A mobile platform for movement tracking based on a fast-execution-time optical-flow algorithm

Rafael de la Rosa-Vidal, Juan A. Leñero-Bardallo, *Member, IEEE*, José-María Guerrero-Rodríguez, and Ángel Rodríguez-Vázquez, *Life Fellow, IEEE*

***Abstract*—A multi-purpose mechanical platform to track moving objects in three-dimensional space has been developed. It is composed of one main microcontroller board that processes all system data, two cameras, three motors, and one secondary microcontroller board to position a platform with three degrees of freedom. The system computes the optical flow and moves the cameras accordingly, tracking motion within the visual scene. The platform operates autonomously. To the best of our knowledge, there are no similar systems reported with low-resolution image sensors and low-cost microcontrollers. Existing solutions rely on personal computers and advanced FPGAs to process image data. This article concludes that the optical flow operation is efficient even using an image sensor with very low resolution. Thus, the system complexity and image data processing are alleviated significantly. The platform can be easily adapted to different application scenarios by adding new peripherals, sensors, or image processing algorithms. A detailed description of the system design and experimental results are provided.**

***Index Terms*—Mobile platform, object tracking, camera positioning, optical flow, image processing.**

I. Introduction

MULTIPLE applications require to control the position of cameras to track moving objects. For instance, drone vision [1], [2] or automotive systems [3], [4] may require to stabilize the camera position to record static images of moving objects. Several authors have also devised sensors and instruments for space navigation that need precise positioning when they navigate. For instance, solar sensors or star trackers have to be continuously oriented directly to the sun or stars to control the attitude of space navigation systems [5], [6].

In parallel, the rise of Artificial Intelligence (AI) has developed many other vision systems [4], [7]–[9] that, processing camera data outputs, can identify moving objects or classify elements in the visual scene. Once the relevant scene visual elements have been identified, the corresponding actuators can be activated to navigate or exchange data with other systems.

Camera positioning to track mobile objects is a complex problem whose difficulty increases exponentially when there is more than one image sensor involved in the operation [10]. Since image quality is not a must in these scenarios where fast and efficient object recognition is the priority, different image sensor architectures are being investigated. Nowadays the spread of event-driven asynchronous image sensors [11], [12] open enticing possibilities to refine the operation by reducing the computational cost [13].

Looking at nature for inspiration, biological organisms, with single or compound eyes, can track moving objects by identifying the optical flow variations. The concept of optical flow detection to estimate movement within the visual scene by biological organisms was proposed in the 1940s by Gibson, [14]. Afterward, many algorithms to compute it were developed by many authors [15]–[18]. However, classical optical flow algorithms have inherently high computational cost because they require processing and storing multiple image frames [16]–[18]. For this reason, real-time execution needs high-speed processing that would limit their implementation in many scenarios that demand low-power systems.

To the best of our knowledge, existing tracking systems based on optical flow computation or movement detection have been implemented on high-performance platforms, using personal computers, utilizing high-level programs to process high image data throughput [1], [3], [10], [19]. That is a limitation to deploy positioning systems on other autonomous systems like drones, satellites, robots, etc. that cannot cope with a high payload and subsystems with high-power consumption. To solve these limitations, we have implemented a compact platform with reduced payload and power consumption. Furthermore, previous authors employ medium or high-resolution arrays to improve the image quality, at the expense of increasing the computational cost. However, as will be discussed in the article, for an optimum scene interpretation, it is not strictly necessary a large number of pixels. Some biological organisms like dragonflies have complex eyes compounded of simpler eyes that capture only a portion of the visual scene [19]–[21].

Previously, the authors presented a live demo with a optical flow tracker [22]. It was an early version of the proposed system with one image sensor, one movement

This work was supported by Proyectos de I+D+i DE entidades públicas – Convocatoria 2020 P20_01206 (VERSO), by Ayudas a Proyectos de I+D+I Programa Operativo FEDER through Project US-1264940 (SPADARCH), by Proyecto Singular de Transferencia del Conocimiento: Ecosistema Innovador con Inteligencia Artificial para Andalucía 2025 RIS3 through Project CEI-07, by Spanish Government MINECO and European Regional Development Fund, (ERDF/FEDER) through Project RTI2018-097088-B-C31, and by ONR grant ONR NICOP N00014-19-1-2156.
Rafael de la Rosa-Vidal was supported by the Spanish Government through Ayudas para la Formación del Profesorado Universitario (FPU) under Grant FPU 01561.

Rafael de la Rosa-Vidal, Juan A. Leñero-Bardallo, and Á. Rodríguez-Vázquez are with the Institute of Microelectronics of Seville (IMSE-CNM), CSIC-Universidad de Sevilla, Av. Américo Vespucio, 28, 41092, Seville, Spain, (E-mails: {rdvidal, jlenero, arodri-vazquez}@us.es).

José M. Guerrero-Rodríguez is with the University of Cádiz, Campus Universitario de Puerto Real, 11519 Cádiz, Spain, (E-mail: josem.guerrero@uca.es)

Manuscript received XXX, 2021; revised December, 2021.

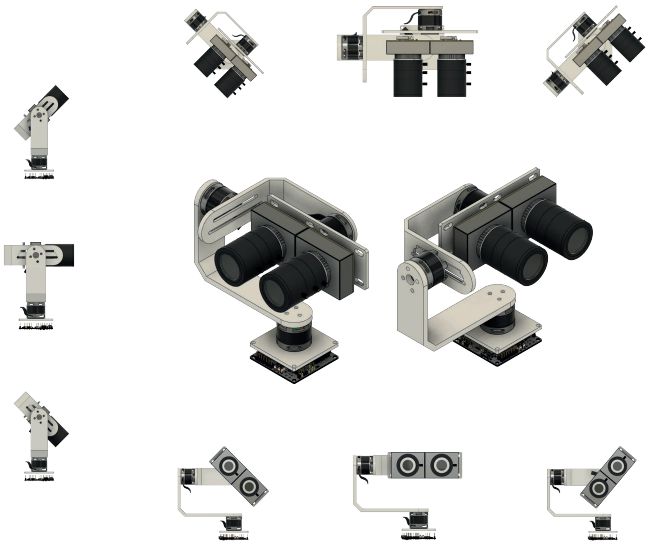


Fig. 1. 3D system model and different ortogonal views of it rotated $\pm 45°$.

degree of freedom, and much less-elaborated mechanical implementation than the one that will be described in this article. The system description was not provided. Extending this previous work, we report design insights about a new moving platform with stereo vision intended to track moving objects employing two low-resolution image sensors, a refined physical design with three degrees of freedom, and lower latency. We have adapted image processing algorithms to the system requirements. Those can be implemented on classic microprocessor-architectures and adapted to process data from image sensors with different nature and applicability. We demonstrate that the operation can be performed with very low-resolution image sensors, involving a very low computational cost that can be executed on a low-cost microcontroller. To the best of our knowledge, tracking involving the optical flow computation by means of the algorithms reported has never been implemented with a microcontroller with low computational capabilities and low-resolution image sensors.

II. System implementation overview

A concept of the system implementation is depicted in Figure 1. The platform can position two cameras intended to track moving objects. Movement is detected by computing the optical flow variations [16]–[18], [23] within the visual scene. The field of view of the two cameras is different to achieve the possibility of implementing stereoscopic vision algorithms to estimate rotations and depth within the visual scene [24]. The two cameras are always moved solidarily. There are three independent motors to position the system with three degrees of freedom (*Pitch*, *Roll* and *Yaw*). The cameras assembly describes a spherical movement similar to a body head movement. There is a microcontroller STM32L476RG that processes the image sensors data. Its mission is to compute the optical flow and determine how the system has to be positioned. There is a dedicated IMU MPU6050 to track the motors positions and move them accordingly.

III. Optical flow detection algorithm

The optical flow can be defined as the apparent movement of luminous intensity patterns within a frame [16], assuming that luminance variations inside the frame are only due to the displacement of such patterns within the frame [25]. It is a relative movement between the objects of the visual scene and an observer [14], [25], [26]. Thus, it can provide very useful information to locate and monitor the position of elements in the space through a frame sequence [27].

Biological organisms compute it to navigate and get across the environment [20], [21]. Although the concept of optical flow computation was proposed many decades ago [14], the first functional algorithms to computed it are more recent [16]–[18], [23]. Traditionally, optical flow computation has been limited by the strong computational requirements associated with real-time algorithm execution. Some authors have already devised systems that track moving objects by computing the optical flow variations [19], [28]. However, to the best of our knowledge, such systems require large computational capabilities that limit their autonomous implementation without using computers. In many cases, complex convolution operations, filtering, storage, and processing of several frames, etc. are required processing steps. Those increase the computational load remarkably with high-resolution pixel arrays.

The basis for the optical flow computation is the motion constraint equation [25]. The light intensity on an instant t of a pixel in the (x, y) coordinates is given by the equation $I(x, y, t)$. If the pixel displaces a distance Δx y Δy in the Cartesian plane during a time interval Δt, the new pixel intensity is given by the function $I(x + \Delta x, y + \Delta y, t + \Delta t)$. Since $I(x, y, t)$ and $I(x + \Delta x, y + \Delta y, t + \Delta t)$ correspond to pixels with identical intensity values, the movement equation, $I\,(x, y, t) = I\,(x + \Delta x, y + \Delta y, t + \Delta t)$, must be satisfied. This assumption is valid for first order approximations where Δx, Δy y Δt are small values.

Among the algorithms to compute the optical flow, three different families of algorithms were considered:

- Differential techniques. They are based on spatio-temporal derivatives computation of image intensity (i.e. Lucas-Kanade [18] and Horn-Schunck [16]).
- Frequency and phase based methods. They are based on the velocity-tuned filters response applied to the frames (i.e. Heeger [29] and Waxman [30]).
- Region-based matching. Those based on the searching of the velocity vector which minimizes the error between two successive frames (i.e. Srinivasan [23]).

For the proposed system implementation, a simplified version of Srinivasan's algorithm [23] proposed by the authors was selected. The algorithm has interesting advantages over the previous ones to embed it on a microcontroller:

- It does not require feature detection inside the frame.
- The optical flow is computed with only one iteration.
- It does not require spatial or temporal filtering. These two operations have a high computational cost.
- The computation is robust to the image noise in non-synthetic images.

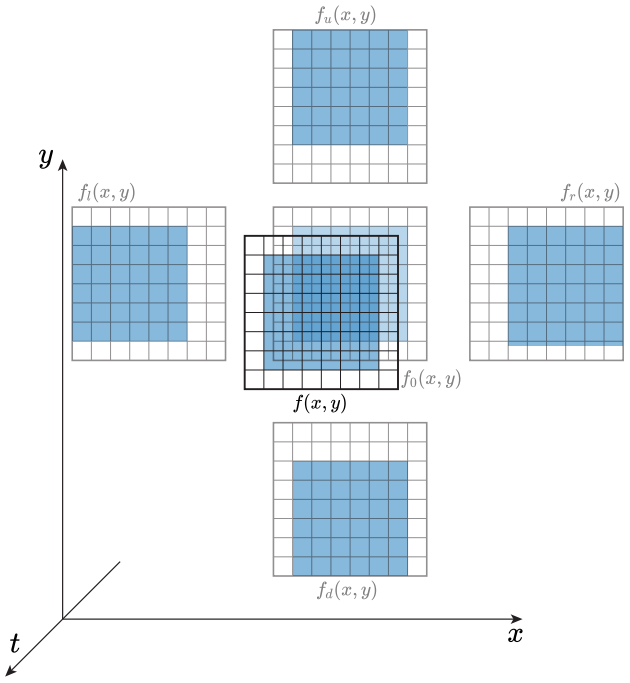


Fig. 2. Example illustrating the bidimensional functions f_l, f_r, f_u, and f_d when considering a frame with 8×8 pixels. A subframe with 6×6 pixels defined by the window function $\Psi(x,y)$ represents a static region of interest. The following values for the displacement parameters were set: $\Delta x_{ref} = \Delta y_{ref} = 1$

To explain the Srinivasan's algorithm operation, let us define a uniform spatial window $\Psi(x,y)$ that selects a region of interest (group of pixels) inside a frame. The pixels intensities inside the window are defined by the function $f(x,y)$. The plane movement in the *x*- and *y*-directions within a time interval Δt is given by $\widehat{\Delta x}$ and $\widehat{\Delta y}$. These two terms represent the optical flow value in each direction. For convenience, let us also denote the function that gives the initial pixel intensity values (before the movement starts at $t = 0$) as $f_0(x,y)$. Also, the functions $f_l(x,y,\Delta t)$ and $f_r(x,y,\Delta t)$ will be defined. They represent the pixel intensities values after a time interval Δt and a frame displacement over the x-axis of a value x_{ref}, i.e.:

$$\begin{aligned} f_l(x,y,\Delta t) &= f_0(x+\Delta x_{ref}, y) \\ f_r(x,y,\Delta t) &= f_0(x-\Delta x_{ref}, y) \end{aligned} \tag{1}$$

Likewise, the functions $f_u(x,y,\Delta t)$ and $f_d(x,y,\Delta t)$ are defined as the pixel intensities after a frame displacement y_{ref} over the *y*-axis:

$$\begin{aligned} f_u(x,y,\Delta t) &= f_0(x, y-\Delta y_{ref}) \\ f_d(x,y,\Delta t) &= f_0(x, y+\Delta y_{ref}) \end{aligned} \tag{2}$$

For simplicity, we will denote the functions of Equations 1 and 2 as f, f_0, f_r, f_l, f_u, f_d. For illustrative purposes, in Figure 2, there is a representation of the different bidimensional functions (f_l, f_r, f_u, and f_d) that are created after a frame displacement in the *x*- and the *y*-directions with the reference values $x_{ref} = y_{ref} = 1$. The frame has a dimension of 8×8 pixels, and the window function $\Psi(x,y)$ defines a subframe with 6×6 pixels representing a static region of interest. Let us assume that the time interval Δt is very short and during it, the pixel intensities do not change. Thus, $f(x,y,\Delta t)$ is just a translation of the original function $f_0(x,y,t)$. Under this assumption, it is possible to express $f(x,y,\Delta t)$ as a function of f_l, f_r, f_u, and f_d, as it is depicted by Equation 3.

$\widehat{f(x,y,t)}$ is an interpolated version of $f(x,y,t)$.

$$\widehat{f} = f_0 + \frac{1}{2}\left(\frac{\widehat{\Delta x}}{\Delta x_{ref}}\right)(f_r - f_l) + \frac{1}{2}\left(\frac{\widehat{\Delta y}}{\Delta y_{ref}}\right)(f_u - f_d) \tag{3}$$

The target is to obtain the values of $\widehat{\Delta x}$ and $\widehat{\Delta y}$ that minimize the error between $\widehat{f(x,y,t)}$ and $f(x,y,t)$. These parameters correspond to the optical flow value. The error minimization between f and $\widehat{f}$ is calculated by optimizing the least quadratic error in the frame region of interest defined by the window function $\Psi(x,y)$. For a frame with $M \times N$ pixels, the error, E, is:

$$E = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left\{\Psi(x,y)\cdot\left[f(x,y) - \widehat{f}(x,y)\right]^2\right\} \tag{4}$$

Where M is the number of pixel rows and N is the number of pixel columns in the frame. In the previous equations, Δx_{ref} and Δy_{ref} are algorithm parameters. They have pixel units and establish the algorithm sensitivity to the optical flow variations. For low-resolution pixel arrays like the one selected in the proposed system implementation, a value of $\Delta x_{ref} = \Delta y_{ref} = 1$ is adequate. This choice also reduces the execution time because no extra calculations are needed when dividing by Δx_{ref} or Δy_{ref}. For larger pixel arrays, Δx_{ref} and Δy_{ref} values must be increased to keep the same algorithm sensitivity to the optical flow variations.

Substituting Equation 3 in Equation 4, the Equation 5 is derived. To minimize the error, the partial derivatives functions referred to $\widehat{\Delta x}$ and $\widehat{\Delta y}$ are forced to be equal to zero, leading to the equation system composed by Equation 6 and Equation 7.

$$E = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left\{\Psi(x,y)\cdot\left\{f - \left[f_0 + \frac{1}{2}\left(\frac{\widehat{\Delta x}}{\Delta x_{ref}}\right)\left(f_r - f_l\right) + \frac{1}{2}\left(\frac{\widehat{\Delta y}}{\Delta y_{ref}}\right)\left(f_u - f_d\right)\right]\right\}^2\right\} \tag{5}$$

$$\left(\frac{\widehat{\Delta x}}{\Delta x_{ref}}\right)\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_r-f_l\right)^2\right]$$
$$+\left(\frac{\widehat{\Delta y}}{\Delta y_{ref}}\right)\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_u-f_d\right)\cdot\left(f_r-f_l\right)\right]$$
$$=2\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f-f_0\right)\cdot\left(f_r-f_l\right)\right] \quad (6)$$

$$\left(\frac{\widehat{\Delta x}}{\Delta x_{ref}}\right)\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_r-f_l\right)\cdot\left(f_u-f_d\right)\right]$$
$$+\left(\frac{\widehat{\Delta y}}{\Delta y_{ref}}\right)\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_u-f_d\right)^2\right]$$
$$=2\cdot\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f-f_0\right)\cdot\left(f_u-f_d\right)\right] \quad (7)$$

The values of $\widehat{\Delta x}$ and $\widehat{\Delta y}$ result from solving the equation system defined by Equations 8 and 9. They provide the optical flow value in the region defined by $\Psi(x,y)$.

$$\widehat{\Delta x}=2\cdot\frac{C\cdot D-B\cdot E}{A\cdot D-B^2}\cdot\Delta x_{ref} \quad (8)$$

$$\widehat{\Delta y}=2\cdot\frac{A\cdot E-B\cdot C}{A\cdot D-B^2}\cdot\Delta y_{ref} \quad (9)$$

where,

$$A=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_r-f_l\right)^2\right] \quad (10)$$

$$B=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_r-f_l\right)\cdot\left(f_u-f_d\right)\right] \quad (11)$$

$$C=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f-f_0\right)\cdot\left(f_r-f_l\right)\right] \quad (12)$$

$$D=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f_u-f_d\right)^2\right] \quad (13)$$

$$E=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\Psi\left(x,y\right)\cdot\left(f-f_0\right)\cdot\left(f_u-f_d\right)\right] \quad (14)$$

These are coefficients defined by the authors that can be calculated while the sensor frame is readout, facilitating the entire system pipeline operation.

There are two situations where the system equation is undetermined:

- The entire frame is exposed to the same illumination value. Consequently, all the pixels intensities are theoretically the same. In this case, all the equation coefficients are null because $f_r=f_l=f_u=f_d$.
- The frame only has one-dimension features (vertical or horizontal). For instance, this can happen whether there is only an horizontal ($f_r=f_l$) or a vertical line ($f_u=f_d$) in the frame. In such cases, there are multiple solutions for the system equation leading to the well-known Aperture Problem [31].

Such situations are detected by examining the parameter values. Optical flow computation is discarded whether the denominator in Equations 8 and 9 is close to zero. However, this case is unlikely because frames incorporate Fixed Pattern Noise (FPN) added by the image sensor that benefits the algorithm operation.

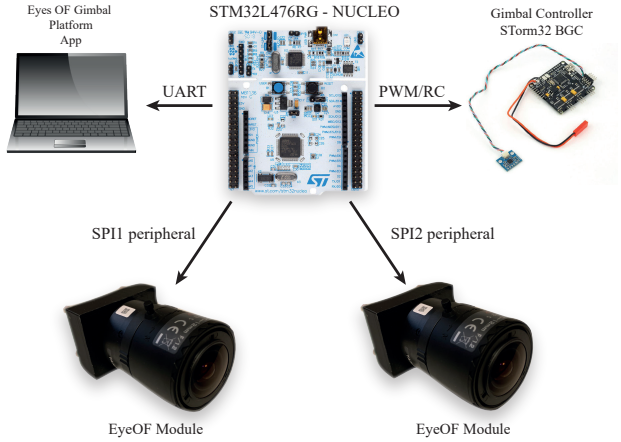


Fig. 3. Diagram showing the system connectivity among the different hardware elements.

IV. Hardware Implementation

In Figure 3, the different hardware elements are shown. The kind of connectivity between them to exchange information is depicted in the diagram. The main component is the STM32L476RG-NUCLEO microcontroller development board from ST Microelectronics. It controls the rest of the modules. It sends/receives data and configuration instructions to/from a PC through a UART interface. It is also connected to two modules called Eye of Optical Flow (EyeOF) that acquire raw images from the visual scene. Finally, the STM32L476RG microcontroller drives using PWM signals three motors with a STorm32 BGC Gimbal controller.

The STM32L476RG-NUCLEO development board integrates as the main device an ARM® Cortex®-M4 core that can operate at 80 MHz consuming 26 mW. It has been conceived for low-power applications and presents multiple SPI interfaces to connect different peripherals. It allows Direct Memory Access (DMA) from the peripherals without using the CPU. With this functionality, pipeline operation between data transfer and data processing is implemented to reduce the system latency.

A. Image acquisition module

Demonstrating that it is feasible to compute the optic flow efficiently with very low-resolution images was one target in this work. With this aim, the low resolution ADNS2610 sensor was selected. This family of sensors was popular a few years

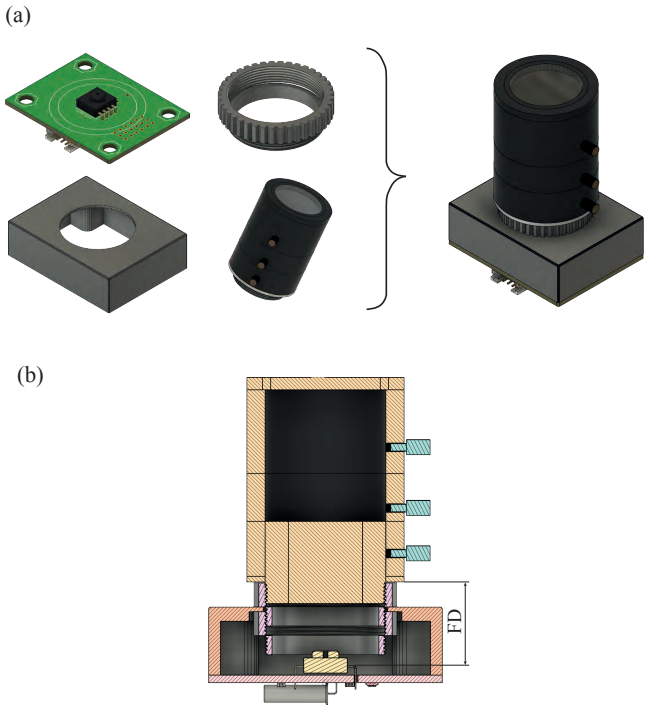


Fig. 4. (a) 3D design of the module EyeOF. (b) Detail of the optics assembled to it. FD is the focal distance from the optics to the sensor.

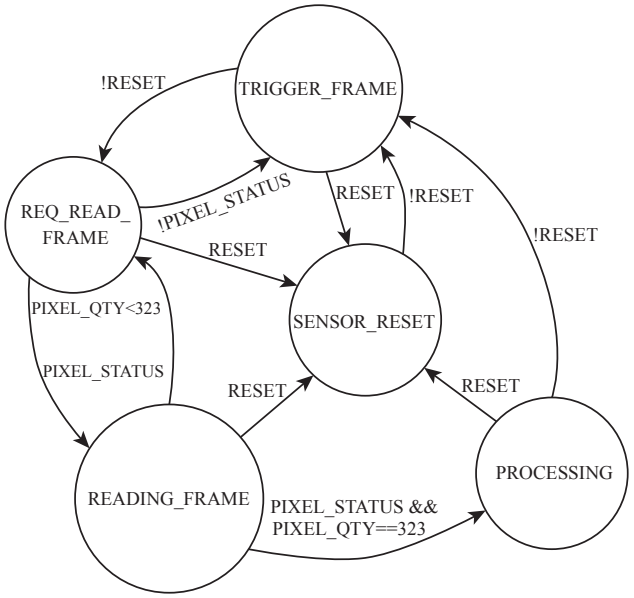


Fig. 5. Diagram of the FSM implemented in the microcontroller for image acquisition and processing.

ago. They were implemented in optic mouses to track the movement of a light source that was solidary with the mouse movement. The sensor is compounded of a low-resolution 18×18 image sensor and a DSP that can implement basic image processing operations. In this work, the ADNS2610 sensor just senses illumination values in the visual scene. Then its data is transmitted to the STM32L476RG microcontroller to compute the optical flow with the algorithm described.

To host the ADNS2610 sensor and its optics, a custom PCB was designed and fabricated. We called it Eye of Optical Flow (EyeOF). The EyeOF module has a 4-wire connector to communicate the sensor with a microcontroller, power the sensor, and to send or receive configuration parameters. In Figure 4.(a), there is a 3D representation of the EyeOF module and its main components. A lens holder can be attached to the EyeOF module (see Figure 4.(b)) to easily mount and remove the sensors optics.

B. Image acquisition and optical flow computation

A Finite State Machine (FSM) was implemented on the microcontroller to control the EyeOF modules in charge of the image acquisition. In Figure 5, there is a diagram illustrating the different transitions between states in the FSM. To speed up the optical flow calculation and to avoid dead times between state transitions, all the required operations are performed while the microcontroller is awaiting data from the EyeOF modules. We provide a brief description of the operations conducted by the FSM:

- *SENSOR RESET*. The EyeOF module and all its data registers are initialized. The module operation starts from this state that is accessible by all the other FSM states.
- *TRIGGER FRAME*. The ADNS2610 sensor is prepared to initiate a frame acquisition. The operation requires to write one control bit in a EyeOF configuration register and wait for the EyeOF module to be prepared to acquire a new frame. In the meantime, all the parameters from the EyeOF module, to be monitored in the user interface, can be transferred to the DMA. The frame pixels will be readout one by one in the upcoming two states.
- *REQ READ FRAME*. In this state, a requirement to readout a pixel is sent to the EyeOF module. In the meantime, if the previous pixel was already readout, the pixel status is checked. If possible, some operations related to the optical flow computation are performed with the previous readout pixels.
- *READING FRAME*. A register containing the pixel data is readout. The number of readout pixels is updated. If there are pending pixels, the FSM comes back to the previous *REQ READ FRAME* state. If the entire frame has been readout, the FSM machine moves to the next state, *PROCESSING*.
- *PROCESSING*. At this state, the two frames required to compute the optical flow are available. Since only two frames are required for the computation, the oldest frame stored in memory is discarded. A register that stores the memory position that corresponds to the new frame and the previous one is updated. The optical flow values are determined and used to correct the motor position. After completing these operations, the FSM returns to the initial *TRIGGER FRAME* state. All the FSM previous steps are repeated to acquire a new frame, compute the optical flow, and update the motors' position.

The maximum frame rate that can be achieved is limited by

TABLE I
TIME REQUIRED FOR THE DATA READING OPERATIONS WITH THE ADNS2610 IMAGE SENSOR.

Operation	Time required
Time to await between pixels consecutive readings, t_1.	50 µs
Time to send a pixel reading requirement and to readout the pixel value, t_2.	600 µs
Time to prepare the image sensor to acquire pixel data and initiate the pixels readout operation, t_3.	700 µs

(a)

17	35	53	71	89	107	125	143	161	179	197	215	233	251	269	287	305	323
16	34	52	70	88	106	124	142	160	178	196	214	232	250	268	286	304	322
15	33	51	69	87	105	123	141	159	177	195	213	231	249	267	285	303	321
14	32	50	68	86	104	122	140	158	176	194	212	230	248	266	284	302	320
13	31	49	67	85	103	121	139	157	175	193	211	229	247	265	283	301	319
12	30	48	66	84	102	120	138	156	174	192	210	228	246	264	282	300	318
11	29	47	65	83	101	119	137	155	173	191	209	227	245	263	281	299	317
10	28	46	64	82	100	118	136	154	172	190	208	226	244	262	280	298	316
9	27	45	63	81	99	117	135	153	171	189	207	225	243	261	279	297	315
8	26	44	62	80	98	116	134	152	170	188	206	224	242	260	278	296	314
7	25	43	61	79	97	115	133	151	169	187	205	223	241	259	277	295	313
6	24	42	60	78	96	114	132	150	168	186	204	222	240	258	276	294	312
5	23	41	59	77	95	113	131	149	167	185	203	221	239	257	275	293	311
4	22	40	58	76	94	112	130	148	166	184	202	220	238	256	274	292	310
3	21	39	57	75	93	111	129	147	165	183	201	219	237	255	273	291	309
2	20	38	56	74	92	110	128	146	164	182	200	218	236	254	272	290	308
1	19	37	55	73	91	109	127	145	163	181	199	217	235	253	271	289	307
0	18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306

(b)

12	30	48	66	8
11	29	47	65	8
10	28	46	64	8
9	27	45	63	8
8	26	44	62	8
7	25	43	61	7
6	24	42	60	7
5	23	41	59	7
4	22	40	58	7
3	21	39	57	7
2	20	38	56	7
1	19	37	55	7
0	18	36	54	7

Fig. 6. a) Table with pixels indexes: in blue, there are the indexes whose data can be used to perform any optical flow computation when they are readout. In red, there are the ones that require to wait for additional data to make a computation. b) Indexes required to calculate the partial sums. Illustration of the order they can be employed to make calculations.

the fact that pixels values has to readout one by one. However, the FSM machine performs all the optical flow computations in parallel while pixels are readout avoiding dead time. Thereafter acquiring a frame, the motors position can be updated, and a new frame can be acquired.

In Table I, the amount of time required for the data reading operations with the ADNS2610 image sensor is reported. Using this information, the amount of time to readout a frame is computed:

$$t_{acq} = (t_1 + t_2) \cdot M \cdot N + t_3 = 211.3 \text{ ms} \tag{15}$$

$$FPS = \frac{1}{t_{acq}} = 4.732 \approx 5 \ fps \tag{16}$$

A maximum frame rate of 5 fps can be achieved with the selected image sensor. This value limits the system latency to track fast moving objects. Such value qualifies to track moving light sources that are far away, i.e. the sun, and objects that do not move fast.

C. Optical flow algorithm implementation

The implementation of the Srinivasan's optical flow algorithm [23] has been devised to avoid a bottleneck in the device operation. There are several computational steps

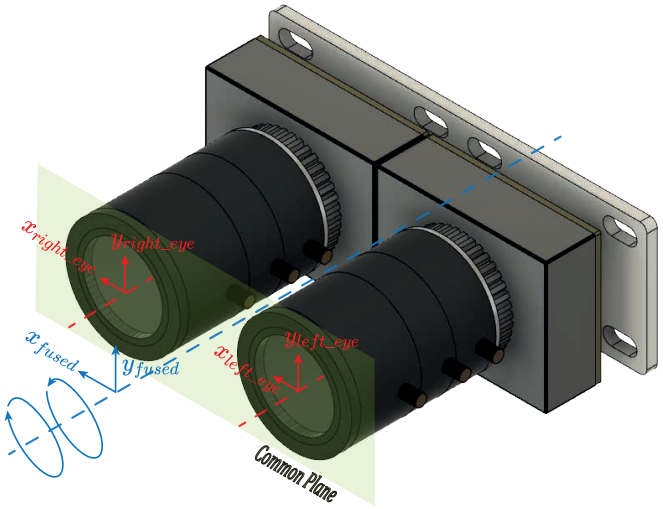


Fig. 7. Spatial arrangement for the EyeOF modules. In red, the reference coordinate systems for each EyeOF module; in blue, the reference coordinate system for the optical flow vectors fusion.

performed in parallel with the pixel readout operations. Thus, there are pipeline operations continuously executed by the microcontroller to speed up the system operation. The algorithm implementation is based on the use of the information provides in the tables of Figure 6.(a-b).

The two frames required for the optical flow computation are stored on two one-dimensional arrays with $M \times N$=18·18 elements. To compute the optical flow, the coefficients A, B, C, D and E, given by Equations 10, 11, 12, 13, and 14, must be calculated. Each coefficient results from the partial sum of $M \times N$ elements. These elements are calculated while the pixel values of the new frames are being readout. To calculate the partial sums, the pixel intensity values are arranged in the memory as it is depicted in Figure 6.(a). For simplicity, in the plots, we have assumed that $\Delta x = \Delta y = 1$. Pixels are always readout in order, starting from the pixel labeled as 1 and finishing with the pixel labeled as 324. In Figure 6.(a), we classify with a color code the indexes. In red, there are the indexes that cannot be immediately used to compute a partial sum when the pixels that represent are readout. Observing the terms of the coefficients equations, it can be understood that until a pixel neighborhood is not readout, any partial sum computation is possible. In blue, there are marked the indexes that can lead to an immediate partial sum computation when they are readout.

In Figure 6.(b), it is illustrated how the different partial sums can be calculated while the different pixels are readout sequentially in the direction the arrow indicates. For the first partial sum computation, the indexes 19, 18, 20, 1, 37 are required; for the second partial sum, the indexes 21, 20, 22, 3, 39 and required; and so on. It can be noticed that to calculate the first partial sum, the first 37^{th} pixels have to be readout. For this reason, the first 36^{th} indexes are marked in red color in Figure 6.(a).

In the proposed system implementation (Figure 7), two coplanar cameras are mounted. This is the minimum amount

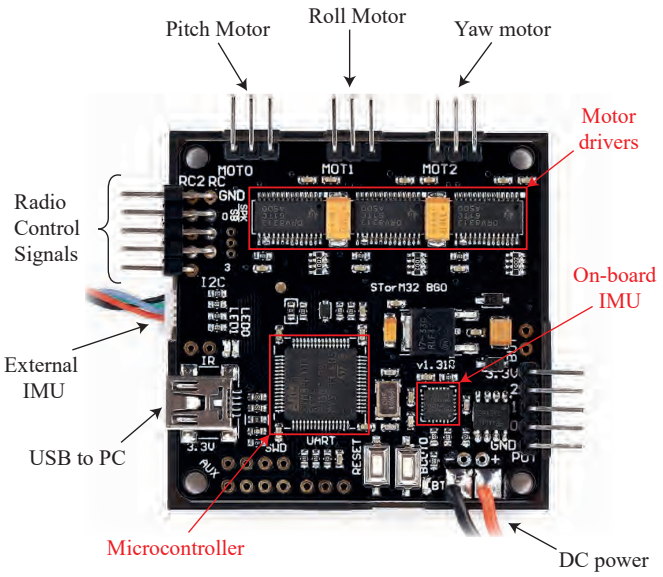


Fig. 8. STorM32 BGC control board. Its main functional modules are highlighted.

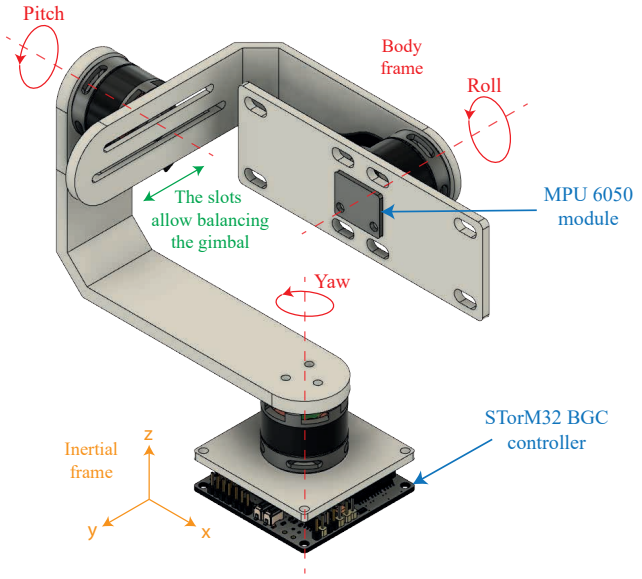


Fig. 9. Platform mechanical design: 3D structural model.

of sensors necessary to implement a three-dimensional object tracking that detects visual scene rotations referred to one axis. Moreover, by increasing the number of cameras, the field of view value is increased. The results of processing the frames captured with each image sensor are two vectors. They indicate the direction and the magnitude of the optical flow: $\overrightarrow{v}_{left} = [\Delta x_{left}, \Delta y_{left}]^T$ and $\overrightarrow{v}_{right} = [\Delta x_{right}, \Delta y_{right}]^T$. Then two vectors are combined leading to:

- An optic flow vector that is the result of of averaging the optical flow vectors provided by each image sensor, i.e., $\overrightarrow{v}_{fused} = \frac{\overrightarrow{v}_{left} + \overrightarrow{v}_{right}}{2}$, as it is depicted in Figure 7.
- The two image sensors are aligned in the x-plane. Hence, the variation of the y-coordinates for the two optical flow vectors, $\Delta y_{fused} = \Delta y_{left} - \Delta y_{right}$ provides information about the visual scene rotation referred to the platform x-axis.

V. Motors and Controllers

Three 2208 brushless DC motors were selected to position the platform. They operate at 90 KV rotation-torque ratio and have 14 poles.

To drive the three motors, the 3-axis STorM32 BGC controller board were selected. The main sub-components and interconnecting pin-outs and are shown in Figure 8. The upper connections are intended for the pitch, roll, and, yaw motor control. At the bottom, there is the power supply connection. We fed the board with a DC power supply with 12 V and 5 A (60 W). On the left side, their connections to three elements: a) Connections to control signals to modify the system position. b) An external Inertial Measurement Unit (IMU) with a gyroscope and accelerometer to determine the module's position. This IMU must be allocated in the same plane as the motors and it is connected to the STorM32 BGC device with a twisted pair. c) An USB port to configure the board from a PC. In the center of the controller board, there is another on-board IMU to detect the board's position when it is installed in moving surfaces, i.e. in a drone. A Graphical User Interface (GUI) is available to tuning the STorM32 BGC microcontroller.

VI. Mechanical Design

In Figure 9, there is a detailed 3D view of the platform design. The mechanical structure is based on the Gimbal system concept. This type of structure is usual for system camera stabilization. For the platform mechanical design, the 3D modeling software *Fusion 360*® from *AutoDesk* was used. This software allows integrating the prior PCB design from the EyeOF modules depicted in Figure 4 with the system mechanical structure. The interaction of the moving structure with the system elements: wires, motors, optics, EyeOF boards was studied in detail before manufacturing. The different platform parts were fabricated with a 3D printer.

For the representation of the position and attitude of a moving object in a three-dimensional space, an inertial reference system was considered. It is fixed and it does not move during the platform operation. Additionally, a local reference system is defined. Its coordinate origin is the system center of gravity. Its axes are disposed as it is depicted in Figure 9 and it is not inertial. Over the local reference system, the attitude of a moving object is defined by quaternions. The first quaternion component represents a rotation angle and the other three components define the axis that is rotated the angle defined by the first vector component.

For the right platform operation, it must be mechanically balanced. This implies that the load's center-mass must be exactly allocated along the rotation axis for each motor. Consequently, the load will be fixed at an optimum equilibrium position to speed up the displacement. Otherwise, vibrations of the gimbal system and consequently added overheating of

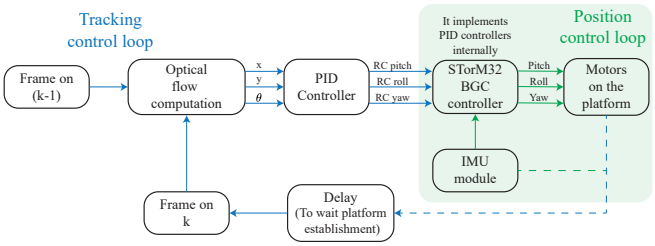


Fig. 10. Diagram with the implemented control loops.

the corresponding motor could also occur. To facilitate the balancing process, slots are present in some parts to balance the center-mass of the pitch axis, as it is depicted in Figure 9. The roll axis will be balanced because the loads allocated over it are identical. For the yaw axis, the equilibrium condition is not so critical because the platform will be held in a vertical position during its operation. Hence, there is not any relevant momentum; all the relevant forces are longitudinal to the motor rotation axis. In Figure 1, the 3D design of the entire platform is shown. The movement range is sufficient for the proposed application scenarios.

VII. Control Loop Implementation

This section describes the system configuration process, its stabilization, and how its parameters are trimmed.

The first step to initialize the STorM32 BGC board is to calibrate the IMU that has been placed next to the EyeOF modules, as it is depicted in Figure 9. The controller GUI is used to define the IMU orientation. With the calibration process, undesired offset values from the gyroscopes measurement are canceled.

The control law is based on the software implementation of three PID regulators to position every specific axis: pitch, roll, and yaw, according to the control loop diagram illustrated in Figure 10. The regulator input is the error between the target position and the current one. The K_P, K_I, and K_D PID parameters were adjusted as described in Listing I. Their values can directly be set using an available microcontroller GUI.

The parameters K_P and K_D are responsible for the dynamic platform response. They have selected to achieve a critically damped response without oscillations. Once the platform is stabilized, the modules EyeOF mounted on it move, tracking the scene visual flow variations.

VIII. User Interface Implementation

To debug the system and monitor its outputs a custom GUI was implemented. This GUI, shown in Figure 11, was programmed with the Microsoft WPF technology and .NET Core framework [32]. Currently, the WPF technology is integrated in .NET Core that aims to be a multiplatform environment compatible with Linux or MacOS. These programming tools allow creating refined and modern user interfaces that can exchange data and commands with custom systems of diverse nature. The GUI's purpose is to monitor

Listing I: PID control parameters adjustment.

1) All the control outputs are disabled, excepting the one corresponding to the pitch axis. Under this configuration, the following adjustments are performed:
   a) The K_D parameter is gradually increased until the system starts vibrating at high frequency. Then the K_D value is decreased until the system vibrations stop. It must be checked that there are no system vibrations for any of its possible axis positions.
   b) The K_I parameter value is increased to the minimum possible value above zero. For the STorM32 BGC microcontroller, this value is 5.
   c) The K_P value is increased until the system starts oscillating at low frequency. At this point, the K_P value is reduced until the system is stable. Again, it must be checked that there no vibrations in any axes position.
   d) The K_I value is increased until the system becomes unstable. In this situation, the motor position in the axis will vary randomly. Then, the K_I value must be reduced until reaching the stability again.
2) The motor that controls the roll axis is activated. The steps 1.(a-d) are repeated.
3) The motor that controls the yaw axis is activated. The steps 1.(a-d) are repeated.

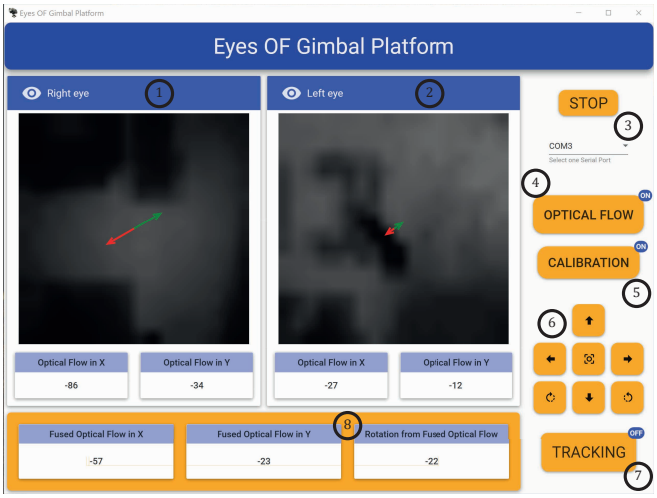


Fig. 11. Eyes OF Gimbal Platform custom interface. The main interface buttons and functionalities are numbered in the plot.

the platform operation, debug it, and illustrate how it works. It must be remarked that the system is autonomous and does not require to be connected to the GUI to operate. In Figure 11, the custom user interface is shown. Its name is *Eyes OF Gimbal Platform*.

The interface can represent simultaneously the two frames rendered by the EyeOF modules. In Figure 12, the face of one of the author is rendered. Although the image sensors pixel resolution is limited (18×18 pixels), some parts of the face can be easily distinguished. In practical situations, we do not target to identify objects within the visual scene. The aim is to detect the optical flow variations provoked by moving objects and track them.

The interface allows to represent arrows vectors over the rendered frames (see Figures 11 and 12). This possibility leads to a very intuitive dynamic representation of optical flow

Fig. 12. Frames with a face captured with the EyeOF modules. In the left one, the optical flow is not shown. In the right one, the result of the optical flow computation and the system position correction to track the movement are represented with two colored arrows, red and green respectively.

computation results, indicating its direction and magnitude. Two colored arrows are displayed over the incoming frames providing information continuously. This representation is necessary to debug the system. The red arrow is the result of the optical flow computation. The arrow length is proportional to the optical flow magnitude. The green arrow indicates the necessary system movement correction to capture a frame equal to the previous one taken as a reference.

Several commands and operations are accessible through the GUI interface. In Figure 11, the most representative windows and interface buttons are numbered. We list them:

- Sections to display the frames acquired by the image sensors, (1), and (2).
- Button to start or stop the platform operation, (3).
- Button to start computing the optical flow taking an initial frame as a reference, (4).
- Button to calibrate the platform ((5)). This operation consists of storing a reference frame to track the movement referred to it.
- Arrows buttons, (6), to force the platform movement in different directions, i.e., South, North, East, and West. These functions are useful to set an initial camera position in the center of the visual scene.
- Tracking mode button, (7). In this operation mode, the platforms track moving objects within the visual scene through the optical flow computation.
- Several items, labeled as (8) that display the numerical value of the optical flow computation.

IX. Experimental Results

The entire system was characterized. Table II summarizes the main system specifications. A photograph of the final system implementation is shown in Figure 13. The entire system is powered with a power supply source operating at 12 V with a power of 60 W.

The system tracking capability was tested in our laboratories. The system can easily track walking people from a distance of 1.5 meters or higher from the platform. Also, its applicability to track bright light sources and walking people was verified.

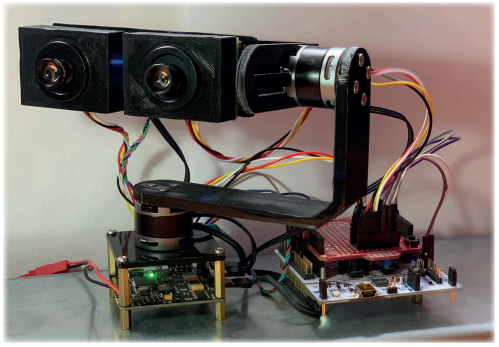


Fig. 13. Photograph showing the final moving platform implementation.

TABLE II
Tracking Platform Main Features.

Functionality	Optical flow computation and object tracking
Microcontroller for data processing	STM32L476RG, ARM Cortex-M4 @ 64 MHz
Microcontroller for motor positioning	STorm32 BGC Gimbal, ARM 32-bit Cortex-M3 @ 72 MHz
Motors	Three 2208 brushless DC motors, 90 KV, 14 poles
Dimensions	160 mm × 125 mm
Image sensor resolution	18 × 18 pixels
Field of view	100°
Latency	450 ms
Frame rate	5 fps
Power consumption	270 mA@12 V
Degrees of freedom	Three (*Pitch*, *Roll*, and *Yaw*)
Stereo vision	Yes
Scalability	Possible to add multiple image sensors
Optional frame postprocessing	Yes

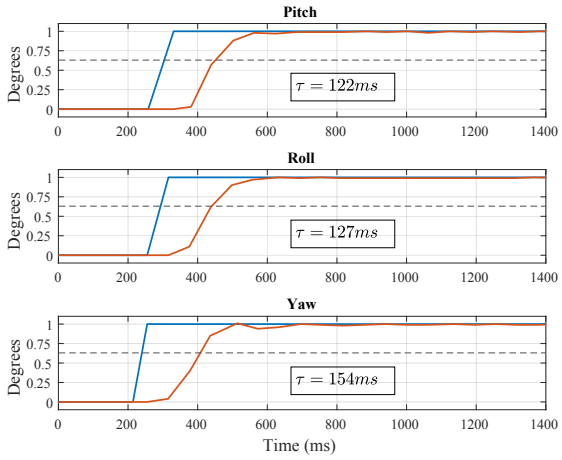
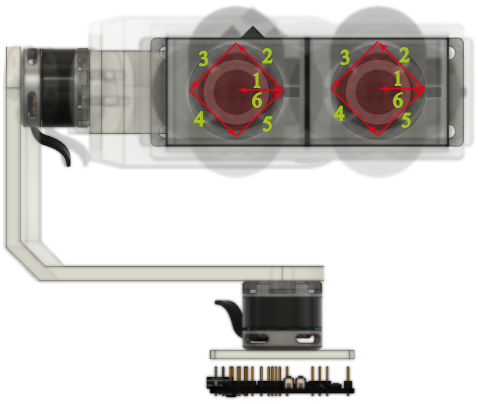


Fig. 14. Experimental motors' unit step-response.

(a)



(b)

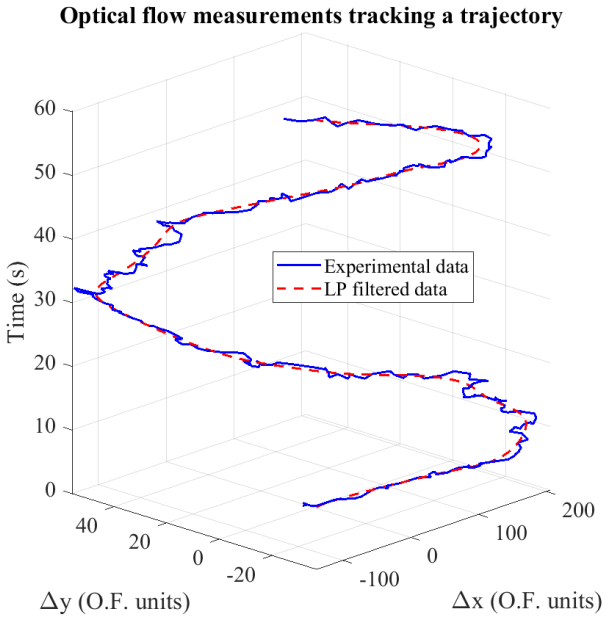


Fig. 15. Optical flow computation with the system following a rhomboidal trajectory. a) System trajectory. b) Experimental data and expected response.

In Figure 14, experimental results illustrating the motors' unit step response are shown. The communication interface allows to timestamp the IMU recorded positions with a 10 ms time resolution. Analyzing this data, it is possible to determine the motors' time constants for a unit step stimulus. Results in Figure 14 show that the motors response is approximately critically damped, as desired. The first-order system time constant is similar for the movement in the pitch and roll axes, and a bit higher in the yaw axis ($\approx$154 ms). This time constant makes the system suitable for applications where the objects do not move at high speed or are far away from their basis.

In Figure 15, the optical flow calculation results when the system was moved across a rhomboidal trajectory are shown. Figure 15.(a) illustrates the trajectory followed by the image sensors centroids. Arrows indicate the movement direction and numbers the order in which each path was taken. In Figure 15.(b), we represent with a blue trace the results of the optical flow computation. The x and y-axes represent a scaled version of the reference frame displacement over these axes, i.e., $\widehat{\Delta x}$ and $\widehat{\Delta y}$, respectively. To avoid floating-point operations, these values are scaled to operate with integers, which require less computation effort from the microcontroller. In the z-axis, temporal information is recorded. The initial and the final optical flow values are the same and equal to zero because the system starts/stops from/at this point. The red trace in Figure 15.(b) represents a low-pass filtered version of the experimental data that is closer to the ideal response expected in the experiment. Optical flow variations must increase/decrease monotonically until a vertex of the rhombus is reached. It can be observed that the system output fluctuates around this filtered data suggesting that the platform's dynamics induces small mechanical oscillations that affect the optical flow computation. Visual scene variations before the system has stabilized itself, induce optical flow computation errors. Thus, there is a trade-off between the system tracking speed and the optical flow computation.

X. Benchmarking and Future Work

Table III benchmarks the proposed system against other related and relevant ones. Comparing the proposed platform to the art, we did not find an autonomous tracking system that performs all the optical flow computation and required operations on a microcontroller device. All previous related works [1], [3], [10], [33] employ image sensors with larger pixel arrays. Personal computers, sometimes combined with advanced FPGAs are needed for the optical flow computation. This is an important limitation for the development of space tracking systems as sun sensors [5], [6] because the amount of power, the system dimensions, and the payload are quite limited in satellites and on-board systems. The work of Floreano et al. [19] demonstrated that it is possible to implement a real-time optical flow computation with modern microcontrollers. However, to the best of our knowledge, autonomous tracking systems based on optical flow detection have not been implemented. Some authors like Conroy et al [34] incorporate dedicated boards to implement the optical flow computation to avoid collisions of autonomous systems in unknown environments. These modules add system complexity and are closed solutions, not being always compatible with a user image sensor choice. Neither can they be easily expanded by adding extra cameras.

Competitive ad-hoc image sensors targeted to detect the optical flow and track small moving objects have been reported [35]–[37]. However, to the best of our knowledge, these previous works are mainly focused on the image sensor implementation. They do not report a detail system integration of the sensors with a mechanical platform and its performance.

One important finding is the demonstration that the optical flow can be computed efficiently with a very low-resolution pixel array. To the best of our knowledge, this has not been explored previously and can lead to further development of autonomous systems with low power consumption, implemented with low-cost microcontrollers. The system scene perception and interpretation can differ from ours, not being necessary to process high-resolution images that, in many cases limit the system performance.

TABLE III
STATE-OF-THE-ART COMPARISON.

Work	This work	Tsai et al. [1]	Ko et al. [3]	Deng et al. [10]	Delbruck et al. [33]
Functionality	Optical flow computation and object tracking	Omnidirectional mobile manipulator	Mobile robotic platform for agricultural applications	Testbed: Visual tracking, 3D control, image processing.	Dynamic object blocking
Dimensions	160 mm × 125 mm	ND	565 mm × 960 mm	Adaptive (testbed surrounded by eigth cameras)	ND
Field of view	100°	90° (H) x 60° (V)	64°	77.32°	ND
Image sensor resolution	18 × 18 pixels	2560 × 720 pixels	ND (> 640 × 480 pixels)	1280 × 720 pixels	128 × 128 pixels
Latency	450 ms	3 s	ND (Real-time navigation)	10 ms	3 ms
Power consumption	270 mA @ 12 V (STM32L476RG ARM microcontroller for data processing and STorm32 BGC Gimbal microcontroller for motor positioning)	ND (One on board Intel i7 laptop, One Arduino controller)	ND (One computer on board)	ND (Two on board Intel i7 computers, Zynq-7000 FPGA, and eight image sensors)	ND (One computer, one HiTec HS-6965 MG digital servo, and one DVS sensor
System degrees of freedom	Three (Pitch, Roll, and Yaw)	Three (Pitch, Roll, and Yaw)	Two (Terrestrial vehicle)	None (It is static)	One (Azimuth)
Number of cameras	Two moving cameras. Stereo vision.	Static camera with stereo vision	One static camera	Eight static cameras	One Dynamic Vision Sensor (DVS)
Scalability	Possible to add multiple image sensors and processing algorithms	Possible to add additional processing algorithms	Possible to add additional processing algorithms	Possible to add multiple image sensors and processing algorithms	Possible to add additional processing algorithms
System adaptability to different application scenarios	Yes	No	Yes	Yes	No

The main system limitation is its high latency response time. This is mainly due to the choice of low-speed image sensors whose pixel data has to be readout one by one until completing the pixel matrix. Since the system dynamics is very fast, it would be possible to operate with image sensors with a higher frame rate. Although this limitation, the system qualifies for the aforementioned application scenarios.

Among the further work, it must be remarked that extra image sensors could be added to the platform to increase the field of view, emulating biological systems with multiple eyes [19]–[21]. Finally, additional image processing algorithms could be embedded on the microcontroller, depending on the system application scenario, given flexibility to adapt the platform to different applications.

XI. CONCLUSIONS

An autonomous platform to track movement has been presented. It detects objects' movement by computing the optical flow variations when they move across the system field of view. A custom optical flow computation algorithm was developed. It can be embedded in microcontrollers with low-computation capabilities. The algorithm can process the images rendered with low-resolution image sensors alleviating, even more, the computational cost. Over similar reported systems, this one operates autonomously, demonstrating that low-resolution image sensors qualify for object tracking. The system can be used as a test platform where new image processing algorithms can be combined with the implemented one. Thus, it can be easily adapted to meet the specific requirements of different application scenarios involving object trajectory tracking and positioning using cameras. System scalability is possible by increasing the number of image sensors, still keeping a reduced computational load.

REFERENCES

[1] C. Tsai, Y. Chou, C. Wong, Y. Lai, and C. Huang, "Visually guided picking control of an omnidirectional mobile manipulator based on end-to-end multi-task imitation learning," *IEEE Access*, vol. 8, pp. 1882–1891, 2020.

[2] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza, "Low computational-complexity algorithms for vision-aided inertial navigation of micro aerial vehicles," *Robotics and Autonomous Systems*, vol. 69, pp. 80–97, 2015.

[3] M. H. Ko, B. Ryuh, K. C. Kim, A. Suprem, and N. P. Mahalik, "Autonomous greenhouse mobile robot driving strategies from system integration perspective: Review and application," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 4, pp. 1705–1716, 2015.

[4] R. Sabzevari and D. Scaramuzza, "Multi-body motion estimation from monocular vehicle-mounted cameras," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 638–651, 2016.

[5] J. A. Leñero-Bardallo, L. Farian, J. M. Guerrero-Rodríguez, R. Carmona-Galán, and Á. Rodríguez-Vázquez, "Sun sensor based on a luminance spiking pixel array," *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6578–6588, Oct. 2017.

[6] L. Farian, P. Häfliger, and J. A. Leñero-Bardallo, "A miniaturized two-axis ultra low latency and low-power sun sensor for attitude determination of micro space probes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1543–1554, May 2018.

[7] C. Cédras and M. Shah, "Motion-based recognition a survey," *Image Vis. Comput.*, vol. 13, pp. 129–155, 1995.

[8] A. Dundar, J. Jin, and E. Culurciello, "Visual tracking with similarity matching ratio," *CoRR*, vol. abs/1209.2696, 2012.

[9] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto, "Unsupervised moving object detection via contextual information separation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 879–888.

[10] H. Deng, Q. Fu, Q. Quan, K. Yang, and K. Cai, "Indoor multi-camera-based testbed for 3-D tracking and control of UAVs," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3139–3156, 2020.

[11] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct. 2014.

[12] J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodríguez-Vázquez, "Applications of event-based image sensors –Review and analysis," *International Journal of Circuit Theory and Applications*, vol. 46, 08 2018.

[13] G. Gallego, T. Delbruck, G. M. Orchard *et al.*, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[14] J. J. Gibson, *The Perception of the Visual World*. Boston: Houghton Mifflin, 1950.

[15] J. J. Koenderink, "Optic flow," *Vision Research*, vol. 26, no. 1, pp. 161–179, 1986.

[16] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981.

[17] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[18] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[19] D. Floreano, R. Pericet-Camara, S. Viollet *et al.*, "Miniature curved artificial compound eyes," *Proceedings of the National Academy of Sciences*, vol. 110, no. 23, pp. 9267–9272, 2013.

[20] P. Anandan, "Measuring visual motion from image sequences" Ph.D. dissertation, University of Massachusetts Amherst, 1987.

[21] J. E. Dowling, *The Retina : An Approachable Part of the Brain*. Cambrigde, Mass: Belknap Press of Harvard University Press, 1987.

[22] R. de la Rosa-Vidal, J. M. Guerrero-Rodríguez, and J. A. Leñero-Bardallo, "Live Demonstration: A Tracking System Based on a Real-Time Bio-Inspired Optical Flow Sensor," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–1.

[23] M. V. Srinivasan, "An image-interpolation technique for the computation of optic flow and egomotion," *Biological Cybernetics*, vol. 71, no. 5, pp. 401–415, 1994.

[24] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S. Ieng, R. Benosman, and B. Linares-Barranco, "Event-driven stereo visual tracking algorithm to solve object occlusion," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4223–4237, 2018.

[25] J. L. Barron and N. a Thacker, "Tutorial: Computing 2D and 3D optical flow," *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, no. 2004, pp. 1–12, 2005.

[26] J. Gibson, *The senses considered as perceptual systems*. Boston: Houghton Mifflin, 1966.

[27] J. J. Gibson, "On the analysis of change in the optic array," *Scandinavian Journal of Psychology*, vol. 18, no. 1, pp. 161–163, 1977.

[28] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 2426–2429.

[29] D. J. Heeger, "Optical flow using spatiotemporal filters," *International journal of computer vision*, vol. 1, no. 4, pp. 279–302, 1988.

[30] A. Waxman, J. Wu, and F. Bergholm, "Convected activation profiles and the measurement of visual motion," *Computer Vision and Pattern*, 1988.

[31] C. C. Pack, "The aperture problem for visual motion and its solution in primate cortex," *Science Progress*, vol. 1, pp. 255–256, 2001.

[32] P. Yosifovich, *Windows Presentation Foundation 4.5 Cookbook*. Birmingham: Packt Publishing, 2012.

[33] T. Delbruck and M. Lang, "Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, p. 223, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2013.00223

[34] J. Conroy, G. Gremillion, B. Ranganathan, and J. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous Robots*, vol. 27, pp. 189–198, 10 2009.

[35] C. Higgins and V. Pant, "A biomimetic VLSI sensor for visual tracking of small moving targets," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 12, pp. 2384–2394, 2004.

[36] S. Mehta and R. Etienne-Cummings, "A simplified normal optical flow measurement CMOS camera," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 6, pp. 1223–1234, 2006.

[37] A. Stocker, "Analog VLSI focal-plane array with dynamic connections for the estimation of piecewise-smooth optical flow," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 963–973, 2004.

Rafael de la Rosa-Vidal Received the B.Sc. degree in electronics engineering from the University of Cadiz, Cadiz, Spain, in 2017 and the M.Sc. degree in microelectronics from the University of Seville, Seville, Spain, in 2020. His work is supported by the Spanish Government through FPU under Grant FPU20/01561. Before he ended the B.Sc. degree, in 2016, he started working as an R&D engineer in a start-up. From July 2018 to September 2020, he worked as a Flight Test Ground Station engineer in Airbus Ground Station (Seville) as subcontracted personnel. He was the main developer of several R&D projects, one of which was exposed in the European Telemetry and Test Conference (ETTC) in 2019. Since October 2020, he undertook a Ph.D. degree in microelectronics in the Microelectronics Institute of Seville. His researching is focused on Address Event Representation (AER) vision systems. Other research interests for him include embedded systems, smart sensors and biomedical systems.

Juan A. Leñero-Bardallo (Member, IEEE) received the M.Sc. degree in telecommunications engineering and the Ph.D. degree in microelectronics from the University of Seville, Seville, Spain, in 2005 and 2010, respectively. After the completion of his Ph.D., he served in several academic institutions and worked for the semiconductor industry at Chronocam inc. From September 2010 to March 2010, he worked as a Postdoctoral Associate at Yale University, New Haven, CT, USA. From March 2010 to August 2013, he was a Postdoctoral Associate at the University of Oslo. From September 2016 to January 2018, he was an Assistant Teacher at the University of Cádiz, Spain. Since February 2018, he is an Associate Professor at the University of Seville, Spain. His main research interests include Address Event Representation (AER) vision systems, frame-based vision sensors, smart sensors, and biomedical systems. He was the Financial and Local Chair of IEEE ISCAS 2020.

José-María Guerrero-Rodríguez received a B.Sc. degree in Electronic Engineering from the University of Cadiz (Spain) in 1987 and a B.Sc. degree in Physics, specialized in Electronics, from UNED University (Madrid, Spain) in 1999. He worked for several electronic sector companies as test-engineer or R&D engineer. Later, he received a Ph.D. in Industrial Electronics from the University of Cadiz (Cadiz, Spain), in 2009. He joined the Engineering School (University of Cadiz) as a professor in the Electronic Area of the Department of Systems Engineering and Electronics, in 1997. His research is focused on electronic instrumentation and sensors devices and AI techniques application on Intelligent Instrumentation.

Ángel Rodríguez-Vázquez (Life Fellow, IEEE) received the Ph.D. degree in Physics-Electronics (Universidad de Sevilla, 1982) with several awards, including the IEEE Rogelio Segovia Torres Award (1981). After stays at the University of California-Berkeley and Texas A&M University, he became a Full Professor of Electronics at the University of Sevilla in 1995. He co-founded the Instituto de Microelectrónica de Sevilla, a joint undertaken of Consejo Superior de Investigaciones Científicas (CSIC) and Universidad de Sevilla, and started a Research Lab on Analog and Mixed-Signal Circuits for Sensors and Communications. He has always been looking for a balance between long-term research and industrial innovation. In 2001, he was the main promotor of AnaFocus Ltd. and served it as CEO until June 2009, when the company reached maturity as a worldwide provider of smart CMOS imagers. He also participated in the foundation of the Hungarian start-up AnaLogic Ltd. He has ten patents filed; AnaFocus started based on his patents on vision chip architectures. His research embraces smart imagers, vision chips, and biomedical circuits, always with an emphasis on system integration. His Lab designed many high-performance mixed-signal chips in the framework of Spanish, European, and USA R&D programs. These included three generations of vision chips, analog front-ends for XDSL MoDems, ADCs for wireless communications, ADCs for automotive sensors, chaotic signals generators, complete MoDems for power-line communications, etc. Many of these chips were state-of-the-art in their respective fields. Some of them entered massive production. He also produced teaching materials on data converters that were delivered to companies and got the Quality Label of EuroPractice. His publications have some 9,900 citations and several awards: the IEEE Guillemin-Cauer Best Paper Award, two Wiley's IJCTA Best Paper Awards, two IEEE ECCTD Best Paper Awards, one IEEE-ISCAS Best Paper Award, one SPIE-IST Electronic Imaging Best Paper Award, the IEEE ISCAS Best Demo-Paper Award, and the IEEE ICECS Best Demo-Paper Award. He has an h-index of 49 and an i10-index of 192 (Google Scholar). He got the 2019 Mac Van Valkenburg award of IEEE-CASS. He has served as Editor for IEEE and non-IEEE journals and is on the committee of several international journals and conferences. He chaired several international IEEE (NDES 1996, CNNA 1996, ECCTD 2007, ESSCIRC 2010, ICECS 2013) and SPIE conferences. He served as VP Region 8 of IEEE CASS (2009-2012) and as Chair of the IEEE CASS Fellow Evaluation Committee (2010, 2012, 2013, 2014, and 2015). He was General Co-Chair of IEEE ISCAS 2020.