

Optimised coordinate generation for Arduino-based focus and motion control system for deployment in time-lapse photography

Johannes Jacobus Myburgh¹, Nicolaas Johannes Luwes²

¹Department of Design and Studio Art, Faculty of Humanities, Central University of Technology, Free State, Bloemfontein, South Africa

²Department of Electrical Engineering, Faculty of Engineering, Central University of Technology, Free State, Bloemfontein, South Africa

Article Info

Article history:

Received Nov 09, 2021

Revised Oct 27, 2022

Accepted Nov 12, 2022

Keywords:

Acceleration

Cinema

Motion control

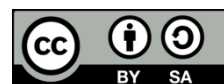
Photography

Time-lapse

ABSTRACT

This paper demonstrated the design and implementation of a unique algorithm used to generate interim coordinates for focus transitions in time-lapse photography. Through a prototype design methodology, the unique algorithm was developed to function within the limitations imposed by a low-cost 8-bit microcontroller for the development of a cost-effective focus and motion control system for deployment in photographic higher education. Through conceptualization and virtualization in MathLab and comparison with values generated in simulated real-world use-case, the proposed algorithm was tested for function and efficacy in generating interim coordinates that included focus coordinate estimation to facilitate smooth non-linear motion with focus transitions during image sequence capture. The 8-bit Arduino Mega 2560 proved capable of completing calculations required by the developed algorithm and generated coordinate data sets identical to sets generated in simulation. The generated coordinated illustrated effective user-defined control over motion and the smoothness of focus transitions. The developed algorithm is thus capable of facilitating the required movement and focus transition coordinate calculation and estimation required while deployed on a low-cost 8-bit microcontroller and is well suited for deployments in time-lapse capture applications.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Johannes Jacobus Myburgh

Department of Design and Studio Art, Faculty of Humanities, Central University of Technology

Free State 3 Pres. Brand Street, Willows, Bloemfontein, 9301, South Africa

Email: myburghj@cut.ac.za

1. INTRODUCTION

“I hear and I forget. I see and I remember. I do and I understand” is a Chinese proverb often echoed in education and historically attributed to scholars such as [1]. In the context of higher education (HE), hands-on practical experience is critical for deeper understanding and knowledge retention. This is even more so in technology-driven fields [2] such as photography, referring specifically to the artistic application of a camera for the capture of both still images and video. This theory has been well-researched and tactile engagement with the subject matter has proven to improve engagement and deeper cognitive processes [3]. Socio-economic issues currently prevalent in developing countries place enormous financial constraints on HE institutions [4], students and graduates alike [5]. This greatly impacts the ability of Universities of Technology (UoTs) to acquire sufficient equipment to allow the required volume of students adequate access needed to gain critical, hands-on practical experience with relevant equipment. Highly specialized tools also often command a premium price. These financial constraints further impact the ability of students and

graduates to purchase personal equipment that would allow them to strike out as entrepreneurs [6]. The shortage or lack of equipment thus has a direct impact on student preparedness and graduates' ability to effectively function in the workplace [7]. One such type of niche tool prominently utilised within the photographic industry is systems used to control the motion and focus of cameras during the capture of both video and still images [8]. One possible solution to the problem of financial constraint UoTs face is an open-source, low-cost focus and motion control system (FMCS), easily assembled from off-the-shelf parts that are predominantly used for prototyping using common system on a chip (SoC) boards such as an Arduino [7].

Low-cost 8-bit SoCs place specific limitations on the complexity of mathematical calculations possible in an expedient manner due to limited processing power and system resources [9]. To circumvent this limitation a unique algorithm that functions within these constraints needed to be developed. This algorithm is necessary to fulfil the requirement of coordinate calculation needed for an FMCS based on an Arduino Mega 2560 to be able to synchronise the movement of multiple stepper motors for the purpose of time-lapse sequence capture. Furthermore, the calculated coordinates need to create the perception of non-linear acceleration and deceleration in the resultant time-lapse video footage generated from captured still images. The non-linear acceleration and deceleration are required to render the resultant media with smooth transitions in focus or movement, which renders it more aesthetically pleasing and less jarring to the viewer [10]. S-curved, non-linear acceleration and deceleration are required for the FMCS to be deemed to perform at an adequate level to serve as a replacement for current commercial offerings [11].

2. LITERATURE AND RELEVANCE OF RESEARCH

Manual, mechanical systems for controlling the focus of photographic systems were first patented in 1917 [12] and subsequently introduced to the industry as a commercial product [13] in the early 1920s. Apparatuses for facilitating the movement of the camera during filming occurred coincidentally [14]. Early apparatuses such as dollies, tracks to slide a heavy cart with a camera mounted on it [14], or jibs, a crane-like device allowing for movement across both vertical and horizontal planes simultaneously [8], were exclusively manual implements [7]. A pioneering exploration of automating the movement of a camera was done by the ubiquitous production company, Industrial Light and Magic, during the production of Star Wars: A New Hope to address specific creative requirements posed by this film, and would forever change filmmaking [10]. Motion control in photography as we know it today is defined as an electromechanical system that allows for the physical movement of the camera and other objects to be digitally recorded, enabling identical successive passes to be photographed [8].

Access to FMCS' would remain exclusive to only the largest Hollywood production houses due to the exorbitant cost, complexity and sheer bulk of the complex tools needed to facilitate control and movement of the camera [11]. This would remain the status quo until the emergence of digital single lens reflex (DSLR) cameras that were capable of recording video shortly after the arrival of the 21st Century [15]. These cameras, which are referred to as video-capable digital single lens reflex cameras (VDSLR's), possessed professional-quality video recording capabilities, a minuscule form factor, and most importantly of all, an infinitesimal price tag when compared to traditional cinema cameras [16]. This revolutionary product created a need for small, lightweight, affordable FMCS' which the industry was quick to explore, with academic research attempting to do the same.

Though product development and prototyping has historically been an exceedingly costly process, it has become accessible to tinkerers and entrepreneurs through the advent and proliferation of open-source technologies, mass production thereof and the free sharing of ideas around product development as proliferated by the 4th industrial revolution [17]. Products such as the ubiquitous Arduino microcontroller have allowed legions of would-be inventors to start to develop new products at very low price points [18], with the Arduino Mega 2560 offering the best value for more complex projects [7]. The popularity of cost-effective do it yourself (DIY) additive manufacturing processes such as fused deposition modelling (FDM) has led to increased production and a subsequent decrease in the price of stepper motors such as the National Electrical Manufacturers Association (NEMA) 17-sized motor, which offers precise incremental control over mechanical motion at very affordable prices [7]. The popularity of FDM printing has had the same effect on the price and accessibility of aluminium extrusions used as both mechanical components and linear guides, as well as lead screws used to facilitate linear motion. The availability and cost-effectiveness of components such as these make them the ideal vehicles for developing a cost-effective FMCS.

The academic exploration of FMCS' has almost exclusively relied on the implementation of cost-effective prototyping tools such as SoCs and stepper motors for precise control of motion, but the vast majority of these implementations' academic research has been in areas outside of the artistic and creative application of photography. A plethora of implementations such as those by [19]-[21] utilize Arduino-based systems to control the camera, its focus and motion purely as a technical tool for documenting data. Examples

like the camera stabilizer by Bryan and Murray [22] or the motorized zoom control of Bagri and Flotov [23] do have potential artistic applications in photography, but these academic exploits are only rudimentary “proof of concept” papers that form part of undergraduate studies, with the projects never being refined to completion. The most definitive academic exploration in the application of SoCs like Arduino for creative application in photography is the works of Hoesl in collaboration with [11], [10], [24] which do mention the importance and impact of acceleration and deceleration during motion control for photography but do not offer insight into the process for facilitating this requirement.

Control over acceleration, travel speed and deceleration of the stepper motors used to facilitate control over the camera’s focus and movements are critical in conveying specific creative intent as well as ensuring the smooth and aesthetically pleasing resultant footage. The initial acceleration from a static position referred to as “ease-in”, and the deceleration at the end of travel, referred to as “ease-out”, fulfil the purpose of making focus adjustment and motion more subtle and less jarring to the viewer [10] as well as decreasing the mechanical forces on the drive system that can cause motion and vibration to become visible in the finale resultant. The speed, acceleration and deceleration of the motors facilitating change also offer the photographer control over the emotional and psychological reaction of the viewer to the resultant footage having a tremendous artistic impact on the footage [11]. This process of calculating the movement and its necessary vector changes is further complicated as multiple motors have to be deployed synchronously to facilitate the required motion and focus shift, with some motions needing to be continuous and linear while others will accelerate and decelerate to reach specific coordinates along the required travel path. Resources such as motor class implementations and Arduino libraries to handle motion implementations do exist, but they were developed to facilitate smooth continuous motion [25] which is more suited to video capture. The most noteworthy example of such an Arduino library is the AccelStepper library developed by McCauley [26]. No such resource could be found that would generate interim coordinates with acceleration for implementation in time-lapse sequence coordinate generation. A timelapse sequence consisting of multiple individual movements, with a still image captured between each movement and the resultant images combined and played back as a video sequence to illustrate the rapid passing of time. This task required an algorithm simple enough to be performed within the limited system resources of a low-cost 8-bit microprocessor, while still being able to expediently generate an s-curved travel path. The sigmoid function proved to be the ideal base for this calculation and this specific implementation.

The base sigmoid function to define a point on a curve is defined [26] and visualized in Figure 1. This function’s angle of incline, midpoint and maximum curvature can be manipulated to result in a variety of coordinate point generation shifts for implementation in time-lapse sequence capture that ranges from linear travel with no acceleration or deceleration, to a time-lapse with a pronounced S-shaped travel curve with equally pronounced acceleration and deceleration.

$$f(x) = \frac{L}{1+e^{-k(x-xmid)}} \quad (1)$$

Where, $xmid$ is the midpoint for x ; L is the curve’s maximum value; k is the steepness of the curve.

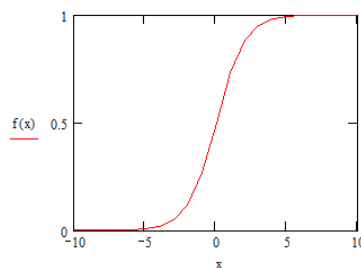


Figure 1. Base sigmoid as calculated with $xmid = 0$, $L = 1$ and $k = 1$

3. RESEARCH METHOD

This section outlines the construction and operation of the prototype low-cost FMCS that can be used as a testbed to assess the function and visual impact of the coordinate generation algorithm on the final resultant time-lapse footage. This is followed by a discussion of the prototype design methodology [27] applied via conceptualization and virtualization used in developing the required interim coordinate generation algorithm. This is accomplished through documenting the development and testing of the position generation algorithm and its implementation for coordinate generation in facilitating transition focus estimation in a simulated and real-world use case implementation.

3.1. Construction and operation of the low-cost FMCS

A low-cost FMCS prototype was constructed using an Arduino Mega 2560, DRV8825 stepper motors drivers, other miscellaneous electrical components and 2 stepper motors as illustrated in Figure 2 and Figure 3 on the following page. One NEMA 17 stepper motor will facilitate moving the camera on a gantry via a TR8×8 lead screw drive mechanism along a 1000 mm long V-slot C-beam which will serve as a short dolly track resulting in approximately 850 mm of total travel. A second NEMA 17 motor is mounted to the camera via a cinema industry-standard 15 mm rail support system and engages the lens via a similarly standard focus drive gear with a modulus of 0.8 using parts designed by the authors and produced via a fused deposition modeling printer. This low-cost FMCS prototype was used to facilitate real-world testing of the coordinate generation algorithm allowing for one degree of motion of the camera along with focus control which required the calculation of a 2-axis coordinate system.



Figure 2. Physical prototype testing (by the authors)

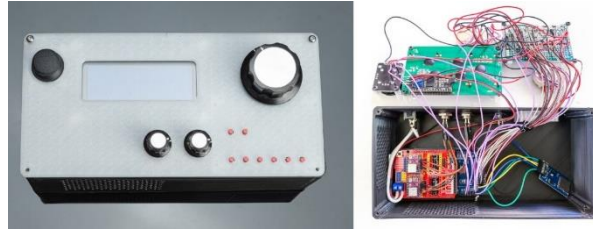


Figure 3. External view and internal wiring of FMCS control unit prototype (by the authors)

The FMCS control unit would allow the user to define maximum travel extents for both motors (defined as NEMA17max, NEMA17min, focusmax, focusmin in later calculations) as well as between two and six sequential coordinate positions for the two axes of movement by allowing the user to jog the motors into position and storing the current position to the control unit memory via dedicated buttons for later recall. The prototype control unit in Figure 3 and the accompanying pins assignment diagram in Table 1 illustrate how this level of input is achieved. These user-defined coordinates are referred to as the start point (pmin), point 1 through point 5 and maximum travel point (pmax) in this text. The user-defined coordinates are then stored for later use via two user-defined capture modes. The first is capturing the progression of the camera through these sequential coordinates as one continuous sequence of motion for capturing a video sequence at a user-defined speed. The second option is a time-lapse sequence breaking up the sequence into multiple individual moves with the control unit triggering the camera to capture a still image at user-defined intervals between each movement. The FMCS interfaces with the camera, a Nikon D750 equipped with a Nikon AF-S 50 mm F1.8 G lens through a proprietary connector cable (Nikon MD-DC2) modified with a male 3.5 mm stereo jack plug to interface with the FMCS.

Table 1. Functions assigned to the Arduino Mega 2560 pins (by the authors)

Analogue pin assignment			
10	Joystick 1 - X R		
11	Joystick 1 - Y R		
12	Slide speed pot		
13	Focus speed pot		
14	Focus control pot		
Digital pin assignment			
20	Display data	21	Display clock
30	Joystick 2 button (unassigned)	31	Camera trigger optocoupler
32	Joystick 1 button (menu selection joystick)	33	External input optocoupler
34	Waypoint 6 button	35	Waypoint 6 light emitting diode (LED)
36	Waypoint 5 button	37	Waypoint 5 light emitting diode
38	Waypoint 4 button	39	Waypoint 4 light emitting diode
40	Waypoint 3 button	41	Waypoint 3 light emitting diode
42	Waypoint 2 button	43	Waypoint 2 light emitting diode
44	Waypoint 1 button	45	Waypoint 1 light emitting diode
46	Hard max button	47	Hard max light emitting diode
48	Hard min button	49	Hard min light emitting diode
50	Secure digital card reader master in slave out (MISO)	SPI	Secure digital card reader master out slave in (MOSI)
52	Secure digital card reader system clock (SCK)		

The FMCS triggers the camera by connecting the signal wire of this connection to the ground wire through the use of a Vishay 4N25 optocoupler. It is in this capture mode that the user-defined coordinate points are used in conjunction with the interim coordinate generation algorithm. This time-lapse function further allows for two more degrees of user input. The first allows the user to define the total number of images to be captured during the sequence (*ShotsRequired*), which directly influences the number of individual movements and images that should make up the complete sequence. The second allows the user to define a percentage variable (*RampPercentage*) that determines how much of the total camera travel should include an initial non-linear acceleration of the camera motion in the final video sequence (5%–50%), or whether the sequence should be performed as one movement at one continuous speed (0%).

3.2. Position coordinate generation

As stated in the previous section, the interim coordinate point generation needs to be calculated from the specific user-defined start, four intermediate and the end coordinates for both axes, at which to stop the movement and trigger the camera to capture an image that forms part of the time-lapse sequence. The operational design specification is that the user must be able to define varying levels of initial acceleration for the motion axis which alters the movement from linear travel to smooth, non-linear acceleration before achieving user-defined travel speed. This function to achieve this *X*-axis movement is achieved by manipulating the base sigmoid function from (1) to:

$$Steps(a) = \left[\left[\frac{1}{1+e^{-c1(a-c2)}} \right] * Nema17Point - \left[\frac{1}{1+e^{-c1(0-c2)}} \right] * Nema17maxPoint \right] * \frac{Nema17maxPoint}{\left[\left[\frac{1}{1+e^{-c1(ShotsRequired-c2)}} \right] * Nema17maxPoint - \left[\frac{1}{1+e^{-c1(0-c2)}} \right] * Nema17maxPoint \right]} \tag{2}$$

Where *a* is the number of required images that influence the number of interim coordinates required (from 0 to total images needed); *steps* is the coordinate points calculated; *NEMA17maxPoint* is the stored max position; *c2* is total images needed + 100 (to get the shape required); *c1* is 0.16 times a *RampPercentage* divided by total images needed; *RampPercentage* is user-defined and is an indicator defining the shift from linear to initially accelerated motion. This function is rounded down as *steps(a)* from *steps(a)*, with an example illustrating *RampPercentage* = 1.

Figure 5 illustrates the linear relationship between the *X*-axis and the *Y*-axis. An example illustrating motion with the user-defined *RampPercentage* = 50 is illustrated in Figure 6. This comparison is made to demonstrate the difference between the extremes of linear progression to the maximum achievable accelerated motion as defined by this implementation. This function is thus an auto-calibration algorithm that generates a defined number of travel coordinates and the linearity of the acceleration/movement curve as specified by the user. Figure 6 illustrates the contrasting, non-linear relationship between the *X*-axis and *Y*-axis after the modified sigmoid function has been applied.

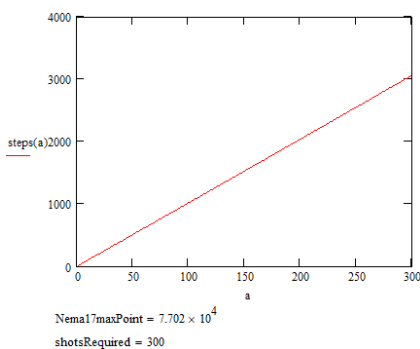


Figure 5. Custom coordinate position generator based on the sigmoid curve with *RampPercentage* = 1

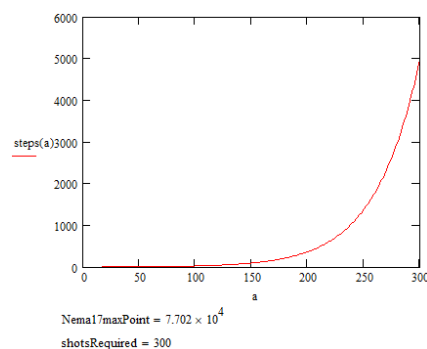


Figure 6. Custom coordinate position generator algorithm based on sigmoid with *RampPercentage* = 50

3.3. Focus estimation

As noted in section 3.1 there are user-defined coordinates set along the *X*-axis which must synchronously coincide with its defined *Y* coordinate in the sequence. The placement of these coordinates within the sequence will be influenced by the linearity of the modified sigmoid function as noted in Figure 6.

Focus transition coordinate estimation is thus a much more complex function influenced by the RampPercentage and this function needs to be designed, simulated, tested and verified for accuracy. The user-defined focus distance coordinates (Y -axis), that coincide with 6 specific coordinates during the progression of the movement axis (X -Axis) need to be considered in this design as their position as calculated in the previous section will no longer be accurate in relation to the X -axis and differ from the user-defined position in the sequence. This will then require the calculation of new estimated focus distance coordinate values to accommodate this shift. The larger the ramp percentage, the more complex the output generations calculation could be for smooth focus transitions. A quantum physics approach was applied, where each coordinate point is given its own “gravity” that influences the priority of its focus set value as a distance from the set value to the calculated value on the movement axes. Phrased differently, the calculated coordinate point’s distance is calculated relative to each defined coordinate and the closer the value gets to the defined coordinate, the more it must use that defined coordinate focus value to transition over to it. This led to multi-level calculation. The first level is the “gravity” calculation for each point. Below is the algorithm for the first set point.

$$movemin(a) = \begin{cases} 100 \left[1 - \left(\frac{steps(a)-pmin}{pmax} \right) \right] & \text{if } 100 \left[1 - \left(\frac{steps(a)-pmin}{pmax} \right) \right] x \leq 100 \\ 100 + \left[100 - \left[100 \left[1 - \left(\frac{steps(a)-pmin}{pmax} \right) \right] \right] \right] & \text{otherwise} \end{cases} \quad (3)$$

Where, $movemin(a)$ is the calculated influence or “gravity” as a percentage towards each coordinate calculated in (2) in relation to the last point ($pmax$); $steps(a)$ is each coordinate calculated in (2); $pmin$ is the position of the first user set coordinate; $pmax$ is the last user-defined coordinate. In other words, the further the calculated time-lapse position moves away from the user-defined coordinate $pmin$, the smaller the percentage of its influence. This function is repeated for all 6 user-defined coordinates calculating each one’s influence on the steps coordinate calculation, with the influence decreasing as distance increases as per Newton’s inverse square law [28]. The next layer of calculation is to determine what user coordinate is the closest to the calculated time-lapse point ($steps(a)$). If for instance, user point 2 is closer to the $steps(a)$ point than user point 1, then user point 2 focus set value must be used as a factor of distance from that point 2. See the (4) for point 2 (that is repeated for all the 6 user points).

$$postmove2(a) = \begin{cases} move2(a) & \text{if } move1(a) < move2(a) > move3(a) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Where, $postmove2(a)$ is the focus priority for point 2 (labelled as min, 1, 3, and 4); $move -$ is the value calculated with (3) (labelled as min, 1, 3, and 4); a is the interim coordinate value from 0 to shots required. There are more rules, with a precalculated layer that influences the final “gravity” of each coordinate. It might be clearer to start with a later layer and discuss the layers that were precalculated. See point 2’s later layer (5) (that is repeated for all the user points min, 1, 2, 3, 4, 5 and max).

$$pp2(a) = \begin{cases} postmove2(a) & \text{if } p1F < p2F > p3F \\ postmove2100(a) & \text{if } p1F \geq p2F \leq p3F \\ ppp(a) & \text{if } p1F < p2F \leq p3F \\ postmove2100(a) & \text{otherwise} \end{cases} \quad (5)$$

Where, $pp2(a)$ is the “gravity” of point 2, $p'n'F$ is the focus value of coordinate ‘ n ’ user-defined to $p'n'$; $postmove2$ is calculated as in (4) (where $postmove3$ would be for $pp3$. If function is repeated for other points; $postmove2100$ is:

$$postmove2(a) = \begin{cases} move2(a) & \text{if } move1(a) < move2(a) > move3(a) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$ppp2$ is:

$$ppp2(a) = \begin{cases} pper2 & \text{if } 0 < pppp2 < pper2 \\ pppp2(a) & \text{otherwise} \end{cases} \quad (7)$$

Where, $pppp2$ is:

$$pppp2(a) = \begin{cases} 100 & \text{if } postmove2(a) > postmove2(a-1) \\ postmove2(a) & \text{otherwise} \end{cases} \quad (8)$$

$$cp2(a) = \begin{cases} cper2 & \text{if } 0 < cpp2 < cper2 \\ cpp2(a) & \text{otherwise} \end{cases} \quad (9)$$

$cp2$ is:

$$cp2(a) = \begin{cases} cper2 & \text{if } 0 < cpp2(a) < cper2 \\ cpp2(a) & \text{otherwise} \end{cases} \quad (10)$$

Where, $cper2$ is:

$$cper2 = \begin{cases} \left\lceil \frac{p1F}{p2F} \right\rceil * 100 & \text{if } 100 < \frac{p2F}{p1F} * 100 \\ \left\lfloor \frac{p2F}{p1F} * 100 \right\rfloor & \text{otherwise} \end{cases} \quad (11)$$

The following (5) is thus a mathematical rule set that selects between the “gravity” calculations as seen from (6) to (10). This is repeated with corresponding values for each of the user-defined coordinates. This ruleset ensures that if the focus value of the previous coordinate is the same as the destination coordinate, no gravity coefficient less than 100 must be used which would have resulted in a dip in the focus transition. In other words, if the two values are the same then there must be no focus transitions. Note that there are five such rules in (5). This equation, which is repeated for each point with their corresponding sub-layers, is then applied to calculate 6 focus ranges as (only focusmin is shown).

$$focusmin(a) = pminF * \frac{ppmin(a)}{100} \quad (12)$$

Where, focus ‘ n ’ is the range value, p ‘ n ’ is the focus value for point ‘ n ’, pp ‘ n ’ is (5) (or similar depending on ‘ n ’). The final range is then calculated (13).

$$final(a) = focusmin(a) + focus1(a) + focus2(a) + focus3(a) + focus4(a) + focusmax(a) \quad (13)$$

A user-defined smoothing filter is applied using averaging.

$$finalave3(a) = \frac{finalave(a) + finalave2(a)}{2} \quad (14)$$

Where, $finalave3$ is the output of filter $finalave$ is:

$$finalave(a) = \frac{\sum_{i=0}^{(filter)} final(a+i)}{filter+1} \quad (15)$$

Where, $finalave$ is the average with a shift in one direction; filter is the filter ratio defined by user; $final$ is calculated from (12); $finalave2$ is:

$$finalave2(a) = \frac{\sum_{i=0}^{(filter)} final(a-i)}{filter+1} \quad (16)$$

Where, $finalave2$ is the average with a shift to the opposite direction; filter is the filter ratio defined by the user; $final$ is calculated from (12). These equations need to be translated into C++ language for compatibility with Arduino. For example, (2) is translated:

$$\begin{aligned} long \ StepPosition = & (((1/(1 + pow(e, (-1) * (c1) * (a - c2)))) * Nema17maxPoint - \\ & ((1)/(1 + pow(e, (-1) * (c1) * (0 - c2))) * Nema17maxPoint)) * \\ & (Nema17maxPoint / (((1)/(1 + pow(e, (-1) * (c1) * (ShotsRequired - c2)))) * \\ & Nema17maxPoint - (1)/(1 + pow(e, (-1) * (c1) * (0 - c2))) * Nema17maxPoint)) \end{aligned} \quad (17)$$

It is important to note the memory required to complete the above calculations for a sufficient number of required images to make the FMCS viable for time-lapse implementation exceeds the specifications of the Arduino Mega 2560. This is overcome by writing the required data sets to an external micro-SD card module via SPI before each calculation. Thus, when this function is called, the (2) is calculated coordinate by coordinate and each value is stored individually on the SD card in a “StepPosion.txt” file. Thereafter each layer calculation is done individually by generating interim coordinates and stored sequentially for the focus transition calculations in a similar manner and written to the micro-SD card.

The same procedure is applied to the filter. Once the required calculation function has been completed and the data set containing the resultant X -axis values with RampPercentage applied and Y -Axis values with focus coordinate estimation applied have been written to the micro-SD card, the FMCS can perform the number of individual movements as defined by the value ShotsRequired from the calculated values stored in the corresponding “.txt” file on the SD card. McCauley’s AccelStepper library, as noted in the literature section, is then utilized to facilitate the required movements in real time, with the corresponding image captures between each movement.

4. RESULTS AND DISCUSSION

The results are generated in the form of a simulated use-case scenario with user-defined coordinate points that approximate an average FMCS deployment. The FMCS system was jogged via user input and the required 6 coordinate points were saved as described in the methodology section. The Arduino Mega 2560-based FMCS was then allowed to calculate the X -axis coordinates with RampPercentage and Y -axis with focus transition estimation applied for a 300-image sequence on the micro-SD card. The original six user-defined coordinates and the calculated coordinate values were then retrieved from the storage media. The user-defined coordinate values were then entered into the simulation in MathLab and the resultant calculated coordinates from the simulation were compared to the coordinates generated by the Arduino Mega 2560. The six user-defined coordinates are tabulated in Table 2 with the user-defined capture settings available in Table 2.

The coordinates values from point (a) of the position calculation from (5) as performed in the simulation (results seen in Figure 6 as Figure 6 had the same range) resulted in identical coordinate values when compared to the coordinates from point (a) as calculated by the Arduino Mega 2560 and stored on the SD. The focus estimation calculation (final(a)). Figure 7 visually illustrates the focus values for the required 300 image capture events from Table 2 and Table 3 with applied gravity of the equivalent coordinate and time-lapse coordinate calculation. Mathematically this expression is accurate, but visually the sharp transitions would result in abrupt movements of the FMCS motors that would be aesthetically displeasing. Thus, the need for filter calculation is observed. The filter results are as.

Figure 8 illustrates the original, pre-filter focus value and the focus value with filtered output applied for the required 300 images captures events. With the filter percentage influencing averaging and thus the extent of smoothing of the resultant curve. As stated at the start of this section the Arduino Mega 2560 based FMCS was used to generate both the user-defined set of six coordinates and the resultant set of 300 coordinates to test the Arduino’s ability to utilise the proposed algorithms. The pre-filter and filtered values generated by the Arduino Mega 2560 matched exactly with the above pre-filter coordinate values and filtered coordinate values as generated by the Arduino Mega 2560 and recorded on the SD card in the corresponding “.txt” file as were generated in the simulation in MathLab. This observation lends credence to the accuracy of the translation of the required calculation to one compatible with the C++ language as well as the efficacy of the algorithm for deployment on Arduino’s limited 8-bit processor.

It must be noted that the filter and focus estimation calculations are made more time-intensive due to the limited system resources of the Arduino and the necessity of writing relatively large data sets to the Micro-SD card. This phenomenon is impacted by the user-defined filter percentage and the number of images in the sequence. The greater the filter percentage or number of images to be captured in the sequence, the more time is required for pre-filter calculation. The observed time required for a 20% filter percentage calculation was approximately 1.3 seconds per image/coordinate point. Fortunately, this calculation occurs prior to sequence capture and once the calculation is complete the sequence is captured in real-time.

Table 2. Point names and user-defined coordinate positions

Point name	X -Axis	Y -Axis
	Motion coordinate value	Focus coordinate value
pmin	0	1453
p1	6800	1833
p2	15410	2125
p3	43010	2324
p4	61000	1735
pmax	77020	2738

Table 3. User-defined capture setting values

Variable	Value
ShotsRequired	300
RampPercentage	50
Filter	10

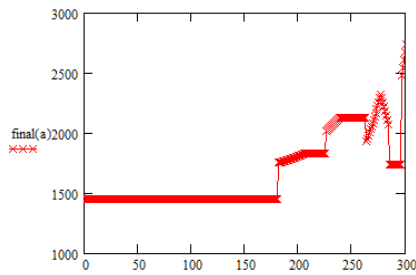


Figure 7. Final(a) calculation

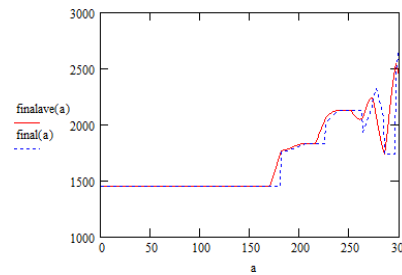


Figure 8. Pre-filter and averaging filter output

5. CONCLUSION

This paper aimed to demonstrate the development and results of an algorithm and implementation method to generate coordinates used for focus transitions during focus and motion control applications in time-lapse photography when deployed on a cost-effective 8-bit SoC. The proposed algorithm utilised a modified sigmoid function to create an auto-calibration algorithm combined with a quantum physics-based gravity calculation to expediently generate coordinate points that included position estimation for focus transitions. The point-by-point calculation procedure was implemented on an 8-bit Arduino Mega 2560. While implemented on this low-cost SoC, the algorithm facilitated user control over the degree of acceleration of camera motion as well the acceleration and deceleration curve of focus transitions. This allowed for the smooth, accurate and synchronous control of two degrees of motion in real time.

The functionality and efficacy of the algorithm were assessed by comparing the coordinate values of a simulated use case created from user input for a capture sequence consisting of six user-defined coordinate points resulting in 300 generated coordinate points as calculated by the Arduino Mega 2560. These coordinates were compared with a coordinate set generated by simulation in MathLab using the same user-defined variables. The curve profile of the generated coordinates was then compared to the profile of the coordinates with linear motion not containing interim focus point estimation and assessed for smoothness.

The 8-bit processor proved capable of completing calculations required by the developed algorithm and generated coordinate values identical to those values generated in the simulation. The generated coordinated illustrated effective user-defined control over the acceleration and deceleration resulting in travel. The developed algorithm is thus capable of facilitating the required focus transition coordinate estimation required while deployed on a low-cost 8-bit Arduino so that it might serve as the main component of a low-cost FMCS for deployment in HE. This low-cost FMCS has the potential to greatly impact HE institutions' ability to offer a more engaging and effective teaching experience to their graduates.

The developed algorithm was optimised for expedient deployment on an 8-bit system with the byproduct of the simplified calculation being large data. These large data sets exceeded the onboard memory of the Arduino Mega 2560 and required being written to supplemental storage. This limitation does leave room to explore further optimisation in deployment to decrease data set size or optimisation for compatibility with other SoC on the market that might offer larger onboard memory sizes or greater processing power at similarly low-price points.





REFERENCES

- [1] J. Needham, *Science and Civilisation in China: History of Scientific thought*, Cambridge, UK: Cambridge University Press, 1956, vol. II. [Online]. Available: <https://www.cambridge.org/id/academic/subjects/history/history-science-and-technology/science-and-civilisation-china-volume-2?format=HB&isbn=9780521058001>
- [2] L. Meda and A. J. Swart, "Graduate attributes in an electrical engineering curriculum: A case study," *International Journal of Engineering Education*, vol. 33, no. 2, pp. 653–661, 2017. [Online]. Available: https://www.researchgate.net/publication/317426930_Graduate_Attributes_in_an_Electrical_Engineering_Curriculum_A_Case_Study
- [3] A. Korbach, P. Ginns, R. Brünken, and B. Park, "Should learners use their hands for learning? Results from an eye-tracking study," *Journal Computer Assisted Learning*, vol. 36, no. 1, pp. 102–113, 2020, doi: 10.1111/jcal.12396.
- [4] A. Kraak and G. Hall, *Transforming Further Education and Training in South Africa, A Case Study of Technical Colleges in KwaZulu-Natal, Volume One: Qualitative Findings and Analysis*, vol. 1, 1999. [Online]. Available: <https://www.hsrcpress.ac.za/books/transforming-further-education-and-training-in-south-africa>
- [5] G. W. -Ouma, "Tuition fees and the challenge of making higher education a popular commodity in South Africa," *Higher Education*, vol. 64, pp. 831–844, 2012, doi: 10.1007/s10734-012-9531-6.
- [6] S. Cassim, P. Soni, and A. M. Karodia, "Entrepreneurship Policy in South Africa," *Arabian Journal of Business and Management Review*, vol. 3, no. 10, pp. 29–43, 2014, doi: 10.12816/0016498.
- [7] J. Myburgh, A. J. Swart, and L. V. Heerden, "Development of a Drive Unit for a Cost-effective Follow-focus Control System," *PalArch's Journal Archaeology Egypt/Egyptology*, vol. 17, no. 9, pp. 6270–6283, 2020. [Online]. Available: <https://archives.palarch.nl/index.php/jae/article/view/5185/5113>
- [8] M. Goi, *American Cinematographer Manual*, 10th ed, Hollywood, California, USA: The ASC Press, 2013. [Online]. Available: <http://www.gouastudio.com/amcinman.pdf>





- [9] J. C. Cluley, *An Introduction to Low Level Programming for Microprocessors*, Macmillan Education, 1987. [Online]. Available: https://books.google.co.id/books/about/An_Introduction_to_Low_Level_Programming.html?id=PElyAAAACAAJ&redir_esc=y
- [10] A. Hoesl, "Understanding and Designing for Control in Camera Operation," Dissertation, Fakultät für Mathematik, Informatik und Statistik, Ludwig-Maximilians-Universität München, München, Germany, 2019. [Online]. Available: https://edoc.ub.uni-muenchen.de/25100/1/Hoesl_Axel.pdf
- [11] A. Hoesl, P. Mörwald, P. Burgdorf, E. Dreßler, and A. Butz, "You' ve Got the Moves, We' ve Got the Motion – Understanding and Designing for Cinematographic Camera Motion Control," *IFIP Conference on Human-Computer Interaction, Human-Computer Interact. - INTERACT 2017*, 2017, vol. 10513, pp. 523–541, doi: 10.1007/978-3-319-67744-6_33.
- [12] J. E. Leonard, *Finder in combination with a camera shifting mechanism for focusing*. 1917, Accessed: Nov. 10, 2022. [Online]. Available: <https://patentimages.storage.googleapis.com/42/e6/ad/f92de04493c6c7/US1297704.pdf>
- [13] Intergovernmental Panel on Climate Change, "Summary for Policymakers," In *Climate Change 2013 – The Physical Science Basis: Working Group I Contribution to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, pp. 1-30, 2014, doi: 10.1017/CBO9781107415324.004.
- [14] M. Winokur and B. W. Holsinger, *The Complete Idiot's Guide to Movies, Flicks, and Film*, in Alpha Books, Indianapolis, 1st ed. 2001. [Online]. Available: <https://www.worldcat.org/title/complete-idiots-guide-to-movies-flicks-and-film/oclc/56408457>
- [15] P. Nuska, "The DSLR Revolution and its Impact on Documentary and Ethnographic Filmmaking," *Visual Ethnography*, vol. 7, no. 2, 2018, doi: 10.12835/ve2018.1-0111.
- [16] P. Nuska, "DSLR Video and its Impact on the Conventions in Filmmaking," Master thesis, Charles University in Prague, Faculty of Arts, Institute of Information Studies and Librarianship, 2015. [Online]. Available: <https://dspace.cuni.cz/bitstream/handle/20.500.11956/2714/120196110.pdf?sequence=1&isAllowed=y>
- [17] E. J. V. Holm, "Makerspaces and Contributions to Entrepreneurship," *Procedia-Social Behavioral Sciences*, vol. 195, pp. 24–31, 2015, doi: 10.1016/j.sbspro.2015.06.167.
- [18] C. Severance, "Massimo Banzi: Building Arduino," *Computer*, vol. 47, no.1, pp. 11-12, 2014, doi: 10.1109/MC.2014.19.
- [19] J. I. Deegan and T. Deegan, "Macrophotography of Fern Gametophytes Using a Focus Stacking System," *Pterodologist*, vol. 5, no. 5, pp. 357–360, 2018. [Online]. Available: <https://www.repository.cam.ac.uk/bitstream/handle/1810/286230/macrophotography%20low%20res.pdf?sequence=1&isAllowed=y>
- [20] C. Chen, P. -C. Chen, J. -P. Jang, Y. -C. Hsu and C. -C. Wang, "Development of an Auto Focusing Stacker for Photomicrography of Marine Microorganisms," *2019 IEEE Underwater Technology (UT)*, 2019, pp. 1-5, doi: 10.1109/UT.2019.8734300.
- [21] P. E. Hertzog and A. J. Swart, "Pigeon presence on PV modules ARE largely random events," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 9, pp. 524–529, 2019, doi: 10.35940/ijitee.I1110.0789S219.
- [22] N. Bryan, S. Chappell, and D. Williams, "Camera Stabilizer," *EE 462 Report, Electrical Engineering Department, California Polytechnic State University, San Luis Obispo*, 2016 Accessed: Nov. 03 2022. [Online]. Available: <https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1369&context=eesp>
- [23] D. Bagri and M. V. Flotow, "Motorized Zoom Control," Project 1314, Engineering Physics 459, Engineering Physics Project Laboratory The University of British Columbia," 2013, Accessed: Nov. 03, 2022. [Online]. Available: <https://open.library.ubc.ca/media/stream/pdf/52966/1.0074484/1>
- [24] A. Hoesl, J. Wagner, and A. Butz, "Delegation impossible?: Towards novel interfaces for camera motion," *CHI EA '15: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 1729–1734, 2015, doi: 10.1145/2702613.2732904.
- [25] M. McCauley, "AccelStepper: AccelStepper library for Arduino," 2017, Accessed: Oct. 09, 2021. [Online]. Available: <https://www.airspayce.com/mikem/arduino/AccelStepper/>
- [26] M.T. Tommiska, Efficient digital implementation of the sigmoid function for reprogrammable logic, *IEE Proceedings-Computers and Digital Techniques*, 2003, vol. 150, no. 6, pp. 403-411, doi: 10.1049/ip-cdt:20030965.
- [27] S. Wensveen and B. Matthews, *Prototypes and prototyping in design research*, Taylor & Francis Group, no. 1, pp. 262–276, 2014, doi: 10.4324/9781315758466-25.
- [28] M. H. Weik, "Inverse square law," in *Computer Science and Communications Dictionary*, pp. 834–834, 2017, doi: 10.1007/1-4020-0613-6_9566.

BIOGRAPHIES OF AUTHORS



Johannes Jacobus Myburgh     Jano is a lecturer in photography, CAD and CAM at the Central University of Technology, Free State. Jano is completing an Masters degree in Art with a specialization in Photography focusing on the development of easily accessible, cost-effective focus and motion control tools for photography. He can be contacted at email: myburghj@cut.ac.za.



Nicolaas Johannes Luwes     Nicolaas is an associate professor in the Department of Electrical, Electronic and Computer Engineering. He is an Activity Leader in the centre for sustainable smart cities. He can be contacted at email: nluwes@cut.ac.za.