An Automated System For High-Throughput Longevity and Healthspan Discovery in *Caenorhabditis elegans*

Benjamin Blue

A dissertation

Submitted in partial fulfillment of the

Requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Matt Kaeberlein, Chair

Alexander Mendenhall

Shane Rea

Program Authorized to Offer Degree:

Pathology

University of Washington

Abstract

An Automated System For High-Throughput Longevity and Healthspan Discovery in *Caenorhabditis elegans*

Benjamin Blue

Chair of the Supervisory Committee:

Matt Kaeberlein

Department of Pathology

Over the last century, the study of aging biology has primarily been advanced through the development and use of animal models that share common molecular hallmarks with human aging. One major model system that has been widely used during the last several decades and through the present day is the roundworm *Ceanorhabditis elegans.* Its short lifespan, easy husbandry, and genetic tractability have allowed it to be easily adapted for studying the molecular biology of aging. For instance, research using *C. elegans* was used to discover the interplay between insulin signaling and rate of aging. However, while *C. elegans* research proved orders of magnitude more efficient (at the cost of a larger evolutionary gap) than using vertebrate models, such as mice or non-human primates, it's use was still hamstrung by the necessity for manually collected lifespans.

During the last decade, the lowering cost and increasing capabilities of digital microscopy and computer vision have led to researchers attempting to automate the

most basic aging related experiment in *C. elegans*: the measurement of an individual's lifespan relative to the population within which it resides. Some examples of this include the Automated Lifespan Machine, The WormMotel, as well as several different microfluidic platforms such as the NemaLife chip. These platforms sought to semi-automate or even fully-automate the collection and recording of lifespan measurements but often had large bottlenecks in their pipeline that precluded them from achieving the scale necessary to robustly advance the field of aging biology. This thesis will present the development and use of a novel robotics platform that, when paired with a suite of AI-powered computer vision, fully automates lifespan analysis of *C. elegans*.

Introduction

Societal trends in aging

For humans, the biological process of aging has remained almost constant across recorded history. Yet, over the last 100 years the development of modern medicine, accompanied by an increased accessibility to it, has resulted in significant improvements in median life expectancy. The average age of residents within the United States is increasing, and the rapidly growing over-65 population is expected to reach 83.7 million by 2050[1]. Although living longer, older individuals still experience significant health challenges, with a progressive decline in the organization and function of a wide variety of tissues and an associated loss in quality of life. Furthermore, an individual's chronological age has been found to be the major risk factor for most diseases that impact mortality[2]. Although global efforts to improve access to health care are still very much necessary, this larger fraction of individuals in developed countries (known sometimes as “the silver wave”) that evade early and midlife causes of mortality are expected to increase and require larger investments in both ongoing healthcare and support in order to maintain a reasonable quality of life. Crucially, there remains a distinction between the absolute measured lifespan of an individual and their quality of health (or healthspan) during that time. Although, the term healthspan is often used too reductively[3], as it broadly integrates many different mutable qualities, it’s common use encompasses the observation that long-life does not necessarily correlate with long periods of general health. In this case, while it is often possible for individuals with access to appropriate medical care to live longer than historically would have been possible, this extension of life is often decoupled from quality of life, and so many individuals undergo long periods of ill-health and low quality-of-life.

Although the therapeutic promise of extending lifespan or healthspan, or both, by targeting the underlying biology of aging has been well validated using animal models, trials using drugs or other treatments to do the same thing in humans have been understandably slow[4]. This is understandable, given the necessarily strict requirements that institutions such as the Federal Drug Authority (FDA) place on harm mitigation and effect validation for any new treatment. Nonetheless, it is notable that this has severely constrained the work to translate interventions well-validated in animal models into humans.

Introduction to aging biology

Although the philosophy of aging, longevity, and lifespan extension extends as far back to the Sumerian myth of Gilgamesh and Enkidu, the last century has seen the formation of modern medical theories of aging. Beginning in the first half of the 20th century, experiments involving the extension of lifespan through dietary restriction would form the basis of many modern approaches[5]. By the 1980s, researchers such as Klass, Johnson and Kenyon had identified genetic pathways that could regulate aging in invertebrate models. This work would then be expanded and encompass multiple model systems from single celled yeast to nonhuman primates[6,78,9,10].

As more and more molecular pathways involved in the regulation of aging were identified, it became increasingly clear that the mechanisms of aging were broadly conserved across most species. In 2012, the seminal paper "Hallmarks of Aging" was published and established 9 key conserved hallmarks that provided a framework for discussion and inquiry for the field[11]. Moreover, this framework has allowed for expanded and informed growth of using invertebrate models as tools for translational research.

Invertebrate research and aging bio

Since the early work into the molecular mechanisms of aging, the last 30 years of research has emphasized the role of invertebrate model systems in aging research. Although rodent models remain a widely used standard in terms of translatability to human research and are even used in the context of drug discovery and validation (ex. The Interventions Testing Program), mouse lifespan alone averages ~3 years and makes such screening extremely expensive and time-consuming. As such, the field has relied on using invertebrate models to accelerate geroscience and leverage the conserved molecular hallmarks of aging[12]. For example, early work into the mechanisms of mTOR and other aging-associated pathways was driven through the use of *Saccharomyces cerevisiae* while the aforementioned work by Kenyon and other researchers to investigate the effects of insulin signaling pathways was done via *C. elegans*. Although mammalian models often remain a bottleneck for validation, the use of invertebrates provides two crucial benefits to this day. Firstly, invertebrate models are generally much shorter lived than mammalian systems. Budding yeast persist for ~20-30 cell divisions while *C. elegans* lives ~2-3 weeks and *Drosophila melanogaster* ~2-3 months. These relatively short lifespans coupled with the generally small animal size and easy culturing drastically accelerate the number of individuals that can be assayed simultaneously via a single researcher. Secondly, the genetics of these invertebrate models have been dissected at-depth through previous research and have relatively standard methodologies for further genetic modifications, labeling of proteins, and gene by gene knockdowns.

C. elegans primer and use in aging experiments

First described in 1897 as a soil-dwelling nematode[13], *C. elegans* would become established in the 1960s as a major model system through the pioneering work of Sydney Brenner[14]. While *C. elegans* and a related species *C. briggsae* had been used in biological research, Brenner's work would establish a central role for the species within the field as well as the specific "N2-Bristol" strain isolated from the UK and which remains the dominant "wildtype" strain used across many of the experiments in the field as well as this thesis. In 1974 Brenner would publish a genetic map of around 100 genes and their phenotypes across 300 mutant lines. This pioneering work would continue and two decades later *C. elegans* would become the first animal to have its genome fully sequenced.

Beginning in the 1970s, *C. elegans* would begin to be adopted for use as a model for aging biology. Klass would provide the groundwork in 1977 by defining culture methods and animal handling techniques that would establish a precedent for *C. elegans* use in lifespan studies[15]. Klass would then begin some of the early genetic screens seeking to identify genes implicated in aging[16], work that would culminate in the identification by Tom E. Johnson of *age*-1 as a pro-longevity mutant[17].

The next major milestone would be the further mapping of the insulin-like-growth-factor signaling pathway as a central axis of *C. elegans* aging[18,19]. By mapping the nearly doubled observed lifespan to the deletion of *daf-2*, these early studies would start the long process to use *C. elegans* genetic screens to untangle the interplay between genetics and the biological processes of aging. As *C. elegans* would find a role in the aging field, the type and number of experiments performed would expand. Biomarkers of age such as stress resistance, changes in gene expression over time, and other physiological traits would be developed and then used to define normative aging in the worm[20–24].

In addition to the well-established library of knowledge and tools that have been developed as part of the inquiry into the genetics of *C. elegans,* several traits of the species contribute to its continued efficacy for research in general, and into the biology of aging in particular. First, *C. elegans* is relatively easy to culture and maintain within a laboratory setting. Populations are generally maintained on agarose plates that are pre-seeded with a bacterial culture of *E. coli* upon which the worms feed. Although rigor must be maintained to ensure environmental control for the animals, a persistent population can be maintained easily. Second, *C. elegans* larvae are highly robust and can survive freeze/thaw cycles. This allows for isolated strains and populations to be frozen, stored, and then recovered even years later. The ability to return to ancestral stock populations so easily is essential for correcting for laboratory adaptation and has also allowed for the development of the *Ceanorhabditis Genetics Center* which stores and distributes nearly all wild-isolated or laboratory derived strains of the species. In doing so, the community of *C. elegans* researchers are able to draw upon nearly all historical work in terms of strain and genetic tool generation. Third, *C. elegans* is a relatively short-lived species. Although its normative lifespan is highly temperature dependent, at standard culturing conditions of 20°C a population's median lifespan will be about 16-18 days, with a maximal reported lifespan for a pro-longevity mutant being less than three months. This allows for relatively efficient experimentation within the context of aging, especially for experiments where lifespan is used an end-point. Fourth, fertilized eggs form a thick shell that is resistant to hypochlorite treatment. In laboratory practice this means that a mixed-stage population can be treated with household bleach to kill and dissolve all animals that have hatched allowing the collection of unhatched eggs in order to systematically age-synchronize a population. Fifth, while there remains debate about how "druggable" *C. elegans* is, due to its relatively thick cuticle, many drug interventions that are efficacious in other species have also been shown to work in *C. elegans,* usually following simple addition of the drug to the agar upon which they reside. While ongoing work is establishing new methodologies of drug delivery using micro-encapsulation, the majority of published literature has relied upon the much simpler method described above. Lastly, over the years *C. elegans* researchers have developed a wide-array of tools that now permit one to undertake with ease such tasks as gene knockdown, genome transformation and the creation of novel mutant lines. Single-gene knockdown in *C. elegans* through feeding RNAi is particularly widely used due the availability of several robust libraries of dsRNA-expressing *E. coli*. Worms process the dsRNA into small-interfering RNA that then targets a gene of interest for knockdown. Such simple epistasis experiments are, for the most part, direct and efficient, and allow researchers to determine a pathway of action of most interventions.

While *C. elegans* have been widely used for aging research, there are several notable limitations to their use. The lack of a blood brain barrier, an adaptive immune system, and dissimilarities in epigenetic regulation preclude investigation into several pathways that are important to human aging[25].

Importance of laboratory automation

Although the relative ease of culturing *C. elegans* has allowed for its widespread adoption within many laboratory settings, the last two decades have seen an even further push for improvement in experimental throughput and rigor, mainly through automation. Most *C. elegans* related publications to-date continue to utilize laboratory techniques that have remained relatively unchanged for decades. Primarily, *C. elegans* are manipulated through either pipet transfer of populations or through picking of individuals using a wire (or hair) instrument. As *C. elegans* are relatively small, even historical techniques such as these afford a much higher throughput in terms of number of individuals per researcher that can be maintained relative to the majority of other animal model systems. However, these techniques still require that experimenters visually assay each sub-population and replicate every day from experiment start to the point at which all animals are deceased. The limitations of such an approach include an upper capacity of a researcher of one to two thousand animals at a time as well as the associated risks of human error in each experiment. In most cases, this methodology also limits the data generated to only that of the total lifespan for each animal under observation. While metrics of health and life-history can also be collected using manual techniques and human observation, they are commonly performed as secondary experiments following an initial lifespan screen. Until the advent of automated technology, this remained the rate-limiting step across most experimental designs.

Automated technologies broadly fall into two categories: those that employ microfluidic devices and those that utilize a plate-based surface for high-throughput imaging. Dozens of microfluidic devices have been developed that allow worms to be kept in liquid media and assayed by light microscopy based systems[26]. Such systems use photolithography to manufacture a silicon-polymer based "chip" that is usually affixed to a glass surface[27]. Most chip designs consist of an inlet and exit port connected by channels through which liquid can flow and that are sized and shaped to allow passage of *C. elegans* up to a point at which they are intended to be trapped. As silicon polymers are optically transparent and gas permeable, *C. elegans* can theoretically be housed for their entire natural lifespan, as long as sufficient food sources are flushed through the device. These microfluidic systems have some benefits over traditional solid culture. Through the placement of properly sized sieves, progeny can be flushed out of the observation chamber allowing for survival experiments in the absence of 5-fluorodeoxyuridine (FUDR), which, while often used by many labs in nematode survival experiments, has been shown to affect worm physiology, gene expression and have genetic background specific effects. In addition, microfluidic systems allow for more precise control over environmental factors to which animals are exposed. Although microfluidic systems can be powerful for longitudinal imaging, they also suffer from drawbacks that impair their ability to study aging in nematodes. First, they require culturing animals in an environment they typically do not experience in nature, as *C. elegans* is not an aquatic species but instead live in soil and in rotting fruit. Upon exposure to liquid environments, worms activate multiple stress response pathways and exhibit a fleeing behavior often described as "thrashing" that can have direct effects on aging biology. This means that it can be difficult to reconcile data generated from animals aged in a microfluidic system with that of the vast majority of prior literature in the field, which in general has been carried out on animals cultured on solid agar media and fed *E. coli* OP50. Microfluidic systems also generally require specialized equipment and expertise that most *C. elegans* laboratories are not equipped with, limiting utility among the broader community. Lastly, although the ability of microfluidic systems to deliver highly controlled and automated experimentation is promising, the complexity of the fabrication and use of such devices often impedes throughput from being drastically improved over manual observation.

The second category of higher-throughput lifespan devices involves time-lapse microphotography to record images of the worms at fixed time intervals throughout the lifespan of the worm on solid media. The most developed of these systems, the Lifespan Machine (ALM)[28], uses an array of modified commercial flatbed scanners to repeatedly scan low profile petri dishes sealed onto the glass bed of the scanner with a rubber gasket. This system has been implemented by a small number of external labs, including the 3 sites of the National Institute on Aging *C. elegans* Intervention Testing Program[29]. However, automating survival analysis using the ALM has proven to have several limitations. First, the system relies on modified flatbed transparency film scanners for imaging. The scanners used in the initial publication (Epson v700) are no longer manufactured, and while the designers of the ALM have issued updates to support newer scanners (Epson v800, v850), the bottleneck of using commercial products that are adapted from their original purpose presents an ongoing limitation. Second, the temporal resolution of the scanned images is suboptimal. While transparency scanners have very high spatial resolution, in an ALM at normal capacity they are only capable of capturing an image of a single experimental plate once an hour. Additionally, ALM protocols call for a modified version of nematode growth media (reduced calcium) that differs from media used in previous aging studies in the worm in order to maintain optimal optics. Furthermore, under normal operation, an ALM scanner's light source travels the length of the scanner bed 4 times an hour, taking between 10-12 minutes per scan on the original model of scanner. This results in long exposures to intense light which has been reported to have effects upon lifespan within populations housed on the ALM and even drug-specific effects. Finally, the physically closed format of the scanner, which presented thermal management challenges to the system's designers, also limits extension of the device to experiments that require the worms to be placed in novel and alternative environmental conditions (e.g. gas exposure). It has been previously reported that hypoxia can increase lifespan. We experienced firsthand an inability to modify the atmosphere of plates sealed onto a flatbed scanner, making hypoxia-type experiments wholly intractable in the ALM system.

Additional systems that rely on solid media culturing techniques include the WormMotel[30], a system that uses a high resolution camera to monitor single worms arrayed into individual wells in custom PDMS plate format. This system has been coupled with a plate handling robot to make it high throughput, and since it relies on a camera is able to record the individually housed animal's movements throughout life to generate longitudinal healthspan measures. While the WormMotel system is elegant and powerful, the necessity to house animals in manufactured custom plates and picking/seeding individual worms into single wells would make it difficult to use for high-throughput studies such as compound or genetic screens. Additionally, the necessity of a plate handling robot to increase throughput limits its availability to many labs.

Although the push for laboratory and experimental automation is often contextualized within the context of increased throughput, an often overlooked benefit is the increased rigor and reproducibility that is simultaneously provided. This is because the need for human intervention is reduced. One study estimated that the monetary cost of flawed or irreproducible research in the United States alone was over $28 billion per year[29]. Perhaps an even larger detriment from the flawed research is the human cost - the lost person-hours of the research scientists involved, and the harm to patients that arises from biomedical studies that lead to erroneous translational work and potentially dangerous treatments. While there are numerous reasons why *C. elegans* work can be difficult to replicate between research groups (animal staging, lab adaptations, media inconsistencies or contamination, experimenter error, etc.)[29], use of automated technologies to conduct research should, in principle, alleviate many of these issues.

The current suite of automated tools relies on digital microscopy and computer vision to record and assess experimental data. Historically, a researcher sitting at a microscope studying worms was limited in the number of replicates they could perform and also, they had to make judgement calls on what data to collected and photograph, and how to interpret it. Modern tools of automation remove this burden and bias. Video cameras can record data for as long as there is digital storage space, and across a far greater number of replicate samples. This allows researchers to return to ‘ground-truth’ images or videos, should inconsistencies present themselves, instead of repeating an experiment. Not only does this increase efficiency, but it also allows for dissemination of raw data collected during experiments between groups.

Although no fully-automated system has been developed, the reduction in human participation in data collection does significantly reduce the potential for error. Pitfalls can still occur prior to the use of automated tools or at points where human interaction is required. In the case of *C. elegans* experiments, a likely persistent source of error is anything involved on the experimental preparation side (hypochlorite treatment, inconsistent food exposure, improper FUDR use, etc.)[31] prior to the initiation of automated monitoring. Despite these limitations, automation drastically increases the potential for rigorous and reproducible research to be performed at a scale hitherto unknown.

Chapter One:

The WormBot, an open source robotics platform for survival and behavior analysis in *C. elegans*

Abstract

C. elegans is a popular organism for aging research owing to its highly conserved molecular pathways, short lifespan, small size, and extensive genetic and reverse genetic resources. Here we describe the WormBot, an open-source robotic image capture platform capable of conducting 144 parallel *C. elegans* survival and behavioral phenotyping experiments. The WormBot uses standard 12-well tissue culture plates suitable for solid agar media and is built from commercially available robotics hardware. The WormBot is controlled by a web-based interface allowing control and monitoring of experiments from any internet-connected device. The standard WormBot hardware features the ability to take both time-lapse bright field images and real-time video micrographs, allowing investigators to measure lifespan and healthspan metrics as worms age. The open-source nature of the hardware and software will allow users to extend the platform and implement new software and hardware features. This extensibility, coupled with the low cost and simplicity of the system, allows the automation of *C. elegans* survival analysis even in small laboratory settings with modest budgets.

Introduction

Over the last 30 years the roundworm *C. elegans* has provided vital insights into the genetic control of longevity. Studies of aging in the worm have capitalized on their small size, short lifespan, exquisitely defined cell lineage, and large genetic tool-set to identify specific genetic factors and the tissues in which they operate to influence the rate of aging[34]. Many of the genetic pathways now known to control aging were first identified in the nematode and then later determined to play a role in mammalian systems[32,33]. Nearly all of this progress was made using manual lifespan collection techniques, in which investigators examined animals grown on solid media in petri dishes daily to determine if worms had died since the previous time point by prodding the worms with a stick (platinum wire) to elicit movement.

More recently, several automated approaches to performing survival analyses have been developed[10,14,35,36]. These typically rely on photographic monitoring of spontaneous nematode movement; however, methods based on vital dyes or disruption of intestinal permeability have also been reported[37,38]. Generally, the imaging-based approaches fall into two categories. First, several microfluidic devices have been developed that allow worms to be kept in liquid media and assayed by light microscopy-based systems[35,39] Microfluidic systems have some benefits over traditional solid culture. Through the placement of properly sized sieves, progeny can be flushed out of the observation chamber, allowing for survival experiments in the absence of 5-fluorodeoxyuridine (FUDR), which, while traditionally used in nematode survival experiments, has been shown to affect worm physiology and have genetic background specific effects[40,41]. In addition, microfluidic systems allow for more precise control over the environmental factors to which the animals are exposed. Although microfluidic systems can be robust for longitudinal imaging, they also suffer from drawbacks that impair their ability to study aging in nematodes. First, they require culturing animals in an environment they typically do not experience in nature, as C. elegans is not an aquatic species but lives in soil and in rotting fruit[42]. Upon exposure to liquid environments, worms activate multiple stress response pathways[43] and exhibit a fleeing behavior often described as "thrashing" that can directly affect aging systems[43]. This means that it can be difficult to reconcile data generated from animals aged in a microfluidic system with that of the vast majority of prior literature in the field, which has been carried out on animals cultured on solid agar media and fed *E. coli* OP50[14]. Finally, microfluidic systems generally require specialized equipment and expertise that most *C. elegans* laboratories are not equipped with, limiting utility among the broader community.

The second category of higher-throughput lifespan devices involves time-lapse microphotography to record images of the worms at fixed time intervals throughout the worm's lifespan on solid media. The most developed of these systems, the Lifespan Machine (ALM)[28], uses an array of modified flatbed scanners to repeatedly scan low-profile petri dishes sealed onto the scanner's glass bed with a rubber gasket. This system has been implemented by a small number of external labs, including the three sites of the National Institute on Aging *C. elegans* Intervention Testing Program[35]. However, our attempts to automate survival analysis using the ALM identified some issues that ultimately led us to develop an alternative system. First, the system relies on modified flatbed transparency film scanners for imaging. The scanners used in the initial publication are no longer manufactured, and while the designers of the ALM have issued updates to support newer scanners, we were concerned that rapid changes in the digital photography world could affect the long-term viability of sourcing components for the ALM. Second, the temporal resolution of the scanned images is suboptimal. While transparency scanners have a very high spatial resolution, in an ALM at average capacity, they can only capture an image of a single experimental plate once an hour.

Additionally, ALM protocols call for a modified version of nematode growth media (reduced calcium) that differs from media used in previous aging studies in the worm[5]. Furthermore, under regular operation, an ALM scanner's light source travels the length of the scanner bed four times an hour, taking between 10-12 minutes per scan on the original scanner model. This results in long exposures to intense light, which has been reported to have effects on lifespan in the ALM[35]. Finally, the physically closed format of the scanner, which presented thermal management challenges to the system's designers[5], also limits the extension of the device to experiments that require the worms to be placed in alternative gas environments. We have previously reported that hypoxia can increase lifespan[44,45] and the limited gas diffusion of plates sealed on a scanner, as well as the inability to modify the atmosphere of the plates sealed to a flatbed scanner, was intractable for our experimental needs. Other systems that rely on solid media culture include the WormMotel[30], which uses a high-resolution camera to monitor single worms arrayed into individual wells in custom PDMS plate format. This system has been coupled with a plate handling robot to make it high throughput, and since it relies on a camera can record the individually housed animal's movements throughout life to generate longitudinal healthspan measures. While the WormMotel system is elegant and powerful, we felt the in-house manufactured custom plates and picking/seeding individual worms into single wells would make it difficult to use for high-throughput studies such as drug screens. Additionally, the necessity of the plate handling robot to increase throughput would limit its availability to many labs. We therefore designed a new system built from consumer robotics hardware that allows for solid media culture and standard 12-well tissue culture plates to provide high-throughput automated survival analysis for *C. elegans*.

Methods

Hardware construction

A MakeBlock XY plotter robotics kit (MakeBlock.com, Shenzhen, China) was purchased from amazon.com and assembled according to the manufacturer's instructions until page 21 of the build manual

(https://github.com/Makeblock-official/XY-Plotter-2.0/blob/master/XY%20Plotter%20V2.0%20Assembly%20Instruction.pdf). Following this step, assembly diverged, and a complete list of additional parts is available at http://wormbot.org/bom.xls. Complete video build tutorials are available at http://wormbot.org and on https://www.youtube.com/channel/UCcbtj864r6CXAUZOVe4vCSg. Optional, additional 3d printed components were printed using 1.75mm PLA filament on a Monoprice MakerUltimate 3D printer (Monoprice, Rancho Cucamonga, CA). STL files are available at wormbot.org, thingiverse.com. A 24 x 24 x ⅜ inch acrylic panel was purchased from TAP plastics (tapplastics.com, Oakland,CA) and machined by Front Panel Express (frontpanelexpress.com, Seattle, WA). CAD files for machining are available at http://wormbot.org/wormbot.fpd.

Software

All of the WormBot's code is open-source and available on Github (http://github.com/JasonNPitt/wormbot). The WormBot codebase is primarily written in C++ and should function on any POSIX operating system. However, all testing has been done using Ubuntu 16.04 (Canonical LTD, London, UK). Wormbot software relies on the OpenCV library[20] (opencv.org), the FFMPEG project (ffmpeg.org), the Arduino platform[21], and the Apache web server[22] (httpd.apache.org). Details of other code dependencies are available via Github. Wormbot software can be installed by cloning the git repository (> git clone http://github.com/JasonNPitt/wormbot) followed by running the install script (> sudo ~/wormbot/INSTALL). More detailed software installation instructions are available at wormbot.org and in the supporting online information.

Media preparation and RNAi experiments

12-well tissue culture plates were purchased from Genesee Scientific (Cat #25-101, geneseesci.com, El Cajon, CA). Each well-contained 3mL of sterilized Nematode Growth Medium supplemented with 100 units/mL Nystatin to prevent fungal growth and a final concentration of 50uM FUDR to prevent progeny development. For RNA interference (RNAi) experiments, media also contained 100ug/mL of ampicillin, 10ug/mL tetracycline, and 2mM IPTG. For all experiments, worms were synchronized by 10-

minute hypochlorite treatment (1% sodium hypochlorite, 250mM potassium hydroxide, and water) followed by four washes in M9 buffer to isolate eggs, followed by an overnight hatch-off in 6mL of M9 in a 60mm petri dish sealed with parafilm in a 20C incubator (Torrey Pines Scientific, Torrey Pines, CA). RNAi clones were taken from the Vidal and Ahringer RNAi feeding libraries[46,47], sequence verified and kept as -80C frozen stocks before use. dsRNA induction was performed by growing RNAi feeding strains overnight in media containing antibiotics but without IPTG. The following morning, stationary phase cultures were diluted 4-fold with fresh media containing 2mM IPTG and grown for 4 hours at 37°C on a shaker. Following induction, cells were pelleted and resuspended in the original overnight culture volume (4x concentrate) and seeded onto RNAi NGM media. Following 20uL of seeding with RNAi bacteria, 12-well plates were allowed to dry for 20-60 minutes with their lids off in a laminar flow hood to allow the seeding solution to evaporate. RNAi experiments began by placing synchronized L1 worms on 10cm RNAi plates until they reached the L4 larval stage when they were either hand-picked or liquid transferred (in M9) to the seeded 12-well RNAi plates containing FUDR. Throughout the experiment, plate humidity was maintained by filling the interstitial spaces of the 12-well plate with double distilled water and refilling as necessary.

Toxin exposure

Potassium cyanide (KCN) was purchased from Acros Organics (ThermoFIsher, Waltham, MA) and resuspended at 100mM in 100mM Sodium Hydroxide. Hydrogen cyanide (HCN) exposure was performed as described by Gallagher and Manoil[48], with modifications for the 12-well plate format. Briefly, in a fume hood, 12-well plate lids were ringed with petroleum jelly from a syringe, and a 100uL drop of the 100mM KCN solution was placed in the interstitial space of the 12-well dish near a 100uL drop of 200mM hydrochloric acid. The lid was sealed, and the plate tipped to mix the two drops liberating HCN gas into the sealed chamber and placed immediately onto the robot. Strains used in this study were provided by the Caenorhabditis Genetics Center (Madison, WI), supported by the NIH Office of Research Infrastructure Programs (P40 OD010440).

Healthspan/Behavior Measures

The WormBot can record brief 30 frame/sec movies each day of all 144 wells in an experiment. To assay worm movement in Figure 6C, one well from each of the 12 plates was randomly selected, and the WormBot application ***MovieTrace*** was used to analyze 1250 frames from the selected wells for all 20 days of the experiment. The resulting 240 ***MovieTrace*** output images were loaded into the Gnu image processing application (http://www.gimp.org) with the center portion of the well containing the worms masked off and the total number of blue pixels measured, divided by the total number of worms (green objects) observed, divided by the number of seconds of video used to produce the image. This mean pixel/worm/s value was then converted to um^2 by photographing a micrometer with the WormBot to determine the actual pixel size.

Results

Hardware Overview

The WormBot system (http://wormbot.org) is a benchtop gantry robot that moves a USB microscope camera over a transparent acrylic optical table to image wells containing agar and nematodes in standard tissue culture plates (Figure 1a). The camera is coupled to a transillumination arm carrying an LED, optical tube, and reflector that provides off-axis illumination and high contrast brightfield imaging. The system supports any camera that provides Video4Linux driver support, but the current software release and data presented here are based on a 1080p (1920x1080 pixel) USB 2.0 web camera with an f/2 4.8mm S-mount micro video lens that yields images with a pixel size of 24.4 microns. While the system can work with higher resolution cameras, the extra storage for the increased data generated and decreased light sensitivity have led us to favor standard HD cameras, which provide adequate resolution to resolve movements in adult worms. Additionally, these 1080p cameras are highly affordable and widely available at online retailers.

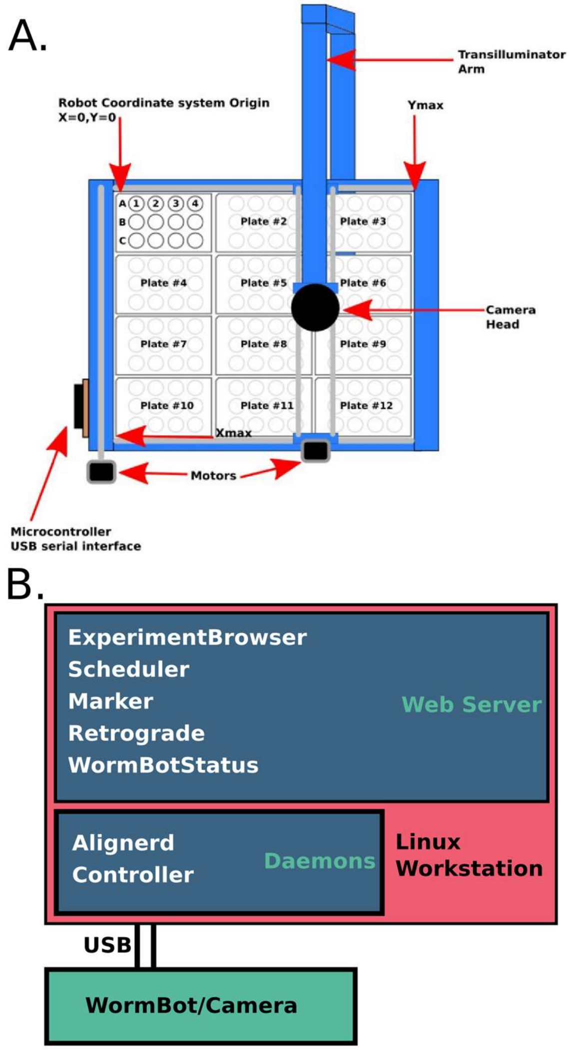

Figure 1. A. Top down diagram of the WormBot system. Plates are held in a 3 by 4 grid by steel dowel pins on a clear acrylic optical table. The camera head and transilluminator arm's motion is powered by a pair of stepper motors that are controlled by a microcontroller board that is connected a linux workstation via USB. B. Software components of the WormBot system. A pair of daemons handles image acquisition and image alignment. User interaction is handled through web server based tools that can be accessed by any web browser.

The main hardware of the WormBot system consists of an XY-plotter extruded aluminum robotics kit made by the company MakeBlock (MakeBlock, makeblock.com, Shenzhen, China). The WormBot system simply replaces the drawing head of the XY plotter robot with a camera and attaches a transillumination arm to the camera head, much like a standard compound microscope with a very long and curved arm. Due to this arm's size and mass distribution, a 3kg lead diving weight is placed on the camera head to serve as a counterbalance and reduce torsion on the linear bearings affording free movement of the camera head and transillumination arm. The movement of the camera head is MXL belt driven and powered by two NEMA 17 stepper motors running in an open loop control configuration by a clone of the Arduino UNO microcontroller board and stepper motor control boards that are provided in the MakeBlock kit. The WormBot's aluminum frame is attached to a ⅜-inch thick acrylic panel drilled to provide mounting holes for the robot chassis, four extruded aluminum or 3d printed legs, and an array of ⅛-inch steel dowel pins. These dowel pins lock 12 standard microtiter plates into position on the optical table. Using 12-well plates allows the system to analyze 144 individual wells. While this acrylic plate could be drilled by hand on a milling machine, it can also be ordered from an online machine shop (Front Panel Express, Seattle, WA, see Supporting Online Information for CAD files for ordering or drilling the panel). For complete video construction tutorials and assembly documentation, see http://wormbot.org.

The WormBot system is connected via two USB cables (one for the camera and one for the robot) to a single Linux workstation. It is advisable to install multiple large hard drives into this workstation as the 1080p cameras generate approximately 50 GB per day at total capacity while 4kUHD cameras generate up to 300 GB of data per day.

Software Overview

The WormBot hardware communicates via USB to a set of software daemons on the workstation (Figure 1b). The ***controller*** daemon coordinates the robot hardware's movement and captures the camera's image data. The captured images are stored as PNG files and served to the ***alignerd*** that uses an ECC image alignment algorithm[40] built into the OpenCV library to align each time-point for a well to previous timepoints. This step is necessary as the spatial resolution of the robot kit stepper motors running in an open loop configuration is limited; therefore the images must be aligned to reduce frame-to-frame noise.

The WormBot is designed to be used by an entire laboratory group; therefore, we designed the software's user interface to work inside a web browser so that any internet-connected device could be used to control the robot and analyze data on the system from any location. This also allows data to be shared easily between lab members and collaborators. The workstation connected to the WormBot runs a standard Apache Web Server (The Apache Software Foundation, Wakefield, MA) installed and configured automatically when the WormBot software package is installed. Experiments are started and stopped on the WormBot using the ***scheduler*** application (Figure 2a, /cgi-bin/scheduler). The ***scheduler*** page is broken into two parts. The upper red portion lists currently running experiments that can be individually stopped or clicked on to access the ***marker*** application (Figure 2b). The lower green portion lists available plate slots on the WormBot and allows users to add new experiments to the robot's joblist. The WormBot performs two types of data collection that can be used simultaneously. *Time-lapse*: where the robot takes a single image of a well every 10 minutes, and *Daily-Monitor*: where once per day, the robot will position the camera over the well and take a 1-5 minute video at 30 frames per second. We typically rely on time-lapse data for determining the time of death and daily monitor videos for healthspan, behavior, or other movement metrics, but they could be used interchangeably. The ***experimentbrowser*** application (Figure 2c., /cgi-bin/experimentbrowser) allows users to see all of the experiments that are currently stored on the server and to back them up or delete them.

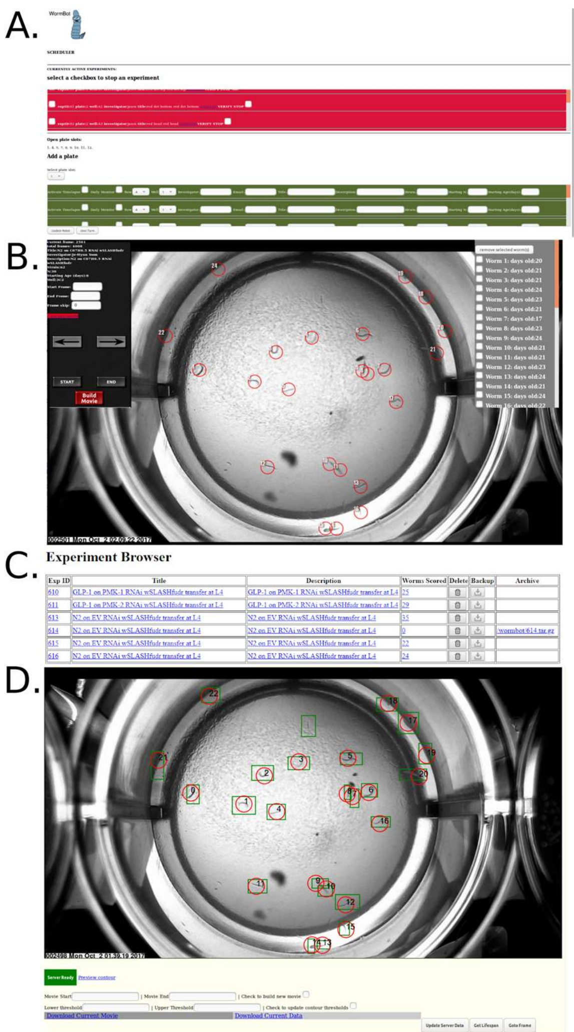

Figure 2. Web applications for the WormBot system. A.The ***scheduler*** application allows users to start and stop experiments. B. The ***marker*** application is designed for manual scoring of lifespan timelapse data and generation of annotated timelapse movie files. C. ***ExperimentBrowser*** displays all of the experiments stored on the server and allows for data management. D. The ***retrograde*** application allows for manual or automated lifespan scoring. Users mark the corpses (green rectangles) at the end of

the time-lapse data and the software automatically determines the death point (red circles) of the enclosed worm object.

In order to generate survival data, the WormBot system provides two different applications. ***Marker*** is a cgi based web application that can run on any internet-connected device, like an iPad, smartphone, or web browser (Figure 2d, cgi-bin/marker?loadedexpID=X). ***Marker*** allows the user to manually scroll through the stack of time-lapse images and click on a worm when it ceases all movements (see discussion section below for a discussion of how death should be determined). Marking a death event produces a red circle which the ***marker*** application can also embed and export as a timelapse movie of the entire experiment (see supporting online VideoS1). The application also lists basic experimental statistics and provides the data in an OASIS compatible format[49] for further analysis. We have written another application, ***retrograde*** (Figure 2e., /retrograde/retrograde.html), which attempts to automate the process of identifying the time of death, alleviating the necessary time investment of manually annotating images. ***Retrograde*** is a javascript application that must be run on a web browser, controlled with a three-button mouse, and provides the same basic manual scoring functionality as the ***marker*** application. For data to be fed into ***retrograde*** prediction algorithm, the user goes to the end of the time-lapse image data and uses the mouse to drag boxes around each worm corpse. The user then hits a button, and the software moves backward in time to find the point at which the objects in the boxes begin to move and marks them as death points. ***Retrograde*** relies on OpenCV's image thresholding and a Canny edge detection algorithm to detect worm-sized objects in the user-defined bounding boxes and determine when they have moved outside of the noise thresholds that are required due to time-point to time-point image variability.

In order to validate the ability of the WormBot to detect differences in *C. elegans* survival, we used RNAi to disrupt members of the well-described insulin/IGF signaling pathway[42,50] (Figure 3). Results from the WormBot were compared to control animals that were kept in identical 12-well plates stored next to the WormBot but were scored by hand using a worm pick and dissecting microscope. Both in empty vector controls and in worms treated with dsRNA targeting the longevity-promoting transcription factor *daf-16,* results were not statistically different when scored by traditional methods or the WormBot ***marker*** application, which allows human users to determine the death points from the time-lapse data manually (Figure 3). We next tested to see if knocking down the *C. elegans* insulin-like growth factor receptor DAF-2, which results in a full lifespan extension, could be detected on the WormBot. When analyzed by hand or with the ***marker*** application, *daf-2* RNAi resulted in longer lifespans compared with EV controls, but *daf-2* RNAi animals were even longer lived when assayed on the WormBot (Figure 3). While the ***marker*** application yields results similar to traditional methods, it still requires a large amount of user analysis to generate a survival curve. The ***retrograde*** application attempts to remove much of this user effort by leveraging computer vision approaches to detect when worms cease movement. In all cases, the ***retrograde*** application was able to correctly detect the differences between the three treatments (Figure 3 bottom right); however, in all cases, ***retrograde*** overestimated the lifespan compared to traditional hand scoring (Figure 3, green curves).

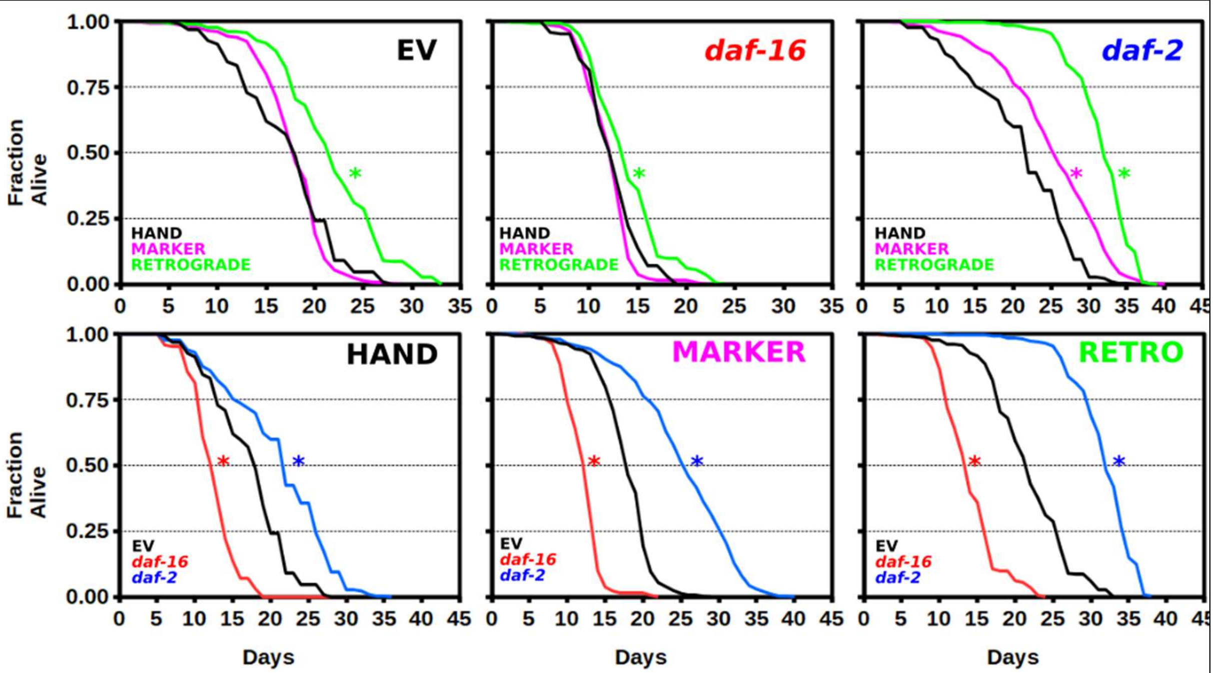

Figure 3. Variation in wildtype survival due to inhibition of insulin signalling by RNAi as analyzed on the wormbot or by traditional methods. Top row and bottom row contain the same 9 survival curves sorted by RNAi treatment (top) or by analysis type (bottom row). Wildtype worms were fed *E. coli* expressing double stranded RNA for the insulin growth factor receptor DAF-2 or it's downstream effector, the FOXO transcription factor DAF-16. All worms were cultured in 12-well plates starting at the L4 larval stage. Plates were scored either by traditional hand scoring with a dissecting microscope (black curves on top row and bottom left panel), using the ***marker*** application from the WormBot system (magenta curves on top row and bottom middle panel), or with the automatic feature of the Retrograde application (green curves top row and bottom left panel). * indicates P value < 0.05 using Log-Rank Test.

Sample to-sample variability was an issue for the designers of the Lifespan Machine, owing to the closed format of the scanners and their tendency to trap heat. Since the WormBot was able to detect changes in lifespan due to disruption of a canonical aging pathway, we next looked at the variation in the lifespan of wild-type animals across all of the wells of the robot in a large number of individuals (Figure 4a). Perhaps due to the open and symmetrical design of the WormBot, we find that when binning wells across all 12 plates of the robot, we detect no difference in mean lifespan in any of the 12 well positions (Figure 4b, type I ANOVA, F=1.24 p=0.27). The average temperature variation between the WormBot wells and the surrounding environment was less than 0.04°C, with the system running a complete 144 experiments (Table S2).

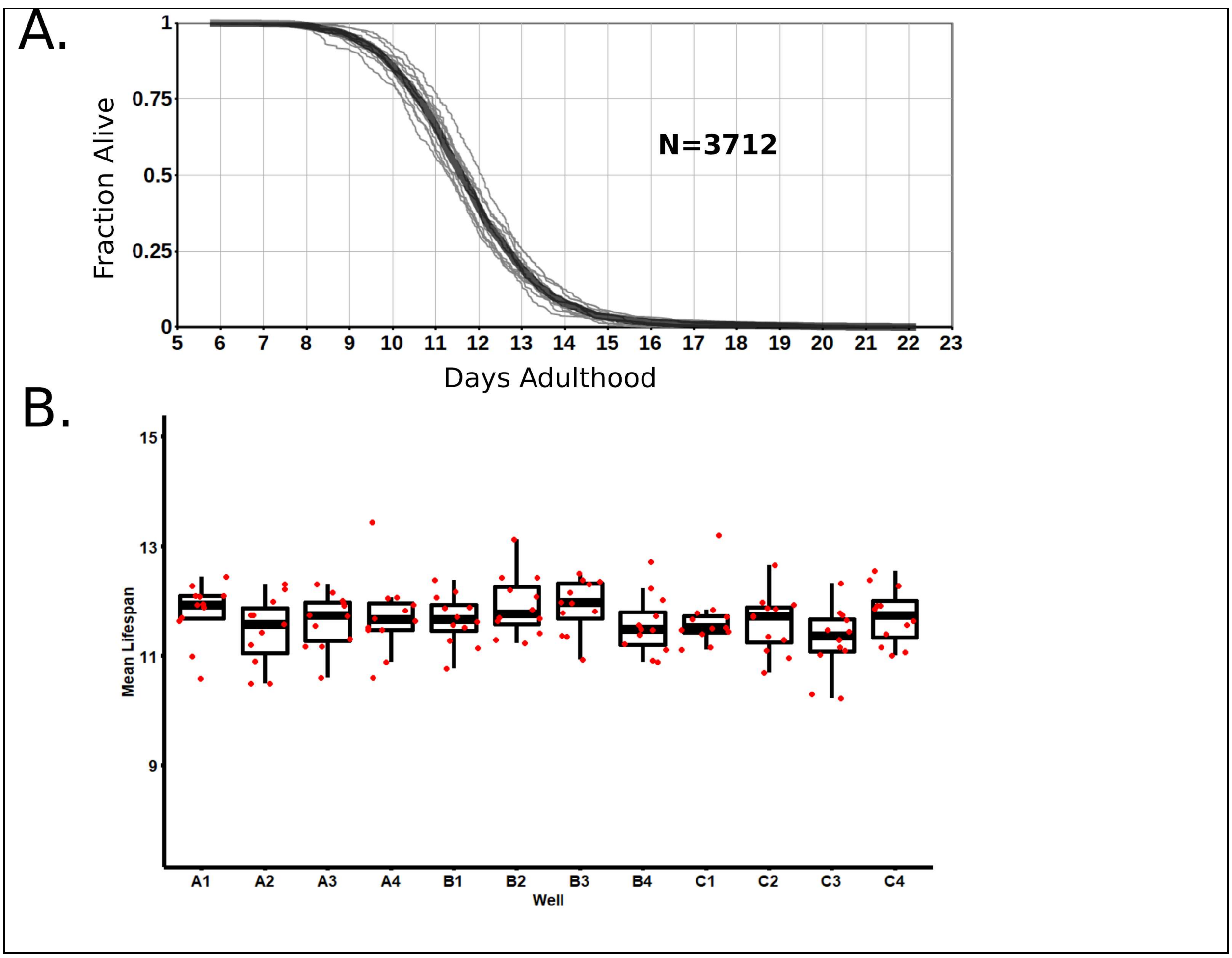

Figure 4. Experimental variability in wildtype aging. A. 143 wells of N2 grown on OP50 either treated with 1% water or 1% DMSO treatment had no significant effect on outcome and curves from binned plates (Gray lines) are shown along with mean of all wells (Black line). Fractional day output was used and timepoints are separated by 10 minutes. B. Box and whisker plots of mean lifespan for individual well positions (red dots,A1,A2, etc) were binned and mean lifespans for binned groups was found to not differ significantly.

One of the critical features of robotic image acquisition is that it allows for finer temporal resolution survival data than is practical with human-acquired data sets. This feature is useful when dealing with highly toxic drugs or sensitive strains. From an experimental drug or genetic screening standpoint, this is powerful because these shorter assays afford the ability to increase throughput significantly. For example, resistance to high-temperature stress has previously been shown to positively correlate with increased longevity, and exposure to 35°C will kill wild-type worms within 10 hours, while the long-lived *daf-2(e1370)* strain can survive up to 18 hours[51]. In order to test the ability of the WormBot to perform high-resolution survival studies, we exposed *C. elegans* to the lethal toxin HCN. Previous work on the susceptibility of worms to the human pathogen *Pseudomonas aeruginosa* (PAO1) identified bacterial production of HCN as being responsible for the fast paralytic killing of the worms when exposed to the pathogen, and resistance screens identified the prolylhydroxylase EGL-9 as conferring resistance to lethal HCN exposure[48]. Subsequent work showed this resistance was due to activation of the hypoxia-responsive transcription factor HIF-1 and its downstream targets[36]. We placed L4 larval stage worms, with loss of function mutations in either EGL-9 or HIF-1 into sealed 12-well plates in the presence of 270 ug of HCN and placed them onto the WormBot (Figure 5). After 2000 minutes (200-time points), the data were analyzed. While only two *egl-9* animals died during the observed period, all of the *hif-1* animals were dead within 1300 minutes, and 75% of wildtype animals were dead by 2000 minutes. These data show that the WormBot can provide high-resolution survival data for even highly toxic and fast-acting compounds such as HCN.

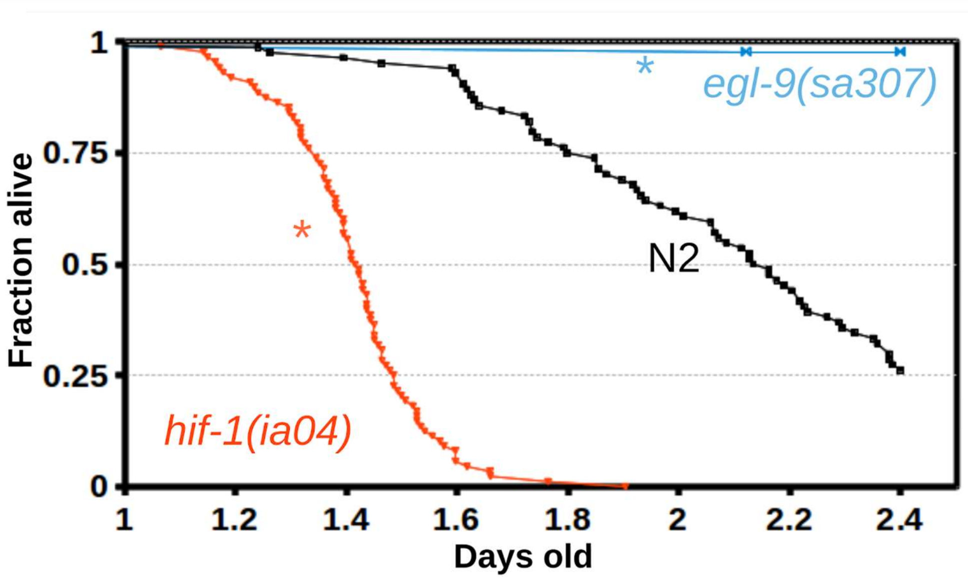

Figure 5. High-resolution survival data in HCN atmosphere over 2000 minutes. Wildtype (N2), *elg-9(sa307)*, and *hif-1(ia04)* worms were placed into a sealed 12-well plate with an acidified solution of KCN to liberate HCN gas. The WormBot's fractional day output was used to generate curves with a 10 minute (0.007 day) temporal resolution starting from the time of HCN exposure at L4 (one day old animals). * indicates P value < 0.05 using Log-Rank test.

While the maximum and mean lifespan are valuable metrics to infer organismal health, the geroscience community has recently begun to favor the concept of healthspan, rather than lifespan, as a more useful metric to evaluate the effectiveness of anti-aging interventions[43]. Specifically, the idea that interventions that extend lifespan but do not also reduce disease burden or moribundity in animal models are not translationally relevant. While the healthspan concept presently lacks the same rigorous definition as lifespan, central to this concept is the ability to measure "health" metrics as the organism's chronological age increases[52]. Health metrics that have declined during normative aging in *C. elegans* include crawling, feeding, reproduction, mechanosensation, and olfaction[41,53–58]. When equipped with a high-resolution camera, the WormBot is able to detect laid embryos (Figure S4), but our standard low-resolution cameras cannot resolve eggs unambiguously. However, spontaneous movement and orientation to aversive or attractive odorants could be quantified. In order to visualize the physical activity of animals, we used the *dailymonitor* feature of the WormBot software to acquire real-time video (30 frames/second) of the worms each day for the entire course of the experiment. These .avi video files can be used with any of the widely available worm tracking packages that have been previously described[59]. However, because implementing many of these trackers requires its separate installation and batch processing procedures, we developed a straightforward command line application for the WormBot system called ***plateExplorer*** that processes all the dailyMonitor movies for an experiment and runs the same worm detection algorithms as our ***retrograde*** application to produce a graphical output of the paths worms take during the recorded movies (Figure 6a). These image files can then be mined with standard image processing applications (see Materials and Methods) to provide data about worm movement throughout the lifespan experiment (Figure 6b). As worms age, their spontaneous movement peaks around days 2-3 and then slowly declines. Similar spontaneous lifetime movement profiles have been observed in the WormMotel[38], and we observe it in our wildtype controls (Figure 6c).

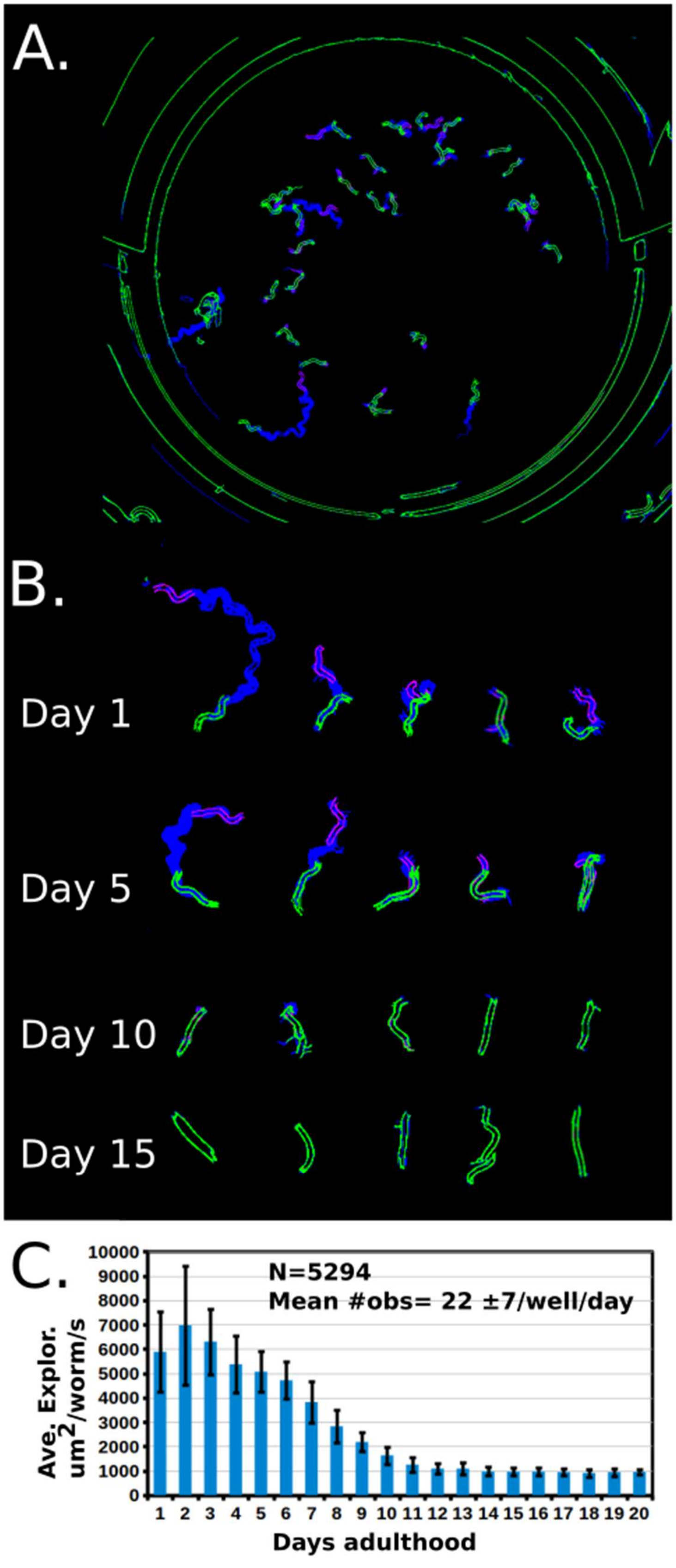
A.
B.
Day 1
Day 5
Day 10
Day 15
C.
N=5294
Mean #obs= 22 ±7/well/day
Ave. Explor. um2/worm/s
Days adulthood
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Figure 6. Lifetime activity data. Plate exploration was scored using *dailyMonitor* movies from the N2 control data in figure 4. A. Output of the WormBot application ***plateExplorer*** to produce image file outputs showing the region of the well worms explored during the recorded time frame. B. Samples of zoomed in traces from days 1, 5, 10 and 15 of the same experimental well. Worm position at the start of the movie is marked in the green channel, the ending position in the red channel and all intervening frames (1250 in this analysis) summed into the blue channel. C. 12 wells were randomly sampled from each of the 12-plates (143 experiments) in Figure 4 and each of their daily traces combined to generate the composite curve showing the average number of um^2 explored by a worm per second as a function of age.

Discussion

Here we describe the WormBot, an inexpensive, open source robotics system for high-throughput lifespan and behavioral phenotyping in *C. elegans*. We find that the semi-automated lifespan scoring features of the WormBot software are able to differentiate between RNAi treatments known to increase and decrease lifespan, and that human annotated WormBot data is comparable to traditional manual lifespan methods at substantially reduced time and effort. In addition to quantifying mortality under standard conditions, we show that the WormBot can easily resolve survival under highly stressful conditions such as hydrogen cyanide exposure, and can be used to quantify simple healthspan metrics such as motility. Obvious additional applications of the WormBot include pathogen sensitivity/resistance and chemosensory (avoidance/attractant) assays. Thus, we anticipate that the WormBot has the potential to greatly accelerate the pace of *C. elegans* research in a variety of contexts.

We performed a direct comparison of data obtained on the WormBot to manual lifespan determination by traditional methods. In all cases, the WormBot data was comparable to manually obtained lifespan data and successfully discriminated short-lived (*daf-16*) and long-lived (*daf-2*) conditions from wild type control. The fully automated ***retrograde*** survival analysis yielded survival curves that were consistently longer-lived than manual annotation of the WormBot data or traditional hand scoring. We do not yet fully understand the reasons for this; however, because the relative differences in lifespan are maintained, we believe that the automated system is suitable for screening purposes. Because the WormBot data is stored as a series of image files, the data can always be re-analyzed using human annotation. Of interest, the human annotated WormBot data yielded lifespan effects that were not significantly different from traditional manual methodology for both control and *daf-16(RNAi)* conditions, and slightly longer lifespan values for *daf-2(RNAi)*. We speculate that this may result from the damage caused by repeated manual prodding over the long lifespans of *daf-2* animals when the experiment is carried out using traditional methods.

The WormBot design appears to be robust against position effects and thermal variation from well-to-well and plate-to-plate. When samples were binned by well position, we observed no statistically significant differences between mean lifespan across the 12 possible well positions on the robot.

A major unmet challenge for the current WormBot system, as well as other automated lifespan systems for *C. elegans*, is an inability to unambiguously identify the precise time of death for each animal. While fully automated survival analysis is a goal we hope to achieve in the future, human curation of the time-lapse image data currently provides survival curves that closely match traditional manual lifespan analysis. To do this, a person scores the time point immediately following the last observed spontaneous movement as a “death” point. While still requiring some human effort, we have found that this approach requires less than 20% of the time required for traditional manual lifespan experiments and can easily be performed by undergraduate students with significantly less training required.

Extending past this stage of development, the next generation systems we are currently designing will feature higher resolution optics and fluorescence detection to observe tagged intracellular proteins, in order to study age-related changes in proteins, organelles and tissues. However even with brightfield microscopy and limited spatial resolution of the current system, we can detect the cause of death in some animals. In addition to age associated vulval integrity defects[60] we have detected several instances of premature worm death that were caused by pathogenic fungal infections (Video S1). Here again, time-lapse observations provide information that would likely have been overlooked in traditional hand scoring. While the current WormBot platform is geared largely toward survival analysis, it could be used for any assay that could be scored with a low power brightfield microscope. For example, a WormBot was recently built in order to study magnetotaxis in the worm (Andres Gadea-Vidal, pers commun) and, conceivably, any assay that involves determining the position of worms on a plate could be adapted to the WormBot. The open-source nature of the WormBot software and the flexibility of the open-source hardware design will allow for future enhancements and upgrades of an already very useful system.

Finally, science is in the midst of a reproducibility crisis[32,61]. The causes of this crisis are multifactorial, but range from simple statistical errors, like inadequate sample size, to outright data fabrication in the face of dwindling pay lines. Systems like the WormBot that automatically create archival time stamped digital observations that can easily be blinded, shared and reexamined are an important tool in addressing this crisis. In this aspect, automation itself is also beneficial as it is designed to be carried out by machines and is therefore necessarily more highly controlled owing to the limitations of these mechanical systems. Automation also allows for larger sample sizes, decreased experimenter bias and fewer human errors during data collection. All of these factors point to laboratory automation tools like the WormBot revolutionizing the scale, rigor and scope of science in the upcoming decades.

Chapter Two:

Utilization of the WormBot for compound and genetic intervention screening in *C. elegans*

Abstract

Reducing disease burden and delaying senescence through targeting the underlying molecular mechanisms of aging is a promising area of research that has been hampered by limited throughput. Although several distinct genetic targets have been identified along with potential therapeutic interventions, much of the current work has focused on these existing pathways. Here we present two avenues of work that seeks to expand on the how the WormBot platform can improve genetic and compound intervention screening. First, we present a comparison between DrugAge reported data and data generated via the WormBot platform as well as an examination of its role as a central hub of aging intervention information. Second, we use a novel machine learning approach to identify putative genetic modulators of Alzheimer's Disease (AD) from human gene expression data and then validate them in a *C. elegans* AD model. These two approaches highlight the utility of the WormBot system and provide further validation of its role in the automation of invertebrate lifespan experimentation.

Introduction

Geroscience

Although the field of Geroscience is relatively new, the goal of understanding how and why individuals age far predates even modern medical science. As advances in the molecular biology of aging identified conserved pathways that, when perturbed, fundamentally changed the rate and trajectory of aging[62]. When coupled with the existing observation that the major risk factor for many chronic diseases was an individual's overall age, the theory grew to encompass the idea that disease burden could be alleviated through perturbation of aging mechanisms as opposed to directly targeting a specific pathology. One advantage of the geroscience paradigm is that it innately addresses the interconnectivity of different systems operating within an organism[62].

Alzheimer's Disease

Alzheimer's Disease (AD) is one of the most prevalent age-associate diseases and as of time-of-publication remains without a standard treatment. Currently AD affects ten percent of individuals in the U.S. and with the growing number of individuals in that age bracket the human and financial costs are expected to drastically increase[63]. Geroscience studies have shown promise that one potential avenue for treating AD is to target the underlying molecular causes of aging and thereby reduce the risk of onset and potentially progression of disease. This avenue is particularly attractive given that the underlying pathophysiological causes of AD are likely multi-factorial and highly complex and a single-target solution might only be partially effective. Although current work investigating many potential causative agents such as Aβ, tau and phosphor-tau is vital to long-term efforts of treating the disease, broad spectrum age-targeting approaches are likely to provide some solutions on a realistic timeline to treatment.

Although mouse models of Alzheimer's Disease exist, they suffer from the common drawback that almost all rodent models have, namely that experiments are cost-intensive and take several years on average to complete. As a result, several invertebrate models of AD have been successfully developed. One common *C. elegans* models is that of GMC101[64]. This model expresses human $A\beta_{1-42}$ in the body wall muscle and results in a progressive and age-associated paralysis and has been validated through the use of amyloid binding dyes as well as several interventions that have shown promise in pre-clinical trials. However, care must be taken to correctly contextualize the utility of this model. As *C. elegans* do not possess a brain or complex nervous system and since the GMC101 model itself only expresses $A\beta_{1-42}$ in the body wall muscle, linking any results directly to human neuronal function is highly ill-advised. Instead, GMC101 is better contextualized as a model of Aβ proteotoxicity rather than a disease model as a whole. While not explored here, one additional potential avenue that would better model AD would to express Aβ neuronally alongside other phosphor-tau[65,66]. However, as the role of invertebrate models is to either broadly screen using high throughput methods or leverage the tractable and well understood genetics of a given model organism, the benefits of GMC101 as a model well outweigh the deficits.

High throughput drug screening in invertebrates

Although the relatively easy culturing and experimental properties of *C. elegans* has allowed for widespread use within the laboratory setting, the last two decades have seen multiple advancements at increasing throughput and rigor through automation.

Most of *C. elegans* publications to-date have utilized historical laboratory techniques that have been relatively unchanged for decades. Primarily, *C. elegans* are manipulated through either pipet transfer of populations or through picking of individuals using a wire (or hair) instrument[15]. As *C. elegans* are relatively small, even historical techniques such as these afforded a much higher throughput in terms of number of individuals per researcher than the majority of other animals model systems[67]. However, these techniques would still entail experimenters visually assaying each sub-population and replicate every day from experiment start to the point at which all animals were deceased. The limitations of such an approach include an upper capacity of a researcher of one to two thousand animals at a time as well as the associated risks of human error in each experiment. In most cases, this methodology also limits the data generated to only that of the total lifespan for each animal under observation. While metrics of health and life-history can also be collected using manual techniques and human observation, they are commonly performed as secondary experiments following an initial lifespan screen. Until the advent of automated technology, this remained the rate-limiting step across most experimental designs. In 2019, the Kaeberlein lab published the designs for the WormBot, an image capture platform designed to automate the collection of lifespan data from *C. elegans* experiments[68]. This system was designed around the model of plate-based assays that have been the norm since the early days of *C. elegans* work, with the caveat that data recording was performed via computer vision as opposed to experimenter observation. This data would take the form of either timelapse images collected every 10 minutes or 1 minute recordings taken once a day. These datapoints can either be analyzed via semi-automated computer vision or through manual annotations using a custom web based interface. Notably, this methodology allows for easy remote analysis, re-analysis, and quality control methods.

DrugAge database and compilation of *C. elegans* work

In 2017, Barardo et al created the DrugAge database to compile and report the efforts of the geroscience field to screen through compound interventions[69]. While this database likely is enriched for positive reporting and does not encompass the entirety of all compound interventions tested for their effects on lifespan, it is to-date the most comprehensive repository of such information. Using the more recent build of the database (at time of writing) there were still only 1096 compounds tested across 38 potential animal models. In particular, while 716 compounds were reported in *C. elegans* and 102 were reported in mouse models, only 9 compounds had been reported to have a positive effect in both animal systems. Although some work has been performed using the reported DrugAge database to predict novel anti-aging compounds, this dataset realistically encompasses only a small fraction of the potential compounds of interest[70]. This underscores the necessity for broader and either unbiased or semi-biased screening to more fully develop the predictive models.

Use of AI to identify targets for screening

Although the potential of using databases of age-modulating compounds remains relatively undeveloped, the scope of data available to mine from human omics data is much broader. In partnership with lab of Su-In Lee in the University of Washington School of Computer Science, we used their PAUSE pipeline to experimentally validate a deep-learning approach to identifying novel potential modulators of AD[71]. This methodology builds on existing deep-learning methodologies to enable fully unsupervised analysis of gene expression datasets. In particular, PAUSE was applied to post-mortem brain RNA-seq measurements from multiple sources including ROSMAP (Religious Orders Study/Memory and Aging Project), ACT (Adult Changes in Thought), MSBB (Mount Sinai Brain Bank), and HBTRC (Harvard Brain Tissue Resource Center). Using PAUSE to predict pathways of importance to the progression of Alzheimer's Disease, we found that the major source of variation was due to the Hallmark Oxidative Phosphorylation pathway. As mitochondrial mechanisms have been strongly implicated in the pathogenesis of AD[72], this highlights the strength of PAUSE to generate clinically relevant targets. Following the identification of this pathway, further exploration of specific genetic signatures was performed and several genes encoding for mitochondrial respiratory Complex I were identified (NDUFS3, NDUFA3, NDUFS7, and NDUFA2) and selected as part of our invertebrate validation pipeline.

Methods

Identification of C. elegans homologs

To enable biological testing of the human genes identified using our computational analysis, we obtained the reciprocal best hits (RBHs) (between human and *C. elegans*. We first identified ~~all~~ unique protein sequences for each potential marker gene using the biomaRt R package available on CRAN[73]. Then, we used the NCBI BLAST tool to identify the *C. elegans* orthologs for each complete human protein query sequence[74,75]. We downloaded the *C. elegans* protein sequences from wormbase.org/species/c_elegans. We took into account only the protein pairs mapped from human to *C. elegans* with a BLAST e-value smaller than 10^{-30}. For each *C.*

elegans isoform, we identified the corresponding human genes, again using the NCBI BLAST tool, and used only the orthologs that achieved a BLAST e-value smaller than 10^{-30}. This process resulted in high-confidence RBHs for us to test in *C. elegans*.

C. elegans strain, cultivation, and RNAi treatment

Standard procedures for *C. elegans* strain maintenance and manipulation were used, as previously described[76]. All experiments were performed using the GMC101 strain expressing the human $A\beta_{1-42}$ peptide under the unc-54 promoter[77]. Experimental worm populations of GMC101 animals were obtained from the *Caenorhabditis Genetics Center* (CGC) and cultivated on NGM plates with OP50 *E. coli* at 15C. Care was taken to ensure that the animals were never starved and the plates remained free of contamination.

The gene-specific RNAi clones were obtained from the commercial Ahringer or Vidal *C. elegans* RNAi-feeding libraries (BioScience, Nottingham, UK). Each bacterial clone was struck-out onto LB plates containing carbenicillin (50 ug/ml) and tetracycline (10 ug/ml). Single colonies were then seeded into 5 ml LB + carbenicillin (50 ug/ml) and tetracycline (10 ug/ml) for growth overnight on a 37C rotator. 100 ul of each overnight culture was then inoculated into 10 ml of LB containing carbenicillin (50 ug/ml) and tetracycline (10 ug/ml) and IPTG (5mM) and incubated on a 37C rotator for 4 hours. Each bacterial growth was then centrifuged at 3500 X G for 25 minutes, decanted, and the pellet resuspended in 0.5 ml of LB containing carbenicillin (50 ug/ml), tetracycline (10 ug/ml), and IPTG (5 mM). To verify RNAi conditions' plasmid DNA, each RNAi clone was purified and assessed through PCR (polymerase chain reaction) with sequence-specific primers or through Sanger sequencing.

Nematode lifespan and paralysis assays

Paralysis assays were performed by visually inspecting recordings of the animals to determine if they were capable of normal locomotion or if they were paralyzed and unable to transit the agar plate. A custom built robotic system, the WormBot[68], was equipped with a digital camera and used to obtain images of individual wells of a 12 well-plate at 10-minute intervals over the entire course of the experiment. Each well contained 25-40 *C. elegans*. Using a custom-built web interface that enabled manual annotation of serial images from each plate, the age at which each animal motility stopped could be easily determined. For wildtype or normatively aging animals, time-of-death was determined based on total cessation of any movement, the last of which is usually a small twitch in anterior or posterior of a given animal. For the transgenic Aβ model line GMC101, the time of paralysis onset for each individual animal was determined via cessation of gross locomotion but not total absence of twitching behavior. Statistical significance of mean paralysis time-points between RNAi conditions was determined by a weighted log-rank test.

For the GMC101 transgenic animals, prior to loading on the experimental plates, animal populations were propagated on high-growth plates seeded with NA22 bacteria. Worm populations were developmentally synchronized by hypochlorite treatment, and the remaining eggs were deposited on unseeded plates overnight. Synchronized larval stage 1 animals were washed off unseeded plates and moved onto standard *C. elegans* RNAi plates containing carbenicillin (50 mg/ml), tetracycline (10 mg/ml), and IPTG (5 mM) 48h at 20C. These developmentally synchronized, late larval stage 4-populations were then washed and transferred to their respective RNAi conditions on 12-well plates.

We used standard RNAi conditions plus FuDR (100 ug/ml) to prevent progeny and nystatin (200 mg/ml) to prevent fungal growth.26 Each RNAi condition was tested in 2-3 wells as technical replicates. At least three biological replicates, each started on different weeks, were conducted for each RNAi clone. Animals were maintained at ambient room temperature (22-24C) over the course of the paralysis assay.

For N2 animals, prior to loading on experimental plates, animal populations were grown on NGM plates seeded with OP50. Following hypochlorite treatment to synchronize the populations, arrested L1s were left on a rotator in S-Basal solution overnight and then plated on NGM plates seeded with OP50 the next day. Experimental plates were prepared by mixing the drug into molten NGM agar (containing FUDR 100 ug/ml and Nystatin 200 mg/ml) at the desired concentration and then immediately pouring into 12-well cell culture plates using a stereological pipet. Agar is then allowed to solidify on the benchtop for 1 day. An overnight of OP50 *E. coli* is inoculated and grown in LB-media and then concentrated 20X via centrifugation. Afterwards, 75uL of concentrated OP50 is then added to each well. Then, developmentally synchronized, late larval stage 4-populations are transferred to each well via liquid pipet until there are 25-35 animals in each well. Lastly, animal-containing plates are transferred to the WormBot and images are collected while ambient room temperature is maintained (22-24C).

Results

DrugAge analysis

Using the most recent online build of the DrugAge database, the raw data was downloaded and imported into RStudio. In brief, the most recent build incorporates 3265 individual entries (where each entry is a report of a given compound and dosage from a single publication) across 1096 unique compounds. Of the total number of unique compounds, only 597 compounds were reported more than a single time. Conversely, the compound reported most often, Resveratrol, was reported 166 times in total. Ethanol, the next most commonly reported, encompasses only 52 separate entries. To better understand the amount of variation across the scope of the DrugAge database, each unique 'compound-by-concentration' pair was ordered by the number of appearances in the database and the variations in lifespan across each combination were reported (Figure 1)

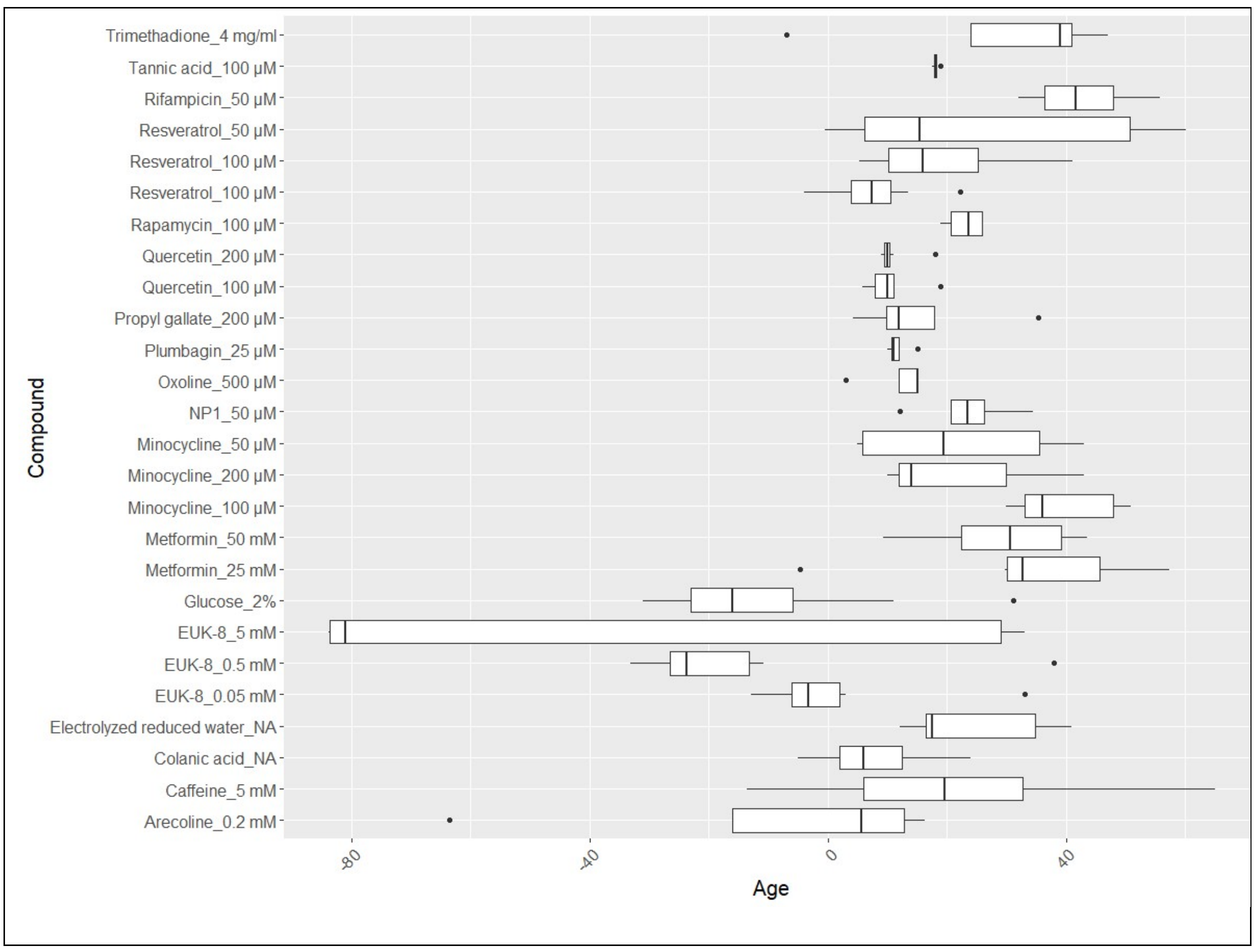

Figure 1. Variation in lifespan reporting across the most repeated drug/concentration combinations. DrugAge database was exported and filtered by the most repeated drug/concetration combinations. Boxplots were generated for each combination showing the reported percent effect for each.

We then performed a screen of compound interventions primarily derived from natural products available over-the-counter in the U.S but with additional highly promising compounds from DrugAge included as well (Figure 2A). Each intervention was tested in triplicate and then analyzed via the WormBot's interface by an experimenter. We also compared the maximal reported %-change in lifespan for each drug with what was determined via our internal screen (Figure 2B) and found that DrugAge results were generally larger than what was observed on the robot.

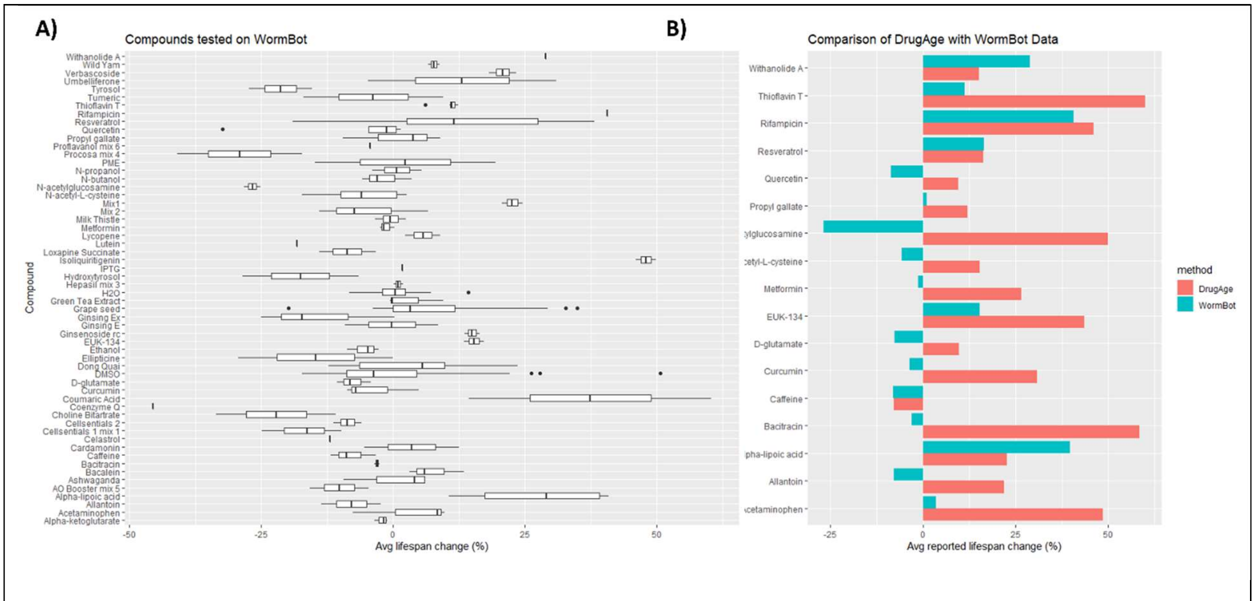

Figure 2. Natural product screening on the WormBot platform. A) A compilation of natural products and several synthetic compounds were tested for their effects upon *C. elegans* lifespan. 30 animals are tested per well and the average lifespan change relative to the matched

solvent controls are shown. B) From the compounds tested in 2A, matching reported effects from DrugAge are shown as average percent change relative to control. DrugAge effects are from the maximum reported *C. elegans* lifespan.

PAUSE screening

Unlike the single-cell datasets with clear ground truth perturbations and well-studied downstream effects examined in the prior analyses, the biology underlying the AMP-AD Alzheimer's is less clearly characterized. Therefore, in order to verify that the genes and pathways identified by PAUSE are biologically-relevant, we could not reference a known ground truth, and had to experimentally validate our findings.

To gain insight into the biological relevance of the genes identified by our PAUSE analysis of the Alzheimer's brain expression data, we used the nematode *C. elegans*, a well-established animal model of Aβ proteotoxicity. We conducted experiments with GMC101, a transgenic worm line displaying an age-associated aggregation of human $A\beta_{1-42}$ peptide resulting in rapid onset of age-associated paralysis. A stringent reciprocal best hits (RBH) approach (BLAST e-value <= 10^-30; details in Methods) was used to identify nematode orthologs for human genes.

We used RNAi via bacterial feeding to reduce expression of 13 *C. elegans* genes encoding homologs of human Complex I proteins in animals expressing toxic Aβ. This resulted in a striking delay in paralysis relative to control animals treated with empty vector (EV) RNAi (Figure XXX) (details in Methods). Suppression of paralysis by knockdown of Complex I genes was comparable, or often exceeded, the suppression of paralysis by RNAi knockdown of the insulin-like receptor gene daf-2 (Figure 3), one of the strongest known suppressors of Aβ toxicity in worms.

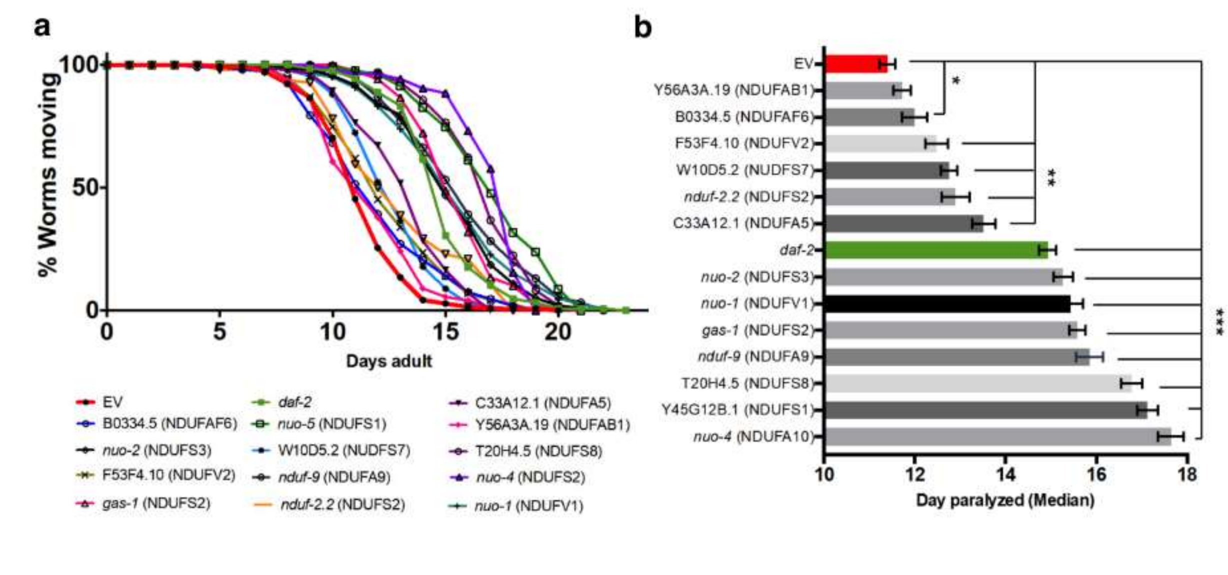

Fig 3. *C. elegans* Aβ proteotoxicity assay validates importance of mitochondrial Complex I genes. a, Paralysis curves for the reciprocal best orthologs of human Complex I genes. All tested RNAi conditions significantly suppress paralysis. b, The same data as in a plotted as the median day of paralysis for each population of worms. Half of the conditions showed even stronger suppression than daf-2 RNAi conditions. Error bars indicate standard error of the mean across experiments. * p-value < 0.05. ** p-value < 0.01. *** p-value < 0.001

Following this initial screen, a larger subset of genes were screened with the several of the identified Complex I genes acting as positive controls (in addition to the positive control of *daf-2*). Several genes were identified that worked as well as the positive controls including isp-1 and cyc-1 (Figure 4). Interestingly, both of these genes are also involved in mitochondrial function, providing further evidence for the hypothesis that mitochondrial health is a potent modulator of Aβ-associated proteotoxicity in the GMC101 model.

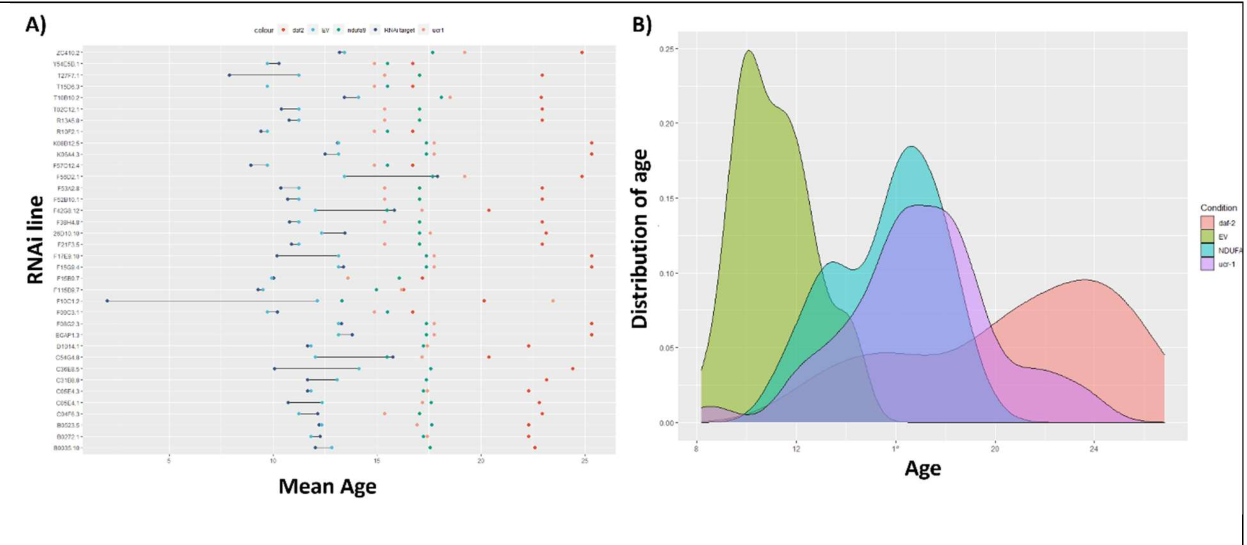

Figure 4. *C. elegans* RNAi screening identifies additional mitochondrial complex 1 genes as modulators of the Aβ-induced paralysis. a) Cleveland dot plot of 36 genes knocked down through RNAi feeding. b) Density plot of control RNAi effects.

Discussion

As the use of *C. elegans* ushered in many of the discoveries of the modern era of molecular genetics, it's integration with automated technologies and machine learning promises the next era of geroscience will be guided, in part, by its use as well. Until the last decade, any study utilizing *C. elegans* was constrained by the amount person-time that could be applied. With the advent of screening platforms such as the WormBot, truly large scale aging-intervention screening can occur. As can be seen from the analysis of the DrugAge database, the overall throughput of reported lifespan studies is extremely limited. For instance, ~5% of the reported observations on DrugAge are all resveratrol. This enrichment for previously identified compounds of interest is understandable, and even necessary to an extent, but with limited screening resources such enrichment comes at the cost of unbiased or even semi-biased screening. Furthermore, even though DrugAge is drawn from peer-reviewed sources, across multiple entries for the same compound tested at the same concentration in the same species, large variations in lifespan are still reported. While this is partially indicative of the difficulties in ensuring reproducibility in experimentation[29] it could also be compounded by less than optimal data mining. For instance, in *C. elegans* alone, 50 uM Resveratrol is reported to extend lifespan by as much as 50% and as little as 0%.

The variation of lifespans for a single drug concentration for a model organism that has a relatively standardized experimental paradigm is nothing new[29,32]. Prior work has shown that simple variations in laboratory practices can produce significantly different lifespan results. In *C. elegans*, poor larval staging, inconsistent compound delivery, improper nutrient control and interactions with FUDR can all result in increased variation within an experimental population. As the majority of previous *C. elegans* work was collected via human observation, checks into the rigor used in experimentation necessitate performing additional replicates. However, one solution that eases this burden is the use of digital microscopy and other forms of laboratory automation. Groundwork laid by the Automated Lifespan Machine and WorMotel systems was continued through the development of the WormBot. By using these systems, interpretable recordings of biological data are recorded and stored for as long as desired. This allows for a much higher degree of quality control for the most common laboratory errors as images and videos can be assessed for animal starvation, contamination by either fungal or bacterial cultures, or gross changes in the plate microenvironment. Furthermore, rather than a single experimenter's notes being the ground-truth for data recording, digital images can be analyzed by multiple experimenters to correct for bias.

While the results of our natural product screen do show variation in lifespan, our data was independently analyzed by multiple experimenters and then subsequently checked by a separate individual. This ensures that variation was due to underlying biological consequences as opposed to experimenter bias. Additionally, the ranges in lifespans are within tolerances reported by the Ceanorhabditis Intervention Testing Program for lifespans collected by a single laboratory. Furthermore, although several notable compounds derived from DrugAge also extended lifespan in our trials, the discrepancies between DrugAge and WormBot experiments for other compounds highlight the benefits of using digital microscopy. For WormBot based experiments, quality control was able to be performed so that negative results were verified whereas the majority of peer reviewed *C. elegans* lifespans do not have this opportunity as they were performed using historical techniques.

The complementary side to leveraging *C. elegans* as a model for intervention screenings is to use its tractable genetics to perform genetic interventions. In the case of our WormBot platform, we used a deep-learning pipeline to mine human RNA-seq databases for genes implicated in the progression of AD pathogenesis. Several genes involved in mitochondrial function were identified as the strongest possible putative targets. Through RNAi feeding we did single-gene knockdown in conjunction with a *C. elegans* model of Aβ-proteotoxicity to validate the PAUSE deep learning approach.

From the initial screen, 13 genes that were worm orthologs of the human genes of interest were identified to delay the onset of Aβ-induced paralysis. Following this approach we broadened our dataset outward from our initial putative targets to 36 additional genetic targets and found several that also delayed paralysis. Interestingly, in the second wave of screening all genes that, when knocked down, delayed paralysis are also involved in mitochondrial function.

These two trials represent the potential impact of adoption of the WormBot technology. The first, a natural product screen enriched for compounds previously reported in the DrugAge database, shows the power of the system to remove barriers to data interpretation that historical methods are unable to tackle. Here we show that the WormBot provides a suite of much needed tools to improve rigor and reproducibility in the context of *C. elegans* aging experiments.

Chapter Three:

Worm-YOLO: A novel machine learning pipeline for high-throughput analysis of *C. elegans* lifespans.

Abstract:

C. elegans has been a workhorse within the field of aging biology due to its short lifespan, easy culturing, and robust genetic tools. However, while some automated tools have been developed, the necessity of manual or semi-manual assays has limited the scope of what a single researcher can accomplish. Using our previously published robotic image capture platform (The WormBot) for data collection, we further developed a novel machine learning-based approach to analyze and extract lifespan data and health span measures from *C. elegans* within the context of drug discovery. Here we present the pipeline for analyzing results from several compound screens and show how Worm-YOLO accurately recapitulates manually analyzed data while also allowing for new and more comprehensive mining of behavioral and morphological metrics of health. The increased efficiency from automated analysis coupled with the ability to mine high-dimensional data allows for a holistic evaluation of each compound tested, linking metrics of health with end-of-life outcomes within the context of total lifespan. Furthermore, we also show how our pipeline allows for analysis of several of the *C. elegans* genetic models of aging associated diseases such as Alzheimer's Disease.

Introduction

C. elegans rationale and aging background

Ceanorhabditis elegans has long been a workhorse of the aging field. It's short lifespan, relatively easy culturing, and genetic tractability have provided a model system that can be flexibly adapted to many of the diverse questions asked in the study of aging biology. With the advent of modern computer vision and the increased feasibility of adapting machine learning to laboratory use, *C. elegans* is even better suited to tackle the complex questions that underlie aging biology. Here we present a novel machine learning approach to analyzing image data from our previously published robotic system: the WormBot. This custom-developed pipeline lays the groundwork for higher throughput intervention screening, in-depth profiling of aging biomarkers, and increased rigor and reproducibility.

The impact of aging on human health and disease progression is well-documented yet only a fraction of potential chemical effectors of age have been screened in any model system. One obvious limitation is that in the absence of effective early life biomarkers, aging studies often require aged animals. This has led to a trade-off in research design where the more-translational species (mice, rats, dogs) require cumbersome and lengthy studies while short-lived species (yeast, flies, worms) offer higher throughput at the cost of evolutionary distance. While each model has its benefits, *C. elegans* has emerged as a strong candidate for high-throughput drug screening due to its short lifespan and easy maintenance. However, despite the development of automated tools to increase throughput, only 395 unique compounds have been reported on DrugAge for their effects on lifespan in *C. elegans* at time of writing.

Laboratory Automation

To-date, the bulk of historical *C. elegans* lifespan experiments across the field have been performed via an experimenter manually surveying an individual's response to physical stimulation via manipulation via wire or hair pick. However, the increased ease and efficacy of digital microscopy has allowed for the development of several platforms that aim to automate this process. Some examples of this include the Automated Lifespan Machine[28,78], The WormMotel[38], as well as several different microfluidic platforms such as the NemaLife chip[34,79,80]. To better improve throughput in this model system the Kaeberlein lab originally developed the WormBot, an automated image capture system. A single installation of this device monitors 144 individual *C. elegans* lifespan experiments and the system is easily scalable to multiple installations per laboratory.

Early experiments performed on the WormBot utilized a combination of manually annotated data and traditional computer vision to semi-automate experimental analysis. To fully automate this process and increase the depth of data that we could analyze, we developed Worm-YOLO, a neural network pipeline that automatically analyzes timelapse images of *C. elegans* experiments for longevity as well as several relevant health metrics. This pipeline produces results comparable to human annotation while being completely unsupervised. Furthermore, while developed in concert with the WormBot platform, our machine learning pipeline is flexible enough to be able to analyze data from other image capture platforms such as the Automated Lifespan Machine.

C. elegans trackers and image analysis

Many of the previous platforms for *C. elegans* automation were co-developed with computer vision analysis pipelines. For instance, the ALM platform uses traditional computer vision to detect foreground objects likely to be worms and then validates those detections through the extraction and analysis of morphological features. Afterwards, worms are identified and posture tracked until time-of-death is determined[28]. Following this, the ALM pipeline often necessitates the use of a manual interface to remove false-positives of non-worm objects or correct putative time-of-death calls. This system has been the most widely adopted of any single *C. elegans* automation technique with multiple labs using the ALM in addition to the three sites of the CITP. Similarly, both the Stress Chip and WorMotel also rely on traditional background/foreground detection pipelines.

Outside of these computational pipelines published as part of hardware packages, there have also been several standalone software suites designed to be used across imaging platforms. Several notable examples include Rex Kerr's worm tracker as well as the Tierpsy tracker system[81,82]. These tracking and analysis platforms excel at the extraction of morphological characteristics but are hampered by several limitations. First, they often require relatively high-resolution cameras that are often cost-prohibitive and challenging to operate. This precludes scaling such trackers for use in high throughput screening. Second, traditional computer vision models are sensitive to deviations in image quality or unexpected deviations from the imaging conditions around which they were designed (brightness, field of view, contrast, etc). Although they can generate exceptionally rigorous data within the laboratory settings in which they were designed, attempts to adopt such techniques by other labs often produces sub-optimal data due to changes in hardware, techniques, or environmental conditions.

The last decade of advancements in artificial intelligence has seen the rise of convolutional neural networks as the best-in-class technique for automated image analysis. Notably, neural networks are an efficient way to integrate and detect signals from highly complex data; in this case images collected of *C. elegans*. Broadly, neural networks are defined by three main characteristics: the interconnection pattern between layers of neurons, the learning process for updating weights, and the activation function that converts a neuron's weighted input to its output[83]. One particularly noteworthy model is known as You Only Look Once (YOLO)[84]. This model performs the dual functions of detection and classification simultaneously, and at best-in-class speeds relative to other convolutional networks. Given that high-throughput invertebrate research entails the detection of relatively small objects (worms) in a high-resolution image, this processing speed is essential. Our methodology involves creating a novel training pipeline for YOLO to create a Worm-YOLO algorithm for analyzing timelapse recordings of *C. elegans*. In doing so we create a flexible tool that is minimally constrained by the idiosyncrasies of individual imaging hardware set-ups. Furthermore, even though we provide the tools necessary to manually check and quality control (QC) Worm-YOLO's output, the data presented herein was all analyzed fully autonomously.

Methods

Animal handling

Worm culture was performed as previously described[68]. Briefly, populations are age synchronized via hydrochloride treatment and then grown to L4 larval stage on NGM growth plates seeded with OP50 *E. coli*. Age synchronized L4s are then liquid transferred into 12-well cell culture plates that also contain 3mL of NGM agar seeded with a 20X concentration of OP50. For experiments using chemical interventions, compounds or solvent controls are pipetted onto the surface of the agar 4 hours before the transfer of animals. Once prepared, plates are parafilm-sealed and placed on a WormBot. The experimenter then enters in the necessary metadata into the web-interface and collection of data is subsequently started. A high-resolution image is captured every 10-minutes from every well and once a day, each well is videoed for 2-5 minutes in order to create a daily-monitor movie. This creates both the timelapse series of images that are used to determine time-of-death as well as the daily movie in which animal behavior and health can be monitored under higher temporal-resolution.

Worm-YOLO training

The Worm-YOLO neural network is a novel implementation of the YOLOv3 architecture. This convolutional neural network detects and identifies objects at the same time and allows for fast analysis of images. To train Worm-YOLO to identify worms, a dataset of hand-curated ground truth *C. elegans* data was created. Using a custom MATLAB interface, we trained our initial dataset with 1068 images hand annotated for worm locations. The interface loads an image from a WormBot experiment from a directory after which the user identifies each animal by eye and draws a bounding box around it. After each image is finished, the program saves the curated csv file alongside the base image to a new directory and then loads a new image in for manual annotation.

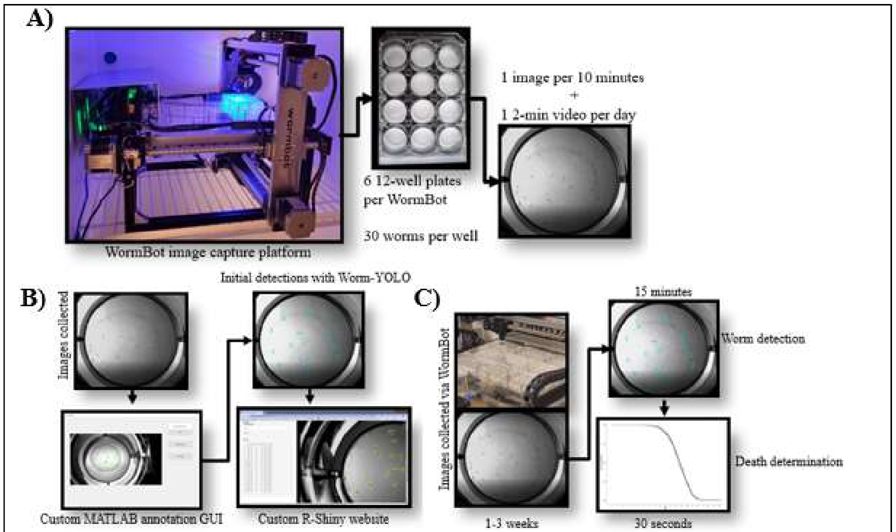

Figure 1. Worm-YOLO developmental pipeline. A) Schematic of the WormBot apparatus and experimental pipeline. B) Training methodology of Worm-YOLO. C) Time-of-death analysis utilizing Worm-YOLO detections coupled with tracking and death thresholding.

Following the creation of this dataset, the first implementation of YOLOv3 trained to detect worms was implemented. The dataset images were 1920 x 1080 pixels while the neural network requires images that are 608 x 608. However, because the worms only make up a small fraction of the total area of each image, we could not downsample the image size while maintaining accuracy. Instead, the original image was cut into overlapping 416 x 416 sub-images with full overlapping borders. Then each cut was upsampled to the input size of 608 x 608. This allows us to maintain the resolution of the worms needed for strong bounding box prediction and accuracy. While overlapping the sub-images increases the total computation time for each image, it ensures that worms on the border between cuts are properly accounted for. We then use non-maximal suppression to filter any duplicate boxes for the same worm. After 100 training epochs, we evaluated our first of Worm-YOLO to be ~90-95% accurate on the test set.

In order to increase the accuracy of Worm-YOLO and create a platform for future training, we developed a feed-forward pipeline. First, images are collected from the WormBot and analyzed for putative bounding boxes of each worm with Worm-YOLO. Second, selections of the individual images are uploaded to an online database alongside the putative calls. These images were selected at random but with a slight weight towards mid-life and late-life timepoints. We then developed a custom web application using R-Shiny that allows for remote users to correct for false positives, incorrect size and shape of bounding boxed, and unidentified but physically present worms. The preparation for this application entailed uploading randomly selected frames to one Dropbox folder while the matching detections were stored in a second folder in the format of a ".csv" file. As a user began the annotation process, an image would be displayed in their web-browser and the putative annotations from YOLO overlayed as bounding boxes. They then could add, delete or alter the existing bounding boxes and the resulting annotations would automatically be saved to a third Dropbox folder as a new image was loaded. Following this, the corrected datasets are used to retrain Worm-YOLO. This pipeline allows for both engagement with community scientists and remote researchers (particularly during periods of enforced remote work due to COVID-19) while also increasing the accuracy and throughput of training.

Time-of-death analysis

Following the identification, we employed a modified version of SORT (Simple Online Realtime Tracking) to track individual animals between images. SORT uses a Kalman filter paired with the Hungarian algorithm to identify worms and then track their movement between timepoints. As the lifespan videos that are captured by the WormBot system have an imaging interval of 10-minutes, early-life animals transit too large of distances in that interval to effectively track. However, as animals enter mid to late life, their movement between frames slows to the point at which they individuals can be identified and tracked via SORT. Our methodology initiates SORT at the end of the experiment when animals are presumed dead and motionless. It then works backward and tracks individuals as they transition from dead to slow-moving and until spontaneous movement exceeds the tracking capabilities of SORT. However, while SORT produces best-in-class tracks for animals, small changes in lighting or movement of the camera can cause a single worms track to be broken into multiple segments with unique IDs. Following the SORT process, we employ a series of algorithms that examine each tracks' starting and ending bounding box and determine if it's the same animal separated into two unique IDs. If such a situation is identified, the tracks are merged into the same unique ID.

After each animal is tracked from the onset of slow movement until the end of image capture, Worm-YOLO uses the location and shape of the bounding box over time to approximate animal activity. Starting from the earliest observation of each animal, the algorithm compares the bounding box overlap between each observed frame and the next. We then used a rigorous testing methodology to determine a threshold for how long a bounding box must exhibit no change in movement or shape before being the animal it defines is defined as dead. These thresholds were defined through use of manually annotated death times as compared with the output of Worm-YOLO.

While the train/test datasets for Worm-YOLO indicate a sufficient level of accuracy to replace manual annotation, altered imaging conditions or quality control can require the ability to view and even correct the output. In order to ensure that Worm-YOLO retained flexibility across different image systems and user groups, we developed a custom MATLAB interface for viewing and changing the death-times determined via the pipeline. This interface allows a user to load a timelapse video with the locations and times of each worm death overlayed. Afterwards, the user can manually add, delete, or change death times for any erroneous calls.

In addition to the lifespan determination derived from the 10-minute timelapse images, it is possible to also use the Worm-YOLO pipeline to detect changes in behavior as a function of age. Using the methodology described above (Worm-YOLO followed by SORT) on the 3-minute daily monitor movies, we are able to track worms in real-time once a day. Using the Trajr package for Python, metrics of motion and behavior are then extracted. These include, but are not limited to, average and maximal speed, average and maximal acceleration, sinuosity of track and total distance traveled.

Results

Accuracy Relative to Manual Calls

The accuracy of Worm-YOLO can be assessed by first comparing the object detection of the neural network against a testing dataset of analyzed ground truth images. We analyzed 1200+ images and found that individual bounding box detection was 95-98% depending on imaging conditions. We acknowledge that no system is likely to be 100% accurate but given the historical methodology of manually testing worm lifespans and the accrued human error, the accuracy of Worm-YOLO is more than sufficient for the experimental needs. The second measure of accuracy is comparing the measured lifespan of both individual animals and populations of animals using YOLO as compared to the human eye. We generated a testing dataset of 19 experimental replicates started as one biological cohort. Each replicate contained ~20 animals and were started from the same age-synchronized population. Using the Worm-YOLO methodology described above and the manual annotation process described in the original WormBot publication, we compared the median lifespan between each method of lifespan determination. We found that that the median lifespan difference for all experiments pooled was <12 hours (Figure 2A). Relative to the temporal resolution afforded by either manual checks (daily) or previously developed lifespan determination software (hourly), this shows the effectiveness of Worm-YOLO paired with the WormBot image capture system. Furthermore, despite the challenges of identifying and tracking individual animals over the dynamic field of view containing multiple worms, the difference of N-value for the pooled Worm-YOLO calls and that of manual scoring is only 2%.

To ensure that Worm-YOLO performed accurately across multiple strains and backgrounds, we tested its efficacy at detecting delta-shifts in median lifespan across mutants that were previously observed to have a different lifespan across treatment groups. First we compared N2 to a relatively short-lived and long-lived mutants, rde3 and hsp17 respectively. We found that while the median time-of-death calls were right-shifted by ~1 day on average, the relative difference in lifespan between the N2 controls and either of the mutants were preserved (Figure 2B). Similarly, another commonly used *C. elegans* model is GMC101, which expresses A$\beta_{1\text{-}42}$ and exhibits an age-dependent paralysis. We similarly tested control GMC101 relative to animals exposed to 25mM Metformin which has been reported to delay the onset of paralysis by 10-20%. As paralysis is phenotypically different than death, we utilized a different set of slow-moving and bounding box overlap thresholds. We then compared manual annotations to Worm-YOLO and found that while the overall median time-to-paralysis events were shifted between each method, the overall delta between control and treatment was preserved (Figure 2C). As this methodology is improved we expect the Worm-YOLO results to more precisely mimic those of human annotation. However, this indicates that Worm-YOLO can already be utilized for screening large datasets of interventions.

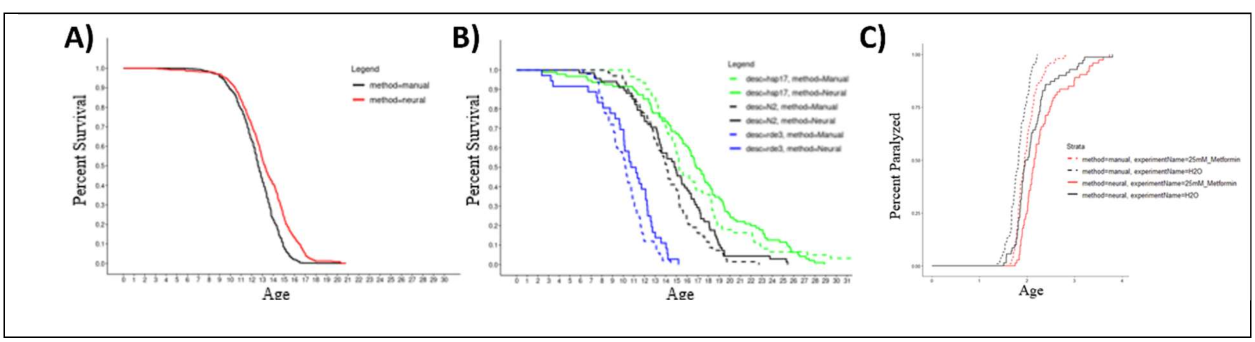

Figure 2. Worm-YOLO replicates manual lifespan and paralysis calls. A) Worm-YOLO vs manual annotation of WormBot images from N2 experiments. B) Worm-YOLO vs manual annotation of WormBot images from N2, hsp17, and rde3 experiments. C) Worm-YOLO vs manual annotation of WormBot images from GMC101 paralysis experiments.

As compound interventions can change imaging conditions on a given plate, we tested if Worm-YOLO could accurately detect the same shifts in lifespan that a human experimenter would while analyzing a timelapse recording. We tested 100uM concentrations of each drug relative to a 1% DMSO control. After manually scoring the data using the WormBot's interface, we then used Worm-YOLO and found that similar delta-shifts in lifespan were observed between all 3 known modulators of lifespan[35,70] irrespective of which methodology was used (Figure 3).

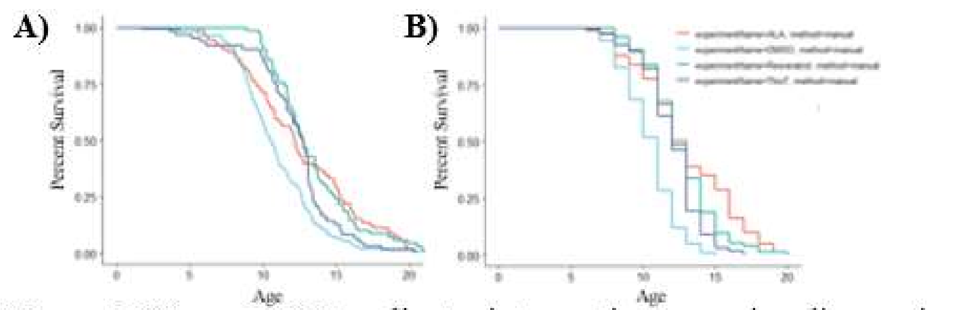

Figure 3: Worm-YOLO replicates intervention screening discoveries.

Figure 3. Worm-YOLO accurately measures lifespans of compound exposed animals. A) Worm-YOLO lifespan curves for animals given Alpha-lipoic acid, resveratrol, and thioflavin-T relative to animals given a solvent control of 1% DMSO. Manual lifespan curves for animals given Alpha-lipoic acid, resveratrol, and thioflavin-T relative to animals given a solvent control of 1% DMSO.

While there have been several successful worm-tracking software packages published[81,82], a common constraint of these systems is the necessary pairing of a specific hardware apparatus to each software package. As such, without investment in a very similar imaging system to what the software package was designed to analyze images from, results are often inconsistent. One benefit to Worm-YOLO is that it is broadly applicable across a range of imaging systems. Leveraging the robust capabilities of the YOLO architecture, we've found that worms can be easily identified from images collected from off-the-shelf electronic microscopes as well as other lifespan machines such as the Automated Lifespan Machine. Image data from the ALM was obtained with permission from the Phillips lab alongside the lifespan calls as obtained through the ALM methodology. We found that while the thresholding values for how long a worm must remain motionless had to be adjusted for the slower frame rate, the rest of the Worm-YOLO pipeline was easily applied and produced statistically similar data.

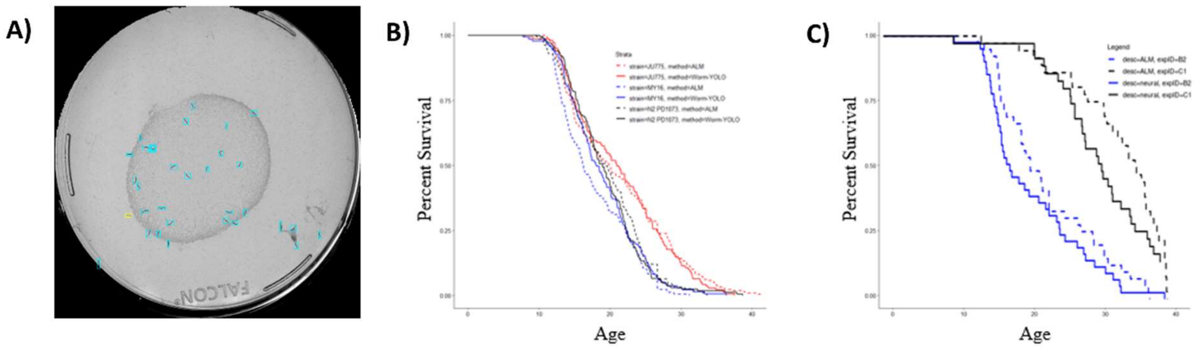

Figure 4. Worm-YOLO as applied to the Automated Lifespan Machine platform. A) Detections of animals using Worm-YOLO on ALM images. B) ALM pipeline data vs Worm-YOLO data between 3 strains. C) Compound intervention data as analyzed by ALM and Worm-YOLO.

Looking to test for potential novel targets of AD, we set out to test a unbiased portion of a commonly available library of FDA-approved drugs for their effects upon the GMC101 phenotype. For each compound we performed 2 matched replicates of a solvent-only negative control, a positive control of 25mM metformin, the compound of interest at 10uM and the compound at 10uM plus 25mM metformin. Normalizing to the negative control we found several drugs that delayed the onset of the GMC101 phenotype by more than metformin. These results are currently blinded due to ongoing IP protection and licensing.

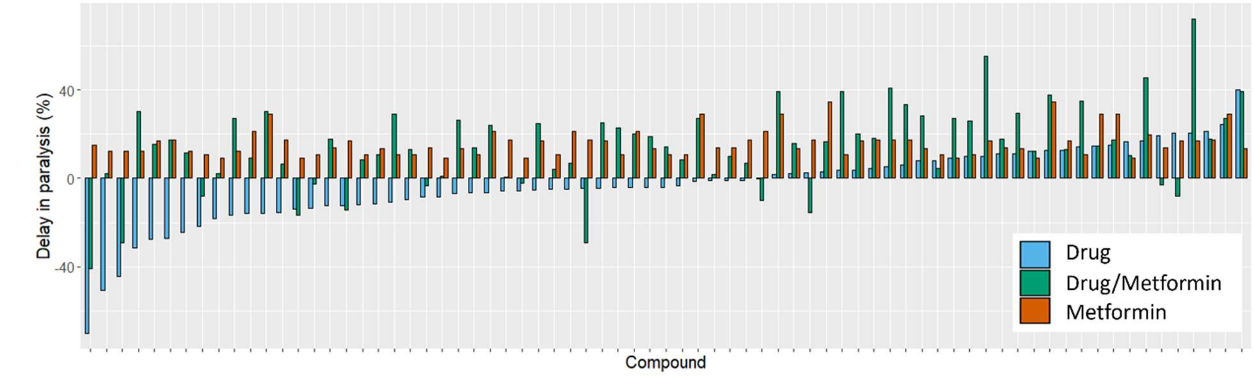

Figure 5. Screen of FDA approved drugs for their efficacy on GMC101.

To better understand the interplay of health and aging we utilized the 1-2 minute recordings that are taken each day that animals are on the WormBot. After using our trained model of YOLOv3 to detect animals, we then used SORT to track individuals. As these movies are real-time, tracking animals continuously over the recorded video is much less error prone. 6 independent replicates of 25 animals were analyzed every other day from days 4-12 of life for 8 different features (Figure 6A). By performing a PCA upon the representative data we found that across the observed motility characteristics, as animals aged their range of behavioral phenotypes canalized.

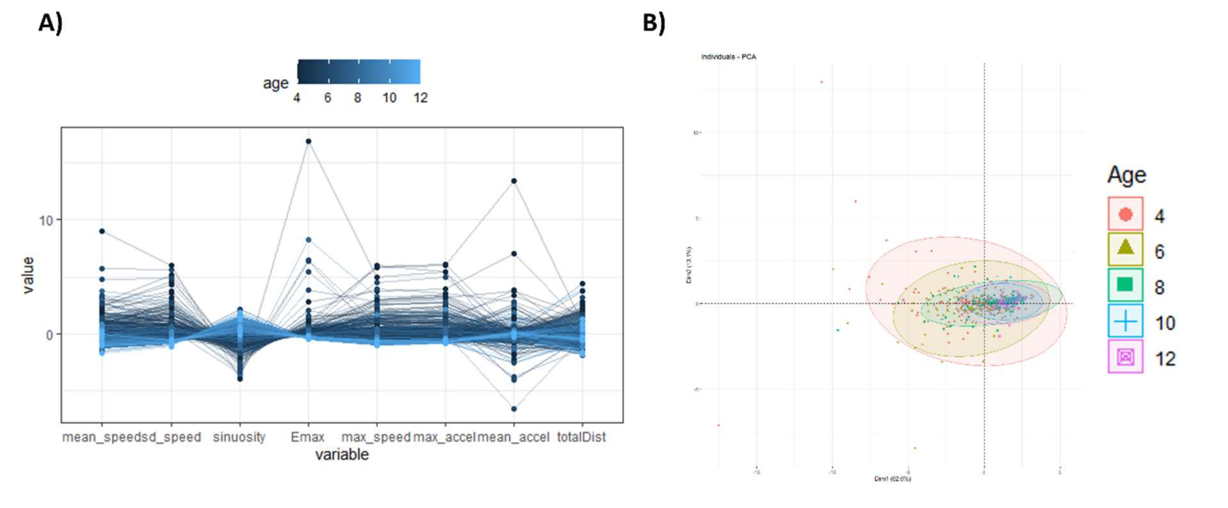

Figure 6. Healthspan of analysis of N2 worms. A) Eight metrics of motility and behavior are analyzed every other day of adulthood. Values are normalized and ordered by the chronological age of the animals. B) PCA plot grouping observed behavior by age. While young animals exhibit a broader range of behaviors, old animals were found to have very stereotyped and compressed range of phenotypes.

Discussion

Here we describe the Worm-YOLO analysis pipeline for fully automated data analysis from images and videos captured from the WormBot platform as well as the ALM. Through the use of this software suite, worms in a timelapse stack of images are detected, tracked, and assayed for several phenotypic behaviors. Predominantly, this software is designed to measure the individual lifespans of worms within populations under normal lab culturing conditions. As we show, Worm-YOLO accurately recapitulates the lifespan measurements of human annotators with a small margin of error. Extensions from Worm-YOLO also hold the promise of more accurately quantifying other physiological and behavioral biomarkers of aging with the goal of better describing as many life-history traits as possible.

Our main benchmark for the effectiveness of Worm-YOLO was a train/test dataset consisting of 605 N2 animals across 32 experiments. We show that in terms of population median's Worm-YOLO time-of-death calls are consistently within 12 hours of human annotations. Furthermore, the total n-value difference between manual and neural calls was only 12 animals in total or less than 2% of the total population. One notable caveat to this method of testing Worm-YOLO's accuracy is that it presupposes that the human annotations are the ground-truth. Although the dataset in question was generated through highly skilled researchers carefully annotating the images, human error is always a possibility. Although our original WormBot paper[68] showed that on-bot lifespans replicated those that were conducted using traditional manual methods, there is one major difference between the two paradigms. Under fully manual conditions, a worms alive/dead status is most commonly determined by whether or not it reacts to a physical stimuli (generally a touch with a wire or hair pick). In contrast, automated image capture technologies do not have the capability to physically interact with individual animals. Although a difference is not suggested by the data, it is important to recognize two fundamentally different approaches to lifespan determination are being compared[85]. One caveat is that *C. elegans* do respond to photo-stimuli and most image systems, the WormBot included, do use interstitial light during data collection. While likely not as direct a stimulus as physical interaction, this intermittent light does likely trigger some movement in the exposed animals. One further avenue of research that can disentangle this distinction in time-of-death analysis would be to incorporate vital dyes or fluorescent labels onto a WormBot outfitted with a fluorescent camera. This methodology could provide a metric for cessation of metabolic activity that could then be linked to the phenotypic data used in determining time-of-death.

Further extensions of Worm-YOLO include both genetic and compound intervention screening. Using several genetic mutant lines linked to short, normal, and long lifespan, we validate that Worm-YOLO is capable of recapitulating manually annotated data that detects a difference in median lifespan across a range of genetic mutants and even the GMC101 model of Alzheimer's Disease. Furthermore, it also detected similar changes in lifespan between drug treatments previously shown to have an effect on lifespan.

In order to rapidly screen for the effects of compounds upon aging pathways, we elected to use a *C. elegans* model of amyloid-beta proteotoxicity that has a progressive and age-associated paralysis phenotype that manifests after several days. In contrast to wildtype *C. elegans* that take several weeks to screen for age associated phenotypes, this model (GMC101) can take less than a week. In order for Worm-YOLO to accurately capture the onset of paralysis as opposed to senescence and death, we adjusted the parameters for how animals were tracked to create a model for the new phenotype. As GMC101 exhibit a cessation of gross locomotion while maintaining small movements in the head and tail we lowered both the bounding box area of overlap as well as the number of frames slow moving before an animal would change from a healthy state to the one of interest. We found that this methodology accurately measured the onset of Aβ induced paralysis in GMC101 animals. Following this, we identified that metformin, a known modulator of aging, consistently delayed the onset of paralysis by ~15-20%. We then conducted a screen through the ApexBio L2120 FDA approved drug library looking compounds that delayed the onset of paralysis either alone or that would have a synergistic effect with Metformin. In our initial screen we tested 75 drugs and found 7 that delayed the onset of paralysis as much as metformin alone and another 8 that had a synergistic effect in combination with metformin.

An extension of this work is the quantification of biomarkers of aging in *C. elegans*. Several biomarkers of ageing have previously been shown to be predictors of biological age as well as time-until-death[86]. These include such metrics as body morphology, growth rate, fecundity, resistance to stressors and pathogens, animal behavior and interactions with environmental cues. One novel aspect of using the Worm-YOLO pipeline and several of its extensions is simultaneous recording of both normative lifespans as well as several of these aforementioned metrics when combined with the WormBot platform. We found that animal behavior canalized with age and that a population of young animals could be separated from a population of old animals based on extracted behavioral traits. Further extensions of this include extracting metrics of body morphology and more advanced behavioral modeling across more drug and genetic interventions.

In conclusion, we present here the first iteration of Worm-YOLO, a neural network-based approach designed to analyze images and videos recorded of the WormBot image capture platform but capable of being broadly applicable to other experimental platforms. As addressed previously, the field of aging research has been hampered through low-throughput assays and results that have been unable to be reproduced. The goal of the WormBot and Worm-YOLO platform is to enable consistent and accurate analysis of *C. elegans* data while leveraging the strengths of the model system to allow easy scaling and higher-throughput. It is our hope that these technologies help catalyze further improvements to rigor and reproducibility while driving the field of geroscience into a new era of experimental capacity.

Conclusion

This thesis presents several novel methods and technologies that will advance the use of *C. elegans* within the context of studying the biology of aging. Our goal has been to create a platform and pipeline that leverages the best in automated technology, computer vision, and machine learning that reduces the potential for human error while also enhancing the throughput of the field. We first present the development of the WormBot platform, then novel uses for the platform for *C. elegans* studies, and lastly the development and use of the Worm-YOLO analysis pipeline.

While the material presented herein shows the potential of the WormBot and Worm-YOLO systems, there are several notable drawbacks to the implementation to this pipeline. These drawbacks consist of both technical limitations derived from the design of the WormBot as well as biological constraints imposed by the use of *C. elegans* as a model organism.

Firstly, the use of *C. elegans* as a model for rapid screening has some inherent flaws. The lack of a complex central nervous system or an adaptive immune system create several large and notable translational gaps that separate *C. elegans* from humans as an aging model. Further complicating the use of *C. elegans* is that it's small size prevent the easy collection of longitudinal biological samples that are often comprise vertebrate biomarkers of aging (blood draws, tissue biopsies etc.). Although the extraction of relevant biomarkers from digital recordings of *C. elegans* show promise to alleviate that gap, this will likely remain a persistent and confounding barrier to a more perfect translational system. Additionally, while our work has focused on the use of *C. elegans* as a tool for high-throughput drug discovery and screening, the lack of a defined vector for drug delivery does provide a confounding variable that may prevent the identification of what otherwise would be positive hits. As most experimental methodologies either mix drugs into the agar or add the drug topically onto the bacterial lawn, there question of precisely what concentration of bio-available compound the worms are exposed to remains in question. Furthermore, while manual lifespans afford the ability to transfer worms between plates over the course of the experiment, the WormBot relies on populations persisting on the same plates throughout their entire lives. This will likely enrich for drugs that are resistant to degradation during the duration of the experiment or drugs that may degrade but that have a noticeable effect after intermittent exposure. Additionally, many experiments in *C. elegans* are performed in the presence of live bacteria which may result in some potential hits being mediated through a given compounds interaction with the bacteria as opposed the worm. Lastly, compared to vertebrate models in which drug delivery is often discretely either injection or ingestion, drug delivery in *C. elegans* is experimentally less precise. With the drugs either in the agar or topically on the plate, the drugs may be ingested by the worm or more likely, they may uptake the drug through their cuticle. This will enrich for potential hits that are able to dissolve through the cuticle and potentially limit the effectiveness of screening as many potentially translatable compounds will remain biologically unavailable.

Second, while the WormBot system provides many advancements towards automating large scale *C. elegans* lifespan screening, there are several outstanding technical hardware and software problems left to be overcome. One major drawback is the aforementioned inability to transfer animals between plates mid experiment. While this may be technically feasible, we found that changing plates mid experiment was detrimental to the effectiveness of the automated image analysis pipeline. The WormBot also requires very precise control of environmental variables such as in-plate temperature and humidity. Care must be taken during plate preparation to ensure that the agar's water content is precisely calibrated as too little will cause the plates to desiccate and animals to escape into cracks. Conversely, too much water within the agar will increase the humidity and cause fogging on the lid that prevents worms from being seen. Either option can result in a failure to collect meaningful data.

Another technical hurdle is the failure rate of the WormBot hardware itself. While drastic improvements have been made since the earliest models, hardware failure still does occur and can cause loss of experimental data. The major sources of this are misalignment of the camera between subsequent frames, hard-drive malfunction, and malfunctions in the lighting source. Misalignment of the camera between frames will result in even stationary worms appearing to have moved and causes much of the downstream image processing to fail. While this can be remedied post hoc through manual annotation of the timelapse series, this is unproductive at scale. While we have re-engineered many of the WormBot's moving apparatus to reduce the likelihood that this will occur, the best prevention is daily monitoring of images collected. When this does occur, simply rechecking the wiring and recalibrating the plate locations often solves the problem. The hard-drive and it's limited space present another recurring problem for the WormBot's effective operation. A fully used WormBot fills up approximately 1% of it's total hard-drive space each day of operation and therefore will need to be backed up at least every 3 months. Although normative operations and rigorous standard operating procedures should prevent the bulk of the instances when the hard-drive runs out of space, trial and error has proved that it remains a large source of failure. Future improvements such as networked storage or a cloud-based storage system could quickly solve this outstanding issue and should be addressed in the next version of the WormBot. Lastly, lighting variability can arise due the adjustable reflector becoming mis-aligned. Poor contrast images and variation in lighting from what the Worm-YOLO neural network was trained against can result in low detection rates and subsequently cause a breakage of downstream parts of the pipeline. However, these mechanical issues are eminently solvable through routine maintenance and intermittent monitoring of the images from WormBots currently under operation. When compared with anecdotal accounts of other technologies developed to increase throughput and experimental rigor of *C. elegans* experiments, the WormBot is easier to run, more consistent in data capture, and adaptable to the challenges of high-throughput drug screening.

Moving forward, further improvements to the WormBots image capture capabilities will allow for increased potential for mining aging associated metrics of health. For instance an additional camera with higher resolution could be included in the design and allow for quantification of anatomical traits that are, in the current model, unobservable. This could include measurements of pharyngeal pumping[20], egg-laying behavior, as well as tissue and organ homeostasis with age[21]. An additional improvement currently in development is the integration of a fluorescent camera which would allow for longitudinal tracking of fluorescently tagged proteins. This would build on existing work in the field that has focused on quantifying how gene expression is linked to lifespan[87] but would have the added benefit of the WormBots scale and longitudinal data collection. In terms of improvements to the Worm-YOLO software, many additional features currently in development hold the promise of wider applicability and increased depth of data mining from static images. While preliminary work has shown the capability of the pipeline to extract health and behavior, this can be further tuned, expanded on, and applied across multiple strains and conditions. While other work has shown the capacity to use predictive biomarkers of age in *C. elegans*[86,88,89], the WormBot and Worm-YOLO pipeline offers drastically scalability of both training and implementation.

In summary, this thesis describes, validates, and provides multiple use-cases for our WormBot platform and Worm-YOLO analysis pipeline. While the last 50 years of research has leveraged *C. elegans* experimental utility to identify genetic pathways that govern aging and describe several compounds that effectively target said pathways, the throughput per researcher has remained relatively stagnant during that period. By building on several recent advances in automation, we drastically increase the efficacy of a single researcher while drastically improving the ability to rigorously retest and re-analyze a single experiment. It is our hope that as the WormBot and Worm-YOLO pipeline achieve broader adoption, large unbiased drug discovery screening will be more achievable and usher in the next age of *C. elegans* discoveries.

Acknowledgements

Jason Pitt, Nolan Strait contributed to the development of the WormBot hardware and operating systems. Elena Vayndorf, Joshua Russel, Ting I Lee, Judy Wu, Raja Estes and Cheyne Littlesun contributed to data collection for experiments performed on the WormBot. Pascal Sturmfels, Paolo Bifulcu, Connor Kaeberlein, Aman Thukrul, Shreya Ramakrishnan and Renusreee Chitella contributed to the software design and training of the Worm-YOLO pipeline. Raja Estes, Faith Berry, Lincoln Pothan, Young Woo Kim, Arash Nikjoo, Annaiz Grimm, Riley Whalen, Manuel Villa, Justin Dillard-Telm, Nehal Hegde, Pulkit Malhotra, Mahimna Pandya, Sophie Liu, Ayush Sharma, Emily Chang, Siobhan Duffy, Sergio Da Silva, Ray Hendricks, Ryan Kelly, Kenny Nguyen, Ashley Liang, Ayush Sharma, Hannah Chen, Kanhai Pandya, Keong Mu Jason Lim, Kevin Yang, Shivani Grandhi, Thomas Wenk, Veronica Hsu, Elysia Quah, Sindhu Narayanan, Jessica Shattuck, Ying Chen, Jason Livingston, Jared Almazan, Bao Nguyen contributed as undergraduate researchers to the data collection for the associated projects herein. Matt Kaeberlein, Bonny Brewer, Alexander Mendenhall, Shane Rea, Alan Herr, and Conrad Liles served as thesis advisors to this document.

Bibliography

1. An Aging Nation: The Older Population in the United States. https://www.census.gov/library/publications/2014/demo/p25-1140.html. Accessed June 26, 2022.

2. Fuentealba M, Dönertaş HM, Williams R, Labbadia J, Thornton JM, Partridge L. Using the drug-protein interactome to identify anti-ageing compounds for humans. Iakoucheva LM, ed. *PLOS Comput Biol*. 2019;15(1):e1006639. doi:10.1371/journal.pcbi.1006639

3. Kaeberlein M. How healthy is the healthspan concept? *GeroScience*. 2018;40(4):361-364. doi:10.1007/s11357-018-0036-9

4. Barzilai N, Crandall JP, Kritchevsky SB, Espeland MA. Metformin as a Tool to Target Aging. *Cell Metab*. 2016;23(6):1060-1065. doi:10.1016/j.cmet.2016.05.011

5. Longo VD, Antebi A, Bartke A, et al. Interventions to Slow Aging in Humans: Are We Ready? *Aging Cell*. 2015;14(4):497-510. doi:10.1111/acel.12338

6. Kapahi P, Kaeberlein M, Hansen M. Dietary restriction and lifespan: Lessons from invertebrate models. *Ageing Res Rev*. 2016. doi:10.1016/j.arr.2016.12.005

7. Austad SN. Comparative biology of aging. In: *Journals of Gerontology - Series A Biological Sciences and Medical Sciences*. ; 2009. doi:10.1093/gerona/gln060

8. Colman RJ, Anderson RM, Johnson SC, et al. Caloric Restriction Delays Disease Onset and Mortality in Rhesus Monkeys. *Science (80-)*. 2009;325(5937):201-204. doi:10.1126/science.1173635

9. Mattison JA, Colman RJ, Beasley TM, et al. Caloric restriction improves health and survival of rhesus monkeys. *Nat Commun*. 2017;8:14063. doi:10.1038/ncomms14063

10. Bitto A, Wang AM, Bennett CF, Kaeberlein M. Biochemical Genetic Pathways that Modulate Aging in Multiple Species. doi:10.1101/cshperspect.a025114

11. Ló Pez-Otín C, Blasco MA, Partridge L, Serrano M, Kroemer G. The Hallmarks of Aging. *Cell*. 2013;153:1194-1217. doi:10.1016/j.cell.2013.05.039

12. Fontana L, Partridge L, Longo VD. Extending Healthy Life Span—From Yeast to Humans. http://science.sciencemag.org/content/sci/328/5976/321.full.pdf. Accessed May 25, 2017.

13. Félix MA, Braendle C. The natural history of Caenorhabditis elegans. *Curr Biol*. 2010;20(22):R965-R969. doi:10.1016/J.CUB.2010.09.050

14. Brenner S. The genetics of Caenorhabditis elegans. *Genetics*. 1974;77(1):71-94. doi:10.1093/GENETICS/77.1.71

15. Klass MR. Aging in the nematode Caenorhabditis elegans: Major biological and environmental factors influencing life span. *Mech Ageing Dev*. 1977;6(C):413-429. doi:10.1016/0047-6374(77)90043-4

16. Klass MR. A method for the isolation of longevity mutants in the nematode Caenorhabditis elegans and initial results. *Mech Ageing Dev*. 1983;22(3-4):279-286. doi:10.1016/0047-6374(83)90082-9

17. Johnson TE, Wood WB. Genetic analysis of life-span in Caenorhabditis elegans.

Proc Natl Acad Sci U S A. 1982;79(21):6603-6607. doi:10.1073/PNAS.79.21.6603

18. Wolkow CA, Kimura KD, Lee MS, Ruvkun G. Regulation of C. elegans life-span by insulinlike signaling in the nervous system. *Science*. 2000;290(5489):147-150. doi:10.1126/SCIENCE.290.5489.147

19. Tissenbaum HA, Ruvkun G. An insulin-like signaling pathway affects both longevity and reproduction in Caenorhabditis elegans. *Genetics*. 1998;148(2):703-717. doi:10.1093/genetics/148.2.703

20. Russell JC, Burnaevskiy N, Ma B, et al. Electrophysiological Measures of Aging Pharynx Function in C. elegans Reveal Enhanced Organ Functionality in Older, Long-lived Mutants. *J Gerontol A Biol Sci Med Sci*. 2019;74(8):1173-1179. doi:10.1093/GERONA/GLX230

21. Son HG, Altintas O, Kim EJE, Kwon S, Lee SJ V. Age-dependent changes and biomarkers of aging in Caenorhabditis elegans. *Aging Cell*. 2019;18(2):e12853. doi:10.1111/ACEL.12853

22. Cypser JR, Wu D, Park S-K, et al. Predicting longevity in C. elegans: Fertility, mobility and gene expression. *Mech Ageing Dev*. 2013;134(7):291-297. doi:10.1016/j.mad.2013.02.003

23. Mendenhall AR, Tedesco PM, Taylor LD, Lowe A, Cypser JR, Johnson TE. Expression of a single-copy hsp-16.2 reporter predicts life span. *Journals Gerontol - Ser A Biol Sci Med Sci*. 2012. doi:10.1093/gerona/glr225

24. Mendenhall AR, Tedesco PM, Sands B, Johnson TE, Brent R. Single Cell

Quantification of Reporter Gene Expression in Live Adult Caenorhabditis elegans Reveals Reproducible Cell-Specific Expression Patterns and Underlying Biological Variation. *PLoS One*. 2015;10(5). doi:10.1371/journal.pone.0124289

25. Zhang S, Li F, Zhou T, Wang G, Li Z. Caenorhabditis elegans as a Useful Model for Studying Aging Mutations. *Front Endocrinol (Lausanne)*. 2020;11:731. doi:10.3389/FENDO.2020.554994/BIBTEX

26. Pittman WE, Sinha DB, Zhang WB, Kinser HE, Pincus Z. A simple culture system for long-term imaging of individual C. elegans. *Lab Chip*. 2017;17(22):3909-3920. doi:10.1039/c7lc00916j

27. EPG Chip Draft.

28. Stroustrup N, Ulmschneider BE, Nash ZM, López-Moyado IF, Apfeld J, Fontana W. The Caenorhabditis elegans Lifespan Machine. *Nat Methods*. 2013;10(7):665-670. doi:10.1038/nmeth.2475

29. Lucanic M, Plummer WT, Chen E, et al. Impact of genetic background and experimental reproducibility on identifying chemical compounds with robust longevity effects. *Nat Commun*. 2017;8:14256. doi:10.1038/ncomms14256

30. Churgin MA, Jung S-K, Yu C-C, Chen X, Raizen DM, Fang-Yen C. Longitudinal imaging of *Caenorhabditis elegans* in a microfabricated device reveals variation in behavioral decline during aging. *Elife*. 2017;6:e26652. doi:10.7554/eLife.26652

31. Gruber J, Ng LF, Poovathingal SK, Halliwell B. Deceptively simple but simply deceptive – Caenorhabditis elegans lifespan studies: Considerations for aging

and antioxidant effects. *FEBS Lett*. 2009;583(21):3377-3387. doi:10.1016/J.FEBSLET.2009.09.051

32. Baker M, Penny D. 1,500 scientists lift the lid on reproducibility. *Nature*. 2016;533(7604):452-454. doi:10.1038/533452A

33. Bansal A, Zhu LJ, Yen K, Tissenbaum HA. Uncoupling lifespan and healthspan in Caenorhabditis elegans longevity mutants. *Proc Natl Acad Sci U S A*. 2015;112(3):E277-E286. doi:10.1073/PNAS.1412192112

34. Banse SA, Blue BW, Robinson KJ, Jarrett CM, Phillips PC. The Stress-Chip: A microfluidic platform for stress analysis in Caenorhabditis elegans. *PLoS One*. 2019;14(5). doi:10.1371/JOURNAL.PONE.0216283

35. Banse SA, Lucanic M, Sedore CA, et al. Automated lifespan determination across Caenorhabditis strains and species reveals assay-specific effects of chemical interventions. *GeroScience*. 2019;41(6):945-960. doi:10.1007/S11357-019-00108-9

36. Budde MW, Roth MB. Hydrogen sulfide increases hypoxia-inducible factor-1 activity independently of von Hippel-Lindau tumor suppressor-1 in C. elegans. *Mol Biol Cell*. 2010;21(1):212-217. doi:10.1091/MBC.E09-03-0199

37. Burnaevskiy N, Chen S, Mailig M, et al. Reactivation of RNA metabolism underlies somatic restoration after adult reproductive diapause in C. elegans. *Elife*. 2018;7. doi:10.7554/ELIFE.36194

38. Churgin MA, Jung SK, Yu CC, Chen X, Raizen DM, Fang-Yen C. Longitudinal

imaging of caenorhabditis elegans in a microfabricated device reveals variation in behavioral decline during aging. *Elife*. 2017;6. doi:10.7554/eLife.26652

39. D'Ausilio A. Arduino: a low-cost multipurpose lab equipment. *Behav Res Methods*. 2012;44(2):305-313. doi:10.3758/S13428-011-0163-Z

40. Evangelidis GD, Psarakis EZ. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Trans Pattern Anal Mach Intell*. 2008;30(10):1858-1865. doi:10.1109/TPAMI.2008.113

41. Ewald CY, Castillo-Quan JI, Blackwell TK. Untangling Longevity, Dauer, and Healthspan in Caenorhabditis elegans Insulin/IGF-1-Signalling. *Gerontology*. 2018;64(1):96-104. doi:10.1159/000480504

42. Finch CE, Ruvkun G. The genetics of aging. *Annu Rev Genomics Hum Genet*. 2001;2:435-462. doi:10.1146/ANNUREV.GENOM.2.1.435

43. Cohen AA, Luyten W, Gogol M, et al. Health and Aging: Unifying Concepts, Scores, Biomarkers and Pathways. *Aging Dis*. 2019;10(4):883-900. doi:10.14336/AD.2018.1030

44. Mehta R, Steinkraus KA, Sutphin GL, et al. Proteasomal regulation of the hypoxic response modulates aging in C. elegans. *Science*. 2009;324(5931):1196-1198. doi:10.1126/SCIENCE.1173507

45. Leiser SF, Fletcher M, Begun A, Kaeberlein M. Life-span extension from hypoxia in Caenorhabditis elegans requires both HIF-1 and DAF-16 and is antagonized by SKN-1. *J Gerontol A Biol Sci Med Sci*. 2013;68(10):1135-1144.

doi:10.1093/GERONA/GLT016

46. Rual JF, Ceron J, Koreth J, et al. Toward improving Caenorhabditis elegans phenome mapping with an ORFeome-based RNAi library. *Genome Res*. 2004;14(10B):2162-2168. doi:10.1101/GR.2505604

47. Kamath RS, Martinez-Campos M, Zipperlen P, Fraser AG, Ahringer J. Effectiveness of specific RNA-mediated interference through ingested double-stranded RNA in Caenorhabditis elegans. *Genome Biol*. 2001;2(1):research0002.1. doi:10.1186/GB-2000-2-1-RESEARCH0002

48. Gallagher LA, Manoil C. Pseudomonas aeruginosa PAO1 kills Caenorhabditis elegans by cyanide poisoning. *J Bacteriol*. 2001;183(21):6207-6214. doi:10.1128/JB.183.21.6207-6214.2001

49. Han SK, Lee D, Lee H, et al. OASIS 2: online application for survival analysis 2 with features for the analysis of maximal lifespan and healthspan in aging research. *Oncotarget*. 2016;7(35):56147-56152. doi:10.18632/ONCOTARGET.11269

50. Kapahi P, Kaeberlein M, Hansen M. Dietary restriction and lifespan: Lessons from invertebrate models. *Ageing Res Rev*. 2017;39:3-14. doi:10.1016/J.ARR.2016.12.005

51. Gems D, Sutton AJ, Sundermeyer ML, et al. Two pleiotropic classes of daf-2 mutation affect larval arrest, adult behavior, reproduction and longevity in Caenorhabditis elegans. *Genetics*. 1998;150(1):129-155. doi:10.1093/GENETICS/150.1.129

52. Kaeberlein M. How healthy is the healthspan concept? *GeroScience*. 2018;40(4):361-364. doi:10.1007/S11357-018-0036-9

53. Hahm J-H, Kim S, DiLoreto R, et al. *C. elegans* maximum velocity correlates with healthspan and is maintained in worms with an insulin receptor mutation. *Nat Commun*. 2015. doi:10.1038/ncomms9919

54. Mukherjee S, Russell JC, Carr DT, et al. Systems biology approach to late-onset Alzheimer's disease genome-wide association study identifies novel candidate genes validated using brain expression data and Caenorhabditis elegans experiments. *Alzheimer's Dement*. February 2017. doi:10.1016/j.jalz.2017.01.016

55. Bansal A, Zhu LJ, Yen K, Tissenbaum HA. Uncoupling lifespan and healthspan in Caenorhabditis elegans longevity mutants. doi:10.1073/pnas.1412192112

56. Pan CL, Peng CY, Chen CH, McIntire S. Genetic analysis of age-dependent defects of the Caenorhabditis elegans touch receptor neurons. *Proc Natl Acad Sci U S A*. 2011;108(22):9274-9279. doi:10.1073/PNAS.1011711108

57. Leinwand SG, Yang CJ, Bazopoulou D, Chronis N, Srinivasan J, Chalasani SH. Circuit mechanisms encoding odors and driving aging-associated behavioral declines in Caenorhabditis elegans. *Elife*. 2015;4(September 2015). doi:10.7554/ELIFE.10181

58. Rollins JA, Howard AC, Dobbins SK, Washburn EH, Rogers AN. Assessing Health Span in Caenorhabditis elegans: Lessons From Short-Lived Mutants. *J Gerontol A Biol Sci Med Sci*. 2017;72(4):473-480. doi:10.1093/gerona/glw248

59. Husson SJ, Costa WS, Schmitt C, Gottschalk A. Keeping track of worm trackers. *WormBook*. 2013:1-17. doi:10.1895/WORMBOOK.1.156.1

60. Leiser SF, Jafari G, Primitivo M, et al. Age-associated vulval integrity is an important marker of nematode healthspan. *Age (Omaha)*. 2016;38(5-6):419-431. doi:10.1007/s11357-016-9936-8

61. Prinz F, Schlange T, Asadullah K. Believe it or not: how much can we rely on published data on potential drug targets? *Nat Rev Drug Discov*. 2011;10(9):712-713. doi:10.1038/NRD3439-C1

62. Sierra F. The Emergence of Geroscience as an Interdisciplinary Approach to the Enhancement of Health Span and Life Span. *Cold Spring Harb Perspect Med*. 2016;6(4). doi:10.1101/CSHPERSPECT.A025163

63. Sperling RA, Jack CR, Aisen PS, Aisen PS. Testing the right target and right drug at the right stage. *Sci Transl Med*. 2011;3(111):111cm33. doi:10.1126/scitranslmed.3002609

64. Mccoll G, Roberts BR, Pukala TL, et al. Utility of an improved model of amyloid-beta (Aβ1-42) toxicity in Caenorhabditis elegans for drug screening for Alzheimer's disease. *Mol Neurodegener*. 2012;7(1):1-9. doi:10.1186/1750-1326-7-57/FIGURES/5

65. Taylor LM, McMillan PJ, Liachko NF, et al. Pathological phosphorylation of tau and TDP-43 by TTBK1 and TTBK2 drives neurodegeneration. *Mol Neurodegener*. 2018;13(1):7. doi:10.1186/s13024-018-0237-9

66. Latimer CS, Stair JG, Hincks JC, et al. TDP-43 promotes tau accumulation and selective neurotoxicity in bigenic Caenorhabditis elegans. *Dis Model Mech*. 2022;15(4). doi:10.1242/DMM.049323

67. Johnson TE. Advantages and disadvantages of Caenorhabditis elegans for aging research. *Exp Gerontol*. 2003;38(11-12):1329-1332. doi:10.1016/J.EXGER.2003.10.020

68. Pitt JN, Strait NL, Vayndorf EM, et al. WormBot, an open-source robotics platform for survival and behavior analysis in C. elegans. *GeroScience*. 2019;41(6):961-973. doi:10.1007/S11357-019-00124-9

69. Barardo D, Thornton D, Thoppil H, et al. The DrugAge database of aging-related drugs. *Aging Cell*. 2017;16(3):594-597. doi:10.1111/acel.12585

70. Berkel C, Cacan E. A collective analysis of lifespan-extending compounds in diverse model organisms, and of species whose lifespan can be extended the most by the application of compounds. *Biogerontology*. 2021;22(6):639-653. doi:10.1007/s10522-021-09941-y

71. Janizek JD, Spiro A, Celik S, et al. Principled feature attribution for unsupervised gene expression analysis. *bioRxiv*. May 2022:2022.05.03.490535. doi:10.1101/2022.05.03.490535

72. Wang W, Zhao F, Ma X, Perry G, Zhu X. Mitochondria dysfunction in the pathogenesis of Alzheimer’s disease: recent advances. *Mol Neurodegener 2020 151*. 2020;15(1):1-22. doi:10.1186/S13024-020-00376-6

73. Durinck S, Moreau Y, Kasprzyk A, et al. BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*. 2005;21(16):3439-3440. doi:10.1093/BIOINFORMATICS/BTI525

74. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *J Mol Biol*. 1990;215(3):403-410. doi:10.1016/S0022-2836(05)80360-2

75. Wheeler DL, Barrett T, Benson DA, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*. 2008;36(Database issue):D13. doi:10.1093/NAR/GKM1000

76. Sutphin GL, Kaeberlein M. Measuring Caenorhabditis elegans Life Span on Solid Media. *JoVE (Journal Vis Exp*. 2009;(27):e1152. doi:10.3791/1152

77. McColl G, Roberts BR, Pukala TL, et al. Utility of an improved model of amyloid-beta (Aβ1-42) toxicity in Caenorhabditis elegans for drug screening for Alzheimer's disease. *Mol Neurodegener*. 2012;7(1):57. doi:10.1186/1750-1326-7-57

78. Abbott M, Banse SA, Melentijevic I, et al. A simplified design for the C. elegans lifespan machine. *J Biol Methods*. 2020;7(4):e137. doi:10.14440/JBM.2020.332

79. Rahman M, Edwards H, Birze N, et al. NemaLife chip: a micropillar-based microfluidic culture device optimized for aging studies in crawling C. elegans. *Sci Reports 2020 101*. 2020;10(1):1-19. doi:10.1038/s41598-020-73002-6

80. Saberi-Bosari S, Huayta J, San-Miguel A. A microfluidic platform for lifelong high-

resolution and high throughput imaging of subtle aging phenotypes in C. elegans. *Lab Chip*. 2018;18(20):3090-3100. doi:10.1039/C8LC00655E

81. Javer A, Currie M, Lee CW, et al. An open source platform for analyzing and sharing worm behavior data. *Nat Methods*. 2018;15(9):645. doi:10.1038/S41592-018-0112-1

82. Swierczek NA, Giles AC, Rankin CH, Kerr RA. High-Throughput Behavioral Analysis in C. elegans. *Nat Methods*. 2011;8(7):592. doi:10.1038/NMETH.1625

83. Macukow B. Neural networks-state of art, brief history, basic models and architecture. *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)*. 2016;9842 LNCS:3-14. doi:10.1007/978-3-319-45378-1_1/FIGURES/10

84. Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 2015;2016-December:779-788. doi:10.48550/arxiv.1506.02640

85. Sarnoski EA, Song R, Ertekin E, Koonce N, Acar M. Fundamental Characteristics of Single-Cell Aging in Diploid Yeast. *iScience*. 2018;7:96-109. https://linkinghub.elsevier.com/retrieve/pii/S2589004218301238. Accessed September 30, 2018.

86. Zhang WB, Sinha DB, Pittman WE, Hvatum E, Stroustrup N, Pincus Z. Extended Twilight among Isogenic C. elegans Causes a Disproportionate Scaling between Lifespan and Health. *Cell Syst*. 2016;3(4):333-345.e4. doi:10.1016/J.CELS.2016.09.003

87. Mendenhall A, Crane MM, Tedesco PM, Johnson TE, Brent R. Caenorhabditis elegans Genes Affecting Interindividual Variation in Life-span Biomarker Gene Expression. *Journals Gerontol Ser A*. 2017;72(10):1305-1310. doi:10.1093/gerona/glw349

88. Restif C, Ibáñez-Ventoso C, Vora MM, Guo S, Metaxas D, Driscoll M. CeleST: Computer Vision Software for Quantitative Analysis of C. elegans Swim Behavior Reveals Novel Features of Locomotion. Prlic A, ed. *PLoS Comput Biol*. 2014;10(7):e1003702. doi:10.1371/journal.pcbi.1003702

89. Statzer C, Reichert P, Dual J, Ewald CY. Longevity interventions temporally scale healthspan in Caenorhabditis elegans. *iScience*. 2022;25(3):103983. doi:10.1016/j.isci.2022.103983

Appendix

R-Scripts

R-Shiny Lachesis

```
library(shiny)

library(tidyverse)

library(ggplot2)

library(magrittr)

library(survminer)

library(survival)

library(openxlsx)

#Lachesis


#create data frame







# Define UI for data upload app ----

ui <- fluidPage(


  tags$head(

    tags$style(HTML("hr {border-top: 1px solid #000000;}"))

  ),


  # App title ----

  titlePanel("Lachesis"),


  # Sidebar layout with input and output definitions ----

  sidebarLayout(
```

```
# Sidebar panel for inputs ----
sidebarPanel(
  #img(src='worm.png', align = "right",height = 70, width = 200),

  # Input: Select a file ----
  selectInput("dataformat","Select input data format",
           c("WormBot" = "wormbot",
            "TidyVerse" = "tidyverseformat")),
  fileInput("file1", "Choose CSV File",
          multiple = TRUE,
          accept = c("text/csv",
                 "text/comma-separated-values,text/plain",
                 ".csv")),
  p("Select which groupings to graph and analyze"),
  uiOutput("analysisSelect"),
  hr(),
  selectInput("isCumHz","Select graph type",
           c("Survival" = "NULL",
            "Cumulative events" = "event",
            "Cumulative hazard" = "cumhaz")),


  #download graph button
  numericInput("graphX", "Graph size in inches (x-axis)",value= 10),
  numericInput("graphY", "Graph size in inches (y-axis)",value= 6),

  downloadButton("graph", label = "Download graph"),

  #download graph button
```

```
    downloadButton("stats", label = "Download stats"),

    selectInput("fileForm","Select image file type",
                c("PNG"="png",
                  "TIFF"="tiff",
                  "PDF"="pdf",
                  "SVG"=".svg")),
    br(),
    #save filename
    textInput("savename","Enter Savename"),
    hr(),

    #save filename
    br(),
    uiOutput("colorselect")

),

# Main panel for displaying outputs ----
mainPanel(

    # Output: Data file ----
    #tableOutput("contents")
    tabsetPanel(type="tabs",
                tabPanel("Survival curve",
                         plotOutput("plot",width = "700px", height = "500px"),
                         plotOutput("coxplot",width = "700px", height = "500px"),
                         numericInput("xmin", "X axis min:",value= 0),
                         numericInput("xmax", "X axis max:",value= 30),
                         textInput("xAxisLabel","X-axis label",value=""),
```

```
            textInput("yAxisLabel","Y-axis label",value = ""),
            selectInput("legPos","Select legend position",
                        c("Top right"= "right",
                          "Top left" = "left",
                          "None" = "none")),
            checkboxInput("confInt","Display confidence intervals"),
            checkboxInput("medLine","Display median lines"),
            textInput("tickmarks","Tick mark intervals",value = "1")),
        tabPanel("Summary and Statistics",
            h4("Summary"),
            tableOutput("summary"),
            h4("Pairwise p-values"),
            tableOutput("survfit"),
            br(),
            selectInput("corrmethod","Select p-value adjustment",
                        c("None" = "none",
                          "Bonferroni" = "bonferroni",
                          "Holm" =  "holm",
                          "Hommel" = "hommel",
                          "Hochberg" = "hochberg",
                          "Benjamini-Hochberg" = "BH",
                          "Benjamini-Yekutieli" = "BY"))
            ),
        tabPanel("Raw Data",
            downloadButton("tidyversedata", label = "Download TidyVerse
Dataset"),
            tableOutput("contents")
            ),
        tabPanel("Instructions",
```

```
                         p("Lifespan plotter accepts comma delimited .csv files as input.
Files can either be in WormBot format or in TidyVerse Format."),
                         br(),
                         p("WormBot format is shown below with # signs denoting
conditional fields.
                           Afterwards, each row should have an age in the first column,
then either
                           a 1 for a death event or a 0 for a censored animal in the second
column."),
                         br(),
                         tableOutput("exampleTable"),
                         br(),
                         downloadButton("exampleFile", label = "Download example
wormbot file"),
                         br(),
                         p("TidyVerse format is a .csv in which the first row is the column
names and each animal is given unique row that includes a column for the age-at-
death. AT LEAST
                          one conditional column must be included with a conditional field
such as 'strain' with entries for every row
                          Please note the age-at-death column name must be formatted as
'age'. If some animals were lost from observation, include a column named'censored'
                          with either a '0' for a censored animal or a '1' for an observed
death."))
         )
      )
   )
)


# Define server logic to read selected file ----
server <- function(input, output) {


   ####
```

```
#creates and updates the table as necessary.
create_table <- reactive({


    if(input$dataformat == 'wormbot'){
    dataf  <- read.csv(input$file1$datapath, header=FALSE, comment.char="%",
stringsAsFactors=FALSE)
    breakpoint <- dataf[1,1]
    breaks <- grep(breakpoint,dataf$V1) #find # signs
    breaksfull <- breaks
    connames <-dataf[breaks,1]
    nWells <- length(breaks)
    breaks[(length(breaks)+1)] <- length(dataf$V1)+1 #find endpoint of document and
add it to breaks


    #for each well
    for (i in 1L:nWells) {
        SingleWell <- slice(dataf,breaks[i]:(breaks[i+1]-1)) #extract data
        NumRows <- length(SingleWell[,1])
        Cons <- grep("#",SingleWell$V1)
        ConTable <-slice(SingleWell,Cons[1]:length(Cons))
        Anustart <- length(Cons)
        colnames(SingleWell) <- c("age","censored")
        SingleWell <- SingleWell[-1:-Anustart,]
        SingleWell <- as_tibble(SingleWell)
        SingleWell %>% mutate(age = as.numeric(age),censored =
as.numeric(censored))
        #SingleWell <- unlist(mapply(rep, SingleWell$age, SingleWell$censored))
        SingleWell <- as_tibble(SingleWell)
        colnames(SingleWell)<- c("age","censored")
        #SingleWell$censored <- 1
```

```
        #for each field exported
        for (k in Cons){
          NewCol <- character()
          ConName <- str_remove(ConTable[k,1],"\\#")
          ConName <- gsub(" ", "_", ConName, fixed = TRUE)
          NewCol[1:(Anustart)] <- ConName
          NewCol[(Anustart+1):NumRows] <- ConTable[k,2]
          NewCol <- as_tibble(NewCol)
          colnames(NewCol)<-NewCol[[1,1]]
          NewCol <- NewCol[-1:-Anustart,]
          SingleWell <- add_column(SingleWell,NewCol)
        }

        #if first well then create totalTable variable
        if (i == 1L) {
          totalTable <- SingleWell
        }
        #if subsequent wells append new data to totalTable
        if (i > 1L) {
          totalTable <- bind_rows(totalTable,SingleWell)
        }
      }
      #book-keeping to make sure that variables are kept as numeric despite data-
wrangling
      totalTable <- mutate(totalTable, age = as.numeric(age),censored =
as.numeric(censored))
      #print(totalTable)
      return(totalTable)
      }
      else{
```

```
            dataf  <- read.csv(input$file1$datapath, header=TRUE,
stringsAsFactors=FALSE)
            if("rls" %in% colnames(dataf))
            {
               dataf<- dataf %>% rename(age = rls)
            }
            if("RLS" %in% colnames(dataf))
            {
               dataf<- dataf %>% rename(age = RLS)
            }
            if("censored" %in% colnames(dataf)==FALSE)
            {
               dataf$censored <- 1
            }
            dataf <- dataf %>% relocate(censored) %>% relocate(age)
            totalTable <- dataf %>% mutate(age = as.numeric(age), censored =
as.numeric(censored))

         }

    })

    xmarks <- reactive({
       xint <- as.numeric(input$tickmarks)
       return(xint)
    })

    showcon <- reactive({
       req(input$file1)
       totalTableF <- create_table()
```

```
    conditionNames <- colnames(totalTableF)[3:length(colnames(totalTableF))]
    conditionBool <- character()
    varname <- character()
    for(i in 1:length(conditionNames)){

        varname[i] <- paste0('chk_',gsub(" ", "", conditionNames[i], fixed = TRUE))
        if(input[[varname[i]]]==TRUE){
            conditionBool[i] <- conditionNames[i]
        }
        else{
            conditionBool[i] <- "NULL"
        }
    }
    conditionSurv <- paste(conditionBool,collapse = " + ")
})

###
#creates analysis selector objects for each condition
output$analysisSelect <- renderUI({
    req(input$file1)
    totalTableF <- create_table()
    conditionNames <- colnames(totalTableF)[3:length(colnames(totalTableF))]
    lapply(conditionNames, function(x){
        checkboxInput(paste0('chk_',gsub(" ", "", x, fixed = TRUE)),label = x)
    })
})



###
#creates color selector objects for each condition
```

```
    output$colorselect <- renderUI({
        req(input$file1)
        totalTableF <- create_table()
        fit3 <- eval(parse(text = paste("survfit(Surv(age, censored) ~ ", showcon(), ",
data=totalTableF)")))
        #print(showcon())
        namesCh <- gsub("description=","",names(fit3$strata))
        namesCh <- gsub("description=","",namesCh)

        lapply(namesCh, function(x){
            fluidRow(
                textInput(paste0('labtxt_',gsub(" ", "", x, fixed = TRUE)),
                          label = NULL,
                          value = x),
                selectInput(paste0('lab_',gsub(" ", "", x, fixed = TRUE)),
                            label = NULL,
                            choices =
c("black","gray","blue","green","orange","purple","red","cyan","yellow","goldenrod","darkg
reen","violet","coral")),
                selectInput(paste0('line_',gsub(" ", "", x, fixed = TRUE)),
                            label = NULL,
                            choices = c("solid","dashed","dotted")),
                checkboxInput(paste0('omit_',gsub(" ", "", x, fixed = TRUE)),
                              label = 'Remove Curve'),



                hr()
            )
        })
    })
```

```
###
#show raw data
output$contents <- renderTable({
   req(input$file1)
   totalTableF <- create_table()
   return(totalTableF)


})


###
#plot graph and update when necessary
plotInput <-  reactive({
   req(input$file1)
   totalTableF <- create_table()
   fit3 <- eval(parse(text = paste("survfit(Surv(age, censored) ~ ", showcon(), ",
data=totalTableF)")))
   #if (nrow(distinct(select(totalTableF,description))) > 1){
   namesCh <- gsub("description=","",names(fit3$strata))
   #}
   #else{
   #    namesT <- distinct(select(totalTableF,description))
   #    namesCh <- namesT$description
   #}


   cLabels <- character()
   varname <- character()


   for(i in 1:length(namesCh)){
      varname[i] <- paste0('labtxt_',gsub(" ", "", namesCh[i], fixed = TRUE))
      cLabels[i] <- input[[varname[i]]]
```

```
}

cPallete <- character()
varname2 <- character()
for(i in 1:length(namesCh)){
   varname2[i] <- paste0('lab_',gsub(" ", "", namesCh[i], fixed = TRUE))
   cPallete[i] <- input[[varname2[i]]]
}
cLine <- character()
varname3 <- character()
for(i in 1:length(namesCh)){
   varname3[i] <- paste0('line_',gsub(" ", "", namesCh[i], fixed = TRUE))
   cLine[i] <- input[[varname3[i]]]
}

cOpaque <- numeric()
varname4 <- character()
for(i in 1:length(namesCh)){
   varname4[i] <- paste0('omit_',gsub(" ", "", namesCh[i], fixed = TRUE))
   if(input[[varname4[i]]] == TRUE){
      cOpaque[i] <- 0
   } else {
      cOpaque[i] <- 1
   }
}

#cLabels[cOpaque==0] <- "    "
cPallete[cOpaque==0] <- "white"
cLine[cOpaque==0]<-"blank"
```

```
if(input$isCumHz=="event"){

finalPlot <- ggsurvplot(fit3, data = totalTableF, conf.int = input$confInt,ylim =
c(0,1.1),xlim=c(input$xmin,input$xmax),xlab= input$xAxisLabel,ylab =
input$yAxisLabel,palette = alpha(cPallete,cOpaque),legend = legendPos(),legend.labs
= cLabels, linetype = cLine, legend.title = "Legend",title=
input$savename,break.x.by=xmarks(),break.y.by = 0.1,surv.median.line =
displayMedian(),fun = input$isCumHz,censor = FALSE)

}

else if(input$isCumHz=="NULL"){

finalPlot <- ggsurvplot(fit3, data = totalTableF, conf.int = input$confInt,ylim =
c(0,1.1),xlim=c(input$xmin,input$xmax),xlab= input$xAxisLabel,ylab =
input$yAxisLabel,palette = alpha(cPallete,cOpaque),legend = legendPos(),legend.labs
= cLabels, linetype = cLine, legend.title = "Legend",title=
input$savename,break.x.by=xmarks(),break.y.by = 0.1,surv.median.line =
displayMedian(),censor = FALSE)


}else{

finalPlot <- ggsurvplot(fit3, data = totalTableF, conf.int =
input$confInt,xlim=c(input$xmin,input$xmax),xlab= input$xAxisLabel,ylab =
input$yAxisLabel,palette = alpha(cPallete,cOpaque),legend = legendPos(),legend.labs
= cLabels, linetype = cLine, legend.title = "Legend",title=
input$savename,break.x.by=xmarks(),surv.median.line = displayMedian(),fun =
input$isCumHz,censor = FALSE)


}

return(finalPlot)


})



###

#displays graph in GUI
```

```
output$plot <- renderPlot({
    print(plotInput())
})


###
#reactive input for median line control
displayMedian <- reactive({
    if(input$medLine == TRUE){
        return("v")
    }
    else {
        return("none")
    }
})

###
#reactive function for legend position in graph
legendPos <- reactive({
    if(input$legPos == "right"){
        val <- c(.8,.8)
    }
    else if(input$legPos=="left"){
        val <- c(.2,.8)
    }
    else {
        val <- "none"
    }

    return(val)
```

```
  })


  ###
  #creates statistical table
  statsInput <-  reactive({
    req(input$file1)
    totalTableF <- create_table()


    fitsurfdif <- eval(parse(text = paste("pairwise_survdiff(Surv(age, censored) ~ ",
showcon(), ",data=totalTableF,p.adjust.method = input$corrmethod")))


    #fitsurfdif <- pairwise_survdiff(Surv(age, censored) ~ description,
data=totalTableF,p.adjust.method = input$corrmethod)
    fitpval <- fitsurfdif$p.value
    z <-1
    newTable <- tibble(Treatments=character(),pvalue=numeric())
    for(i in seq(1,ncol(fitpval))){
      for(k in seq(1,nrow(fitpval))){


        if(is.na(fitpval[k,i]) == FALSE){
          newTable[z,1] <- paste0(colnames(fitpval)[i]," vs ",rownames(fitpval)[k])
          newTable[z,2]<-fitpval[k,i]
          z <- z +1
        }
      }
    }
```

```
    return(newTable)
  })


  ###
  #creates summary table
  summaryInput <- reactive({
    req(input$file1)
    totalTableF <- create_table()
    fit <- eval(parse(text = paste("survfit(Surv(age, censored) ~ ", showcon(), ",
data=totalTableF)")))
    #fit <- survfit(Surv(age, censored) ~ description, data=totalTableF)
    lok <- surv_median(fit)
    lok <- surv_median(fit)

    for( i in seq(1L,length(fit$n))){
      lok[i,5]<-fit$n[i]
    }
    lok[ , c(1,2,3,4,5)] <- lok[ , c(1,2,5,3,4)]



    namesCh <- gsub("description=","",names(fit$strata))

    cLabels <- character()
    varname <- character()

    for(i in 1:length(namesCh)){
      varname[i] <- paste0('labtxt_',gsub(" ", "", namesCh[i], fixed = TRUE))
      lok[i,1] <- input[[varname[i]]]
```

```
    }

    colnames(lok)[3] <- "censored"
    colnames(lok)[1] <- "description"
    colnames(lok)[4] <- "95% lower confidence limit"
    colnames(lok)[5] <- "95% upper confidence limit"

    return(lok)
})

###
#displays summary
output$summary <- renderTable({
    summaryInput()
},digits = 3)

###
#displays  stats
output$survfit <- renderTable({
    statsInput()
},rownames = TRUE, digits = 3)


###
#creates excel book object
excelBook <- reactive({
    wrkBook <- createWorkbook()
    addWorksheet(wrkBook,"Summary")
    addWorksheet(wrkBook,"Statistics")
    addWorksheet(wrkBook,"Raw_Data")
```

```
    writeData(wrkBook,"Summary",summaryInput())
    writeData(wrkBook,"Statistics",statsInput())
    writeData(wrkBook,"Raw_Data",create_table())
    return(wrkBook)
  })

  ###
  #saves excel book object
  output$stats <- downloadHandler(
    filename = function() {
      paste(input$savename, "stats.xlsx", sep = "_")
    },
    content = function(file) {
      saveWorkbook(excelBook(),file,overwrite = TRUE)
    }
  )

  ###
  #saves graph handler
  output$graph <- downloadHandler(
    filename = function() {paste(input$savename, "_graph.",input$fileForm, sep = "")},
    #filename = "test.png",
    content = function(file) {
      ggsave(file, plot =  print(plotInput()), device = input$fileForm,width = 
input$graphX, height = input$graphY,dpi=1200)
    }
  )

  ###
  #outputs example table
```

```
    output$exampleTable <- renderTable({
        newTable <- tibble(column1 =
c("%break","#title","#treatment",5,6,6,10,11,12,13,14,14,14,15,"%break","#title","#treatm
ent",7,7,8,9,10,10,10,13), column2 =
c("","CoolData","DMSO",1,1,0,1,1,1,1,1,1,0,1,"","CoolerData","EtOH",1,1,0,1,1,1,0,1))


    })


    exampleTableP <- reactive({
        newTable <- tibble(column1 =
c("%break","#title","#treatment",5,6,6,10,11,12,13,14,14,14,15,"%break","#title","#treatm
ent",7,7,8,9,10,10,10,13), column2 =
c("","CoolData","DMSO",1,1,0,1,1,1,1,1,1,0,1,"","CoolerData","EtOH",1,1,0,1,1,1,0,1))
        colnames(newTable) <- c("%","%")
        return(newTable)
    })


    output$exampleFile <- downloadHandler(
        filename = "exampledataset.csv",
        content = function(file) {
            write.csv(exampleTableP(), file, row.names = FALSE, quote=FALSE)
        }
    )



}
```

```
# Create Shiny app ----
shinyApp(ui, server)
```

R-Shiny Neural Trainer

```
# Adapted from https://github.com/kyamagu/bbox-annotator/
# Edited original JS to add color_list as an option
# ...should be the same length as labels
# ...and controls the color of the rectangle
# ...will probably be broken for input_method = "fixed" or "text"
# Also added color as a value in each rectangle entry



js <- '
  $(document).ready(function() {
    // define options to pass to bounding box constructor
     var options = {
       url: "https://www.dropbox.com/s/s8vh1wdpok15lz5/worm.PNG?raw=1",
       input_method: "select",
       labels: [""],
       color_list:  [""],
       onchange: function(entries) {
            Shiny.onInputChange("rectCoord", JSON.stringify(entries, null, "  "));
       }
     };

     // Initialize the bounding-box annotator.
     var annotator = new BBoxAnnotator(options);

     // Initialize the reset button.
     $("#reset_button").click(function(e) {
```

```
    annotator.clear_all();
  })

  // define function to reset the bbox
  // ...upon choosing new label category or new url
  function reset_bbox(options) {
    document.getElementById("bbox_annotator").setAttribute("style", "display:inline-block");
    $(".image_frame").remove();
    annotator = new BBoxAnnotator(options);
  }

  // update image url from shiny
  Shiny.addCustomMessageHandler("change-img-url", function(url) {
    options.url = url;
    options.width = null;
    options.height = null;
    reset_bbox(options);
  });

  // update colors and categories from shiny
  Shiny.addCustomMessageHandler("update-category-list", function(vals) {
    options.labels = Object.values(vals);
    options.color_list = Object.keys(vals);
    reset_bbox(options);
  });

  //update list from CSV
  Shiny.addCustomMessageHandler("rectCoord", function(value) {
    Shiny.setInputValue("rectCoord", value);
```

```
          });


        // redraw rectangles based on list of entries
        Shiny.addCustomMessageHandler("redraw-rects", function(vals) {
          var arr = JSON.parse(vals)
          arr.forEach(function(rect){
            annotator.add_entry(rect);
          });
          if (annotator.onchange) {
            annotator.onchange(annotator.entries);
          }
        });
    });
'
```

```
library(shiny)
library(tidyverse)
library(ggplot2)
library(magrittr)
library(survminer)
library(survival)
library(openxlsx)
library(rdrop2)
library(jsonlite)
library(stringr)
```

```r
token <- readRDS("droptoken.rds")
drop_auth(rdstoken = "droptoken.rds")
#change below for new dropbox folders
drop_imgdir <- "/WormYOLO_training/raw_imgs/"
#drop_csvdir <- "/Najafian Collab/NN_pretrain_csv/"
drop_imgdirdone <- "/WormYOLO_training/post_imgs/"
drop_csvdirdone <- "/WormYOLO_training/post_csvs/"

#totalsize <- nrow(drop_dir(drop_imgdir,dtoken = token))
#randnumber <- sample(totalsize,1)
#imageName <- drop_dir(drop_imgdir,dtoken = token)[randnumber,2]
#print(paste0(imageName,"start"))

round_df <- function(x, digits) {
  # round all numeric variables
  # x: data frame
  # digits: number of digits to round
  numeric_columns <- sapply(x, mode) == 'numeric'
  x[numeric_columns] <-  round(x[numeric_columns], digits)
  x
}

# Define UI for application that draws a histogram
ui <- fluidPage(
  tags$head(tags$script(HTML(js))),
          tags$head(
            tags$script(src = "bbox_annotation.js")
          )),
  titlePanel("Neural Network Training"),
```

```
  sidebarLayout(
    sidebarPanel(
      actionButton("goButton", "Start!"),
      actionButton("doneButton", "Next!"),

      selectInput("category_type", "Label Category", c("Worms", "geography")),
      div(HTML(
        '<input id="reset_button" type="reset" />'
      )),
      HTML(
        '<input id="annotation_data" name="annotation_data" type="hidden" />'
      ),
      hr(),
      h4("Entries"),
      verbatimTextOutput("rectCoordOutput")

    ),
    mainPanel(div(id = "bbox_annotator", style = "display:inline-block"))
  )
)




# Define server logic required to draw a histogram
server <- function(input, output, session) {
  imgNum <- reactiveVal()
  # user choices
  output$rectCoordOutput <- renderPrint({
```

```
  if(!is.null(input$rectCoord)) {
    as.data.frame(jsonlite::fromJSON(input$rectCoord))
  }
})



# send chosen URL from shiny to JS on start
observeEvent(input$goButton, {
  #get URL for dropbox and load image
  totalsize <- nrow(drop_dir(drop_imgdir,dtoken = token))
  randinput <- sample(totalsize,1)
  imgNum(randinput)
  imageName <- drop_dir(drop_imgdir,dtoken = token)[imgNum(),2]

  Sys.setenv(TZ="GMT")
  tpoint <- Sys.time()
  tpoint <- tpoint + 60
  tpoint1 <- gsub(" GMT","",tpoint)
  tpoint2 <- gsub(" ","T",tpoint1)
  tpoint3 <- paste0(tpoint2,"Z")
  dropPath <- paste0(drop_imgdir,imageName)
  fileURL <- drop_share(path = dropPath, expires = tpoint3,dtoken = token)
  realURL <- fileURL$url
  realULRcorr <- gsub("dl=0","raw=1",realURL)
  session$sendCustomMessage("change-img-url", realULRcorr)

  #get pre-annoted csv and load data
  #csvNewName <- gsub(".png","_NN.csv",imageName)
  #dropcsvPath <- paste0(drop_csvdir,csvNewName)
```

```
    #new_data <- drop_read_csv(dropcsvPath,dtoken = token)
    #new_data1<-subset(new_data,select=-c(dataCells1,dataCells2))
    #new_data1 <- round_df(new_data1,0)
    #new_data2 <- new_data1
    #new_data2$F <- "worm"
    #new_data2$G <- "yellow"
    #colLabels <- c("left","top","width","height","label","color")
    #colnames(new_data2) <- colLabels
    #jsonfile <- toJSON(new_data2,pretty=TRUE)
    #jsonfile2 <- str_replace_all(jsonfile, "[\r\n]" , "\n")
    #testfile <- head(new_data2,3)
    #jsontestfile <- "[\n  {\n    \"left\": 195,\n    \"top\": 172,\n    \"width\": 77,\n    \"height\":
49,\n    \"label\": \"worm\",\n    \"color\": \"yellow\"\n  },\n  {\n    \"left\": 199,\n    \"top\":
259,\n    \"width\": 67,\n    \"height\": 51,\n    \"label\": \"worm\",\n    \"color\": \"yellow\"\n
},\n  {\n    \"left\": 195,\n    \"top\": 331,\n    \"width\": 55,\n    \"height\": 46,\n    \"label\":
\"worm\",\n    \"color\": \"yellow\"\n  }\n]"


    #send message to load pre-annotted bounding boxes into the annotater program
    #session$sendCustomMessage("redraw-rects", jsonfile2)
    #print(paste0(imageName,"afterload"))


  })


  # send chosen URL from shiny to JS on next frame
  observeEvent(input$doneButton, {


    #get data and reformat csv file for YOLO
    imgNumber <- imgNum()
    imageName <- drop_dir(drop_imgdir, dtoken = token)[imgNum(),2]
    if(!is.null(input$rectCoord)) {
      csvfinished <- as.data.frame(jsonlite::fromJSON(input$rectCoord))
```

```
}
csvfinished1 <- subset(csvfinished,select=-c(color))
#df[, Variable] <- NA
##csvfinished1$dataCells1 <- imageName
csvfinished1[, "dataCells1"] <- imageName
csvfinished1 <- csvfinished1[c("dataCells1", "label", "left", "top", "width", "height")]
colLabelsFinal <-
c("dataCells1","dataCells2","dataCells3","dataCells4","dataCells5","dataCells6")
colnames(csvfinished1) <- colLabelsFinal
csvName <- gsub(".png","",imageName)
write.csv(csvfinished1, file = paste0(csvName,"_annotated.csv"))

#upload csv file
drop_upload(paste0(csvName,"_annotated.csv"),path = drop_csvdirdone, dtoken =
token)
unlink(paste0(csvName,"_annotated.csv"))
#move image file to finished directory
currentFile <- paste0(drop_imgdir,imageName)
newFile <- paste0(drop_imgdirdone,imageName)
drop_move(currentFile, newFile, dtoken = token)



#get URL for dropbox and load image
Sys.setenv(TZ="GMT")
tpoint <- Sys.time()
tpoint <- tpoint + 60
tpoint1 <- gsub(" GMT","",tpoint)
tpoint2 <- gsub(" ","T",tpoint1)
tpoint3 <- paste0(tpoint2,"Z")
totalsize <- nrow(drop_dir(drop_imgdir,dtoken = token))
```

```
  randnumber <- sample(totalsize,1)
  imgNum(randnumber)
  imageName <- drop_dir(drop_imgdir, dtoken = token)[imgNum(),2]
  dropPath <- paste0(drop_imgdir,imageName)
  fileURL <- drop_share(path = dropPath, expires = tpoint3, dtoken = token)
  realURL <- fileURL$url
  realULRcorr <- gsub("dl=0","raw=1",realURL)
  session$sendCustomMessage("change-img-url", realULRcorr)

  #get pre-annoted csv and load data
  #csvNewName <- gsub(".png","_NN.csv",imageName)
  #dropcsvPath <- paste0(drop_csvdir,csvNewName)
  #new_data <- drop_read_csv(dropcsvPath, dtoken = token)
  #new_data1<-subset(new_data,select=-c(dataCells1,dataCells2))
  #new_data1 <- round_df(new_data1,0)
  #new_data2 <- new_data1
  #new_data2$F <- "worm"
  #new_data2$G <- "yellow"
  #colLabels <- c("left","top","width","height","label","color")
  #colnames(new_data2) <- colLabels
  #jsonfile <- toJSON(new_data2,pretty=TRUE)
  #jsonfile2 <- str_replace_all(jsonfile, "[\r\n]" , "\n")

  #send message to load pre-annotted bounding boxes into the annotater program
  #session$sendCustomMessage("redraw-rects", jsonfile2)
})

#send chosen category list from shiny to JS
observeEvent(input$category_type, {
  vals <- switch(input$category_type,
```

```
            Worms = list("yellow" = "worm","green" = "bacteria","blue" = "well"),
            geography = list("grey" = "well",
                             "brown" = "bacteria",
                             "tan" = "something else")
    )
    #update category list
    session$sendCustomMessage("update-category-list", vals)
    #redraw rectangles
    session$sendCustomMessage("redraw-rects", input$rectCoord)
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

Python Scripts

For a compilation of all the Worm-YOLO pipeline, please refer to the projects GitHub at https://github.com/paolobif/Worm-Yolo.

Worm-YOLO Video Annotator

```
import os
import sys
import cv2
import csv
import tqdm
import pandas as pd
from yolov3_core import *
from sort.sort import *
```

```
## initialize model
class YoloToCSV():
    def __init__(self, model, frame, frame_count):
        """
        model is a model object from YoloModelLatest class.
        img_path is the complete path to the image
        """
        self.model = model
        #self.img_path = img_path
        self.img = frame
        self.img_path = frame_count

    def get_annotations(self):
        # pass through img processor. Image and cut size.
        img_size = 416
        outputs = self.model.pass_model(self.img)
        self.outputs = outputs
        if outputs:
            #print(outputs)
            outputs = non_max_suppression_post(outputs, overlapThresh=0.1)
        return outputs

    def write_to_csv(self, out_path):
        """Writes outputs to CSV at out_path"""
        img_name = self.img_path
        outputs = self.get_annotations()
        df = self.pd_for_csv(outputs, img_name)
        #outputsSort = self.sort_update(outputs)
```

```
        ##print(outputsSort)
        #if self.img_path > 1:
        #    df = self.pd_for_sort_output(outputsSort, img_name)
        df.to_csv(out_path, mode='a', header=False, index=None)
        print(f"Wrote {img_name} to csv!")

    def draw_on_im(self, out_path, writer, text=None):
        """Takes img, then coordinates for bounding box, and optional text as arg"""
        img = self.img
        for output in self.outputs:
            output = [int(n) for n in output]
            x1, y1, x2, y2, *_ = output
            # Draw rectangles
            cv2.rectangle(img, (x1,y1), (x2,y2), (255,255,0), 2)
            if text is not None:
                cv2.putText(img, text, (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, col,
2)
        ## write image to path
        #cv2.imwrite(out_path, img)
        writer.write(img)
    # creates pandas df for easy csv saving.
    @staticmethod
    def pd_for_csv(outputs, img_name = "name"):
        """Converts tensors to list that is added to pd df for easy writing to csv"""
        csv_outputs = []
        for output in outputs:
            x1, y1, x2, y2, *_ = output
            w = abs(x2-x1)
            h = abs(y2-y1)
```

```
            csv_outputs.append([img_name, x1.tolist(), y1.tolist(), w.tolist(), h.tolist(),
"worm"]) # ideally change to list earlier bc now outputs is a mix of tensors and lists....
        out_df = pd.DataFrame(csv_outputs)
        # change header to datacells for R-shiny processing
        #out_df =
out_df.set_axis(['dataCells1','dataCells2','dataCells3','dataCells4','dataCells5','class'],
axis=1)
        return out_df


    def sort_update(self, outputs):
        fullOutputs = np.array(outputs)
        boxes_xyxy = fullOutputs[:,:4]
        track_bbs_ids = mot_tracker1.update(boxes_xyxy)
        return(track_bbs_ids)



    def pd_for_sort_output(self, outputs, img_name = "name"):
        csv_outputs = []
        for worm in outputs:
            worm = worm.astype(np.int32)
            x1 = worm[0]
            y1 = worm[1]
            x2 = worm[2]
            y2 = worm[3]
            name = worm[4]
            csv_outputs.append([img_name, name, x1, y1 ,x2, y2])
        out_df = pd.DataFrame(csv_outputs)
        return out_df


if __name__ == "__main__":
    # declare source directory and out path
```

```
"""
IMG_DIR is path to the folder with the images
OUT_PATH is the path to the csv file you would like the output to go to
i.e './output/sample.csv'
"""
VID_FOLD_PATH = sys.argv[1]
OUT_FOLD_PATH = sys.argv[2]



vid_list = os.listdir(VID_FOLD_PATH)
print(vid_list)
for vid_name in vid_list:

    VID_PATH = os.path.join(VID_FOLD_PATH, vid_name)
    OUT_PATH = OUT_FOLD_PATH

    ## Declare settings for nn
    ## make sure to change these prarameters for your work enviroment
    settings = {'model_def': "cfg/yolov3-spp-1cls.cfg",
                'weights_path': "weights/416_1_4_full_best200ep.pt",
                'class_path': "cfg/classes.names",
                'img_size': 608,
                'iou_thres': 0.4,
                'no_gpu': True,
                'conf_thres': 0.1,
                'batch_size': 6,
                'augment': None,
                'classes': None}

    model = YoloModelLatest(settings)
```

```
#img_list = os.listdir(IMG_DIR)


vid = cv2.VideoCapture(VID_PATH)
total_frame_count = vid.get(cv2.CAP_PROP_FRAME_COUNT)
video_name = os.path.basename(VID_PATH).strip('.avi')


csv_out_path = f"{os.path.join(OUT_PATH, video_name)}.csv"
out_video_path =
f"{OUT_PATH}/{os.path.basename(VID_PATH).strip('.avi')}_yolo.avi"


while (1):
    ret, frame = vid.read()
    frame_count = vid.get(cv2.CAP_PROP_POS_FRAMES)
    if frame_count == 1:
        height, width, channels = frame.shape
        print(height, width)
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(out_video_path, fourcc, 10, (width, height), True)
    ToCSV = YoloToCSV(model, frame, frame_count)
    ToCSV.write_to_csv(csv_out_path)
    img_out_path =  f"{os.path.join(OUT_PATH, video_name)}_{frame_count}.png"
    ToCSV.draw_on_im(out_video_path,writer)

writer.release()
```

Worm-YOLO Processing

```
from __future__ import division

import os
import sys
import cv2
import csv
import tqdm
import pandas as pd
from sort.sort import *
import random
import numpy as np
import time
from matplotlib import pyplot as plt
from re import sub
from utils.utils import *
import fnmatch
import scipy.optimize
from skimage.metrics import structural_similarity as ssim
import time
from tqdm import tqdm




def analyzeSORT(df,threshold,slow_move,delta_overlap):
    vc = df.label.value_counts()
    test = vc[vc > threshold].index.tolist()
    deadboxes = []
    deathspots = []
```

```
for ID in test:
    filtval = df['label'] ==ID
    interim = df[filtval]
    interimD = []
    interim2 = np.asarray(interim)
    fill = 0
    deadcount = 0
    alivecount = 0
    isdead = 0
    x1E = interim['x1'].iloc[0]
    y1E = interim['y1'].iloc[0]
    x2E = interim['x2'].iloc[0]
    y2E = interim['y2'].iloc[0]
    for row in reversed(interim2):
        frameNA, x1A, y1A, x2A, y2A, labelA, deltaA, catagoryA, *_ = row
        if fill > 1:
            boxA = [x1A, y1A, x2A, y2A]
            boxB = [x1B, y1B, x2B, y2B]
            deltaA = bb_intersection_over_union(boxA, boxB)
            if deltaA > delta_overlap:
                deadcount+=abs(frameNA-frameNB)
            if deadcount > slow_move:
                if deadboxes == []:
                    deathspots.append([frameNA, x1A, y1A, x2A, y2A,labelA])
                    deadboxes.append([x1A, y1A, x2A, y2A])
                    isdead = 1
                    x1Z = x1A
                    y1Z = y1A
                    x2Z = x2A
                    y2Z = y2A
```

```
        else:
            notunique = 0
            for box in deadboxes:
                #print(box)
                x1D, y1D, x2D, y2D, *_ = box
                boxD = [x1D, y1D, x2D, y2D]
                deltaD = bb_intersection_over_union(boxA, boxD)
                if deltaD > 0.2:
                    notunique = 1
            if notunique == 0:
                deathspots.append([frameNA, x1A, y1A, x2A, y2A,labelA])
                deadboxes.append([x1A, y1A, x2A, y2A])
                isdead = 1
                x1Z = x1A
                y1Z = y1A
                x2Z = x2A
                y2Z = y2A
    if isdead==1 and deadcount > slow_move:
        boxA = [x1B, y1B, x2B, y2B]
        boxZ = [x1Z, y1Z, x2Z, y2Z]
        deltaZ = bb_intersection_over_union(boxA, boxZ)
        if deltaZ < 0.4:
            #alivecount+=abs(frameNA-frameNB)
            #print(frameNA,frameNB,deltaZ,deadcount,labelA)
            #if alivecount > 1:
            deadcount = 1
            deathspots =  deathspots[:-1]
            deadboxes = deadboxes[:-1]
            isdead=0
    #if deadcount > 5*slow_move:
```

```
                # print("broke cause of long",frameNA,frameNB,deadcount)
                 #break
            frameNB, x1B, y1B, x2B, y2B,labelB, deltaB, catagoryB, *_ = row
            fill +=1



    csv_outputs = pd.DataFrame(deathspots, columns = ['frame','x1A', 'y1A', 'x2A',
'y2A','labelA'])
    return(csv_outputs)



def bb_intersection_over_union(boxA, boxB):
    # determine the (x, y)-coordinates of the intersection rectangle
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    # compute the area of intersection rectangle
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)
    # compute the area of both the prediction and ground-truth
    # rectangles
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)
    # compute the intersection over union by taking the intersection
    # area and dividing it by the sum of prediction + ground-truth
    # areas - the interesection area
    if float(boxAArea + boxBArea - interArea) != 0:

        iou = interArea / float(boxAArea + boxBArea - interArea)
    else:
```

```
        iou=0
    # return the intersection over union value
    return iou


if __name__ == "__main__":

    CSV_FOLD_PATH = sys.argv[1] #folder of YOLO outputs
    OUT_FOLD_PATH = sys.argv[2]

    #ARGS
    threshold = 2 #number of frames a worm has to be tracked in order to be analyzed
    slow_move = 5 #number of frames overlapping by 'delta_overlap' before being called
dead or paralyzed (15ish=dead,5=paralyzed)
    delta_overlap = 0.8 #%overlap to be called motionless (.95 for dead, .8 for paralyzed
    max_frame = 5000

    csv_list = os.listdir(CSV_FOLD_PATH)
    csv_list = list(filter(lambda f: f.endswith('.csv'), csv_list))

    print(csv_list)
    csvindex = 0
    #loop through list of CSVs
    for csv_name in csv_list:
        csvindex +=1
```

```
    print("starting", csv_name,"which is ",csvindex,"of",len(csv_list))
    start_time = time.time()
    csv_PATH = os.path.join(CSV_FOLD_PATH, csv_name)
    OUT_PATH = os.path.join(OUT_FOLD_PATH, csv_name)

    #read csv and reformat
    df = pd.read_csv(csv_PATH,names=('frame', 'x1', 'y1', 'w','h','label'))
    df['x2']=df[['x1','w']].sum(axis=1)
    df['y2']=df[['y1','h']].sum(axis=1)
    df = df[['frame','x1', 'y1', 'x2', 'y2']]
    filtval = df['frame'] < max_frame
    df = df[filtval]

    unique = df["frame"].unique()

    #initialize sort tracker and create container
    mot_tracker1 =Sort(max_age=0, min_hits=0, iou_threshold=0.3)  #see SORT
documentation NEEDS TUNING
    sort_outputs = []
    print("sorting")

    #sort from end of experiment backwards
    for x in reversed(unique):
        frame = int(x)
        filtval = df['frame'] == x
        boxes_xyxy = np.asarray(df[filtval])[:,1:5]
        track_bbs_ids = mot_tracker1.update(boxes_xyxy)
        for output in track_bbs_ids:
            x1, y1, x2, y2, label, *_ = output
```

```
            sort_outputs.append([x.tolist(), x1.tolist(), y1.tolist(), x2.tolist(),y2.tolist(),label.tolist()])
        sort_outputs = pd.DataFrame(sort_outputs)
        sort_outputs.columns = ['frame','x1', 'y1', 'x2', 'y2','label']
        vc = sort_outputs.label.value_counts()
        test = vc[vc > 2].index.tolist()
        sort_outputs = sort_outputs[sort_outputs['label'].isin(test)]



        #create container dataframes for maximum and minimum frames for each label
        dfmin = pd.DataFrame(columns=['frame', 'x1', 'y1','x2','y2','label'])
        dfmax = pd.DataFrame(columns=['frame', 'x1', 'y1','x2','y2','label'])
        print("creating max/min arrays")
        #for each label, create max and minimum arrays. This is done dumbly feel free to improve via vectorization.
        uniqueTracks = sort_outputs["label"].unique()
        for track in uniqueTracks:
            filtval = sort_outputs['label'] == track
            trackdf = sort_outputs[filtval]
            max_value = trackdf['frame'].max()
            min_value = trackdf['frame'].min()
            trackmin = sort_outputs[(sort_outputs["label"]==track) & (sort_outputs["frame"]==min_value)]
            trackmax = sort_outputs[(sort_outputs["label"]==track) & (sort_outputs["frame"]==max_value)]
            dfmin = dfmin.append(trackmin, ignore_index = True)
            dfmax = dfmax.append(trackmax, ignore_index = True)


        #take necessary arrays and reformat all to correct data type (int)
        #also, remove any SORT label that was observed less than twice. This drastically lowers computation time.
```

```
#this filter could be removed if the linking step was more improved
sort2 = sort_outputs
vc = sort2.label.value_counts()
test = vc[vc > 2].index.tolist()
sort2 = sort2[sort2['label'].isin(test)]
sort2 = sort2.apply(pd.to_numeric)
sort2 = sort2.apply(np.int64)
uniqueTracks = sort2["label"].unique()
dfmax = dfmax.apply(pd.to_numeric)
dfmax = dfmax.apply(np.int64)
dfmin = dfmin.apply(pd.to_numeric)
dfmin = dfmin.apply(np.int64)

print("linking")




#this section is kinda a mess but it works
#goes from labels at the end of the experiment and progresses toward the beginning
#for a given label (labelmin)
#gets the miniumum (earliest) frame it was observed, then loops through the maxiumum frames of other labels
#if there is sufficient overlap between this min and a max, overwrites the 'label' value in all arrays of the max with the 'label' of the min
#if no overlap is found within a threshold of time (2 days curr), break loop and move to next frame

#needs tuning and iteration on time and overlap thresholds to figure out optimum
itersMin = 1
```

```
while itersMin < len(uniqueTracks):
    labelX = uniqueTracks[itersMin]
    #print(labelX)
    wormA = dfmin[dfmin['label'] == labelX]
    #print(wormA)
    frameA = wormA['frame'].iloc[-1]
    x1A = wormA['x1'].iloc[-1]
    y1A = wormA['y1'].iloc[-1]
    x2A = wormA['x2'].iloc[-1]
    y2A = wormA['y2'].iloc[-1]
    labelA = wormA['label'].iloc[-1]
    itersMin +=1
    itersMax = itersMin
    #print(len(uniqueTracks))
    while itersMax < len(uniqueTracks):
        uniqueTracks[uniqueTracks>labelA]
        labelY = uniqueTracks[itersMax]
        #print(labelY)
        wormB = dfmax[dfmax['label'] == labelY]
        frameB = wormB['frame'].iloc[-1]
        x1B = wormB['x1'].iloc[-1]
        y1B = wormB['y1'].iloc[-1]
        x2B = wormB['x2'].iloc[-1]
        y2B = wormB['y2'].iloc[-1]
        labelB = wormB['label'].iloc[-1]
        itersMax+=1
        if frameB < frameA and labelA != labelB:
            boxA = [x1A, y1A, x2A, y2A]
            boxB = [x1B, y1B, x2B, y2B]
            delta = bb_intersection_over_union(boxA, boxB)
```

```
            if abs(frameA-frameB)<5:
                deltathresh = 0.2
            elif abs(frameA-frameB)<36:
                deltathresh = 0.4
            else:
                deltathresh = 0.7
            if delta > deltathresh:
                #print('changing ',labelB,' to ',labelA,' at frame:',frameB)
                sort2 = sort2.replace({'label': labelB}, labelA)
                dfmax =dfmax.replace({'label': labelB}, labelA)
                dfmin = dfmin.replace({'label': labelB}, labelA)

                #dfmax = np.where(dfmax == labelB, labelA, dfmax)
                #dfmin = np.where(dfmin == labelB, labelA, dfmin)
                uniqueTracks[uniqueTracks==labelB]=labelA
                #maxOver.append(labelB)
                break
            if frameB < (frameA-144):
                break



    #print(sort2)
    #reformat output to be accepted into analyze sort
    csv_outputs = pd.DataFrame(sort2)
    csv_outputs.columns = ['frame', 'x1', 'y1', 'x2', 'y2','label']
    csv_outputs['delta'] = 0
    csv_outputs['catagory'] = 'alive'

    #analyze for death
```

```
outputs = analyzeSORT(csv_outputs,threshold,slow_move,delta_overlap)
#print(outputs)
outputs['expID'] = os.path.basename(csv_PATH).strip('.csv')

#export and move to next csv file
pd.DataFrame(outputs).to_csv(OUT_PATH, mode='w', header=True, index=None)
print('finished in:',time.time()-start_time,'seconds')
```