual components [7]. It has also been shown to have excellent noise rejection characteristics [37]. It is worth noting that these algorithms are non-stochastic. That is, for a given input data and parameter set, the algorithms will produce the same results. Therefore, in mathematical terms, it is said to be deterministic.

Both BIV09 and BIV19 were applied as rotational velocity estimators. The dataset on which these algorithms were tuned consisted of 12-bit monochrome panoramic videos of 16 different scenes captured at 11 different rotational velocities by a single degree-of-freedom (rotation/yaw about Z axis) robotic platform. For each velocity and scene pair, the input frames were processed by both algorithms, and each algorithm estimates the amount of optic flow energy that was present for that combination. The full dataset consists of 1001 frames per recording, and was captured at a nominal frame rate of 100.027 frames per second. Examples of indoor and outdoor scenes featuring both sparse (large image areas with no information, such as the sky) and dense feature distributions throughout the scenes can be seen in Fig. 2. See [8], [38] for full dataset methodological information.

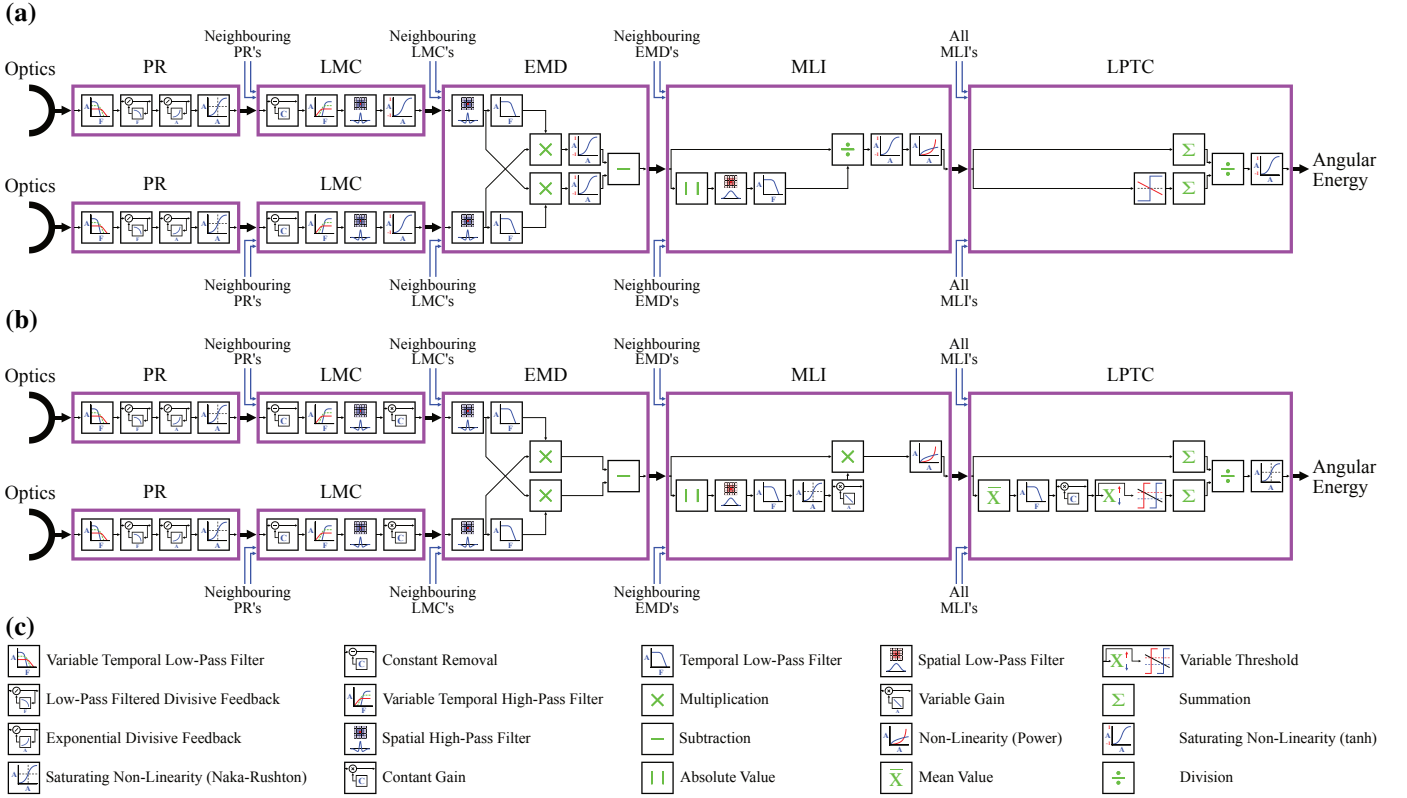


Fig. 1. Diagrammatic representation of: (a) The existing biologically-inspired vision algorithm from [7], BIV09; (b) Our further elaborated biologically-inspired vision algorithm from [8], BIV19; and (c) The legend associated with these diagrammatic representations. The algorithm contains models of the photoreceptors (PR), lamina monopolar cells (LMC), elementary motion detectors (EMD), medulla-lobula interneurons (MLI), and lobula plate tangential cells (LPTC) modelled as H1-H5 horizontally-selective cells.²

III. METHODOLOGY

A. Tuning Algorithm Being Used

This work used an Adaptive Genetic Algorithm (AGA) as outlined in [27]. Due to the non-linear relationship between each gene, which themselves have a non-linear impact on model performance, it was entirely expected that this model would be multi-modal within the solution space.

1) *Fitness Function*: Arguably the most important aspect of evolutionary computation algorithms is the fitness function to which it adheres. We have used a published metric [8], known as an adjusted geometric score, or G_{adj} , that quantifies the ability of an algorithm to statistically discriminate between optical flow outputs across different scene and velocity pairs. First, the distinctiveness of a response is calculated:

$$D_i = \text{PPD}_i * \frac{(P50_i - P50_{i-1})}{(P50_i - P5_i) + (P95_{i-1} - P50_{i-1})} \quad (1)$$

Where i is the index of the current rotational velocity, $i-1$ is the previous index, D is the distinctiveness score, $P5$ is the 5th percentile, $P50$ is the 50th percentile (median), $P95$ is the 95th percentile, and the goal is $\max(D_i)$. The PPD_i term is the number of test points per decade, calculated using:

$$\text{PPD}_i = \frac{1}{\log_{10}(\nu_i) - \log_{10}(\nu_{i-1})} \quad (2)$$

Where ν_i is the rotational velocity at index i , and ν_{i-1} is the rotational velocity at index $i-1$. This correction was required to account for unevenly spaced sampling velocities. To account for non-Gaussian distributions of the data, the raw geometric mean of D , \bar{G}_{raw} , was calculated across the entire dataset by means of a natural logarithm transform:

$$\bar{G}_{raw} = \exp\left(\frac{1}{N_v - 1} \sum_{i=2}^{N_v} (\ln(D_i))\right) \quad (3)$$

Where N_v is the number of velocities. \bar{G}_{raw} is thus a measure of how the statistical distribution of motion energy responses differ between test velocities. The geometric confidence intervals were also calculated using a natural logarithm transform and are used to demonstrate the variability of \bar{G}_{raw} throughout the range of tested velocities. The final metric, the adjusted geometric score G_{adj} , accounts for the confidence range:

$$G_{adj} = \bar{G}_{raw} - (G_U - G_L) \quad (4)$$

Where G_U and G_L are the upper (95%) and lower (5%) geometric confidence intervals, respectively. G_{adj} is a measure

²Adapted from Image and Vision Computing, Volume 92, Phillip S.M. Skelton, Anthony Finn, Russell S.A. Brinkworth, Consistent estimation of rotational optical flow in real environments using a biologically-inspired vision algorithm on embedded hardware, Copyright (2019), with permission from Elsevier.



Fig. 2. Example scenes from the panoramic rotational motion dataset used from [8] after they have been processed by the BIV19 photoreceptor model and adjusted for display purposes. Please see online version for full details in the images, specifically the pixelation caused by the low optical resolution of 180x36. The density of the scenes, namely sparse (large areas of no structure) and dense (mostly structured), refers to the Local Contrast Factor (LCF) measurement shown in [8]. Row 1: Sparse outdoor scene (large sections of sky). Row 2: Dense outdoor scene. Row 3: Sparse indoor scene (large sections of blank walls). Row 4: Dense indoor scene.

of not only how well an algorithm can discriminate between test velocities, but also the consistency of those discriminations across the dataset. It is scale and unit invariant and thus requires no standardisation of algorithm outputs, allowing for direct statistical comparison between numerous algorithms.

2) *Parent Selection:* Stochastic Universal Sampling (SUS) [39] was used for parent selection. The SUS method, shown in Fig. 3, samples the population at evenly spaced intervals with a random starting offset. This method gives weaker children a chance to become parents, which may produce a stronger child in the following generation. As the children were all hermaphroditic, parent selection was only constrained to being unique between each pair. The number of children selected as parents using the SUS method was empirically set to 10% of the population size.

3) *Crossover Operation:* The probability of the crossover operation being applied to a parental pair was implemented as per the work of [27]:

$$p_c = \begin{cases} k_1 (f_{max} - f') / (f_{max} - \bar{f}), & f' \geq \bar{f} \\ k_3, & f' < \bar{f} \end{cases} \quad (5)$$

Where p_c is the probability of crossover, k_1 and k_3 are user-defined constants, f_{max} is the maximum population fitness for that generation, f' is the largest fitness of the parental pair, and \bar{f} is the mean population fitness. As per the recommendations in [27], $k_1 = k_3 = 1.0$.

The crossover operation was implemented as a 2-parent uniform crossover where each parent contributed equally to the

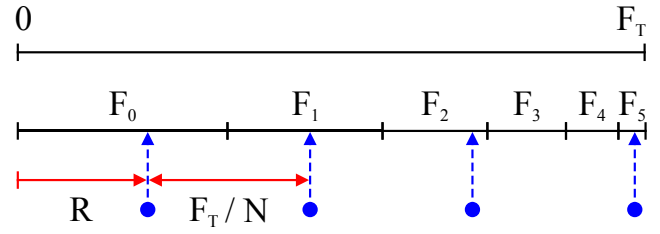


Fig. 3. Diagrammatic representation of Stochastic Universal Sampling, where F_T is the total population fitness, F_i denotes the fitness of the i^{th} child in a list sorted by descending fitness, N is the number of stochastic parents to be selected, and the random starting point $R \in \mathbb{R}[0, F_T/N]$. The lower dots represent the parent selection points.

genome of the child. The crossover points, or k-points, were calculated using a uniform integer distribution with a pseudo-random number generator generating an array of binary flags the length of the genome. If the number of k-points generated was equal to the number of k-points required — half the length of the genome in this work — those k-points were used. This method generated a random set of k-points without bias towards the higher-order genes that would be present when trying to force a fixed number of k-points sequentially.

4) *Mutation Operation:* The mutation operation was implemented in two distinct ways. First, the probability of the mutation operation being applied to a child was implemented as per the work of [27]:

$$p_m = \begin{cases} k_2 (f_{max} - f) / (f_{max} - \bar{f}), & f \geq \bar{f} \\ k_4, & f < \bar{f} \end{cases} \quad (6)$$

Where f is the fitness of the child being mutated. As per [27], $k_2 = k_4 = 0.5$ drove the probability of a genome being mutated. Parents with weaker phenotypes within their generation would possess a mutation probability that approaches k_4 , whereas stronger parents are capped at k_2 . To disrupt the gene pool, a global mutation of 0.5% probability was applied.

5) *Premature Termination:* For a child to be a valid member of the population, it must phenotype to a positive, non-zero fitness. To optimise the processing throughput of the tuning process, the suitability of each child is periodically examined. After all scenes at a given velocity have been processed by the child, the percentiles of the responses, which forms a part of (1) which is a component of the fitness score calculated using (4), can be used to validate the continued existence of that child. If the 5th percentile of the responses falls below 0, then the child is terminated. A child could be prematurely terminated anywhere from $i_v = 1$ through to $i_v = N_v$, but was most likely to occur at $i_v = 1$, as shown later in Sec. IV-F.

B. Genome

The genome for this study consisted of 36 unique genes that represented different parameters of the computer vision algorithm being tuned. These genes were all represented as IEEE-754 32-bit floating-point numbers to match the computer vision algorithm implementation. Genes were encoded

to fixed-point numbers and decoded back to floating-point numbers similarly to Alg. 1 for all gene generation and uniqueness checking operations.

1) *Genome Boundaries*: The majority of genes, although presenting varying levels of epistasis with other genes, did not have strongly linked boundary conditions with other genes. This was due to the non-linear relationships between the individual component models within the algorithm; and even within the individual component models there are non-linear relationships between mathematical functions. However, certain genes have direct relationships to surrounding genes, such as a maximum gain following a minimum gain, where the maximum gain must be a higher value than the minimum gain to avoid signal inversion or exponential runaway.

For each gene in the genome, the boundaries were set in one of two ways. First, the majority of boundaries were set based on known mathematical limitations, such as that which occurs in a temporal low-pass filter:

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1} \quad (7)$$

Where y_i is the filtered output at frame index i , x_i is the measured value of the system being filtered at frame index i , y_{i-1} is the filtered output at the previous frame index, and α is the smoothing factor given by:

$$\alpha = \frac{2\pi f_c}{2\pi f_c + f_s} \quad (8)$$

Where f_s is the sampling frequency (in this case frame rate) and f_c is the cut-off frequency of the filter. By definition $0 \leq \alpha \leq 1$, however, the gene boundaries were set to $0.001 \leq \alpha \leq 1.251$ as $\alpha = 0$ is contextually inappropriate, $\alpha > 1.0$ was clamped to 1.0 as this provided a 20% chance that the filtering would be disabled, and this provided equal spacing between boundaries (human readability). See results in Sec. IV-E, specifically Fig. 8a, for representation of this.

Another mathematical constraint is a linear gain. It may be desirable to attenuate ($g < 1.0$), maintain ($g = 1.0$), or amplify ($g > 1.0$) a signal. While exact signal magnitude is often less important than the mean or median in the floating-point domain, follow-on mathematical operations, such as an offset removal or Naka-Rushton non-linearity transform, have specific signal operating ranges required for correct operation.

The second kind of boundary condition is arbitrary values. Unlike mathematical boundaries, an offset removal or intermediate gain are generally not mathematically bounded. Instead, their boundaries are set based on an understanding of the behaviour of the computer vision algorithm, and are constantly adjusted based on the behaviour of the tuning.

While the specific alleles of some genes could be informed based on biological recordings, as was the case with many of the photoreceptor values used in historical models [40], this assumes 100% mimicry of the biological system; something that is not currently technologically possible.

2) *Genome Seeding*: For studies where the genomes of all children within the first generation are randomly generated,

Algorithm 1: Generation of uniformly distributed random floating-point number between boundaries.

Input : $B_{L,F}$ - Floating-point lower boundary.
Input : $B_{U,F}$ - Floating-point upper boundary.
Input : S - Number of uniform steps.
Input : P - Precision of float-to-int conversion.
Output : R_F - Random float uniformly distributed in $[B_{L,F}, B_{U,F}]$

- 1 Convert boundaries to integer representations:
 $B_{L,I}, B_{U,I} \leftarrow P * (B_{L,F}, B_{U,F})$
 - 2 Generate uniformly random integer: $X_I \in \mathbb{N}[0, S]$
 - 3 Calculate integer representation of random float:
 $R_I \leftarrow B_{L,I} + X_I * (B_{U,I} - B_{L,I}) / S$
 - 4 Convert back to float representation: $R_F \leftarrow R_I / P$
-

each gene had a random value assigned using Alg. 1. Certain studies, discussed later, may have used the genome of a child from a previous study as the seed.

3) *Child Uniqueness*: Uniqueness of each child was enforced both within the current generation, and against all previous children that failed to phenotype to a positive fitness³. Although elitism was not directly implemented, it was not explicitly forbidden, so the genome of a child that phenotyped to a positive fitness can appear in a later generation.

4) *Gene Behaviour Types*: Throughout the tuning process, there are different types of behaviour that genes can express. In this research, these are defined as 3 distinct types.

Type A responses occur when a gene prefers a value that is comfortably within the boundaries, ensuring that sufficient room for exploration exists around the operating point that has been determined to be appropriate for that specific genome.

Type B responses occur when a gene has strict mathematical boundaries. For these genes, having a preferential value that is close to, or even on, a boundary is perfectly valid. For example, when the leakiness of a high-pass filter, that is an intentionally non-perfect filter that allows some percentage of DC to pass-through, has been removed (gene = 0.0).

Type C responses occur when a gene has very little, or no, impact on algorithm performance. This tends to occur in situations where parameter sections in components of the model render a particular gene irrelevant. For example, removal of a constant DC offset is inconsequential if a following high-pass filter is ideal and removes all DC components.

C. Tuning Phases

Three distinct phase of work were undertaken for this paper. Each population in each phase consisted of 2000 children. Phase 1 consisted of extensive open-ended tuning of the existing baseline BIV09 algorithm to provide a foundation against which the improvements made to form BIV19 could be gauged. Phase 2 consisted of 30 unique studies, each

³There may be machine-specific errors in IEEE-754 representations, for example 15.2001 can be represented as 15.20009994..., but genome uniqueness was implemented in the integer domain as 15.20009994... rounds to 152001 after integer promotion (10E3 promoter) irrespectively.

containing 2000 children per generation. Due to the significant algorithmic changes implemented on BIV09 to form BIV19, the initial population of all 30 studies were randomised. Finally, Phase 3 consisted of an open-ended continuation of the strongest child from Phase 2, using it as the seed for a longitudinal study. This provided the means to analyse the steady-state behaviour of our tuning algorithm, as well as tune the maximum fitness from the BIV19 algorithm.

The computational resources required for Phase 1 were satisfied by a 24-core computational server over the course of ~ 6 weeks. The extensive resources required to undertake Phase 2 were provided by the University of South Australia’s High Performance Computing cluster, where an allocation of 280 computational cores for 336 hours walltime (14 days) was provided. Several study parameters, shown in Table I, were set around this allocation. Finally, Phase 3 was again undertaken on a local server over ~ 4 weeks. While the dataset analysed has 1001 frames available for each scene and velocity pair, only frames 0 to 200 were used as, from previous work, ~ 200 frames are known to provide sufficient duration for the short-term adaptive elements within BIV19 to stabilise [8].

D. Software Libraries

The C++ API of the OpenCV computer vision library (version 4.1.0), was used for both BIV09 and BIV19. The genetic algorithm was implemented from scratch in C++11-compliant code, with OpenMPI 4.0.1 providing the means to utilise distributed computing resources.

IV. RESULTS

A. Tuning Phase 1

Phase 1 commenced with BIV09 having a baseline fitness of 5.6 from previous work [8]. BIV09 was then tuned for 350 generations in total. Study termination was manually determined by ensuring all genes were exhibiting the appropriate type of behaviour, and by analysing the maximum fitness

TABLE I
SEVERAL PARAMETERS USED THROUGHOUT THE WORK PRESENTED IN THIS PAPER. PHASE 2, THE EXTENSIVE WIDESPREAD STUDY, HAD ADDITIONAL PARAMETERS RELATING TO RESOURCE ALLOCATIONS.

Description	Key	Value
Number of Genes per Child	N_{genes}	36
Number of Velocities per Child	N_v	11
Number of Scenes per Velocity	N_{sc}	16
Number of Frames per Scene	$N_{f,sc}$	201
\therefore Maximum Frames per Child	$N_{f,c}$	35,376
Number of Children per Generation	N_c	2,000
\therefore Maximum Frames per Generation	$N_{f,g}$	70,752,000
Number of Generations per Study	N_g	200
\therefore Maximum Frames per Study	$N_{f,st}$	14,150,400,000
Parameters Specific to Phase 2		
Number of Studies	N_{st}	30
\therefore Maximum Frames Studied	$N_{f,tot}$	424,512,000,000
Number of CPU Cores Available	N_{cpu}	280
Walltime Available	t_w	336 hours
\therefore Total CPU Time Available	t_{cpu}	94,080 hours

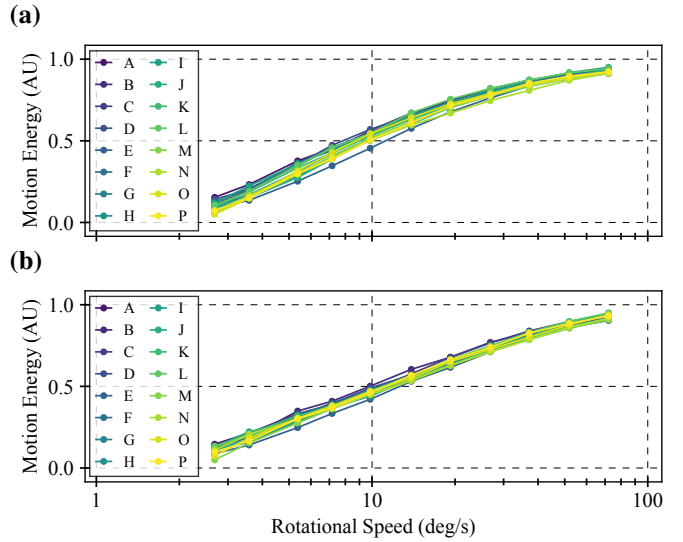


Fig. 4. Characteristic algorithm response curves for: (a) Baseline model performance for the existing biologically inspired vision algorithm, BIV09, using default parameters from literature [7], producing a fitness of 5.600; and (b) Tuned output of BIV09, producing a fitness of 6.515. Legend entries (A-P) denote anonymous scenes within the dataset of [8].

progression over time. The maximum fitness once tuned was 6.515; an improvement of 16.3%. Comparing the fitness of $i_g = 350$ to $i_g = 330$, a change of $\Delta f_{20} = -0.032$ was observed. Looking further into the past, changes of $\Delta f_{40} = -0.015$, $\Delta f_{60} = -0.025$, $\Delta f_{80} = 0.039$, and $\Delta f_{100} = 0.053$ were observed. These two factors — suitable genome boundaries and stability of the fitness over time — resulted in termination of this phase. The characteristic response curve for the baseline BIV09 model can be seen in Fig. 4(a), and BIV09 at the completion of Phase 1 is shown in Fig. 4(b). Visually comparing the shape of the responses between the default and tuned BIV09 outputs, it can be seen that the tuned BIV09 output presents as more of a linear response on the lin-log graph shown. The distribution of the responses at each velocity are also tighter, which is reflected in the improved fitness score.

B. Tuning Phase 2

Phase 2 commenced with 30 independent and randomly generated populations. The fitness response of each study over time can be seen in Fig. 5. From their random starting populations at $i_g = 1$, each study rapidly transitioned to a landscape of higher fitness. By $i_g = 11$, all populations achieved a fitness greater than 4.0. By $i_g = 50$, all populations exceeded 5.0. At the end of $i_g = 200$, the minimum fitness was 5.991, the mean was $\bar{f}_{max} = 6.794 \pm 0.460$, and the maximum was 7.449. Comparing the mean fitnesses at $i_g = 200$ to $i_g = 180$, $\Delta \bar{f}_{20} = 0.025$, shows only minor improvements. Looking further back, $\Delta \bar{f}_{40} = 0.092$, $\Delta \bar{f}_{60} = 0.163$, $\Delta \bar{f}_{80} = 0.256$, and $\Delta \bar{f}_{100} = 0.362$, the fitness improvement is slowing, but has not reached steady-state. The mean of the population at each generation can also be seen to be increasing, showing that diversity throughout the population is being maintained.

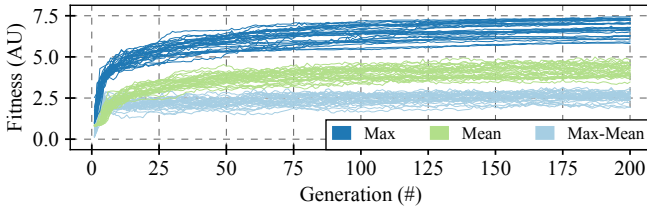


Fig. 5. Strongest child fitness (Max), mean population fitness (Mean), and distance of the mean fitness to the maximum fitness (Max-Mean) for Tuning Phase 2. A total of 30 independent studies were undertaken, all of which had random starting genomes for all 2000 children in each study population. Each independent study rapidly transitions from a low fitness at $i_g = 1$ through to a continually improving fitness by $i_g = 200$. The mean population fitness tracks the maximum, showing that we are maintaining diversity throughout the entire population.

C. Tuning Phase 3

Phase 3 commenced with the strongest child from Phase 2 (strongest child from Study #24) as the initial seed. The characteristic response curve of the strongest genome from Phase 3, and thus the most optimal response of this paper, can be seen in Fig. 6, which produced a fitness of 7.547. Comparing the fitness at $i_g = 200$ to $i_g = 180$, $\Delta f_{20} = -0.007$, shows no meaningful change in fitness. Looking further back, $\Delta f_{40} = 0.029$, $\Delta f_{60} = -0.064$, $\Delta f_{80} = -0.039$, and $\Delta f_{100} = -0.027$, it can be seen that the fitness has reached Pareto-optimality, where no change to one gene can improve the overall fitness. The fitness of 7.547 is a 15.8% increase over the tuned BIV09, and a 2.68% increase on the previously published methodological paper for BIV19 [8] which used a single longitudinal study to loosely tune the genome, but did not elaborate on the tuning or take the tuning to steady-state as this paper has done. This difference is within an acceptable error for repeated independent studies of an algorithm this complex, with factors such as genome boundaries and stepping the cause. Looking at the shape of the BIV19 response, it can be seen that a sigmoidal response on a lin-log graph is present, with both minimum and maximum responses being asymptotic with maximal slope in the middle of the range.

D. Genomic Landscapes

Due to the significant algorithmic changes between BIV09 and BIV19, the genomic landscape for BIV09 will not be shown as it bears no meaningful relationship nor direct comparison to the landscapes of BIV19. For BIV19, looking at the genomes of the strongest child from each individual study from Phase 2 (Fig. 7(a)), it can be seen that each independent study tended towards a very similar genomic landscape. Some variability in the genome was expected from a highly-elaborated biologically-inspired algorithm, where multi-modality of the solutions was an expected response.

The landscape of the strongest child from each generation of the strongest study from Phase 2, Study #24, is shown in Fig. 7(b). The older generations (darker colour) had genomes that were quite different from the younger generations (brighter colour). Certain genes expressed very strong

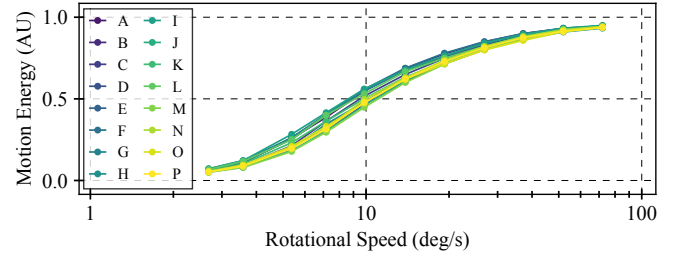


Fig. 6. Characteristic algorithm response curves for the tuned output of our BIV19 algorithm, producing a fitness of 7.547. Legend entries (A-P) denote anonymous scenes within the dataset of [8].

preferences for a given value. For example, Gene #31 has a strong preference regardless of the immediately preceding gene (#30), and the immediately following gene (#32).

Lastly, the landscapes from Phase 3 are shown in Fig. 7(c). These used the strongest child from Phase 2 (Study #24) as the initial population seed. It should be noted that the genome boundaries from Phase 2 were adjusted to improve resolution on certain genes, so the genomic landscape appears slightly different. The initial seed, which had a fitness of 7.449, had a couple of genes (8 and 9, for example) that further improved on the subsequent study, peaking at a fitness of 7.547.

E. Gene Behaviour

Critically analysing the responses of 36 genes is outside the scope of this paper. Instead, 4 individual genes that display one of the characteristic behaviours shown in Sec. III-B4 will be shown. Exact mathematical and algorithmic explanations, including in-depth analyses of all 36 genes and their operating ranges, will be reported at a later date.

The first gene, gene 03 (Fig. 8(a)), represents the smoothing factor α of a filter. This is a perfect example of the adaptability of the biologically-inspired algorithm. While there is a clear allele peak (~ 0.24), there is a wide range of alleles (0 to ~ 0.8) that allow the algorithm to function correctly, albeit in a sub-optimal state. Once the filter reaches the point where it is no longer enabled (shaded section from 0.8 to 1.0, refer to Sec. III-B1), the algorithm enters a drastically degraded state.

The second gene, gene 16 (Fig. 8(b)), represents the amount of DC pass-through that a high-pass filter will allow. As per Sec. III-B4, this gene exhibits Type B behaviour, where the preferential allele is on the lower boundary (0.0), meaning a perfect high-pass filter is preferred. However, as can be seen, several of the older generations had a different response, where the filter allowed varying levels of DC to pass through while still maintaining high operational fitness.

The third gene, gene 20 (Fig. 8(c)), represents the mid-point of a Naka-Rushton non-linearity adjustment on the DC component allowed through a high-pass filter. This gene exhibits Type C behaviour as the filter is having no impact on the performance of the algorithm. With a high-pass filter being tuned to be perfect (no DC pass-through), a dynamic DC pass-through is inconsequential. This is an example of a

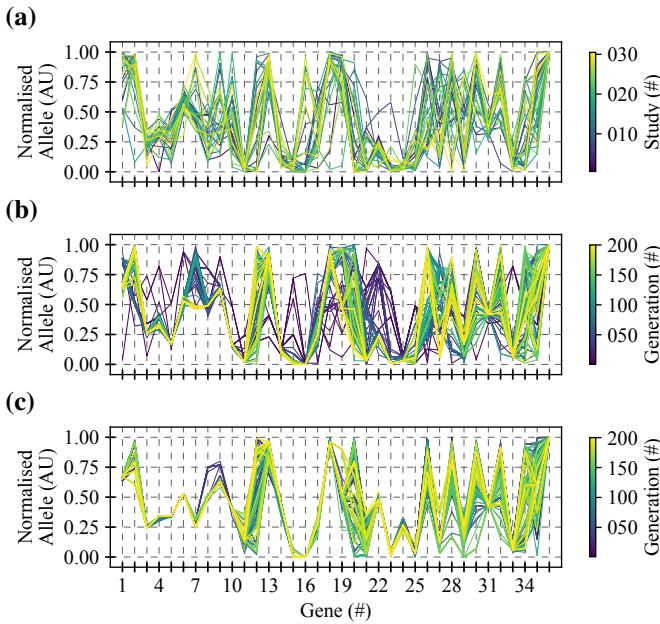


Fig. 7. Genomic landscapes for our BIV19 algorithm, all normalised to $[0.0, 1.0]$ for display purposes, for: (a) Overall strongest child from each study in Phase 2 (30 independent and randomly seeded studies), not necessarily occurring at $i_g = 200$; (b) Generational output of the strongest child from all studies in Phase 2, specifically Study #24; and (c) Generational output of Phase 3, where the strongest child from Phase 2 was used as the seed of the initial population. Note that several genome boundaries were adjusted between (b) and (c) to improve the response of certain genes (see genes 5 - 10 for example), therefore the landscapes will appear different.

redundant gene where the original purpose of the gene has been subverted by the operation of another gene.

The fourth gene, gene 22 (Fig. 8(d)), represents the factor used in a non-linearity adjustment. This gene exhibits Type A behaviour, where it can be seen there is a clear preference for the allele (~ 0.48) while sufficient distance is provided to the boundaries to allow for exploration.

Although BIV09 and BIV19 have identical photoreceptor (PR) models, and near-identical lamina monopolar cell (LMC), models (see Sec. II-A, Fig. 1), the alleles for these two models were drastically different between the two. This was an expected result due to the synergistic behaviour of the models as a whole algorithm. Specifically, the epistasis between seemingly unrelated genes, where one gene can be influential in, or influenced by, a gene from a different part of the algorithm. This kind of non-linear and non-direct interaction is extremely difficult to account for manually. However, it is something that an evolutionary algorithm excels at handling.

F. Processing Statistics

For Phase 2, the average time taken to phenotype each generation was $191.9 \text{ s} \pm 11.3 \text{ s}$. Given the number of frames processed per generation ($61,703,548 \pm 1,937,040$), this results in a throughput of $\sim 320,000$ frames per second, or $\sim 1,140$ frames per second, per core. In contrast, in a real-world deployment on an embedded computer, BIV19 (without pre-processing stages, similarly to the tuning) achieved ~ 400

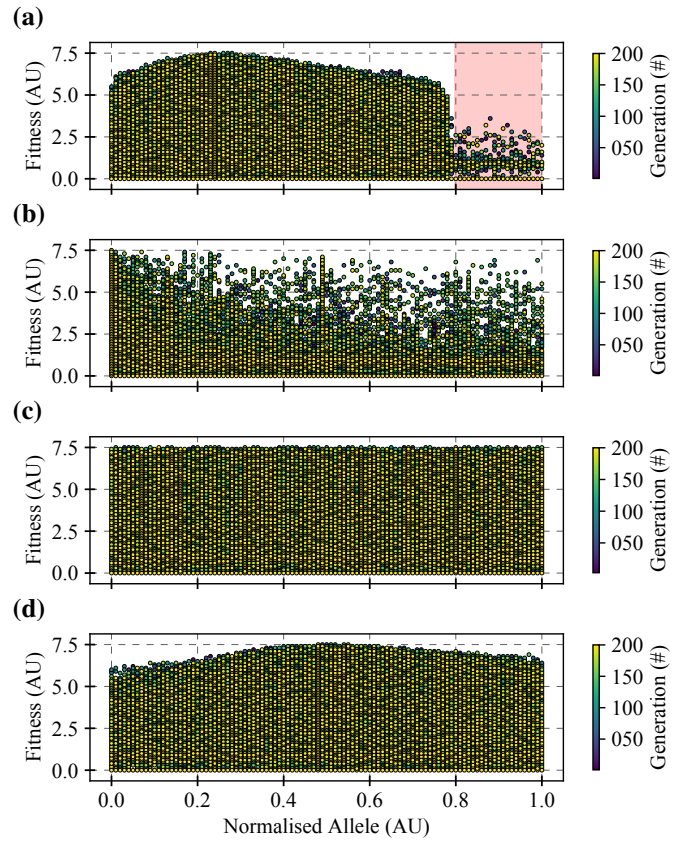


Fig. 8. Example outputs for selective genes from the strongest child of Phase 2. Fitness was calculated using (4). As elitism was not employed, older generations can produce a higher fitness. Full explanation of behaviour types can be found in Sec. III-B. Note that fitness resolution was 3 decimal points in practice, however have been decimated to 1 decimal point to facilitate graphing. (a) Gene 03, the smoothing factor α of a filter, where the shaded section ($\sim 0.8-1.0$) represents the allele values where the filter would have been disabled; (b) Gene 16, the amount of DC that is allowed to pass through a high-pass filter, which exhibits Type B behaviour (allele settled on genome boundary); (c) Gene 20, a linear signal adjustment factor, which exhibits Type C behaviour (allele value has no impact on algorithm fitness); and (d) Gene 22, a non-linearity correction, which exhibits Type A behaviour (allele is comfortably within genome boundary).

frames per second, reducing to ~ 125 frames per second for the entire processing pipeline, allowing for real-time operation [8]. Due to expensive mathematical operations in BIV09 that were removed as part of the improvements to form BIV19, Phase 1 throughput was only ~ 600 frames per second, per core, which normalises to ~ 500 frames per second, per core, for the different computing systems used.

The vast majority of children ($85.64\% \pm 2.36\%$) within the Phase 2 studies went full-term, while a relatively high percentage ($13.94\% \pm 2.33\%$) of children were terminated after failing to phenotype correctly after processing the first velocity. While some terminations did occur at other velocities, their cumulative percentage was very low ($0.42\% \pm 0.12\%$). The total number of frames processed in Phase 2 was $370,221,292,176$, or $12,306,074,027 \pm 294,558,701$ per study, compared to the $424,512,000,000$ available had all children phenotyped positively.

V. DISCUSSION

The work presented here has shown that a complex biologically-inspired computer vision algorithm, featuring 36 parameters with both non-linear responses and interactions, can be tuned with evolutionary algorithms. A baseline algorithm, BIV09 [7], has a fitness of 5.6 from previous work [8]. In previous work, extensive changes were made to BIV09 to form BIV19, and we compared a non-optimally tuned version of BIV19 to a non-optimally tuned BIV09 algorithm and other state-of-the-art algorithms [8]. To accurately quantify the impact of the changes, BIV09 was tuned to Pareto-optimality in this work, achieving a fitness of 6.515; a 16.3% increase. BIV19 was then also tuned to a state of Pareto-optimality, achieving a fitness of 7.547. Not only is this a 15.8% increase over the tuned BIV09 and a 34.8% increase over the baseline BIV09, BIV19 executes ~ 3 times faster than BIV09 on a real-world real-time system [8]. The slight improvement of 2.68% reported here for BIV19 compared to previous results [8] is caused by different genome boundaries and, as the aim of this paper was not to directly improve upon the previously published result but to validate the methodology used, is acceptable.

Unlike a binary or fixed-point genome, the floating-point genome presented a few challenges. First, we had to ensure enough resolution between boundaries was provided while restricting the solution space. Exploratory work could be performed at a lower stepping (50), with later studies at higher steps (200). In this work, we empirically chose 100 steps as this was a balance between enough resolution to see operating point details (Fig. 8(d)), what occurs when a filter is disabled (Fig. 8(a)), and decreased convergence times.

Determination of tuning termination presented another problem. With a binary genome, there are 2 possible states; with an 8-bit genome, 256. With a 32-bit floating-point genome this limitation still exists, but to a much lesser extent. A typical response might be to increase the stepping between boundaries, or restrict the boundaries. While this will drive the fitness higher, it can also decrease generalisability. This problem is common to all forms of machine learning, where performance is dependent upon the diversity of the training dataset [41]. However, for a computer vision algorithm, generalisability might not be important, depending on how specific the task is that is being addressed. This is where a deep understanding of the algorithm being tuned, the application space, and the tuning algorithm itself, is mandatory. While possible to tune the algorithm for higher performance, there is not a single, or even a small, group of parameter values that are uniquely high performing. Rather, there exists a large cohort of parameter values that result in high fitness. This diversity of parameters yielding good results indicates that the system is not likely to be uniquely tuned to this dataset. This has previously been explored where a validation scene different to the training data was used to quantify performance [38].

While the unsuitable children are tracked within each study and uniqueness was brutally enforced at all times, repeated

instances of the tuning could benefit in having awareness of historically unsuitable children, known as *a priori* knowledge transfer [42]. This *a priori* knowledge would remove the re-evaluation of unsuitable children, thereby decreasing solution times. Quantifying the improvement in solution time, however, is not trivial due to the large number of genome combinations for an algorithm as complex as BIV19.

As with lots of machine learning tasks, parameter limitations are often determined by available resources, ultimately cost. In this work, the time taken to process a generation is variable as there is extensive spatial processing within the computer vision algorithm, and the size and strength of these spatial filters is open to adjustment. This, combined with premature termination of unsuitable children, makes it practically impossible to optimise the number of children to the resources available other than $N_c \gg N_{cpu}$. This lack of generalisability in parameter selection is a well-known problem [43].

This work has focussed on a single objective — fitness function maximisation — however multi-objective tuning is often employed in computer vision research [44], [45]. Due to the modularity of BIV09 and BIV19, being they are comprised of distinct and separable models, development of parallel vision pathways can also occur. For example, the small target detection pipeline of many insects shares the photoreceptor (PR) and lamina monopolar cells (LMC) models with the motion estimation pipeline [36]. Each pipeline could be tuned individually (single-objective), or they could be tuned in parallel (multi-objective), where each pipeline shares common PR and LMC genes. This level of interoperability is more biologically plausible, and is the subject of ongoing work.

VI. CONCLUSION

This paper has shown the usefulness of a biologically-inspired evolutionary algorithm in guiding the development and tuning of an elaborated biologically-inspired computer vision algorithm. Working with an algorithm that has 36 unique parameters, most having non-linear impacts on algorithm performance and non-linear interactions with each other, presents some unique challenges for operational tuning. While the value of some genes can be informed from biological recordings, that is only applicable if the digital model is 100% biologically accurate. This is something that is currently difficult to achieve both neurophysiologically (particularly the LMC, EMD, and MLI stages), as well as technologically, meaning using parameters derived from neurophysiological recordings is not currently feasible. As a result, educated guesses to seed a genetic algorithm is one of the only viable approaches. However, by taking inspiration from other aspects of biology, specifically evolution, the continued development of complex algorithms can progress with the knowledge that their parameter sets can in fact be tuned.

ACKNOWLEDGEMENTS

The authors declare that they have no conflicts of interest. The authors thank the University of South Australia for their donation of extensive computational resources on their

High Performance Computing cluster to realise this work. PSMS was supported by an Australian Government Research Training Program Scholarship.

REFERENCES

- [1] C. V. Stewart, "Robust parameter estimation in computer vision," *SIAM review*, vol. 41, no. 3, pp. 513–537, 1999.
- [2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.
- [3] G. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. Goodfellow, and J. Sohl-Dickstein, "Adversarial examples that fool both computer vision and time-limited humans," in *Advances in Neural Information Processing Systems*, 2018, pp. 3910–3920.
- [4] D. B. Fogel, "An evolutionary approach to the traveling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, 1988.
- [5] C.-J. Du and D.-W. Sun, "Learning techniques used in computer vision for food quality evaluation: A review," *Journal of food engineering*, vol. 72, no. 1, pp. 39–55, 2006.
- [6] R. K. Mohanta and B. Sethi, "A review of genetic algorithm application for image segmentation," *Int. J. Comput. Technol. Appl.*, vol. 3, no. 2, pp. 720–723, 2011.
- [7] R. S. Brinkworth and D. C. O'Carroll, "Robust models for optic flow coding in natural scenes inspired by insect biology," *PLoS computational biology*, vol. 5, no. 11, p. e1000555, 2009.
- [8] P. S. Skelton, A. Finn, and R. S. Brinkworth, "Consistent Estimation of Rotational Optical Flow in Real Environments using a Biologically-Inspired Vision Algorithm on Embedded Hardware," *Journal of Image and Vision Computing*, pp. 1–8, 2019.
- [9] N. Franceschini, J.-M. Pichon, and C. Blanes, "From insect vision to robot vision," *Philosophical Transactions of The Royal Society Of London. Series B: Biological Sciences*, vol. 337, no. 1281, pp. 283–294, 1992.
- [10] F. Ruffier and N. Franceschini, "Optic flow regulation: The key to aircraft automatic guidance," *Robotics and Autonomous Systems*, vol. 50, no. 4, pp. 177–194, 2005.
- [11] J. Serres, D. Dray, F. Ruffier, and N. Franceschini, "A vision-based autopilot for a miniature air vehicle: Joint speed control and lateral obstacle avoidance," *Autonomous robots*, vol. 25, no. 1-2, pp. 103–122, 2008.
- [12] A. Schwegmann, J. P. Lindemann, and M. Egelhaaf, "Depth information in natural environments derived from optic flow by insect motion detection system: A model analysis," *Frontiers in computational neuroscience*, vol. 8, p. 83, 2014.
- [13] Z. Michalewicz and M. Schmidt, "Report on Planning/Artificial intelligence techniques and some research issues that are potentially applicable to the UAV pre-mission planning problem," Non-Public Report; Contact Author(s) for Information, 2006.
- [14] R. Hooke and T. A. Jeeves, "“Direct Search” Solution of Numerical and Statistical Problems," *Journal of the ACM (JACM)*, vol. 8, no. 2, pp. 212–229, 1961.
- [15] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.
- [16] R. M. Lewis, V. Torczon, and M. W. Trosset, "Direct search methods: Then and now," *Journal of computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 191–207, 2000.
- [17] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM review*, vol. 45, no. 3, pp. 385–482, 2003.
- [18] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM Journal on optimization*, vol. 9, no. 1, pp. 112–147, 1998.
- [19] F. Walters, "Sequential simplex optimization-An update," 1999.
- [20] W. Swann, "Direct search methods," *Numerical methods for unconstrained optimization*, pp. 13–28, 1972.
- [21] V. Torczon, "On the convergence of the multidirectional search algorithm," *SIAM journal on Optimization*, vol. 1, no. 1, pp. 123–145, 1991.
- [22] D. Pham and D. Karaboga, *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer Science & Business Media, 2012.
- [23] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.
- [24] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001, vol. 16.
- [25] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.
- [26] C. Sharma, S. Sabharwal, and R. Sibal, "A survey on software testing techniques using genetic algorithm," *arXiv preprint arXiv:1411.1154*, 2014.
- [27] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.
- [28] M. Mahmoodabadi and A. Nemati, "A novel adaptive genetic algorithm for global optimization of mathematical test functions and real-world problems," *Engineering Science and Technology, an International Journal*, vol. 19, no. 4, pp. 2002–2021, 2016.
- [29] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2000, pp. 849–858.
- [30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [31] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 503–515, 2003.
- [32] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [33] I. Das, "A preference ordering among various Pareto optimal alternatives," *Structural optimization*, vol. 18, no. 1, pp. 30–35, 1999.
- [34] A. Baresel, H. Sthamer, and M. Schmidt, "Fitness function design to improve evolutionary structural testing," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2002, pp. 1329–1336.
- [35] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 345–370, 2009.
- [36] A. Melville-Smith, A. Finn, and R. S. Brinkworth, "Enhanced micro target detection through local motion feedback in biologically inspired algorithms," in *2019 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2019.
- [37] R. S. Brinkworth and D. C. O'Carroll, "Bio-inspired model for robust motion detection under noisy conditions." IEEE, 2010, pp. 1–8.
- [38] P. S. Skelton, A. Finn, and R. S. Brinkworth, "Real-Time Visual Rotational Velocity Estimation Using a Biologically-Inspired Algorithm on Embedded Hardware," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2017, pp. 1–8.
- [39] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms*, vol. 206, 1987, pp. 14–21.
- [40] E.-L. Mah, R. S. Brinkworth, and D. C. O'Carroll, "Implementation of an elaborated neuromorphic model of a biological photoreceptor," *Biological cybernetics*, vol. 98, no. 5, pp. 357–369, 2008.
- [41] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, 2019.
- [42] W. Y. Choy and B. C. Sanctuary, "Using genetic algorithms with a priori knowledge for quantitative NMR signal analysis," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 4, pp. 685–690, 1998.
- [43] F. Lobo, C. F. Lima, and Z. Michalewicz, *Parameter Setting in Evolutionary Algorithms*. Springer Science & Business Media, 2007, vol. 54.
- [44] J. Delpiano, L. Pizarro, R. Verschae, and J. Ruiz-del-Solar, "Multi-objective optimization for characterization of optical flow methods," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 2. IEEE, 2014, pp. 566–573.
- [45] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2013.