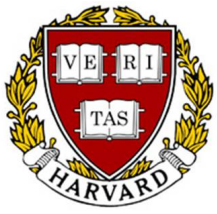


Observation of Insect Collisions

Master Project

Adrien Briod

Section of Microengineering

WINTER 2008

Assistants:
Adam Klaptocz (EPFL)

Harvard Microrobotics Laboratory
Prof. Robert Wood (Harvard)
and Prof. Dario Floreano (EPFL)

Harvard
Faculty of Arts and Sciences (FAS)
School of Engineering and Applied Sciences (SEAS)

FACULTY OF ARTS AND SCIENCE
HARVARD MICROROBOTICS LABORATORY
60 Oxford st.
Cambridge, MA 02138

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

DIPLOMA PROJECT --- WINTER 2008 / 2009			
Title :	Observation of insect collisions		
Candidate :	Adrien Briod	**Section :**	Microtechnique
Professor :	Dario Floreano		
Assistant 1 :	Adam Klaptocz	**Assistant 2 :**	Robert Wood

Project Summary

The goal of the project is to realize a setup for fly collisions observation. The flies are usually very good at avoiding obstacles and maneuvering in complex environments. But, collisions do occur sometimes, and the insects are very good at coping with them and at stabilizing their flight. From the fly collision observation, the goal is to be able to build more robust bio-inspired flying robots in the future.

A setup composed of 6 high-speed cameras is used to track the 3D position and orientation of the fly at 500Hz, thanks to reflective markers glued on the fly (see Figure 1).

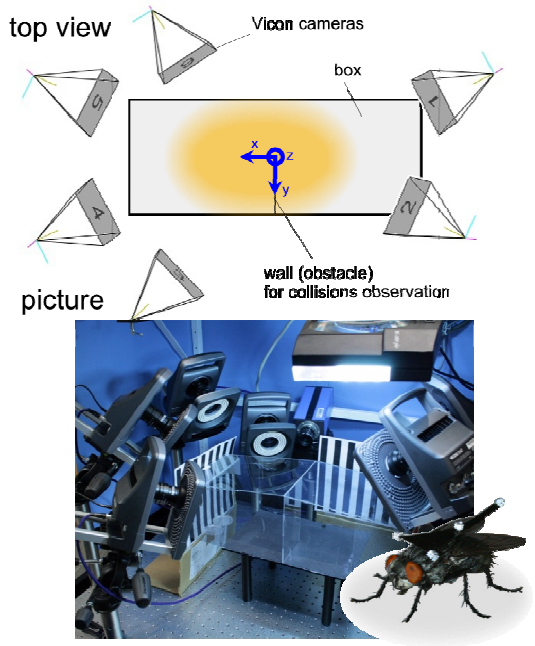


Figure 1: Diagram and picture of the setup for high-speed position tracking. Right: Real fly with a fly suit on which four reflective markers are glued.

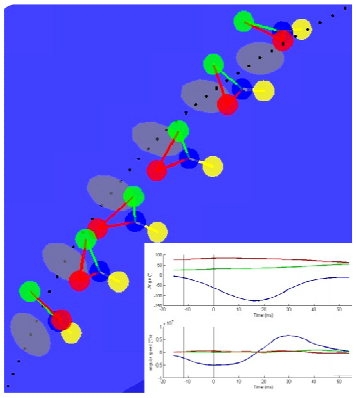


Figure 2: Example of a trajectory of the fly, tracked thanks to the setup shown in Figure 1. Flight data such as linear and angular accelerations in the body axis are computed.

The setup allows to capture sequences automatically as soon as it is detects that the fly moves. Hundreds of automatic captures were recorded that way, and more than fifty flights could be analyzed with a custom program. In addition to the 3D data, video images are recorded by another camera.

It has been found that the fly uses its legs a lot during collisions. As soon as the wall is hit, the fly extends its legs and tries to grab the wall either to land on it, either to slow down and damp the collision so that it has a more stable attitude when bouncing off the wall. Also, to stabilize its flight, the fly has approximately twice more control on the roll axis, with the ability to produce angular accelerations that can go up to $7 \cdot 10^5$ °/s^2. In general, it takes around 70ms for the fly to recover from a collision (i.e. stabilize the angular speeds), which corresponds to about 13 wing beats, and the average accelerations generated for stabilization right after the collision are: around the yaw and pitch axis: $8 \cdot 10^4$ °/s^2, and around the roll axis: $16 \cdot 10^4$ °/s^2.

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR

LABORATORY OF INTELLIGENT SYSTEMS (LIS)
CH - 1015 LAUSANNE

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROJET DE MASTER

Titre: Observation of Insect Collisions

Candidat(s): Adrien Briod (MT)

Professeur: Dario Floreano

Assistant 1: Adam Klaptocz **Assistant 2:** Robert Wood

Donnée & travail demandé:

Some flying insects are remarkably adept at navigating cluttered environments at high speeds. Elegant obstacle avoidance sensors and algorithms have been suggested as models for how insects operate in constrained environments., But obstacles are not always avoided: it is common to observe insects violently crashing into objects with little or no detriment to their flight performance or mechanical integrity. This project will involve staging experiments in which we induce flies to collide with objects in their environment while observing both body and wing kinematics with an off board high speed motion capture system. The end goal is to characterize the dynamics of collisions and subsequent recovery in an attempt to identify salient features (both in the system dynamics and the mechanics) necessary for rapid recovery of a flapping-wing MAV.

Remarques:

Un plan de travail (Gantt chart) sera établi et présenté à l'assistant responsable avant la fin de la deuxième semaine.
Une présentation intermédiaire (10 minutes de présentation et 10 minutes de discussion) de votre travail aura lieu le 24 octobre 2008 à partir de 13 heures. Elle a pour objectifs de donner un rapide résumé du travail déjà effectué, de proposer un plan précis pour la suite du projet et d'en discuter les options principales.
Un rapport, comprenant en son début l'énoncé du travail (présent document), suivi d'un résumé d'une page, devra être rédigé. La page de résumé (A4, seulement recto) doit comporter, en plus de la description du projet et de 1 ou 2 figures représentatives, la date, le nom du laboratoire, le titre du projet, son type (projet de master), vos noms et prénoms ainsi que ceux du professeur responsable et des assistants. Dans le rapport, l'accent sera mis sur la description des expériences et la présentation des résultats obtenus. Une version préliminaire du rapport sera remise à l'assistant au plus tard 10 jours avant la date de rendu final. La version finale sera remise au secrétariat de votre section le 16 janvier 2009 avant avant 12 heures en 3 exemplaires signés et datés.
Une défense de 30 minutes (environ 20 minutes de présentation et démonstration, plus 10 minutes de réponses aux questions) aura lieu dans la période du 2 au 3 février 2009. Vous serez jugé sur les résultats de votre projet tels qu'ils seront présentés dans votre rapport et lors de votre défense.
Tous les documents en version informatique, y compris le rapport (en version source et en version PDF), la page de résumé au format PDF et le document de la présentation orale, ainsi que les sources des différents programmes doivent être gravés sur un CD-ROM et remis à l'assistant au plus tard lors de la défense finale.

Le professeur responsable:	L'assistant responsable:
Signature: Dario Floreano	Signature: Adam Klaptocz

Lausanne, le 7 octobre 2008

Acknowledgments

I first want to thank Prof. Robert Wood and Prof. Dario Floreano, as well as Jean-Christophe Zufferey for having made my stay here at the Harvard Microrobotics Lab possible. I want to sincerely thank Rob and all the lab members for having welcomed me in the lab during these five months, and for the warm atmosphere always present around the fussball table or during the lab meetings. I also want to thank many people for the very interesting discussions and their constant will to help, especially Sangbae, Peter, Mark, Katie and Stacey. Finally, I want to thank Adam for his help with the report and for managing my project at the EPFL.

Contents

1 Introduction

Insects, such as the cockroach or the grasshopper, have already been the source of inspiration for many microrobots, especially in the field of locomotion (for example [1] or [2]). For flying insects, the fly, or rather a few particular families of flies, have already been the subject of many studies (for example in [3]). Their purposes were to understand some of this insect's mechanisms, such as reactions to external stimuli, flight control, obstacle avoidance, etc. This fascinating insect has already been taken as a source of inspiration for bio-inspired robots thanks to these researches , and will most probably continue to be.

The Harvard Microrobotics Laboratory[1] is developing a robotic fly of 60mg, and 3cm wingspan. This flapping MAV is strongly bio-inspired in its mechanical design, using for example a wing transmission similar to the fly's (see [4]). In further steps of the design of the robot, involving for example the flight control, biology can still represent a major source of inspiration, which is what motivates the present project.

In its first part, this project aims at realizing a setup for insect flight observation. In the second part, the flight control of insects enduring collisions will be studied, with the goal of providing a source of inspiration for future robotic designs. The insects are usually very good at avoiding obstacles and maneuvering in complex environments. However, collisions do occur sometimes, and the insects are very good at coping with these strong disturbances and at stabilizing their flight without falling on the ground. The study of insect collisions should eventually help in the design of more robust robots, that could allow imperfect obstacle avoidance and recover from collisions with obstacles that could not be avoided.

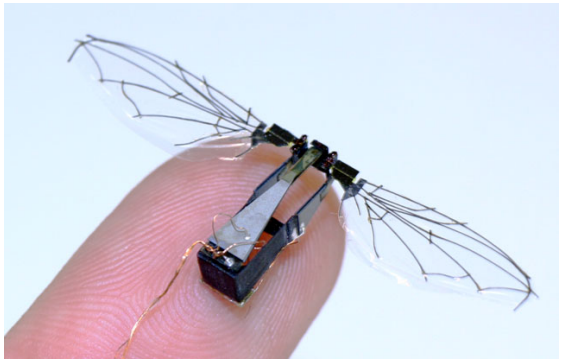


Figure 1: The 3cm wingspan robotic fly built at the Harvard Microrobotics Lab, capable of liftoff with external power.

[1] http://micro.seas.harvard.edu

2 Tracking Setup

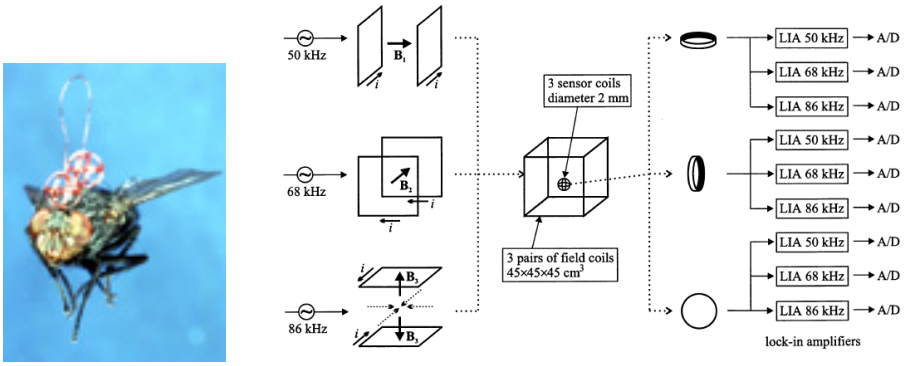
In this chapter, the choice of the hardware equipment used in the experimental setup is discussed. The observation of collisions implies measuring (or tracking) at high speed the position and orientation of freely flying insects. First a state of the art of the setups used for insect observation is presented. Then the chosen solution is discussed and its different elements are presented.

2.1 Setup choice

2.1.1 State of the art

Some existing experimental setups built in an attempt to better understand the fly's behaviors, such as setups involving high-speed position capture or high-speed stimulus generation are presented in this chapter, with an emphasis on experiments involving free flight.

- *Flying Eye*
  This setup built by J.H. van Hateren and Kees Schilstra can record the position and orientation of the fly with an accuracy of 1 mm and 0.5° respectively, at a frequency of 1kHz (see [7]). The sensors consist of small coils glued on the fly and linked by tiny wires to some electronic equipment (see Figure 2). The fly studied is the blowfly (*Calliphora vicina*), which is a fairly large fly. This setup was used for example to study flight maneuvers (in [8]) or the movement of the fly's head relative to the fly's body, which showed in particular that the fly tends to turn its head faster during saccades so that the blur due to the rotation is shorter (see [9]).



(a) (b)

Figure 2: (a) Fly with coils glued on its head and thorax, used in the Flying Eye experiments by J.H. van Hateren and Kees Schilstra. (b) Diagram of the setup allowing to measure the position and orientation of the coils glued on the fly. Three magnetic fields each at different frequencies induce voltages in each coil, which are measured to infer the 3D coordinates and orientation. Images taken from http://hlab.phys.rug.nl/demos/flying_eye/ and [7]

- At the *Drosophila flight control lab*[2] at ETHZ, a free-flying parasitoid fly's 3D position can be tracked in real-time thanks to two pan-tilt cameras (see [10]). The setup allows to change auditory stimuli in function of the fly's position. The advantage of using pan-tilt cameras is that they can cover a large space with a good resolution, thanks to the capacity to orient the camera towards the target in real-time. The small fly (about half a centimeter) was tracked in a room of several cubic meters. The position, but not the orientation of the fly, were obtained at 50Hz thanks to the *Trackit 3D* software. A scheme of the setup is shown in Figure 3.

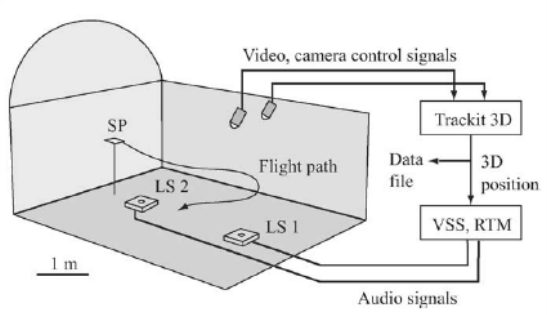


Figure 3: Setup allowing 3D position tracking at 50Hz thanks to two pan-tilt cameras, with the possibility to change auditory stimuli in real-time in function of the fly's position. Image taken from [10]

- *Fly-O-Rama*
  This setup from *Dickinson Lab* at Caltech[3] is composed of 2 cameras that can track the 3D position of a fly at a rate of 30Hz, in a 1m diameter cylinder (see Figure 4). Different patterns can be displayed on the wall that is made up of an array of LEDs. For example, this setup was used in [14] to show that the fly navigates in straight lines, and avoids obstacles with 100ms saccades triggered by asymmetries in the optic flow.

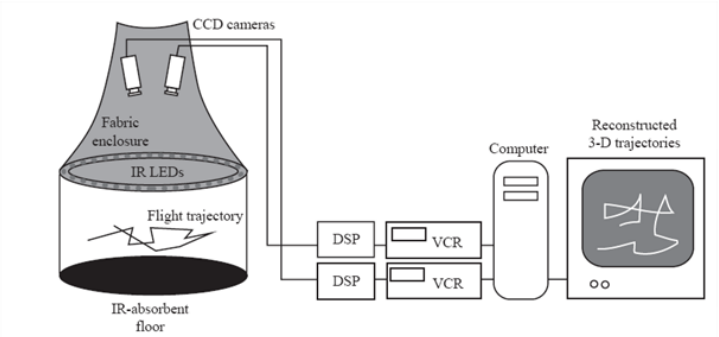
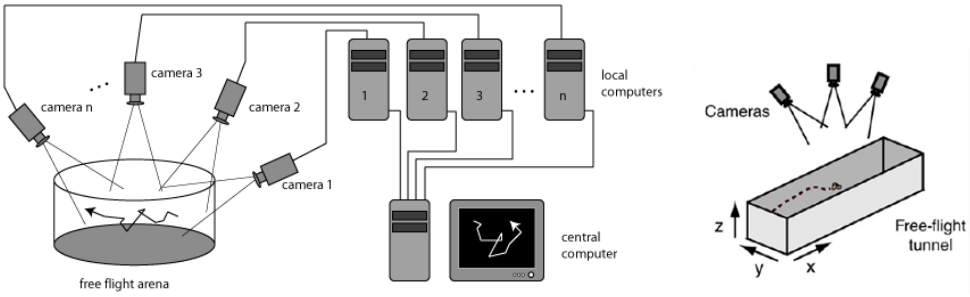


Figure 4: Fly-O-Rama: Two-cameras setup, tracking at 30Hz the 3D position of a *Drosophila melanogaster*. Online and post processing generate the 3D trajectory. Image taken from [14].

[2] http://fly.ini.unizh.ch/joomlas/
[3] http://www.dickinson.caltech.edu/

- *FlyDra*
  This setup from *Dickinson Lab* is composed of 5 cameras and can track in real-time the 3D position and orientation of a fly at a rate of 100Hz. Each camera is connected to its dedicated image processing computer, and the final 3D reconstruction is done on a central computer (see Figure 5(a)). The possible tracking volume is in the order of a 1m diameter cylinder with a height of 1.5m. The same setup is also used to track the trajectory of flies in a 31x31x86cm box, to observe the free-flight behavior in presence of posts of different heights (see Figure 5(b) and [13]).



(a) (b)

Figure 5: (a) FlyDra : setup allowing 3D position and orientation tracking at 100Hz in real-time, thanks to five cameras taking 656x491 px images. Image taken from http://www.dickinson.caltech.edu/Research/MultiTrack. (b) Example of experiment in a 31x31x86cm box, taken from [13]

- *FlyTrack*
  This setup from the *Drosophila flight control lab* allows the real-time 3D position tracking of *Drosophila melanogaster* flies thanks to two cameras, and can display visual stimuli in closed-loop (see [11]). The tracking is done at 50Hz with the *Trackit 3D* software. The insects fly freely in a 30x30x98cm wind tunnel, in which vinegar vapor can be vaporized to attract the fly upwind (see Figure 6).

- A setup of Steven N. Fry composed of 3 high-speed cameras can record flight sequences at 5,000Hz, and afterwards reconstruct the 3D position and orientation of the fly's body and wings. The forces applied by the wings can be inferred from M. H. Dickinson's *Robofly*, an up-scaled robot of the *Drosophila*. For example, in [17], two flight maneuvers are analyzed, and it is shown that very subtle modifications in wing motion are sufficient to make the fly turn. Also, it is shown that the fly has to apply a torque to turn and apply an opposite torque to stop turning, which means that the dynamics of the fly's body are determined by body inertia and not the viscous friction of the air acting on it.

- Other setups have been used to record at high-speed the wing movements of tethered flies. The very small required capture volume allows image capture at higher speeds . For example, the system *DigiWBA* from the *Drosophila flight control lab* can track in 2D the wing motion with a precision of 1° at 6250Hz (see [12]). This

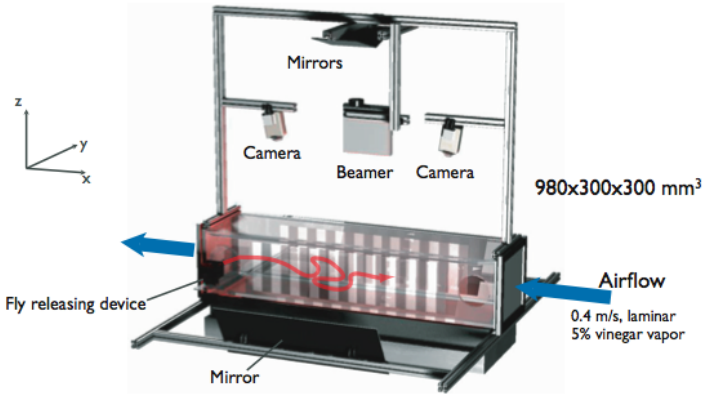


Figure 6: *FlyTrack* : setup allowing 3D position tracking at 50Hz thanks to two cameras, with the possibility to change in real-time visual stimuli in function of the fly's position. Image taken from http://fly.ini.unizh.ch

is done thanks to real-time image processing and a kalman filter that position the ROI (Region of Interest) of the camera for each frame (see Figure 8(a)). This allows the use of a small ROI (about 3,600px total) and therefore increases the frame rate. The *Fly-O-Vision* setup of *Dickinson Lab* can also track wing motions in 2D thanks to an infrared diode and an optical wing beat analyzer (see Figure 8(b)). In [15], it allowed to update in real-time a LED display in function of the wing stroke amplitude, and to create a virtual reality environment for the tethered fly (if it tries to turn left, the display will shift to the right).

2.1.2 Discussion

Depending on the study objectives, the tracking systems presented are used to know the position of the fly with different levels of precisions. In certain cases the orientation or even some body part positions are extracted as well. Apart from the *flying eye* experiment, the tracking of the free flies is done by optical means. This can be explained by the versatility of setups with multiple cameras, since they can be used for large or very small volumes. Also, the increasing power of recent video cameras in terms of framerate and image resolution facilitate high measurement precisions. For example, a free flight can be tracked fairly precisely in 3D at the impressive speed of 5,000Hz thanks to video images (see [17]).

The latter experiment was limited in its capture volume, because only a small window of the video camera's sensor can be read at high-speed. For a given resolution, there is indeed a compromise that has to be done between the capture volume and the framerate: if the speed is increased, the resolution decreases. The capture volume can be increased by using different optics for example, but the resolution then decreases.

Some setups allow the tracking to be done in real-time, but this feature is limited by the data rate at which the cameras can transmit the images to a computer. In fact, many

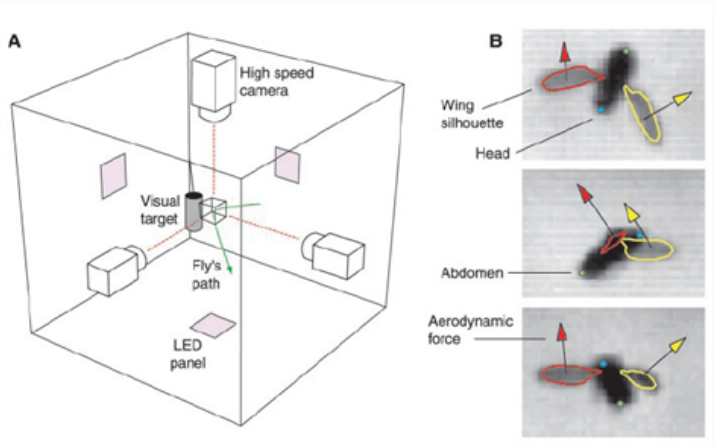
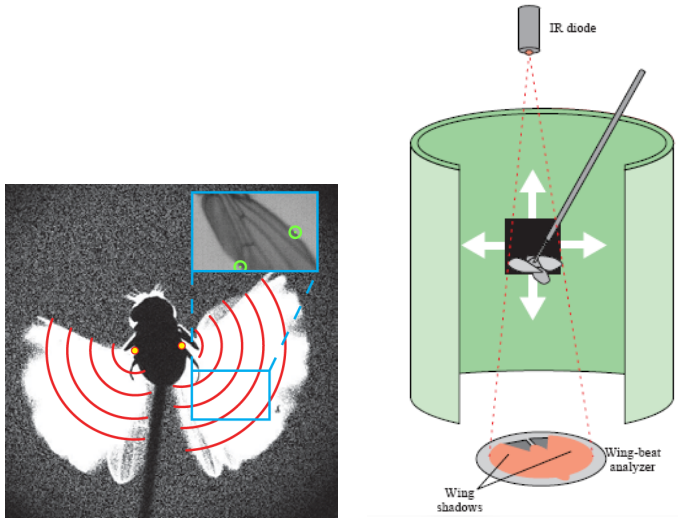


Figure 7: Setup allowing a high speed (5,000Hz) recording of short flights in a small capture volume, and used to analyze the aerodynamics of free flight control. Thanks to the 3 images of 146x88 pixels and manual adjustment of the position of the body parts on the images, the position of the body and the wings can be reconstructed in 3D. Image taken from [17].



(a) (b)

Figure 8: Two examples of wing tracking. a) *DigiWBA*: A camera is used to track at 6250Hz the wing edges in 2D. The high-speed is obtained thanks to a small ROI that is repositioned at each frame thanks to a Kalman filter prediction. Image taken from http://fly.ini.unizh.ch/ b) The shadow of the wings cast by an infrared diode is tracked in real-time thanks to a wing beat analyzer, which allows to close the loop and refresh the display in real time in function of the fly's action. Image taken from [15]

high-speed cameras have a huge internal buffer because they record more data than they can transmit when used at full speed. Therefore, the video-based real-time tracking systems presented in the previous chapter are relatively slow (such as *FlyTrack*), or have limited regions of interest (such as *DigiWBA*), or use multiple computers (such as *FlyDra*) to distribute the task of image processing and to reduce the amount of data re-

ceived by the central computer. Cameras allowing very high framerates usually store the recorded images in internal RAMs so that saving the data is very fast (unlike streaming the data to a computer), but this does not make real-time applications possible.

Using optical means to track the fly's position has the advantage to be passive, if we exclude the fact that an important amount of light has to be provided to the capture volume. Using video cameras for position tracking also has the advantage that it provides more information than just the position : it provides images of the flight from several points of view, and images contain a lot of information. Humans, unlike computers, can understand very easily a video, and it is useful to understanding a flight intuitively. In addition, if the data is not used in real-time, the images can be used to manually improve the tracking if the image processing did not work properly. Also, a video aids in understanding how the fly uses its different body parts, or to clear an ambiguity that cannot be understood simply with the tracking.

The tracking of the position can be done accurately quite easily with cameras, but a precise orientation measurement by image processing is a very difficult task (in particular the measurement of the roll), especially if the images are not of a high resolution. Indeed, it would imply the continuous identification of precise features on the fly's body to infer its orientation. This would need a sharp, high-resolution image and a fair amount of image processing. In addition, it would not work well in cases where the image is slightly out of focus.

The *flying-eye* experiment, which uses active magnetic fields to track the position of the fly, requires the sensor to be glued on the fly and have it constantly connected to some equipment with tiny wires. This is only possible for large species of flies, but it offers a good precision, especially for the orientation.

Since the fly's visual system is only sensible to short-wavelength visible light, light sources with high-wavelengths, such as infrared, are usually used so that it does not affect the fly.

2.1.3 Chosen solution

The insect chosen for the study is the fly, because of its similarities with the robotic fly. To observe collisions, the insect has of course to be able to fly freely. Therefore, a system allowing to record precise free flight sequences is needed. The capture volume has therefore to be large enough to track part of the flight before and after the collision, and also to be fast enough to catch the high accelerations occurring during collisions. In addition, the position and attitude of the insect in the space is one of the most interesting pieces of information to understand some dynamic aspects of the flight and the collision. As we saw, measuring the orientation by optical means without putting a sensor on the fly is very difficult.

The chosen solution is an off-the-shelf setup of cameras, that are optimized for 3D tracking. The solution is different from all other camera solutions presented in the

previous chapter, because the system does not try to find the position of the fly on each image, but rather the position of small reflective markers that are glued onto the fly. This has the following advantages and drawbacks :

- *Advantages*

  - The image processing required to extract the markers from an image is very simple because they appear as very shiny blobs, and the blob centroids detection is pretty accurate. Also, the use of the markers allows a much more precise measurement of the orientation of the fly than it is possible to achieve by image processing on marker-less fly images. Indeed, the markers are clearly identified features, compared to the ones that have to be extracted by feature extraction techniques, and if three or more markers are present, the orientation can clearly be inferred. Finally, it is still possible to detect accurately the shiny markers even when they are a bit blurry, which allows to use the system for larger volumes (the centroid of a slightly blurry marker will still be correctly placed at the center of the marker).

  - The data reduction process (i.e. extracting the blob positions from the image) is simple, therefore it can be carried out by simple processors embedded into the camera itself. This is a key point, allowing to limit the data that has to be transmitted to the central computer, which is the limiting factor of the real-time setups. It also decreases dramatically the data that has to be stored on the computer.

  - Since the size of the image captured by the cameras is not limited by the maximal data transmission rate to the central computer (because some data reduction happens before), the image sensors can have a high resolution (several megapixels) and a high framerate ($> 100 fps$) and still provide tracking in real-time (rather than data transmission rate, the limiting factors are probably the reading of the sensor or the on-board processing).

  - Using an off-the-shelf solution allows to quickly obtain an operational tracking system, as opposed to developing one from scratch.

- *Drawbacks*

  - Several reflective markers have to be glued on the fly, which is only possible for relatively large flies. This also represents additional manipulations to do before the experiments.

  - The extraction of the markers is done in real-time on the camera, and the images are then thrown away. Therefore, no video of the flight is recorded. If the marker extraction did not work properly for more than a few frames, it is hence very difficult to reconstruct the missing sequence. For example, with video images, it would have been easier to manually position the markers on the image in case of bad marker detection.

- An off-the-shelf proprietary system only allows it to be used for what it was designed to. It is for example not possible to adapt the tracking algorithms or change some camera settings other than the ones available in the monitoring software. It also means that its interaction with other elements is limited and not customizable.

As we can see, the use of markers has different advantages, such as higher precision for the position and orientation reconstruction. The on-camera image processing makes possible the real-time tracking without being limited by the large size or high framerate of the captured videos. This setup is therefore appropriate for this project's application, since we need a good tracking precision, and studying large flies is adequate, since they are similar in size to the robotic fly. Finally, the real-time capability will be very useful during this project.

Note that in applications where real-time is not needed, a setup still based on markers and on similar cameras, but using very large buffers to keep all the video frames would be interesting as well. This would allow the use of the same marker detection and reconstruction algorithms in post-processing, with the same precision, but with the advantage of having the images in addition. On the other hand, only very short sequences could be recorded because of the large amount of data generated.

2.2 Hardware

The off-the-shelf commercial system used for 3D tracking is a Vicon MX[4] solution, composed of six high-speed cameras and one Giganet to monitor the cameras and to link them to a computer. In addition, a PCO high-speed video camera is used to capture video images of the flights, since the Vicon system does not record any video. As explained before, video images are still very useful, for an intuitive understanding of the flight or for information on how the body parts are used for example.

The Vicon solution and the PCO camera have different interfaces to third-party software or equipment. This is important because in our case, we want to synchronize the video camera with the Vicon system, as well as the start and stop of the recording. For that purpose, a custom software has been developed to monitor simultaneously the PCO camera and the Vicon system. This program also uses real-time information about the 3D tracking, provided by the Vicon monitoring software, which enables the recording to be automatically triggered in function of the fly's position.

Figure 9 shows an overview of the different modules of the capture setup.

2.2.1 Vicon system

The Vicon system is composed of six high-speed cameras and one Giganet (see Figure 10). The Giganet deals with the communication with the cameras, their synchronization,

[4] http://www.vicon.com/products/viconmx.html

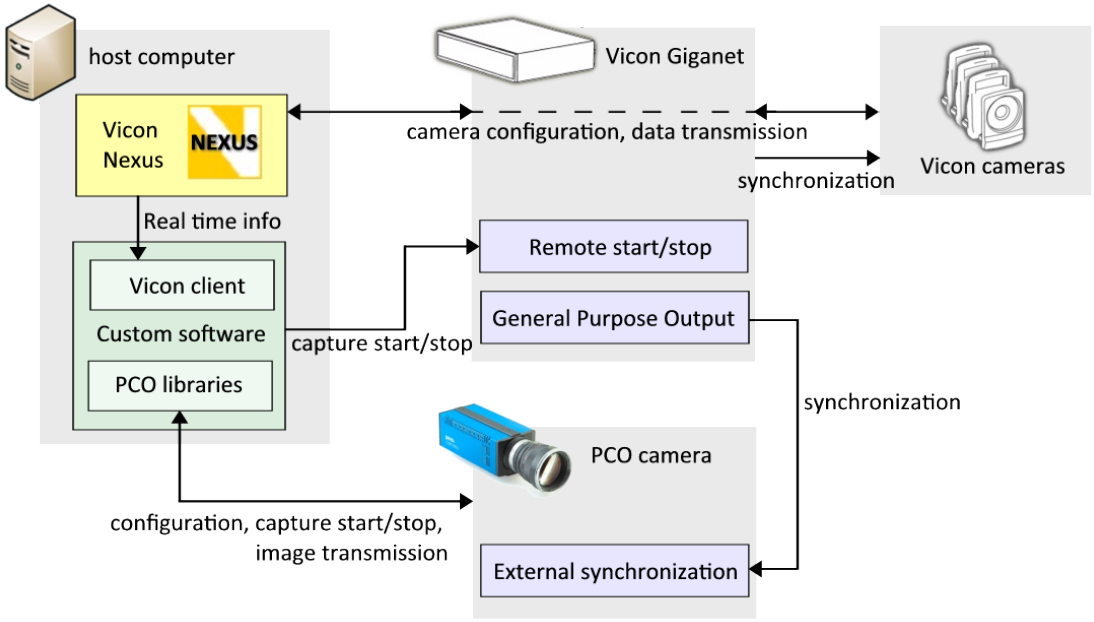


Figure 9: Overview of the capture setup. The PCO camera is synchronized with the Vicon system, and the start/stop of the recording of both the Vicon tracking and the PCO camera is monitored by the custom software, which uses the real-time information from Vicon Nexus.

and with the data transmission to a host computer. It can manage up to 10 cameras, but only 6 were used in our setup. The tracking is based on spherical retroreflective markers put on the subject, and powerful light sources (strobes) illuminating the capture space from each camera. The parameters are then adjusted so that, if possible, only the very bright reflective markers are clearly visible on the image, and can be easily detected. Each marker seen by at least two cameras can then be tracked and reconstructed in 3D (assuming a good calibration beforehand). A Vicon proprietary software, Vicon Nexus, offers a large range of functions to operate the cameras, configure the tracking, capture, reconstruct and analyze the data, etc. The data is available directly in real-time, and it can also be captured (saved) for post-processing.

The frame rates and resolutions of the two different cameras we used are shown in table 1. Their framerate can go up to 2,000 fps accordingly to the manual (see [21]), but the frame rate of the whole system is limited by the Vicon software on the host PC. In our case, Vicon Nexus supports a maximum frame rate of 1,000fps. At high frequencies, only a fraction of the sensor's lines can be read, and the image height is automatically lowered, as shown in table 2. The windowing drops automatically the outside lines of the image, and it is not possible to choose the window manually. We remark that at high frequencies, the field of view of the TX160 camera is dramatically reduced, with for example only a quarter of the original field of view at 500Hz, formed by a 4704x827px image.

The Vicon system is typically used for body motion tracking in life science, or to reproduce certain motions in video games or movies. Using the Vicon system to capture

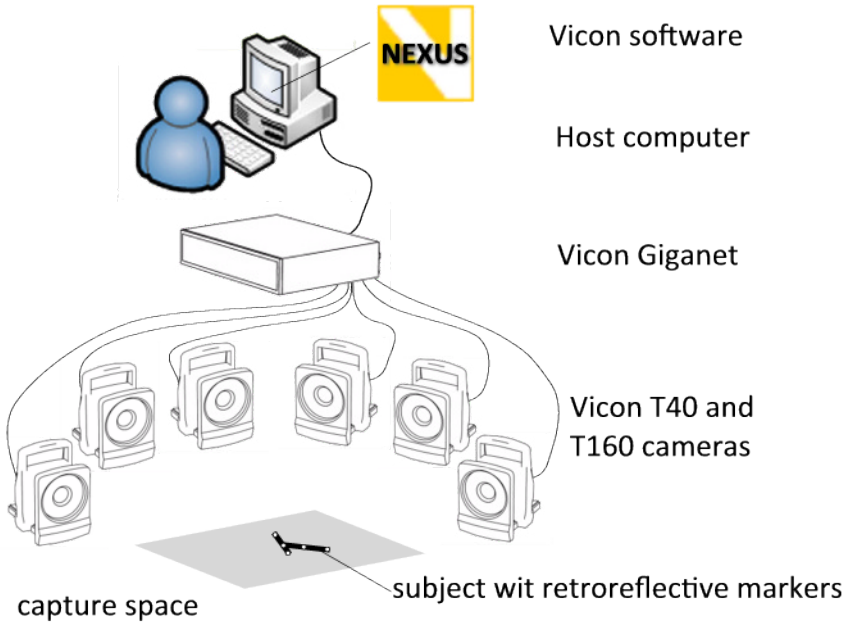


Figure 10: Hardware setup constituting a full Vicon MX tracking solution. Reflective markers are tracked by cameras, which are connected to the Giganet. Its role is in particular to synchronize the cameras, and transmit the data to the host computer. Thanks to the software on the host computer, the tracking can be configured, observed in real time, analyzed, etc..

	TX40	TX160
Resolution	2352x1728 px	4704x3456 px
Nb. of Pixels	4,064,256	16,257,024
frame rate	30-2,000	30-2,000
Max. frame rate at full resolution	370	120

Table 1: Frame rates and resolutions of the Vicon cameras used in the setup. See also datasheet in annexe page 64

fps	TX40	TX160
100	1728px (100%)	3456px (100%)
250	1728px (100%)	1280px (37%)
500	1295px (75%)	827px (24%)
1,000	527px (30%)	350px (10%)

Table 2: Maximal height in pixels (and percentage of the full field of view) of the images of the Vicon cameras, in function of the frame rate

fly motion implies some adaptation, starting with the size and manufacturing of tiny retroreflective markers. Also, the sides and height of the capture volume, that are usually of several meters, are in our case of several centimeters, which implies to position the cameras very close to each other around the volume, and to configure the optic appropriately to the small distances.

The lenses mounted on the TX40 and TX160 cameras have a fixed focal length of 18mm. The sensor sizes for the TX40 and TX160 are respectively of 16.46mm x 12.10mm and 18.35mm x 13.48mm, which gives fields of view of 49° x 37° and 53° x 40° respectively. This means that to capture a volume with a height of 10cm, a TX40 camera has to be placed further than 15cm from the capture volume. The lenses allow to focus from infinity to an object distance down to about 10cm. These lenses provide an appropriate field of view for our application, allowing to come close enough from the capture space, and still be able to focus at that close distance. Being able to position the cameras close to the scene is important, because they will get a maximum of light, which is a key point in high-speed imaging, where the exposure times are very short.

The available aperture range goes from f/2 to f/16. Having the lowest aperture possible allows to increases the depth of field, but the image is proportionally darker. Therefore, it is important to provide as much light as possible, to allow a low aperture and therefore a larger depth of field. Note that even if the image is slightly out of focus, the markers will still be identified, which does not limit the usable capture distance to only the depth of field, but rather a larger zone.

The strobes mounted on each camera are formed of 250 LEDs. They emit NIR[5] light pulses (wavelength of 780nm), synchronized with the capture of the cameras.

The cameras are equipped with 3 processors, that do a certain amount of on-board computations, which decreases the data volume sent to the Giganet and then to the host computer. Indeed, it would be too much data to send in real-time 4 or 16 Megapixel images from many cameras at high frequencies and to the same computer. Therefore, the on-board processing locates the markers in 2D and computes their centroids with a sub-pixel resolution. Then, only the data relative to each marker is transmitted further. This allows to get the data and the markers' reconstruction in real-time on the host computer.

The Giganet is linked to the host computer by a 1Gb/s Ethernet connection. In addition to powering and synchronizing the cameras, and to acting as a relay for the data going between the host PC and the cameras, the Giganet offers a few input/output possibilities for third party devices. Eight configurable outputs can provide for example custom synchronization signals for external components. Also, two input lines can receive signals to start or stop the capture.

2.2.2 PCO camera

To acquire video images of the flights, a PCO 1200hs camera is used (see Figure 11). Its main characteristics are presented in Table 3. A main feature is the possibility to trigger the capture by an external signal. This is what makes possible the synchronization with some other equipment, such as the Vicon system.

[5]Near Infrared

This camera cannot stream the images directly to the PC. The main reason is because too much data is generated when images are recorded at high-speed (up to 1 GB/s), and therefore the data rate of the firewire communication would be too slow (30MB/s in average) to send the images in real time. A "FIFO buffer mode" seems to be available for some PCO cameras, where the images are sent as fast as the communication allows, with the internal RAM of the camera used as a huge buffer. But the only mode available with the the PCO 1200hs is the "recorder mode", where the internal RAM of the camera is used to store the recorded videos, that have then to be manually transferred to the PC if necessary.

Another important feature is the SDK[6], which allows the user to build his own software to fully monitor the camera.

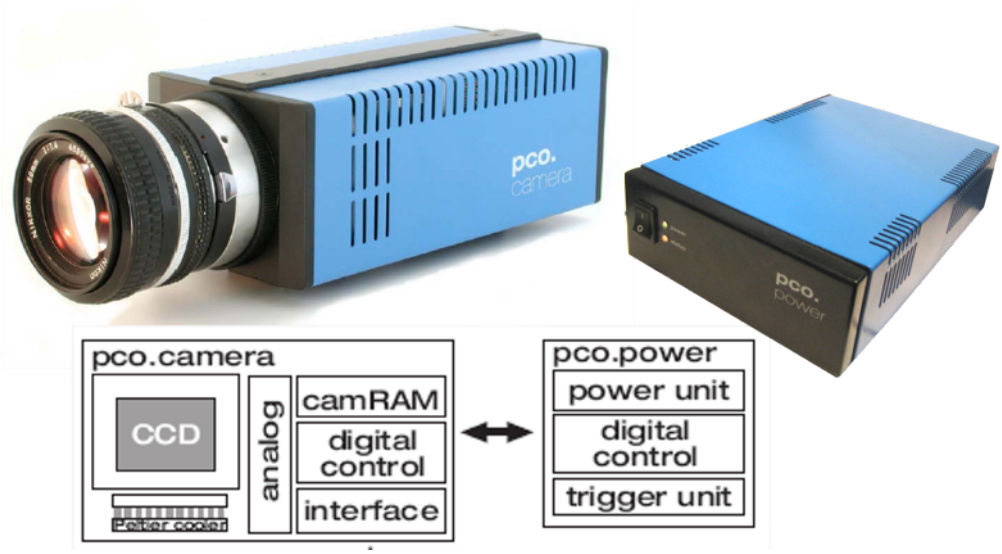


Figure 11: PCO camera and the power unit, with structural overview of the components. The camera is directly interfaced with the computer by firewire, but the external signals such as triggers for synchronization have to be provided to the power unit, which transmits them to the camera. Images taken from [22]

	PCO 1200hs
Resolution	1280x1024 px
frame rate	up to > 10,000 with windowing
Max. frame rate at full resolution	636
Exposure time range	50ns - 5s
Connection to PC	IEE 1394 (firewire)
Internal RAM	4GB
External trigger	TTL signal

Table 3: Some important characteristics of the PCO camera used in the setup. See also datasheet in annexe page 66

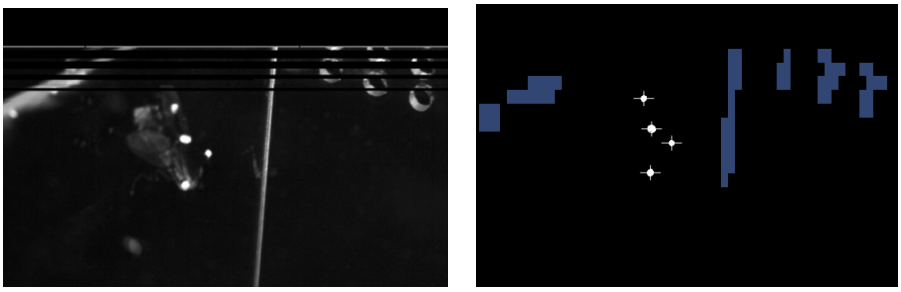
[6]Software Development Kit

2.3 Software

2.3.1 Vicon Nexus

Nexus is the proprietary software monitoring the Vicon system. It offers a large panel of features and options, whose main ones are presented here :

- *Cameras configuration*
  A few configurations are very important for efficient marker tracking. The focus and aperture can be adjusted directly on the lenses. To help with this process, a "preview mode" allows to see in Nexus the black and white image as captured by the sensor of each camera. On the software side, the threshold and the strobe intensity of each camera are the main options to configure to make the markers clearly visible. The threshold basically sets the grayscale level from which the pixels in the image are considered as markers. In addition, an option allows to mask bright spots that could not be removed from the background, so that they are not detected as markers (see Figure 12). Of course, no marker will be detected by the camera when it passes in front of the masked spot, therefore the background behind the capture volume should always be covered with dark covers when possible. Another option allows to specify a minimal circularity parameter, to discard the blobs that are not circular in shape.

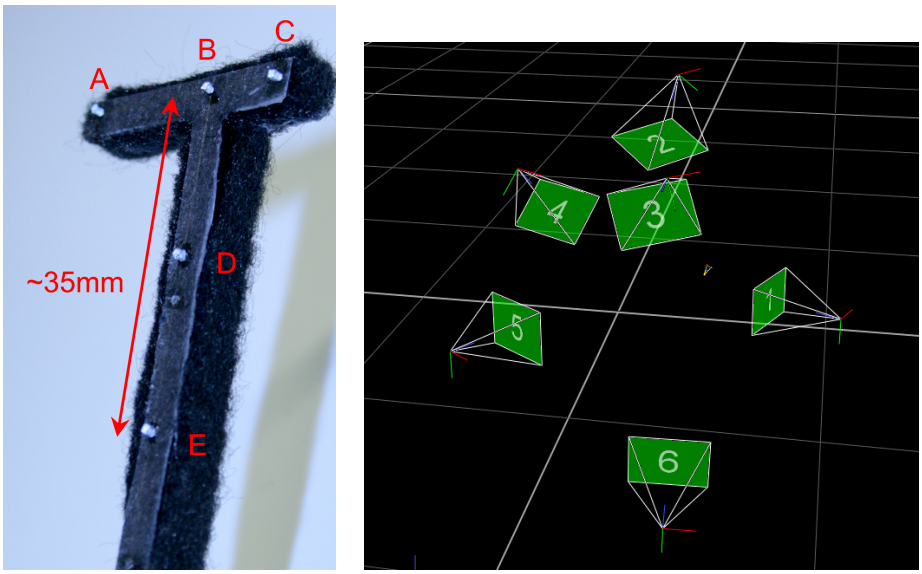


(a) (b)

Figure 12: (a) Preview image of the fly from a vicon camera that has been well configured, with a dark image except the markers that are clearly visible. (b) The markers rendered in the camera view, which also corresponds to what will be saved during a capture (as explained in chapter 2.2.1, not the black and white image is recorded but only information about the markers). For a good tracking, the markers have to be clearly detected in this view. Note that some static bright features of the background can be masked in the camera view.

- *Cameras calibration*
  Nexus manages the calibration, which is a necessary step to make possible the 3D position reconstruction. The calibration can only take place after each camera has been configured to detect clearly the markers in the capture space. This operation positions precisely in the space each camera, so that they will later be able to reconstruct in the space any marker that is seen by at least two cameras. A special wand is used for the calibration, it is an object composed of 5 markers

whose coordinates are precisely known and entered in the software (see Figure 13(a)). The wand has to be moved everywhere in the capture volume, until each camera has recognized the object in 1,000 to 2,000 frames. Since the geometry of the 5 markers wand is known, the software can position relatively to each other all cameras seeing the wand from different angles. The result of a calibration can be seen in Figure 13(b). Many frames are needed to cancel the errors, and to cover the whole space, so that the deformations due to the optics can also be taken into account. The final step is to tell Nexus where the desired origin is in the space. To do that, the wand or another object has to be positioned in the capture volume where the origin is desired. It should be placed on a flat surface, so that the z axis corresponds to the gravity direction.



(a) (b)

Figure 13: a) Object (wand) used during the calibration process, and whose markers' coordinates are precisely known. b) Result of a calibration, with all cameras positioned in the space as they are in reality.

- *Marker reconstruction and subject labeling*
  A few modes allow to choose the amount of processing that is done on the recorded data for the 3D reconstruction. The basic mode is the "Reconstruct" process, which reconstructs the markers in 3D. The position of the cameras is known thanks to the calibration, therefore rays starting from each camera can be traced for each detected blob on the images. An option allows to set the degree of precision with which the rays have to cross for a marker to be constructed.
  An additional level of processing is provided by the "Label" mode, where the software tries to fit a certain group of markers to a known subject. For this, a previous step is necessary, where the subject has to be reconstructed and each marker manually labeled so that Nexus can save the geometry of the subject. An option allows to set the level of flexibility of the subject, that tells whether it is

possible that the markers of a subject can move relatively to each other, or if the markers will stay precisely in place. Figure 14 shows an example of an unlabeled and a labeled subject.
Finally, the "Kinematic Fit" mode tries to fit at best the position of the subject in function of the labeled markers. Instead of returning the position of each marker, this process returns only the position of the origin and orientation of the subject. It is supposedly using a Kalman filter to update the position from frame to frame. Unfortunately, this mode proved to be hard to use and the results were not always satisfying. For example, if the subject is not well fitted, this cannot be manually corrected, and it influences the following frames which will most likely not be fitted correctly either. Finally, the coordinates and orientation are available through the real-time module, but no option allows to export this data in post-processing, which makes the use of this mode useless if other applications are used for data analysis.

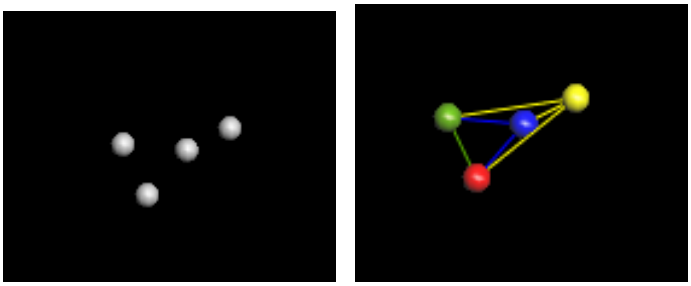


Figure 14: Unlabeled and labeled markers as seen in the 3D view. The labeled markers correspond to a subject whose geometry was previously saved thanks to a manual labeling.

- *Capture and post-processing*
  Some capture options are very interesting, such as the option to save the data a certain amount of time before the capture is actually triggered, or the possibility to control remotely the start or stop of the capture. During a capture, the blobs data sent by each camera is recorded, which allows to reconstruct the markers with different parameters afterwards.
  To work on the recorded sessions, a certain number of post-processing operations exist in addition to the reconstruction and the automatic labeling. It is for example possible to manually unlabel, or label markers that were not correctly labeled automatically in certain frames. Also, it sometimes happens that some markers disappear during a few frames. These gaps can be filled with splines, or by using other markers' trajectories. The markers' coordinates as well are some other information are stored by Nexus in a *.c3d* file, a public domain binary file format used to record synchronized 3D and analog data [7]. This file can be easily used by any other program to use the coordinates of the markers for other purposes.

- *Monitor*

[7] http://www.c3d.org/

This feature can trigger the start or stop of the capture in function of conditions on certain variables. These variables can be markers' position, speed, or acceleration.

- *Real-time server*
  Nexus creates a socket on the host computer, to which any client can connect. Then, if the processing mode is set at least to "Label", the coordinates of the the labeled markers are sent in real-time through the socket. This allows other programs to use the real-time data, which could typically be used to control the tracked subject if it is a robot. That is to say, use the Vicon system as a position sensor. Note that this is the only way Nexus can interact with other softwares, and the socket does not accept incoming commands for example.

2.3.2 Monitoring software

As shown in Figure 9, a custom software has been created to monitor the recording of the video and the capture in Vicon. It uses the real-time data sent by Nexus to know the position of the fly in the box.

The monitoring software has several motivations :

1. Automatic organization of the experiments, with the start and stop of the capture synchronized between the video camera and Vicon.
2. Data recording only when the fly is flying in the field of view.
3. Possibility to leave the setup unattended for several hours, with each flight automatically recorded.
4. Possibility to use the PCO camera in a constantly recording mode allowing to obtain the video of what happened a few seconds before the trigger.

Nexus could have been used to manage only the Vicon system, thanks to the "Monitor" feature, that can be used to automatically start and stop the captures. But the possibilities with this feature are very constrained, in the sense that only simple conditions on a few values are available (markers' coordinates and speeds) to trigger the automatic start or stop of the capture. For example it is not possible to implement a counter to avoid triggering the capture because of glitches, or to implement a filter on the variables, or to create more complex conditions.

Therefore, the custom software decides when the capture of the data should be started, thanks to the real-time data sent by Vicon Nexus. The socket created by Nexus streams the position of each labeled marker of the subjects that are recognized in the capture volume. Unfortunately, the socket does not accept any incoming command, such as ones that could simply start or stop the capture. In fact, the only way to remotely start or stop the capture is by sending electric signals to the Giganet, which implied to use an additional piece of hardware. Two output PINs of a USB to COM port converter are used to send the signals to the Giganet, as shown on Figure 15.

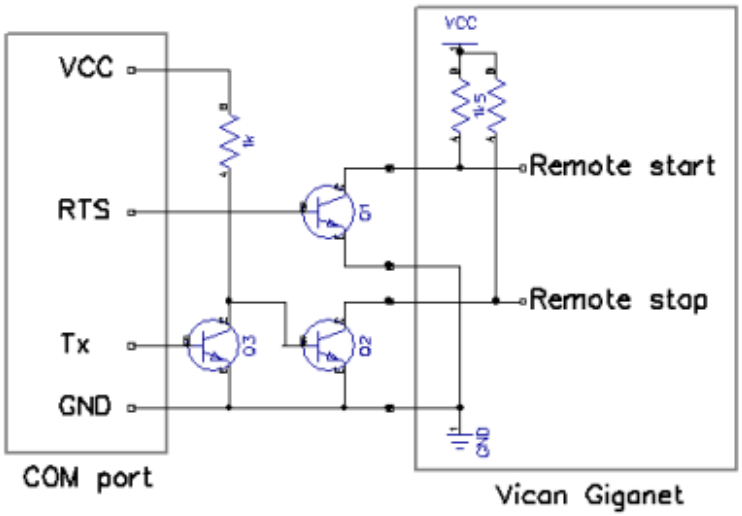


Figure 15: Electric circuit used to generate the signals necessary to remotely start or stop the capture of the Vicon system. The capture is started or stopped when the corresponding line is connected to ground. To control the Remote start, the RTS pin of the COM port is used. It can be easily controllable by software and is therefore usually kept low to deactivate the transistor Q1, and a pulse allows to start the capture. To control the Remote stop, the Tx pin of the COM port is used. Since this pin is usually high, except when data is sent, an additional transistor and resistor are used to reverse the signal. Then, sending a byte with the COM port will make the Tx pin go to zero, and activate transistor Q2 and stop the capture.

To monitor the video camera, the monitoring software provided by PCO, *Camware*, does not offer the desired possibilities. In fact, Camware only allows to manually start or stop the video capture, and once the video is recorded on the camera's internal RAM, it has to be manually imported on the PC's hard drive after each capture. Therefore the custom monitoring software fully manages the video camera to make this process automatic, thanks to the PCO libraries provided in the SDK. The frames' synchronization between the Vicon system and the PCO camera is done thanks to two signals generated by the Vicon Giganet and configured in Nexus. The outputs of the Giganet are connected by BNC cables to the "exp trig" and "acq enbl" inputs of the PCO power unit as shown in Figure 16.

To summarize, the monitoring software manages the communication with the following external elements :

1. The Vicon server, which streams real-time data when Vicon Nexus is tracking a subject.
2. A USB to COM port converter, which is used to send the remote start/stop signals to the Vicon Giganet.
3. The PCO camera, connected by Firewire.

And has the following tasks :

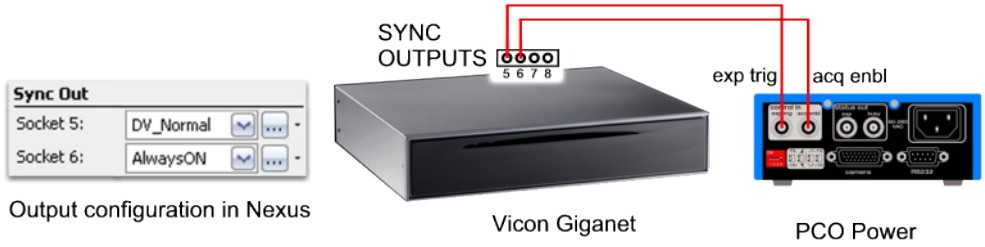


Figure 16: Connection scheme of the synchronization signals from the Vicon Giganet to the PCO power unit. The signals are configured in Nexus. "Exp Trig" is a series of pluses synchronizing the capture of the PCO with the Vicon system. If "Acq Enbl" is high, it indicates that images have to be recorded. The switches on the PCO power box have to be configured in the following fashing: up, down, up and up

1. Process the data sent in real-time by the Vicon server.
2. Trigger the start and stop of the capture of the Vicon system and of the PCO video camera when there is an interesting event.
3. Import the video recorded on the internal RAM of the PCO camera and save it so that it is organized with the Vicon captures.

The monitoring software is programmed in C++ with Visual Studio, and uses the Windows API. Here are the main parts of the program :

- *Graphical User Interface (GUI) and program core* (file: *PCOautocapture.cpp*)
  The GUI allows the user to configure the connection to the external elements, with fields to enter Vicon client's IP address and the COM port number for the remote start/stop module. The GUI also allows to set different parameters for the video recording, such as the type of compression, the capture length in seconds, the file name and folder, and the id of the next video, which is a number incremented at every capture and that is added to the file name. Since Nexus saves each trial with an incrementing id, this feature allows to synchronize the ids of both the Vicon trials and the video recordings. The GUI allows to choose the mode for the video recording, and to activate or not the automatic capture. Some information is displayed in the interface, such as the recording state, the progress of the video importation from the camera to the PC when it occurs, and a few values sent by the Vicon real-time server. A screen shot of the GUI is shown on Figure 17.
  At startup, the program initializes a certain number of threads that will manage the video camera, the vicon client and the COM port.
- *PCO camera control* (file *PCOctrl.cpp)*
  The software uses the PCO DLLs[8] documented in [23] to perform the following important configurations on the camera :
  - The correct image resolution is set. Tests showed that in "external trigger

[8]Dynamic Link Libraries

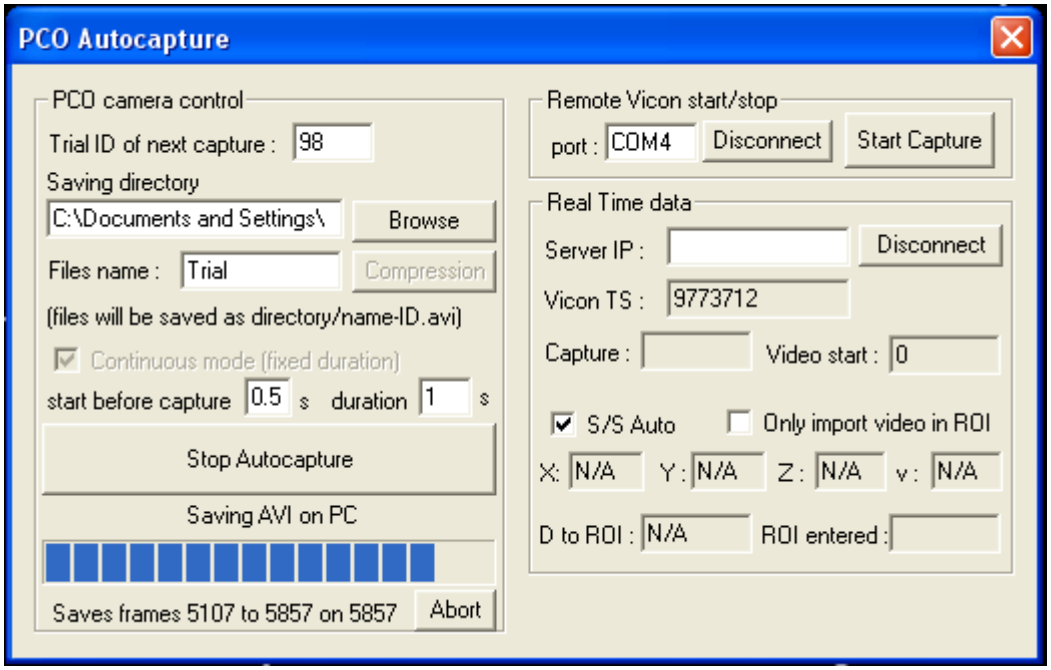


Figure 17: Screen shot of the monitoring software, allowing the configuration of the automatic captures of flights.

mode", the maximal possible resolution in function of the framerate is lower than in normal mode. At 500fps, the maximal image height is of about 500px, and at 1,000fps it is of 250px.

- The internal buffer is set as a ring buffer, which means that the camera overrides the oldest images when the memory is full.
- The external trigger is activated, in order to use the synchronization signal of the Vicon Giganet for the capture of the frames.
- The Recording mode is set to "On", which means that if the "acq enbl" signal is set to high, the camera will be recording images at each rising edge of the "exp trig" signal.

Then, two different modes are programmed :

1. If the Giganet is configured to set to high the "acq enbl" signal only when the capture is active, that means that only the sequence during the capture is recorded by the PCO video camera. In this case, the first mode of the monitoring software will automatically detect when the capture is over, and import the whole sequence from the PCO's internal RAM to the computer's hard drive. This mode could very well be used without the other features of the monitoring software (that is to say, without the automatic triggering of the capture thanks to the real-time data) and could simply be used to import automatically on the computer the videos taken during manual captures done with Nexus (instead of doing it manually each time with Camware).

2. Since the automatic capture is triggered once the fly is flying, and because of the delays in the system, some interesting parts before the beginning of the capture are not recorded with the first mode. To solve that, a second mode in which the videos are continuously recorded is programmed, the "continuous" mode. In this mode, the "acq enbl" signal has to be always high, so that the PCO is continuously recording videos. This allows to recover the video of a few seconds before the start trigger, which can be specified thanks to a text field in the GUI. In continous mode, the start and stop signals have to come from the monitoring software, because otherwise it has no indication of when the capture has started and therefore what sequence has to be imported from the video camera's memory.

While the video is imported on the computer, the recording is disabled. The program imports one image at a time and converts it thanks to a look-up table (LUT) provided by the PCO LUT API (documented in [24]). Once the image is a 3x8bits bitmap image, the video is created with the VfW (Video for Windows) functions (see *AVIFile.cpp*), adding one image after the other to the video file. The importation is relatively slow, since the camera transmits the non-compressed images. To give an idea, the size of a 1.5 seconds 1280x500px sequence recorded at 500fps (750 frame) is 512Mo, and the video takes about 50 seconds to be imported. Note that the PCO's internal memory of 4GB can store no more than 11.7s of video with these parameters. To avoid filling the hard drive with huge files, the monitoring software allows to choose a compression option. The *ffdshow-codec*[9] was generally used to convert the videos in MPEG-4, resulting in unnoticeable losses of quality, but a file size reduction of more than two orders of magnitude (files of 3-5Mo).

- *Vicon client* (file *Viconclient.cpp*)
  Part of this code is based on the samples provided with the Vicon SDK, realizing the connection to the Vicon socket and the extraction of the real-time position data sent by the server. Then, the speed of each marker is computed for each frame, using the previous frame position. A low-pass filter is applied to the speed value, since derivatives are very sensitive to noise. In automatic mode, knowing the speed allows to start the capture automatically when the fly speed goes above a certain threshold (typically 250 mm/s). To avoid unnecessary captures due to glitches, the recording is started after a certain number of frames where the speed is above the threshold (typically 25 frames). An options allows to start the capture only when the fly is at a certain distance of a point of interest (such as the place for which the focus of the video camera has been adjusted). This option is useful in order to reduce the number of Trials recorded and only keep the ones where the fly was flying at a certain place, but tests showed that interesting behaviors could be tracked in the whole capture space and this option was usually not used.
  In continuous mode, where the video of a few seconds before the start signal is

[9] http://sourceforge.net/projects/ffdshow/

saved, there is a configuration to do in Nexus to also save the tracking of a few seconds before the start signal (see Figure 18). To have the video and the tracking starting at the same instant, the number of seconds in the "Capture Before Start" field of Nexus should be the same as in the monitoring software.

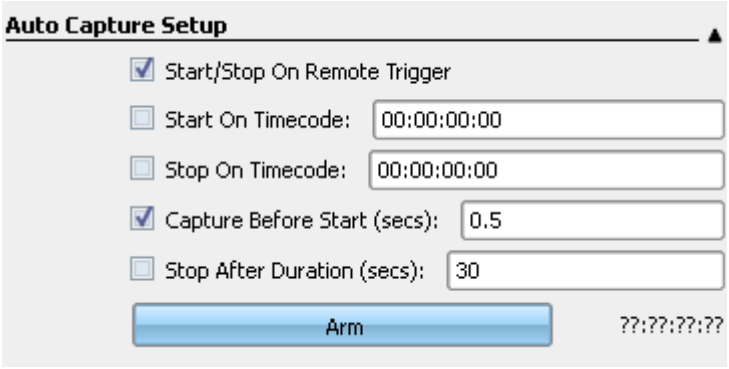


Figure 18: Screen shot of the options that should be used in Nexus for the auto capture. In this case, Nexus will continually save the last seconds of tracking in a buffer, in order to save 0.5s of tracking before a capture is started. This very useful option in Nexus allows to capture the tracking of the flights entirely and should be used with the same value as in the monitoring software, that manages to implement the same feature but for the PCO video camera.

The stop of the capture can be done after a certain number of frames where the fly speed is around zero, which means the flight is over, or after the fly has quit the capture volume. Unfortunately, this does not work well when the option "Capture Before Start" is used in Nexus. The reason is that once the capture starts, Nexus has to save on the hard drive the tracking data present in the buffer (with the tracking of the few seconds before the start) and the data arriving continuously during the capture. This usually makes Nexus lag and it does not send the position of the fly to the monitoring software in real-time anymore. In addition, while it is saving the tracking of the sequence before the start signal, Nexus will send again the position data of this sequence through the real-time socket, which is clearly not the real-time position and could even fool the monitoring software with data where the fly is immobile (since it is a few moments before the flight is detected). It has not been determined if this behavior is a bug or a feature in the real-time module of Vicon Nexus.
For these reasons, in continuous mode, the real-time data is only used to trigger the start of the capture, and the stop signal is simply generated after a fixed amount of time, calculated by a timer (and that can be configured in the interface). The start and stop signals are using the functions of the file *Viconclient_SS.cpp*, which manages the COM port that sends the signals to the Vicon Giganet.

3 Experiments

This section presents how the elements presented in chapter 2 are used to realize the fly collision observations, starting by a presentation of the flies studied in this project.

3.1 The *Sarcophaga* flies

The insect chosen for the study is the fly. First, the dimensions and weight of the robot are similar to those of flies found in different families. In addition, the fly has only one pair of wings (unlike most insects), which is also the case of the robot. Finally, the flies are easy to work with and easy to obtain.

The *Sarcophaga* fly is in the *Sarcophagidae* family (or fleshflies), which is in the Diptera order (all two-winged insects). The *Sarcophagidae* family has similarities with the *Calliphoridae* family (or blowflies). The blowfly was for example the subject of the *flying-eye* experiments (see chapter 2.1.1). The *Sarcophaga* flies were chosen for this project because they were the only large flies easy to buy in the region. The blowflies have a similar size, but were harder to obtain, and their shiny body may also have created unwanted bright reflexions that could disturb the Vicon tracking system.

3.1.1 Life cycle

The flies are usually bought at the pupae stage, by packs of a hundred or more[10]. At the pupae stage, they have accumulated all the necessary food and do not need to be fed till emergence. The pupation lasts about 10 days, during which the larvaes develop into flies. The very interesting fact about the pupae stage is that its development can be dramatically slowed down by decreasing the temperature. For example, the development can almost be paused if the pupaes are placed at a 3-4°C temperature, such as in a fridge. This can be used to delay the flies' emergence of up to 2-3 weeks. If kept longer in the fridge, the fly may not emerge, or be malformed (for example, chances are the wings would not be in good shape and the fly not able to fly). It is also advised to indicate to the supplier that fresh pupaes are desired, otherwise risks are that pupaes stored for a long time in the fridge will be shipped (fly pupaes are usually not used to produce flies, but rather to provide habitats to the *Nasonia* parasitic). Once they have emerged, the flies will first spend up to one day resting and waiting for their wings to become fully expanded.

The flies live between 2 and 3 weeks, and can be fed with water and sugar. They spend most of their time resting or walking. Once in a while they fly until they touch a few times the walls of their box. The flies, received from two different suppliers, all showed a pretty calm behavior and were surprisingly fearless, even when objects were approached quickly toward them. Indeed, even the flies that could escape from their boxes where

[10]Suppliers used: http://wardsci.com and http://www.carolina.com/

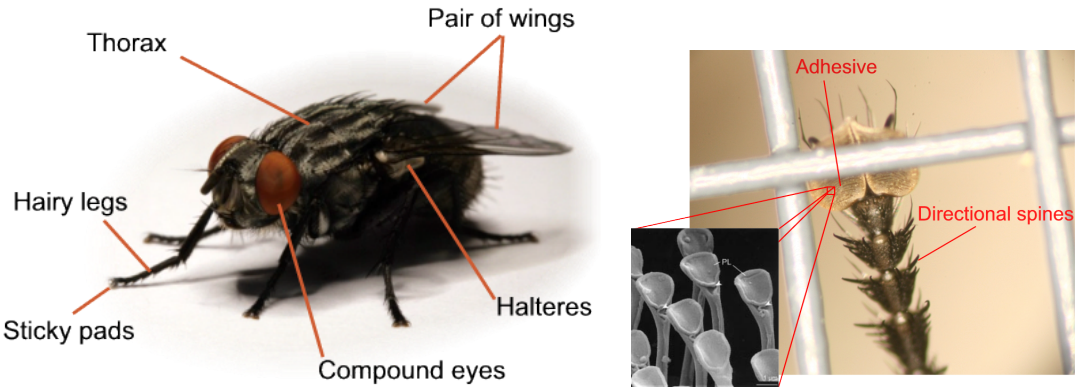
Figure 19: Picture of the flies' habitat, which provides enough space an light.

fairly easy to catch again. It is probable that wild flies would have been slightly more reactive to threats or disturbances. The flies were kept in a 30cm diameter mesh (see Figure 19) to provide them with enough space to fly a little bit. It is possible that flies that are kept in larger boxes are more likely to fly later during the experiments than the flies kept in vials (but this fact was not verified). To observe many flights, it is better if the fly is flying as much as possible, so that more data can be recorded. Attempts to excite the flies and make them fly more included : presenting different kinds of food (fruits, vinegar, insect bait, meat, ...), blowing constant wind in their box, etc. but no solution making the flies fly more than usual was found.

3.1.2 Morphology and sensing

The *Sarcophaga* measures about 13-15mm (from head to abdomen), has a 21-24mm wingspan, and weights about 70mg. The leading edge of the wings measures about 10mm. During flight, the wings are flapping at 170-180Hz. It is hard to differentiate the females from the males (except when they breed), but normally the males have slightly larger eyes. See Figure 20(a) for a picture of the *Sarcophaga* fly.

The fly's pair of eyes is a crucial sensor used for navigation and obstacle avoidance as well as for flight stabilization. The compound eyes are formed by thousands of facets, each comprising an individual photoreceptor and lens. Compared to humans, the fly's optical system provides an extremely low resolution image of a much wider field of view, but at a higher rate. Even if the visual system is low resolution, the fly is able to avoid obstacles by using the optic flow, which they can compute well thanks to the high temporal frequency of their eyes (see [5]). In particular, a fly approaching a contrasted wall can detect an image expansion, which triggers a change in direction (see [14]). In



(a) (b)

Figure 20: (a) A *Sarcophaga* fly, with some important body parts. (a) Zoom on an adhesive pad at the tip of the *Sarcophaga* fly's legs. The adhesion comes from tiny setaes (or hairs) and a liquid secretion. The zoomed image on the setaes comes from [20], a study done on the *Calliphora* fly, very similar to the *Sarcophaga* fly.

addition, the eyes are also used for stabilization and for example responses to visual disturbances were shown (for example in [18]). The latency of the response to a visual stimuli is estimated in [15] to be of about 50ms for the *Drosophila melanogaster*.

The pair of halteres is another crucial sensor for flight stabilization (see [16]). The halteres are characteristic of Dipterans (two-winged insects) and are the result of the evolution of the fly's hindwings. These small club-shaped organs beat in anti-phase with the flapping, and they provide feedback to the muscles steering the wings. During a body rotation, the halteres are deflected because of the Coriolis force. This allows to measure the body rotational speed, like gyroscopes. The haltere nerve pathway is very fast (see [18]) and makes possible extremely fast response to disturbances occurring during flight. In [19], *Calliphora* flies are studied and the latency of the reaction to rotational disturbance is estimated around 5-10ms (1-2 wing beat cycles). It is indicated that the rotation can be recognized in fractions of a wing beat cycle, but in a very coarse manner. This allows a relatively imprecise first reflex, but the correction is then refined in the following milliseconds.

The legs of the fly have a lot of large directional spines, that allow them to walk easily on rough surfaces. But at the tip of the legs, thousands of extremely tiny setae (or hairs) give a grip to any relatively flat surface. This allows the fly to climb walls or to rest on the ceiling of a room. As it is presented in chapter 4.3.1, these adhesive pads are also used a lot during collisions against walls.

3.1.3 Manipulation

To manipulate and work on the flies, they have to be anesthetized so that they do not move. An easy way to do so is to chill them, which slows the flies down, to a point

where they do not move any more. Typically, 3-5 min in the freezer is usually enough to anesthetize a *Sarcophaga* fly. Then, to keep the fly asleep for an extended amount of time (up to one hour), the fly should be kept at a low temperature, but slightly above 0°C. For this purpose, a temperature controlled stage was built, using computer cooling parts (see Figure 21). A thermoelectric cooler (using the Peltier effect) is powered by a regulator using a thermistor (temperature sensor) to control the stage's temperature. The thermoelectric stage, normally used to cool down graphic card GPUs, needs to be water cooled.

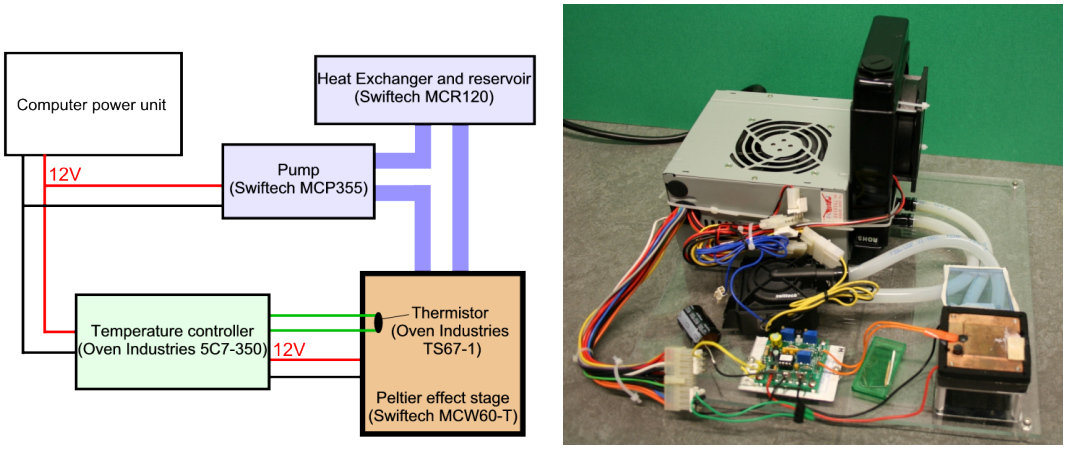


Figure 21: Scheme and picture of the cooling platform built to keep the flies at low temperature, composed of a regulated Peltier effect stage, and a water cooling system.

3.2 Fly suit

To track the fly's position with the Vicon system, reflective markers have to be attached in some way to the fly. To create these, a few materials were tested. A good reflective material should reflect the maximum of light back to its source. Reflective painting and reflective glass beads were tested, but the best material was found to be a good quality reflective tape. A reflective tape is generally composed of transparent glass beads glued on a reflective backing (see Figure 22).

The markers should be small enough not to add too much weight to the fly, but large enough to be well detected in Vicon (i.e. take at least 5-10 pixel on the image). It is also important for the markers to be visible from as many angles as possible. It is therefore not very efficient to simply paste a flat piece of tape on the fly, since it would not be seen well as soon as the camera looks from the side. An ideal marker is rather a 3D shape, ideally a sphere since it is seen as a circle by cameras looking from any orientation, and the centroid of the circles precisely indicates the center of the sphere. A similar 3D shape much easier to build with the reflective tape is a cylinder. The markers were made by rolling tape to form a cylinder, and adding a piece of tape on its top (see Figure 23).

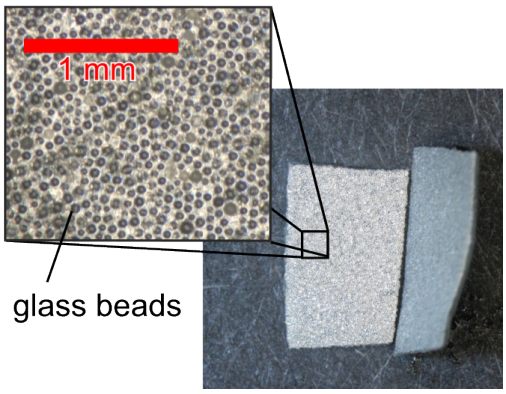


Figure 22: Reflective tape used to create the markers put on the fly. It is formed by a layer of glass beads kept in place in front of a reflective backing. The tape still appears very bright even when it does not face the light source.

The first trials were done by gluing 3 markers on the thorax of the fly (see Figure 24(a)). The recognition of the markers by the Vicon video cameras worked relatively well, but there were still many flight sequences where the tracking was interrupted during several frames. The poor quality of the tracking was often due to the numerous reflections happening on the fly's thorax, which makes impossible the detection of the markers in front of the bright areas. These disturbances are very hard to avoid, since very powerful light sources are used, and almost any relatively flat surface will appear bright if it is aligned as a mirror so that it reflects the light emitted by an opposite strobe, or the own camera's strobe. But to cope with this problem inherent to the way the Vicon system works (with strong light sources, and bright marker detection), the solution is to have as many video cameras as possible that are seeing the marker at the same time, so that at least two cameras can constantly have a clear detection and reconstruct the marker without gaps in the trajectory.

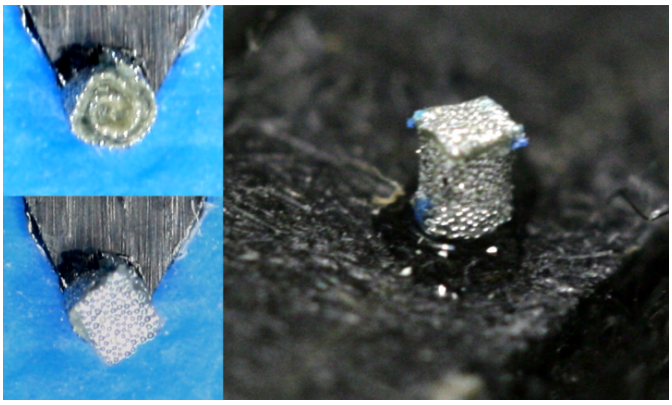
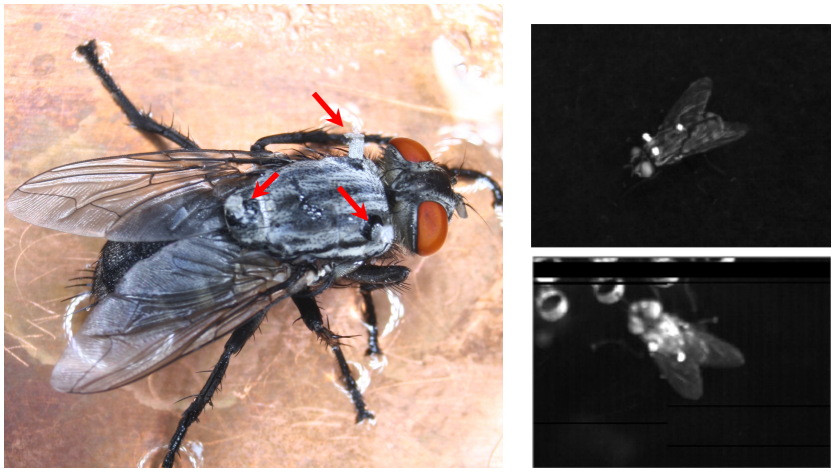


Figure 23: Reflective marker having a cylindrical shape, built with the tape showed in Figure 22. The typical weight of a marker is between 0.2 and 0.5 mg, and the diameter is usually between 0.6 and 1 mm.

This led to the manufacturing of a "fly suit", with a carbon skeleton and four markers fixed on it, as shown in Figure 25(a). The fly suit has a "Y" shape that is non-



(a) (b)

Figure 24: (a) Fly with three reflective markers directly glued on its thorax. A very small quantity of super glue is used. (b) Fly as seen by the Vicon cameras. The markers are easy to detect in some cases, but because of reflections on the thorax, some are sometimes impossible to extract.

symmetrical in the direction along the fly. A suit with four markers is used because Vicon has difficulties recognizing subjects with only three markers. In addition, the marker at the tip of the "tail" allows to identify without any doubt the direction of the fly during the reconstruction process (as opposed to three markers forming a triangle). The suit is designed and placed on the fly so that it does not disturb the wing movements. The drawing of the suit is in annexe page 70.

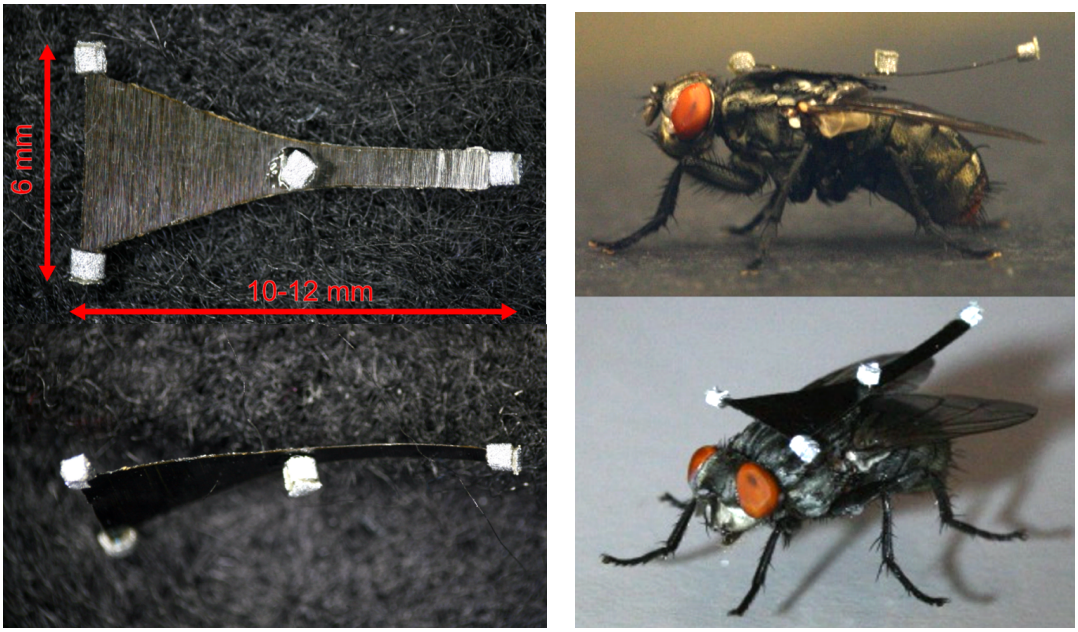
An advantage of using a suit is the fact that the markers' positions relatively to each other are precisely determined by the carbon structure. If the suit is placed the same way on each studied fly, the position of the markers is very similar from fly to fly, which is a great advantage for the further analysis. Also, the flysuit positions the markers in such a way that they are visible from more orientations, and therefore by potentially more Vicon cameras. For example, the two front markers can almost be seen from under the fly, which is not the case of the markers of Figure 24(a) glued on the thorax. As explained before, it is an advantage if a marker is seen by as many video cameras as possible, in case reflections disturb one of the cameras. Another advantage of the fly suit is that it is much easier to glue on the fly's thorax than the three individual small markers. Finally, a same fly suit can be re-used from fly to fly, even though removing the suit from a fly can sometimes be difficult.

A few suit designs are used, some models having one or two carbon layers, or being of different sizes. $50\mu m$ thick unidirectional carbon fibers[11] are used. The carbon fibers, preimpregnated with uncured epoxy, are cut by a high precision laser and then cured to bond the sheets together and solidify the resin. The most satisfactory design is a suit composed of two layers of carbon fibers. The use of these two layers induces a slightly

[11] reference XN50A

curved shape, which makes the suit's tail be further away from the fly's abdomen, which the observations showed to be a lot less disturbing for the fly. Two different sizes (12mm and 10mm) of suits are used, to adapt to the different flies' sizes.

The heaviest model of the fly suit (two carbon layers, 12mm length) weights about 5.4mg. The carbon skeleton weights about 2.7mg, the rest being the markers and the glue. This is a reasonable load for the fly to carry (about 7.5% of the fly's weight). As a comparison, the load supported by a similar fly in the *flying eye* experiment (see chapter 2.1.1) is about 7mg (see [7]). It is reported in [7] that such a load did not appear to hinder normal free flight, as it is the case with the fly suit, which did not appear to disturb the fly's flight or maneuvers.



(a) (b)

Figure 25: (a) Fly suit composed of a two carbon fiber layers skeleton, with four markers glued on it. Models of 10mm and 12mm are used. (b) Fly with suit glued on its thorax. The fly suit is positioned so that it does not obstruct the possible trajectories of the wings. The suit weights 5.4mg and did not appear to affect the flights of the flies, having a typical weight of 70mg.

3.3 Experiments setup

To study fly collisions, the fly is put in a transparent box so that it does not see the walls and eventually collides into them. The box has to be large enough to allow the fly to initiate a normal flight before colliding into a wall. But the box cannot be too large because the Vicon cameras have to be close enough to the fly to see well the reflective markers. A good trade-off is a box with sides of a few tens of centimeters. Transparent boxes are built from 1.5mm thick acrylic sheets.

When placing the Vicon cameras around the box, some constraints due to reflections

arise. Indeed, the strobes present on each camera are very bright, and no marker detection is possible in areas of the image where a camera sees directly the strobe of another camera or a reflection of its strobe or of another camera' strobe. Therefore, the box for the fly has to be designed carefully, since the transparent walls reflect a certain amount of the strobes' light, and they act like mirrors. For example situations shown in Figure 26 should be avoided, which induces consequent constraints. Also, the background behind the box should not be bright, otherwise it can also produce disturbances in the marker detection. Black felt can be used to easily cover the bright or reflective spots in the background. For the same reasons, the bottom of the box is covered with a black non-reflective sheet of cardboard.

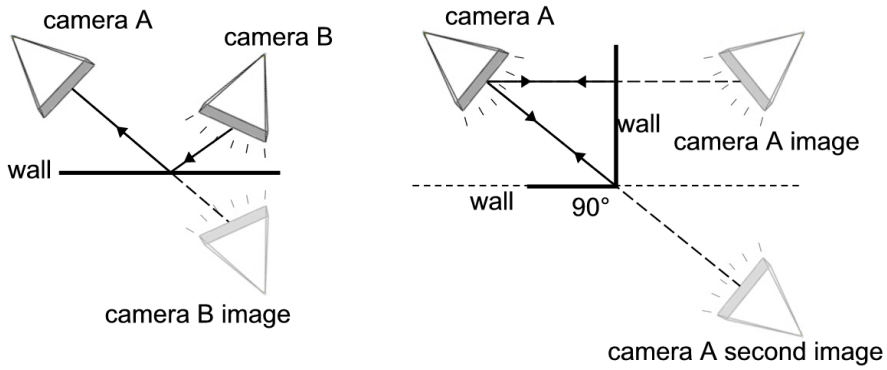


Figure 26: Examples of disturbing reflexions occurring when a camera sees another camera's light, when a camera sees its own direct reflexion or its second reflexion in a situation with a 90° angle.

Mainly two different setups were used to acquire data for this project. The box of the first setup is designed with 95° angles between walls, so that the problem of the reflection of a camera's own light facing a 90° angle is avoided. The camera positions for the first setup are shown in Figure 27. Three cameras are at the box's height, and three are looking from above. Having many cameras at the box's height is hard to manage because of the many possible reflexions against the vertical walls. Also, cameras oriented horizontally can only look at the scene from one main side, otherwise the case happens where two cameras are facing each other. This is why all cameras of this setup are looking from the same main side.

The second setup is realized by taking into account the observations made with the first setup. First, the cameras contributing the most to the tracking are the ones looking from above the box. These are also the cameras that are a little bit further and that offer a larger capture volume. Also, gaps in the capture often happen because of the lack of cameras on one side of the capture volume. The second box is designed so that it makes possible the positioning of four cameras looking from above and from the four corners of the box (see Figure 30), so that a larger and more "efficient" capture volume is provided, with cameras looking from all sides. For such a camera configuration, the ceiling of the new box is made slightly inclined on both sides, so that cameras 4 and 5 do not see the reflections of cameras 1 and 2 for example. The new box is larger, and

collisions are observed on both sides of a wall obstructing half of the way in the middle of the box. The cameras present at the four corners are the TX40s, since they provide the largest field of view at high speed. In fact, the two TX160s have much less impact because of their limited field of view at high frequencies, and contribute very little to the tracking. Figure 29 shows what each camera sees and the windowing happening at 500Hz and 1,000Hz. Longer flight sequences can be observed with this setup because the cameras are placed slightly further from the capture volume on average than for the first setup, and less gaps occur in the capture because of the cameras present all around the box.

It is important that the fly studied in the setup has an environment as favorable for flight as possible. For example, a fly will not want to fly if the environment is too dark. As it can be seen in the picture of Figure 30, a light is placed above the box, to provide a light gradient coming from above (since the fly is suspected to use its occelli to orient itself relatively to the light), and to make sure the surroundings are well lit and provide a contrasted visual feedback to the fly. Contrasted panels were placed around the box to make sure the visual feedback of the fly was good, but this did not appear to make a difference in the fly's behavior during flight, because the surroundings of the setup (the lab room) probably provide all the contrasts needed by the fly to fly normally.

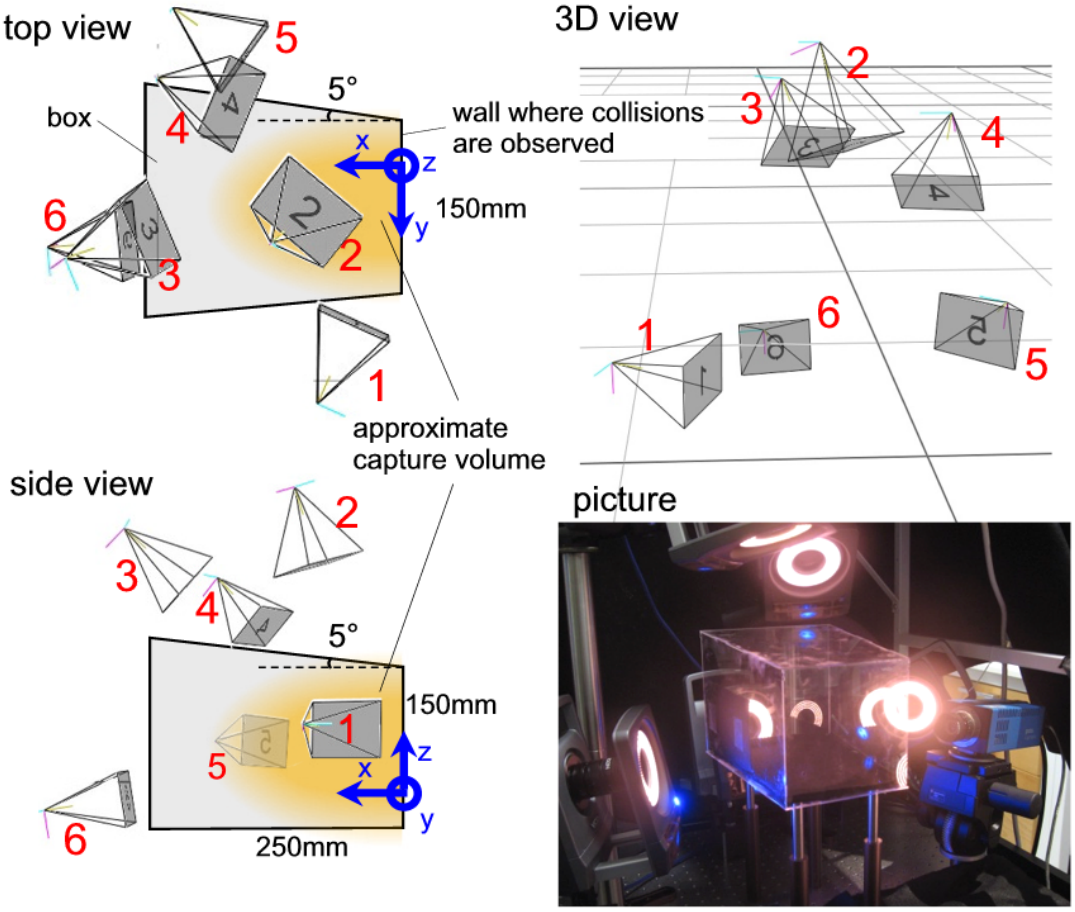


Figure 27: Vicon cameras' positions around the box used for the first setup. Cameras 2 and 5 are TX160s, the others TX40s.

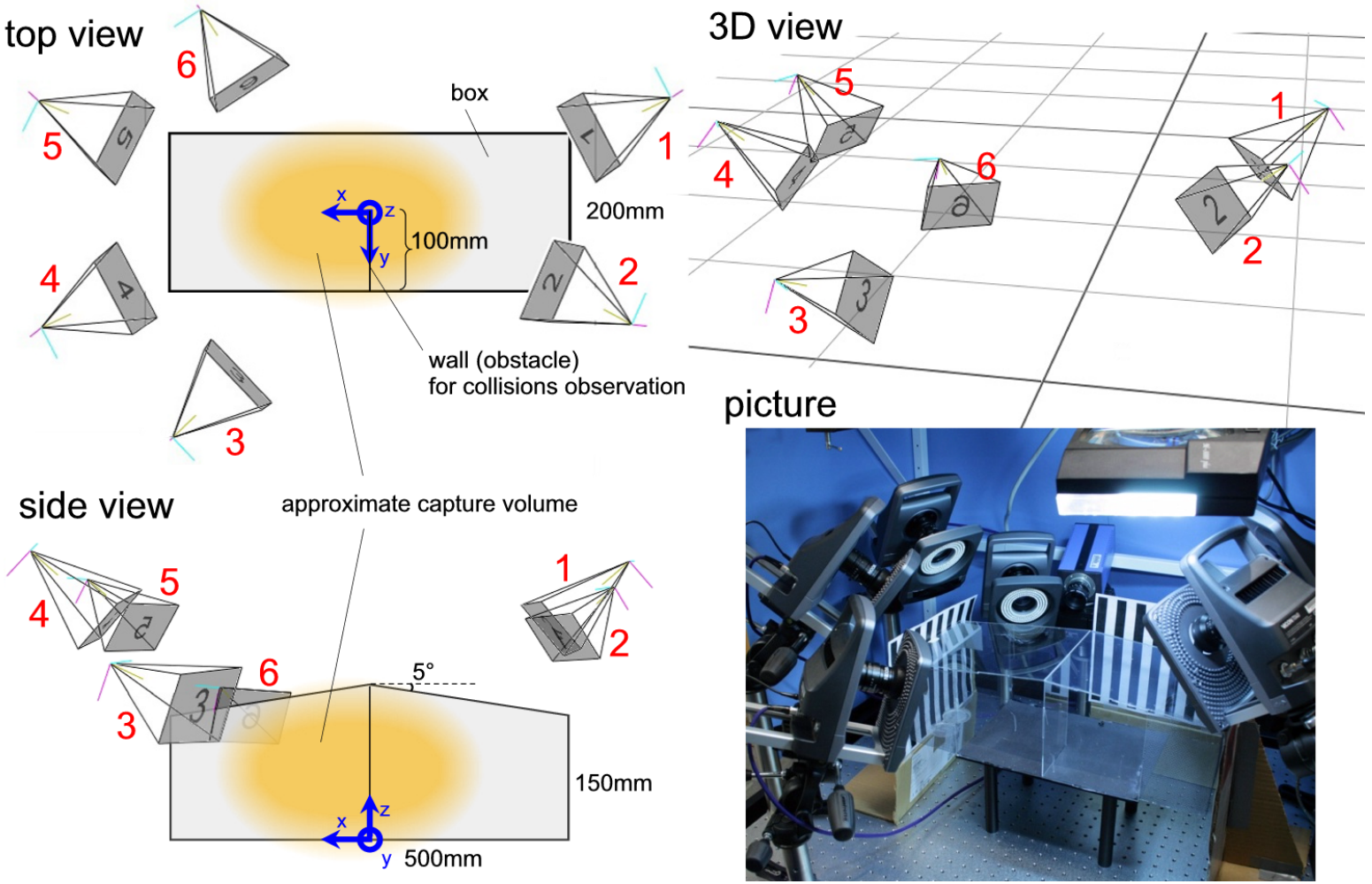


Figure 28: Vicon cameras' positions around the box used for the second setup. The cameras placed at each corner (1, 2, 4 and 5) are TX40s since these are the key positions, and the TX40s provide the larges field of view at high frequencies. In fact, the TX160s are almost useless for this application at high frequencies, because of the windowing of the field of view.

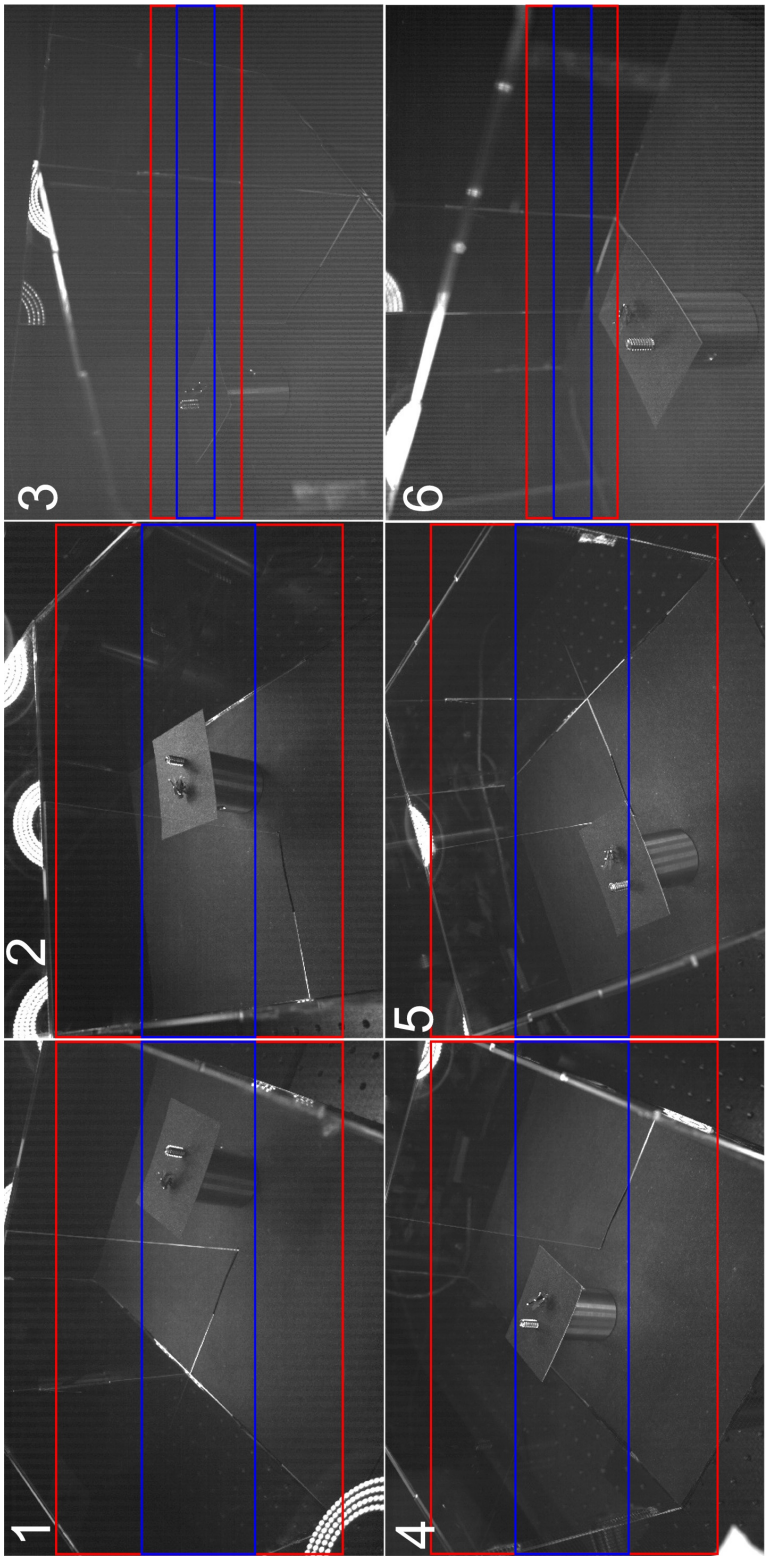


Figure 29: View from each camera present in the setup shown in Figure 30. The windowing happening at 500Hz (red) and 1,000Hz (blue) is drawn on top of the images. Cameras 1, 2, 4 and 6 are TX40s, and cameras 3 and 6 are TX160s.

3.4 Setup preparation and experiments process

Once the cameras and the box have been positioned, the Vicon system has to be configured so that it achieves a good tracking at the desired frequency. The chosen tracking frequency is 500Hz, because it offers a good trade-off between the temporal resolution and the capture volume. Some trials are tracked at 1,000Hz, but the capture volume is in that case too restrained to produce interesting results for the type of observations done in this project. An adapted setup and additional TX40 cameras may make possible very interesting experiences at 1,000Hz.

Before starting the calibration process, the focus, the aperture, the strobe intensity, the gain and the threshold have to be tuned for each camera so that the markers are clearly detected. Then, the calibration has to be performed with a wand that has to be waived in the entire capture volume. Since the setup for fly observation implies capturing markers moving in a box, the calibration has to be done in presence of the box to account for the light deviation due to refraction happening when light goes through the transparent walls. The boxes are built so that the inside is easily accessible to allow to waive the wand inside the box during the calibration. The detailed steps for the setup preparation are described in annexe A.1. After a good calibration, the markers seen by the Vicon cameras can be reconstructed in 3D. For the real-time module to work, the fly suit carried by the fly has to be recognized and labeled by Nexus. Subject creation and automatic labeling is described in annexe A.2.

The fly has to be prepared with a suit one day before the tests, since it was noted that a fly is less active in the hours following an anaesthesia. Also, no water should be provided to the fly during one day to make it slightly more active during the observation sessions. To capture automatically flights, the monitoring software presented in chapter 2.3.2 has to be used. Its configuration is presented in annexe A.3.

The setup can then be left alone during several hours, while it records each flight happening in the capture volume. Sometimes experiments were also run during the night, but it is probably better to let the flies rest in the dark during the night, so that they have a 12 hours day/12 hours night cycle. Otherwise, the flies become a lot less active, perhaps because they are tired (the light in the experiment room is the same day and night and is not the reason).

In normal conditions, the fly was flying in the capture volume and triggering the auto capture every 5 minutes in average. When the experiments were ran during the night, the period between captures was often around 15-20 minutes, which is why the experiments were then only performed during day. These low numbers of flights show why the monitoring software allowing the automatic captures is necessary. It would otherwise not have been possible to wait and launch the captures manually just when the fly is flying. Five flies in total had the final version of the fly suit glued on them and were studied. The number of flights captured automatically with the first and second setups are around 400 and 625 respectively. This corresponds to approximately 150 hours of experiments.

Once the experiments are over, the captures can all be reconstructed using the "batch processing" option of Nexus. Then, the trials have to be analyzed one by one (by looking at the Vicon tracking or the video file) to determine if the capture is a landing or a collision (the "interesting" sequences for this project) and if the flight sequence is complete enough for analysis. A few false detections appeared in the trials captured automatically, that is to say a few captures were triggered because of glitches when the fly was immobile. Otherwise, most of the discarded trials are trials where the tracking done by Vicon is too short or of bad quality. Often this happens when the fly is flying in the borders of the capture volume. But even if the tracking data cannot be used, the video is sometimes still interesting. From the first setup, 33 trials were complete enough and selected for analysis in Matlab, but a much larger amount of trials provide interesting videos. Because of lack of time, not all trials captured with the second setup were analyzed, and on the 280 first captures, 30 trials were selected for analysis in Matlab.

The bad quality of a tracking means that the tracking is incomplete because some markers are not detected during a certain amount of time either because the fly went out of the capture volume, either because it passed in front of spots that are masked to the camera because of reflections for example. Gaps in marker trajectories happen often and if they are not too consequent, they can be filled and the capture is still usable. A tool permits to fill the gaps by splines, or by using another marker's trajectory. It is also possible to replace by splines a part of a trajectory that suffers from glitches. Once the experiments done, there is therefore still a certain amount of work of selection and editing to do before the analysis is made possible.

It is important to have a subject well labeled in Vicon to be able to use the data correctly in Matlab. Indeed, only the labeled markers can be found by the program. Matlab directly reads the ".c3d" files generated by Nexus when the 3D reconstruction is realized. When reconstructing or editing a trial in Nexus, it is therefore enough to use the "Save" button to make the changes available to Matlab.

4 Results

This chapter first presents the tools realized to analyze the results of the experiments, then precision measurements of the tracking setup are shown, followed by some analysis on the collisions themselves.

4.1 Visualization tools

4.1.1 TrackingReplay

The *TrackingReplay* Matlab program allows to analyze the tracking data recorded with Nexus. It has the following motivations :

1. Visualization at the same time of the 3D tracking and the video image taken by the PCO camera.
2. Computation of additional values than just the marker trajectories, such as variables related to the dynamics of the flight.
3. Possibility to plot and display exactly the desired information on a graph, a 3D view, or a video.
4. Implementation of tools specific to collision observation.

TrackingReplay is designed so that it can be used to analyze any kind of tracking data recorded with Vicon, and therefore its application is not only limited to fly collision observation. *TrackingReplay* is composed of a GUI controlling the display of a certain number of views and graphs (Figure 30 shows the different elements). Here are the features of *TrackingReplay* (see annexe A.4 for a description of each file of the program):

- At startup, a user interface allows the user to choose the 3D data (.c3d file) and the video (.avi file) corresponding to the trial to analyze.
- In addition, a configuration file can be associated with each trial. The configuration file allows to configure settings particular to a subject, customize the plots, the views, etc. which makes the program very adaptive to different types of observation (such as a fly suit with 3 markers, and then 4 markers).
- At startup, some processing is done on the marker trajectories. First a filter is applied to smoothen a little bit the marker coordinates. Then, values on the dynamics of the fly are computed, such as the fly's linear and angular accelerations in the body axis. See annexe A.5 for the description of the calculations.
- A scroll bar in the GUI allows to jump to any frame in the trial, two others allow to set the beginning and end of the interesting sequence, and the Play button allows to animate all graphs allowing to watch the sequence at a certain replay speed.

- Scroll bars and buttons allow to mark where the collision happens in the sequence, allowing to use the flight data later in parallel with other trials, all synchronized relatively to the collision instant.
- A 3D view of the subject, allows to visualize the markers, the reconstructed body of the fly (schematized by an ellipsoid), its trajectory, as well as its speed and acceleration.
- The video camera image view can be shifted in time thanks to an offset selector in the GUI. This allows to synchronize perfectly the tracking data and the video since because of various delays during the recording process, the first frame of the video does not always correspond to the exact first frame of the tracking data.
- A view containing four graphs allows to visualize the evolution of many variables during the flight. The views can be changed by using a graph selector in the GUI.
- Once one trial is closed, a .mat file containing the reference to the video file and the configuration file as well as some values related to the state of the GUI, is created. It is used when a trial is re-opened, to load automatically the good video, the good configuration file, and to configure the GUI in the same state as the previous time (start/stop frames, collision frames, play speed, video offset, etc.)
- It is made possible to easily create AVI movies from the different plots. For this, an additional figure is used, on which different views can be arranged. When the option "Record" is selected and the "Play" button pressed, each frame of this figure is recorded, and then the "Save AVI" generates the AVI file.

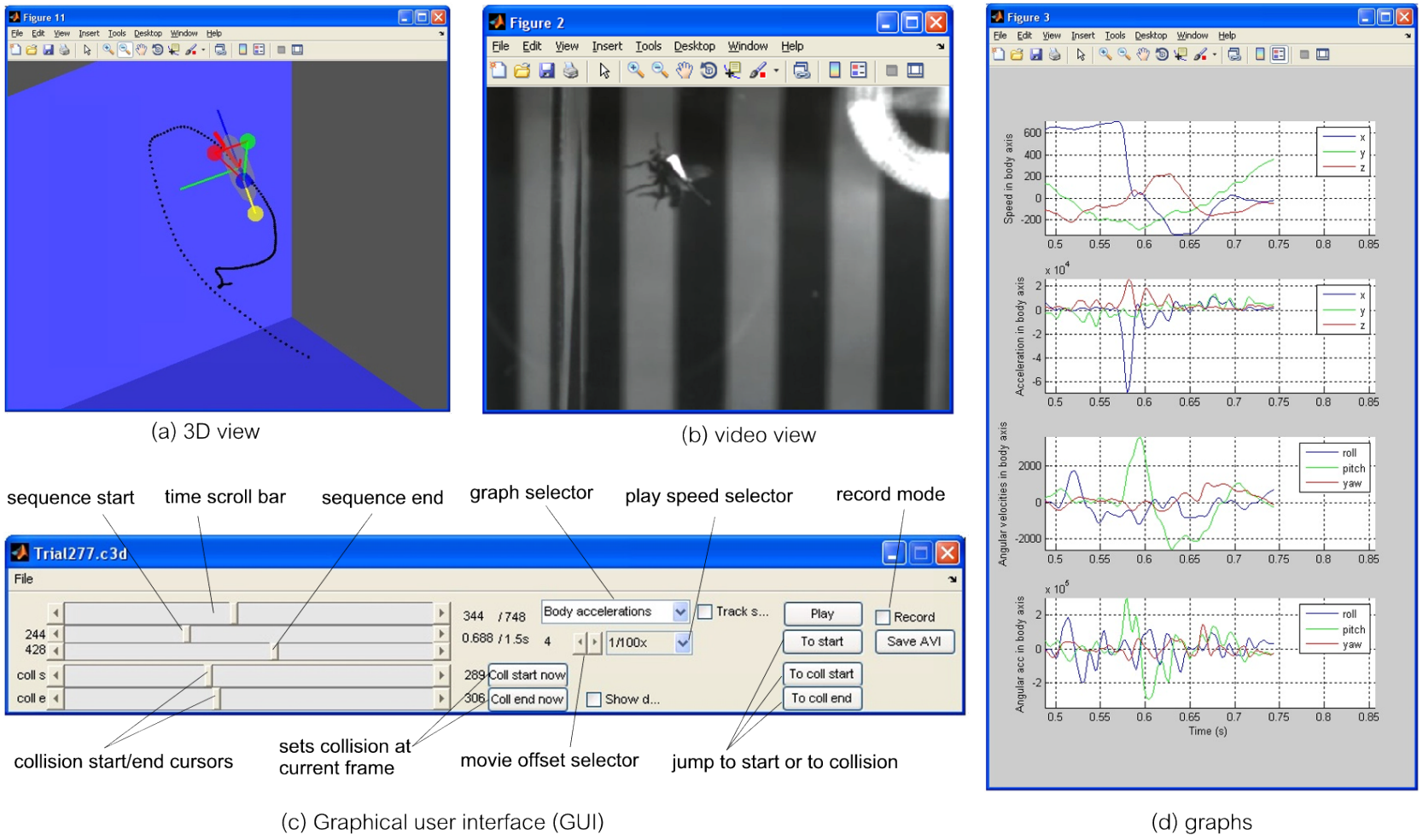


Figure 30: Main windows of the *TrackingReplay* Matlab application used to visualize the flights.

4.1.2 Data_analysis

The *Data_analysis* Matlab program provides a panel of functions allowing to extract and group easily all flight sequences together and compute statistics on all different variables. A GUI also allows to compare all flights and to display different flight variables next to each other for comparison (see Figure 31). In particular, since the instant of the collision against the wall has been marked in all flights with *TrackingReplay* (the first frame where the fly touches the wall, and the first frame where the fly does not touch the wall anymore have been identified), statistics and averages can be computed for the period before and after the collision.

4.2 Tracking quality

In this section, the quality of the tracking done with Vicon is discussed. It is indeed important to have an idea of the precision of the 3D position given by Vicon. To test that, the fly suit used to track the fly's position was fixed to a precise x-y stage. Also, the same camera position as for the flight observation experiments was used, without doing a new calibration. The measure of the precision of the Vicon system completely depends on the camera positions and the calibration quality. This is inherent of the fact that the Vicon system can be used for observations from a large range of distances. For example, to track human motion, the cameras are placed much further away to have a much larger capture volume, but the absolute precision is lower than in the case of this project, where the cameras are extremely close to each other, but where the capture volume is small.

The precision and the accuracy are measured in this chapter. The precision of the 3D position measurement was obtained by tracking immobile markers, and taking the variance of the position. In general, the variance of an immobile marker around its middle position is of $5\mu m$ (averaged on 27 marker positions), which is very precise. But this corresponds to the precision in cases where the same cameras track the marker all along. The major source of imprecision comes in cases where the cameras tracking a marker change. This is mainly due to two facts :

1. No calibration is perfect, especially in the case where the capture volume is very small. Indeed, for the best possible results, very small, uniformly bright, and spherical markers should be used during calibration (and for the tracking) but such markers are very hard to produce. If a calibration is not perfect, when several cameras contribute to the position measurement, the construction rays coming from each camera never intersect at one precise point, and the point the closest to all rays is chosen as the marker's position. This implies that when a camera does not contribute anymore to the position construction, or if a new camera contributes, the marker position will be influenced.

2. The marker center may not be well detected in the images by some cameras. Then, the rays coming from these cameras for the 3D reconstruction are slightly off, but

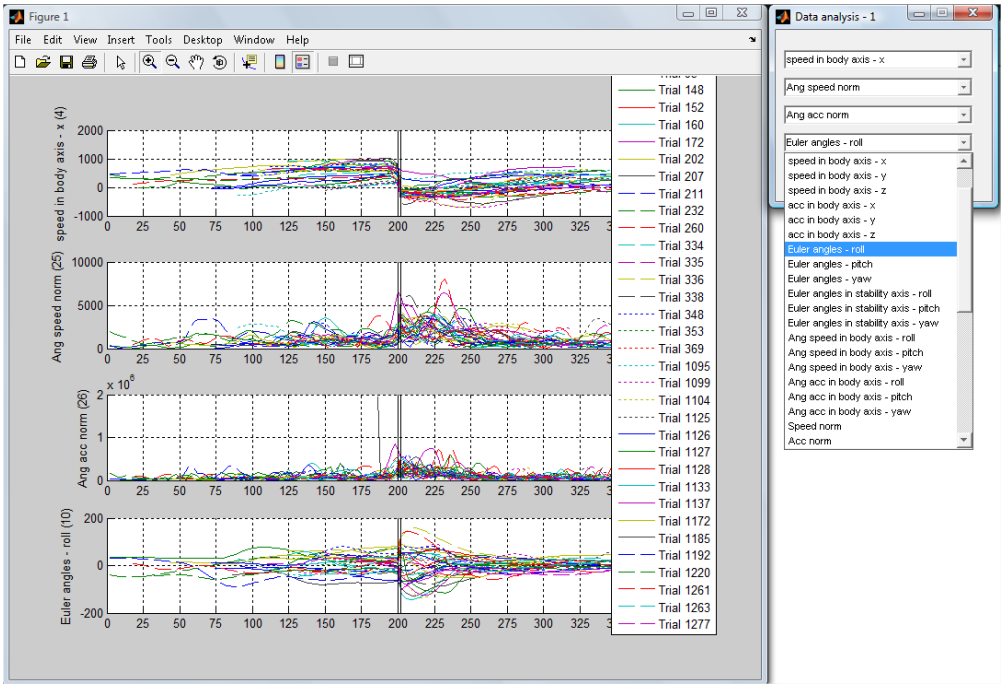


Figure 31: Screenshot of the GUI of the *Data_analysis* program, that allows to plot different flight variables for all flights, time synchronized relatively to the collision instant. The program also provides a set of functions allowing to easily group the data of all flights for different analysis or to generate statistics.

this can be consequent, for example if only two cameras are constructing a marker, and if the angle between these cameras is low (typically, if they are next to each other). A bad detection of the marker's center on an image can come from markers that are not spherical, or that are partially masked depending on the orientation from which they are seen. Also, pixels of a marker that are not as bright as the rest may be suppressed if they are under the threshold.

Using a low value for the "Ray factor" setting in Nexus allows to only keep the markers that can be constructed with rays crossing very precisely at one point. But using a low value for this option implies often that more gaps in the trajectory will appear, or that markers will simply not be tracked at all because no pair of rays can reconstruct them accurately enough. Therefore, a trade off has to be done, but for this project the "Ray factor" setting was put relatively high, to have as many reconstructed markers as possible. In addition, when glitches are remarked, they often can be manually removed in Nexus by using the spline option. Glitches can represent a deviation of up to 0.5mm from the original trajectory. Figure 32 illustrates the precision of a position measurement when no glitch occurs, and when glitches occur.

To give an idea of the accuracy of the system, the fly suit positioned on the x-y stage was moved by steps of 1.27mm (20th of inches). The distances between 28 such steps were in average $1.2762mm \pm 4.2\mu m$ ($\pm$ variance). The distance between these steps is therefore very repeatable, but it is 0.5% larger than it should. This small lack of accuracy in the

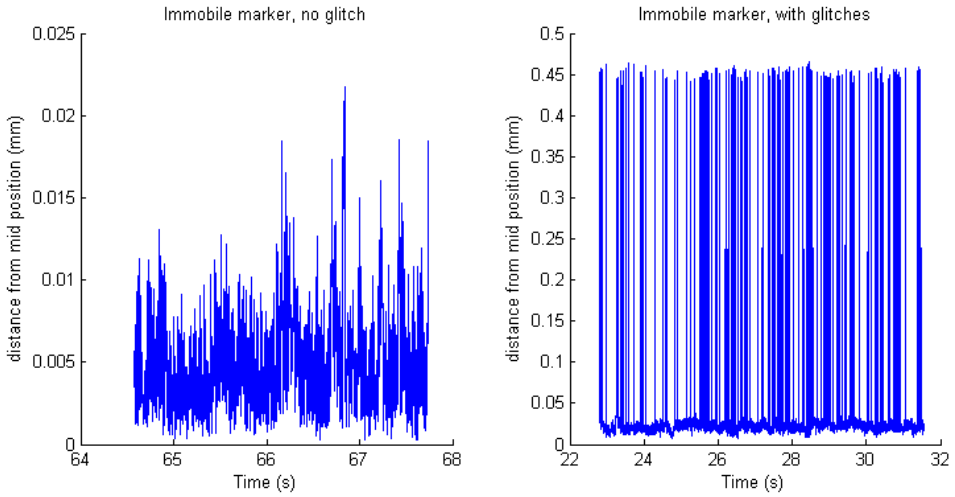


Figure 32: Distance of an immobile marker from its average position, during a few seconds with a 500Hz tracking. The first plot shows only noise, and the precision of the position measurement in this case is of $5\mu m$. The second plot shows the result when the marker is placed purposely so that one of the three cameras contributing to the reconstruction has an imperfect view of the marker, and is at the limit of being included in the reconstruction. The fact that one camera that has a bad measure of the marker's centroid on its image, and is used for the reconstruction only for some frames, and not for others, induces the intermittent marker deviation from one position to the other. These glitches of up to 0.5mm alter greatly the precision, and can happen at a few occurrences on a trajectory of a moving marker, when the cameras tracking the marker change (but not as much as the special situation shown in this Figure).

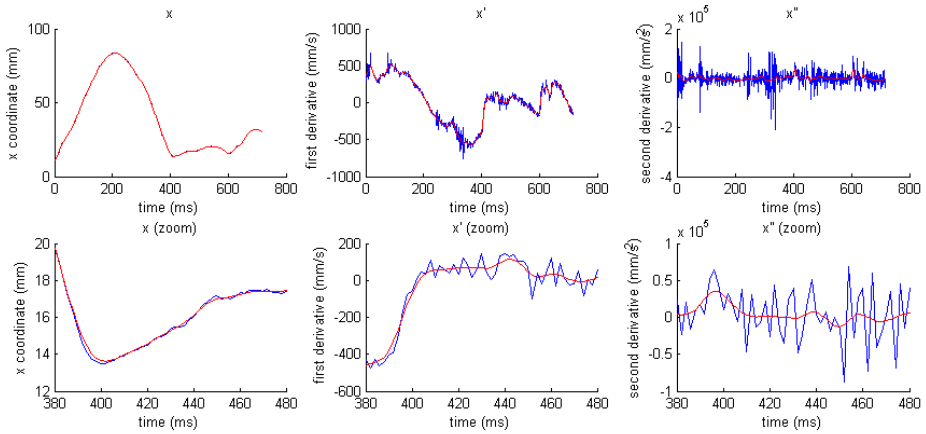


Figure 33: Example of the influence of the Gaussian filter to smoothen the trajectory, and the impact on the derivatives. The smooth speed and acceleration curves are the derivatives of the filtered trajectory, without any additional filter used. Actually, filtering the original trajectories is sufficient to smoothen all the variables inferred from them.

distance measurment comes from the calibration process, during which a wand is used to fix the scale. Indeed, if the distances between the wand's markers are not perfectly correct, the scale might be slightly wrong, but in this case, 0.5% is negligible.

To conclude, the markers quality and the calibration process are key elements in providing a good precision. The main imprecisions are due to the glitches occurring in certain situations. Most of these glitches can be fixed relatively well by editing the trajectory afterwards. Except the glitches, the precision is fairly outstanding as shown before. A Gaussian filter with a variance $\sigma = 4ms$ is used to smoothen the noise, which is necessary when the trajectories are derivated, as it is shown in Figure 33.

4.3 Collision analysis

The boxes used in the setups have transparent walls so that the flies cannot see them, and will more likely collide into them. The environment behind the walls provide the fly with enough contrast so that they can fly normally. When the fly hits the transparent wall, it is therefore a surprise, and can be thought of as a huge perturbation in the middle of the flight. The fascinating reflexes of the fly that allows it to recover from such a perturbation are the main subjects of the present project. For example, the flights where the fly cannot recover from a collision are not studied, also because it would have required a slightly different setup and capture volume.

In addition to the collisions, an intriguing fact is also discussed: at some occurrences the fly acts like it is seeing the transparent wall in advance, and it is able to land on it. Because of the fly's low-resolution eyes, and the fact that almost no feature can be recognized on the transparent wall to compute optic flow, this behavior is surprising. The fly may still be able to see some dirt sometimes present on the walls, or see the

edges of the box and infer that there is an obstacle close to these edges. Some landings though, let think that the fly does not see the wall in advance but is still able to grab the wall and land thanks to quick leg reflexes. A total of 35 tracked collisions were selected to be analyzed (shown in Figure 34) as well as 16 landings.

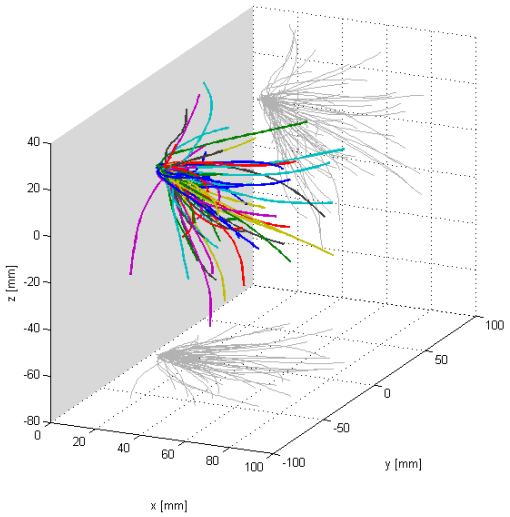


Figure 34: The 35 trajectories where a collision happens, plotted so that the impact against the wall is at the same position for each one. The curves in gray on the bottom and side walls are the projections of each trajectory.

4.3.1 Role of the legs

In most of the collisions analyzed, the fly hits the wall with its head first. The usual pitch angle of the fly during forward flight is around 20°. When it hits a wall with its head, the reaction force provokes therefore a positive moment around the pitch axis, which will usually make the fly have a positive pitch velocity after bouncing off the wall, as it is shown in Figure 35. In some cases where the approach is done from the side, the fly hits the wall with a wing first. This situation implies a very strong disturbance because of the reaction of the force applied by the wing touching the wall, leading often to unrecoverable situations. Most of these unrecoverable situations are when the fly has its back against the wall and tries to flap its wings (see Figure 36). Indeed, flapping the wings in this situation generates a lift force in direction of the wall, which makes it very hard for the fly to go away of the wall, and a recovery is in this case often impossible.

As explained before, two different situations seem to happen when a fly lands on a wall. Either it is able to see the wall before and can plan to land on it, either it does not see it and it looks like a collision at first, except that the fly manages right after the impact to somehow quickly use its legs and grab the wall to land on it. These landings were chosen to be called "reflex landings". The assumption that the fly does not see the wall before such landing comes from the observations on the videos of the fact that the fly usually extends its front legs in a characteristic fashion when it sees the wall in advance,

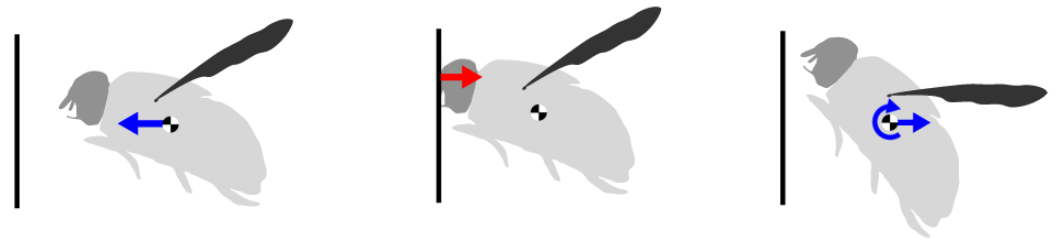
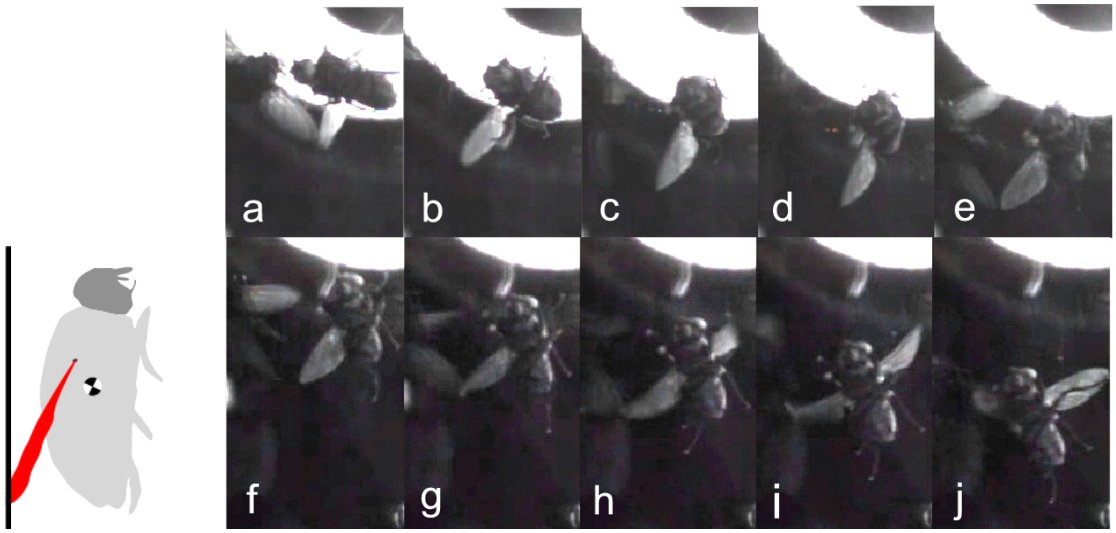


Figure 35: Drawing of the reaction to a collision where the fly hits the wall with its head and does not use its legs. The head bounces off the wall and provokes a relatively high angular speed around the pitch axis.



(a) (b)

Figure 36: (a) Drawing of the critical position where the fly has its back against the wall, which is approximately the only situation from which the fly cannot recover. Indeed, flapping the wings in this position create a lift force partially in the direction of the wall. It is often an approach towards the wall with a high angle of incidence that leads to such a situation. (b) Video frames sampled every 10ms of a flight where the right wing of the fly touches the wall first(frame **a**) which makes the fly's body rotate (frames **b-e**), which leads to the situation where the back of the fly is against the wall, and eventually makes the fly fall (frames **f-j**)

whereas in the "reflex-landing" case the fly usually hits the wall with its head first, like it happens for most of the collisions.

The flights with planned landings and reflex landings were sorted thanks to video observation, and then some values obtained with the tracking are compared to verify the supposition. In Figure 37, it can be seen that before a planned landing against a wall, the fly increases its pitch of 30° in average, starting about 50ms before touching the wall. On the other hand, it can bee seen that before a reflex landing, the fly does not change its attitude (the pitch stays around 20°, like before the collisions), which tends to show that they do not expect the wall in these cases. Figure 39 shows an example of

planned landing agains a wall.

The "reflex landings" are possible thanks to the adhesive pads present at the tip of each leg. Depending on the situation, one single pad has enough grip to allow the fly to grab the wall and then make possible for the other legs to grab the wall as well. In most of the "reflex landing" cases, the fly is able to grab the wall with its two front legs just after its head bounces off the wall. If the legs adhere well enough, they can counter the backward speed resulting from the rebound that tends to make the fly go away of the wall. Finally the body rotation allows the other legs to grab the wall as well (see Figure 38).

Dealing with a collision by the ability to land once a contact is detected is an excellent way to avoid bouncing in a random attitude that may lead to a fall. In fact, according to all collisions observed on the videos, the fly always extends its legs and agitates them as if it was constantly trying to grab something right after a collision. This reflex is happening right after the impact, and if it does not always allow the fly to land, some legs still can grab the wall and it looks like it plays a role in "damping" the collision by decreasing the speed of the rebound.

Even if usually the legs are folded along the body during flight, the observation of the videos shows that the fly is sometimes opening its legs during flight. It is supposed that this behavior is triggered when some visual information lets think that obstacles are present, or when the situation is unclear. For example it happens sometimes after saccades, quick turns during which the fly has no visual feedback and after which the fly may feel uncertain of what is in front of it. An example of these behaviors is showed in Figure 40. Having the legs already extended in unclear situations allows to react much faster in case of collision with a wall, and increases the chances to land on the wall instead of bouncing off. All these elements tend to show that the legs play a great role during a collision, and they are used by the fly as soon as an impact is detected or expected.

4.3.2 Orientation stabilization

After a collision, the speed of the fly is reduced from about 0.7m/s to 0.5m/s. The average pitch angle increases from about 20° to 40-50°, which is clearly related to the speed decrease (see Figure 41(a)). In general, the fly's speed is inversely proportional to the pitch angle. For example, the fly hovers with a pitch around 50°, so that the stroke plane of the flapping is horizontal.

It is interesting to see on Figure 41(b) that after a collision, the linear acceleration of the fly's body stays low, whereas the angular acceleration stays at a relatively high value during 50-70ms. The accelerations, when the fly is in free flight, correspond to the fly's action to control itself. For example, the angular acceleration is proportional to the torque developed by the fly's wings to stabilize the flight. It can be seen that after a collision, the fly develops much higher torques than in normal flight, whereas the linear

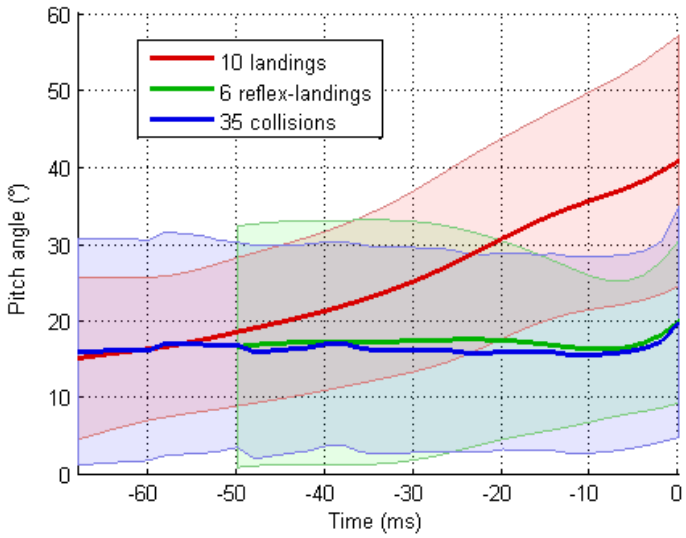


Figure 37: Average and variance (shaded area) of the pitch values before 10 landings, 6 "reflex-landings" and 35 collisions (the collision happens at t=0). It can be clearly seen that the fly increases its pitch about 50ms before a landing, whereas the pitch stays at its normal value before a collision or a "reflex landing". This tends to show that the fly does not see the wall before the "reflex landings" and the collisions, unlike in the landing cases.

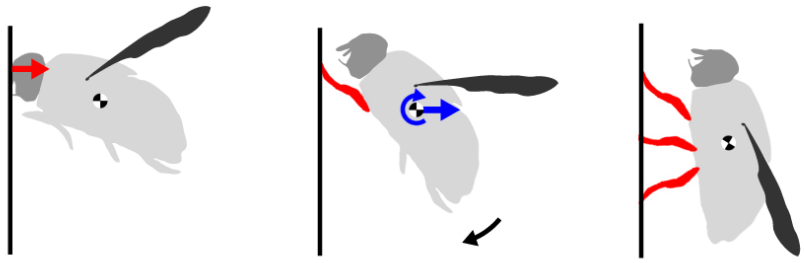


Figure 38: The "reflex landings" happen when the fly collides into a wall, but is still able to grab it and land on it. It is called "reflex", because the legs only have a few ms to grab and adhere to the wall before the fly bounces off. If the fly manages to grab the wall with its front-legs, the backward speed of the fly is then transformed into rotational speed that makes the fly's body turn towards the wall. The "reflex landings" happen most of the time after collisions before which the fly already had its legs partially open, which allows faster reactions to grab the wall.

acceleration value stays low. This means that the fly only tries to control its orientation after a collision, and does not try to correct its trajectory. It does make sense that having a correct orientation and being stabilized is the fly's priority after a collision.

Figure 42 shows for each axis the average angular speed and acceleration before and after a collision. The values before the collision are non-zero on average, because the fly is sometimes in the middle of a turning maneuver. It can be seen that after the collision, the accelerations around the roll axis are in average twice as large as the accelerations around the yaw and pitch axis. This is due to the fact that it is the axis around which

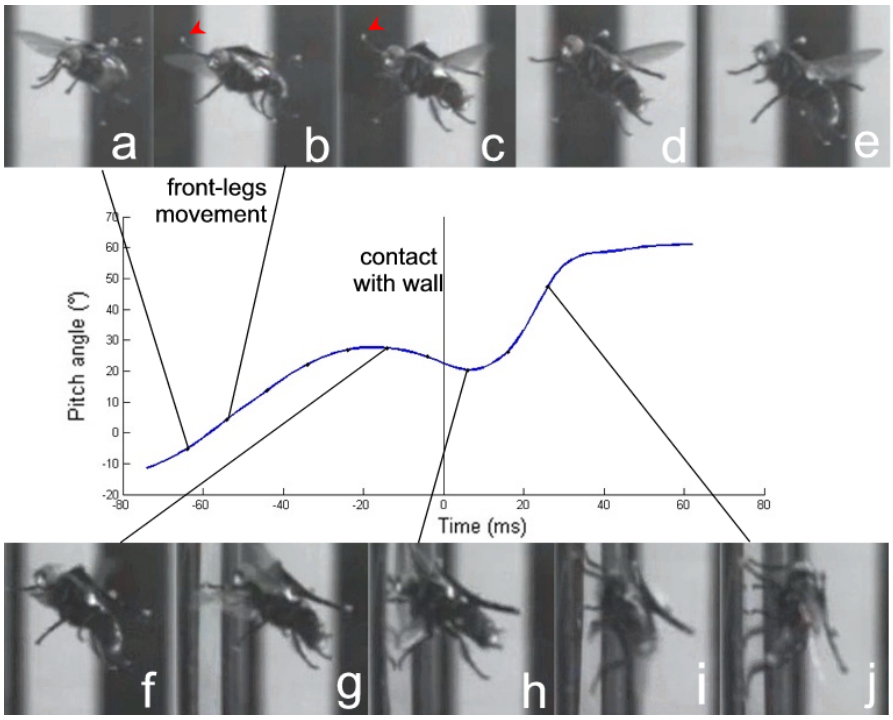


Figure 39: Example of a typical landing sequence. Video frames are showed every 10ms, and the pitch angle is plotted. On frames **b** and **c**, 50ms before the contact with the wall, an upward movement of the front-leg can be observed (indicated by an arrow), which is a behavior observed before other landings, and that tend to confirm that the fly sees the wall in advance. After having increased its pitch of about 30°, the fly touches the wall between frames **g** and **h**.

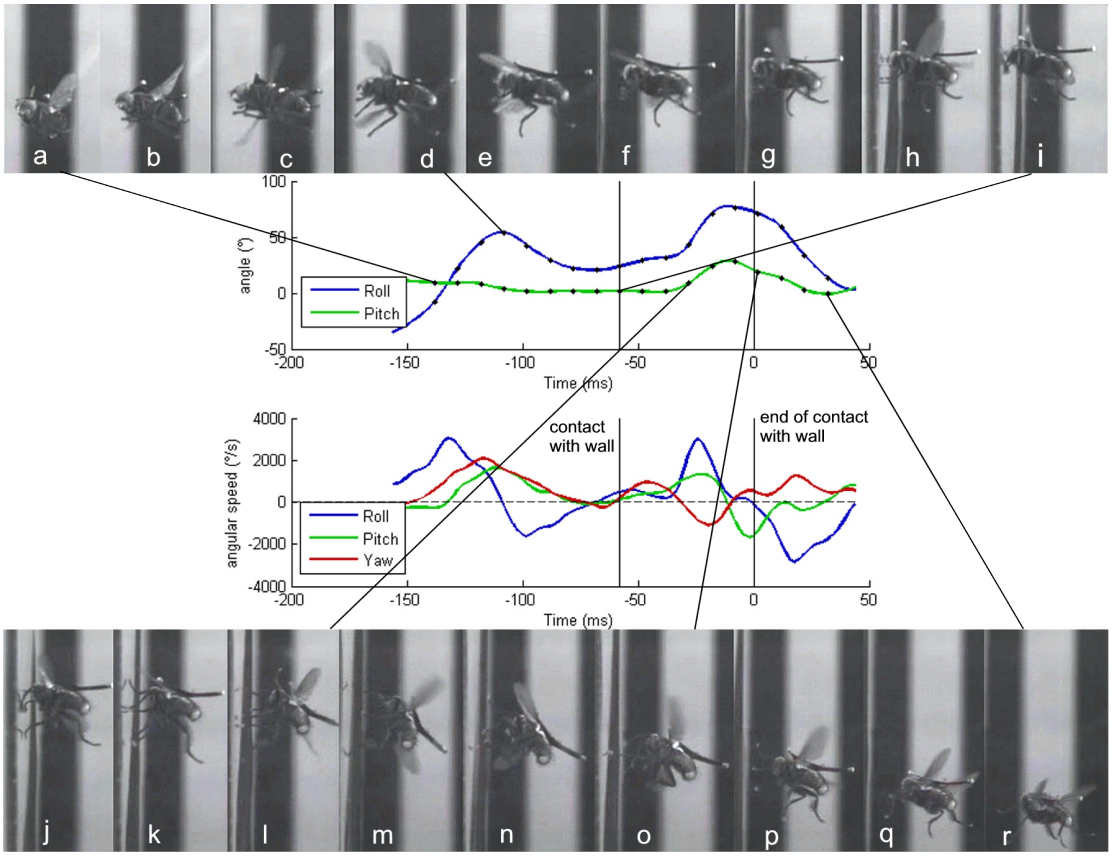
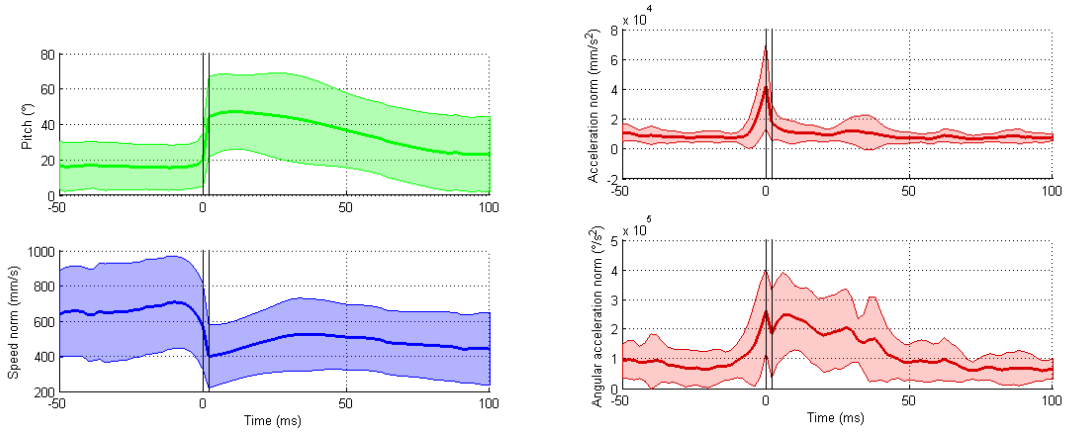


Figure 40: Sequence showing and example of the role of the legs during some collisions. Video frames are showed every 10ms, and angles and angular speeds are plotted. During frames **a-c** the fly is executing a saccade. Right after, probably because it is uncertain of what it sees, it extends its legs (frame **d**), and then flies at 0.3m/s towards the wall. At frame **i**, it hits the wall and thanks to the relatively low speed and the fact that the legs are already extended, it can grab the wall with its legs (frames **j-l**). But the fly is not able to land, and it looses contact with the wall at frame **o**. It still move its legs to try to grab the wall, and finally stabilizes its roll (frames **p-r**). The contact of the fly with the wall lasts 60ms, which is a lot compared to the average. The fact that the fly had his legs extended before the collision allowed it to grab the wall fast and even if it could not land, it could decrease its speed which allowed an easier recovery after loosing contact with the wall.

the fly has the best control, and it can therefore accelerates and decelerates faster its roll angle. The reasons for this are the low body inertia along the roll axis and the position of the wings, that allow to develop high torques around this axis. Note that the low body inertia along this axis makes it also the easiest to disturb.

The average time for the angular speeds and accelerations to come back to a normal value after a collision is of 70ms in average. This corresponds to approximately 12-13 wing beats. The angular speeds happening after a collision are in average more than twice higher than the ones occurring during 60°-90° saccades (see [8]). Maximal speeds of about 5'000°/s for the roll and 3'000°/s for the yaw and pitch were reached several times.

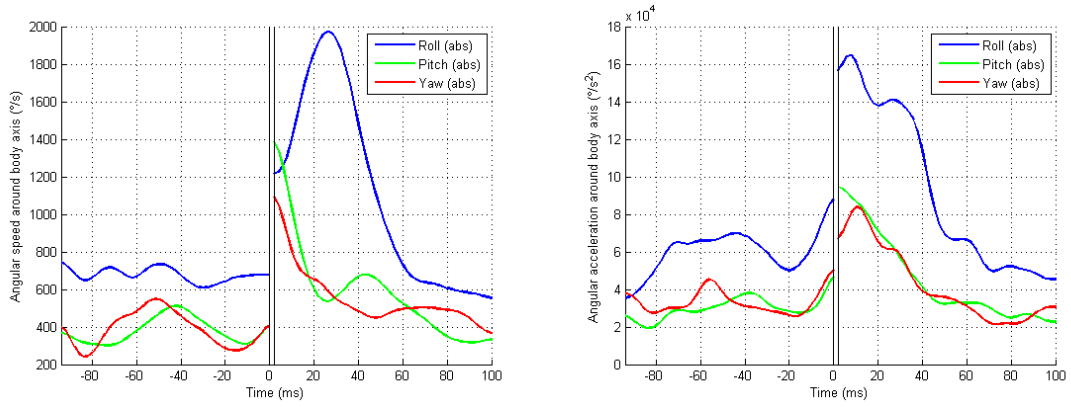


(a) (b)

Figure 41: These plots show the average and the variance of different variables before and after the collisions recorded during 35 trials. The collision happens at t=0, and the data while the fly is in contact with the wall is truncated so that at t=1 it is the the instant when the fly ends the contact with the wall for all 35 trials. (a) Plots of the pitch angle and the linear speed norm. The speed is decreased from 0.7m/s to 0.5m/s in average, whereas the pitch after a collision is usually high, 50°in average. (b) Plots of the average linear acceleration norm and the average angular acceleration norm. The linear acceleration after a collision stays more or less the same after a collision, whereas the angular acceleration norm is much higher during the 50-70ms following the collision. Note: the high increase in the values just before the collision are due to the very high accelerations happening at the collision instant, and since the original data is filtered with a gaussian filter, the data just before the peaks is influenced, but in reality no high acceleration can be observed before the impact with the wall.)

Attempts were done to extract the main characteristics of the stabilization control achieved by the fly after a collision. The idea would be to categorize a certain number of reactions to collisions leading to the flight stabilization. First, principal component analysis techniques were used, but it appeared that the K-means clustering algorithm was a better solution to reach the same kind of result. In this application, K-means is an incremental algorithm that basically finds k principal components associated with k groups of curves, minimizing the distance of each curve to its associated principal component. It has been slightly adapted so that it does not take into account the scale of the curves (i.e. curves with the same shape but scaled differently will still be considered being close to each others and will classified in the same group).

The K-means algorithm applied to the pitch speed (see Figure 43) allows to distinguish two recurring different initial conditions: with positive or negative pitch speeds. This is interesting in the sense that the explanation of Figure 35 tends to say that the pitch speed should always be positive after a collision (see the example sequence of Figure 44). After the observation of each trial individually, it turns out that all the flights identified in the first category (negative initial pitch speed) present a similar use of the legs to damp the collision, and even in certain cases push the wall with the back legs to inverse

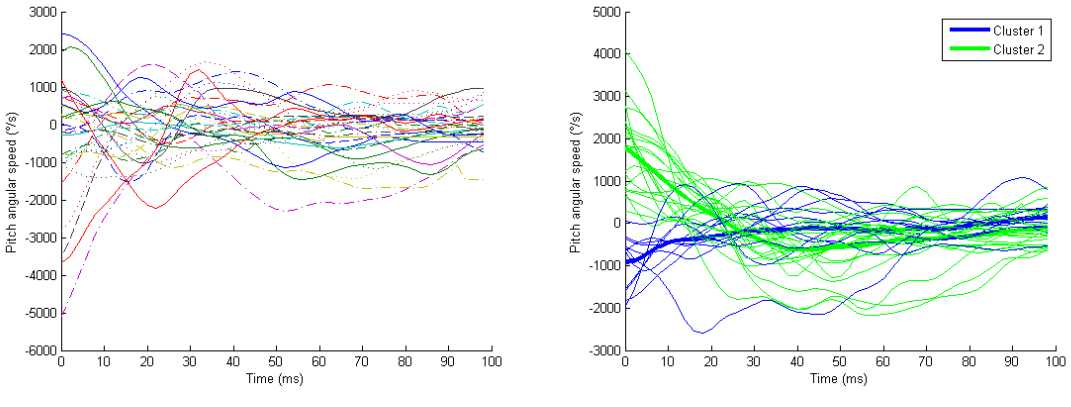


(a) (b)

Figure 42: Average absolute values of the angular speeds (a) and angular accelerations (b) around the 3 axis before and after the collisions. The variances are not shown for clarity. The data is extracted from 35 collisions and the curves are smoothed with a Gaussian filter with a 4ms variance. It can be seen that the control around the roll axis is twice more important than around the other axis.

the pitch speed (see example sequence of Figure 45). Unfortunately, the clustering has not given any real result, and other solutions may have to be looked into for a different analysis of the fly's control.

Figures 44 and 45 show flights where a strong pitch stabilization is needed. It is interesting to compare with Figure 46, where a strong roll stabilization is performed. The maximum acceleration that is applied by the fly around its roll axis is of $7 \cdot 10^5/s^2$, whereas around the pitch axis it is only of $2 \cdot 10^5/s^2$.



(a) (b)

Figure 43: (a) The pitch speed curves just after the collision in 35 trials, used as input to the clustering algorithm, whose goal would be to classify the curves in different families. (b) Result of the clustering into two groups, one group being the pitch speed curves whose initial values are negative, one group whose initial values are positive.

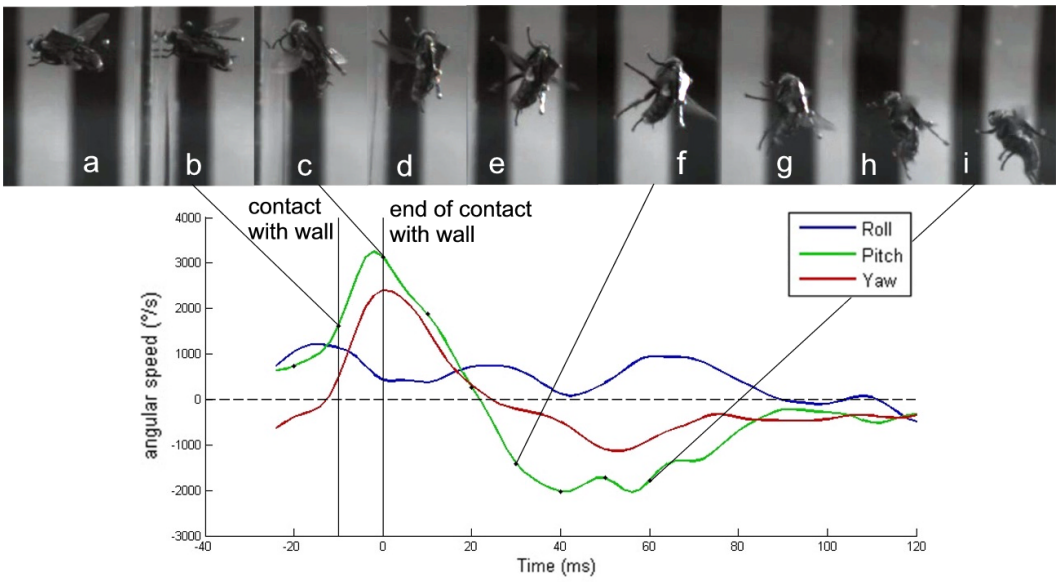


Figure 44: Example of a collision involving a high pitch speed. Video frames are showed every 10ms, and angular speeds are plotted. The fly hits the wall at frame **b** and has already bounced off the wall at frame **c**. Since it does not have the time to extend its leg and grab the wall, the time in contact with the wall is very short (10ms)). The pitch speed resulting from the collision is 3,000˚/s, which is fairly high, and it takes 20ms to the fly to slow the rotation speed down to 0˚/s, and it takes 70 additional ms to stabilize back the pitch at the correct angle. The roll and yaw axis are not too disturbed in this sequence, and their angular speed stays around 0.

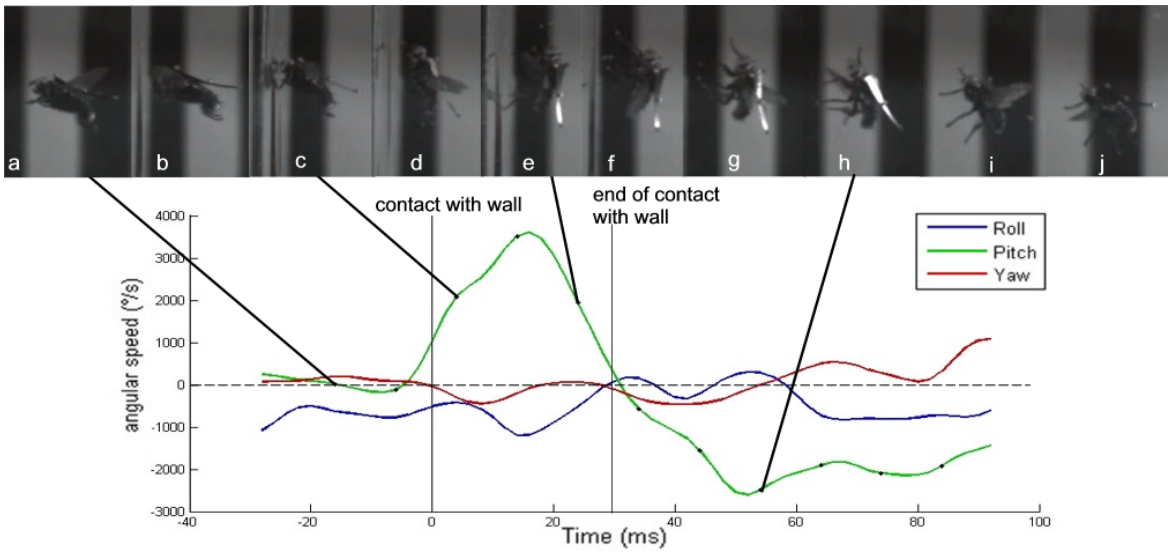


Figure 45: Example where the fly uses its legs to damp the collision and to limit the speed remaining after bouncing off the wall. Video frames are showed every 10ms, and angular speeds are plotted. The fly hits the wall between frames **b** and **c**, and stays in contact with it until a movement of its back legs pushes its body off the wall between frames **e** and **f**. This movement provokes a negative pitch speed that makes the fly turn back to a good pitch angle during frames **f-j**.

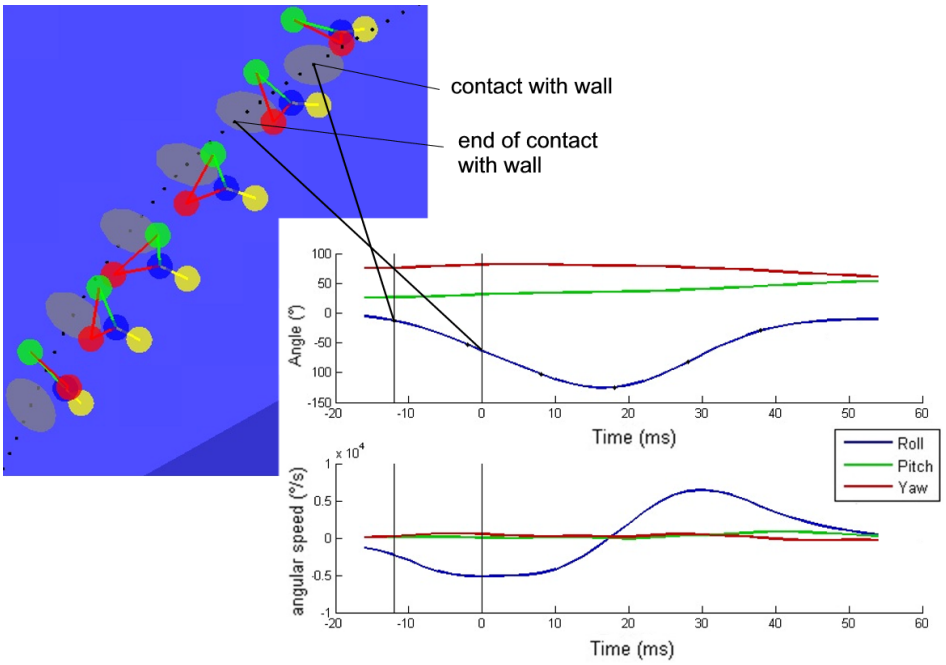


Figure 46: This example shows a special situation, where the fly hits the wall with its right wing, which provokes an extremely strong disturbance on the roll axis. The 3D position of the tracked markers and the reconstructed ellipsoid is plotted every 10ms, in parallel with the angular speeds and accelerations. It can be seen that just after the fly hits the wall, its roll speed is -5'000°/s and its roll angle around -60°. Thanks to a strong counter acceleration ($7 \cdot 10^5/s^2$ at its maximum), it can recover in 50ms, or 9 wing beats.

5 Conclusion

This project, positioned in between engineering and biology, involved to solve a lot of different challenges in different fields. The first one was of course to study living animals, which implies lerning how to work with them. In addition of being a totally unknown piece of hardware, an animal also comes with a very unclear manual, and it does not always work according to the specifications. Typically, it was very hard to actually make the fly fly.

Using a system such as the Vicon solution for freely flying insect observation was quite challenging in the sense that it is normally used at much higher scales. But a good understanding of the system, and an incremental evolution of the setups allowed to finally achieve very good captures of fly flights. The synchronization with a video camera and the monitoring by a custom software made possible fully automatic experiments, providing tracking data and video images of the flights captured at 500Hz.

Custom Matlab programs, allowing an easy visualization and manipulation of the data, helped with the challenge of analyzing the hundreds of captures realized with the setup. A total of 35 collisions with a good quality tracking were selected for analysis, allowing to extract statistics on the flight variables such as the accelerations of the fly.

The accelerations during free flight tell us about the control of the fly, since these accelerations are proportional to the torques generated by the fly to stabilize itself. Right after the collision, the average accelerations generated for attitude stabilization are in the order of $8 \cdot 10^4/s^2$ around the yaw and pitch axis and $16 \cdot 10^4/s^2$ around the roll axis. These are the values that a robot should be able to reproduce in order to be as reactive as the fly (in addition to have as good sensors as the fly as well). The stabilization after a collision takes on average 70ms, or 13 wing beats. The use of the legs also proved to be very important, even allowing the fly to land just after colliding the wall, thanks to quick reflexes and to the adhesive pads present at the tip of the fly's legs.

To conclude, the goal of the project has been achieved: a setup for fly collision observation was built, and a fair amount of collisions could be captured and analyzed. This project also opens the roads to many new potential projects, that could use the same technologies, but with setups designed to study particular elements of interest, or even setups with feedback loops providing stimuli to the fly. The possibilities are endless, and the humans are far from having understood all what the flies have to show.

Cambridge, 16th of January 2009

Adrien Briod

References

[1] A. M. Hoover, E. Steltz, R. S. Fearing, *RoACH: An autonomous 2.4g crawling hexapod robot*

[2] M. Kovac, M. Fuchs, A. Guignard, J.C. Zufferey, D. Floreano, *A miniature 7g jumping robot*

[3] J.-C. Zufferey, A. Beyeler, D. Floreano, *Optic Flow to Steer and Avoid Collisions in 3D*

[4] R.J. Wood, *Design, fabrication, and analysis of a 3DOF, 3cm flapping-wing MAV*

[5] Jean-Christophe Zufferey *Bio-Inspired Flying Robots, Experimental Synthesis of Autonomous Indoor Flyers*

[6] D. R. Nelson, D. B. Barber, T. W. McLain, R. W. Beard, *Vector Field Path Following for Small Unmanned Air Vehicles*

[7] C. Schilstra, J.H. van Hateren, *Using miniature sensor coils for simultaneous measurement of orientation and position of small, fast-moving animals*

[8] C. Schilstra, J.H. van Hateren, *Blowfly flight and optic flow: I. Thorax kinematics and flight dynamics*

[9] C. Schilstra, J.H. van Hateren, *Blowfly flight and optic flow: II. Head movements during flight*

[10] S.N. Fry, P. Muller, H.-J. Baumann, A.D. Straw, M. Bichsel, D. Robert *Context-dependent stimulus presentation to freely moving animals in 3D*

[11] S. N. Fry, N. Rohrseitz, A. D. Straw, M. H. Dickinson *TrackFly: Virtual reality for a behavioral system analysis in free-flying fruit flies*

[12] C. F. Graetzel, S. N. Fry, B. J. Nelson *A 6000 Hz Computer Vision System for Real-Time Wing Beat Analysis of Drosophila*

[13] G. Maimon, A. D. Straw, M. H. Dickinson *A Simple Vision-Based Algorithm for Decision Making in Flying Drosophila*

[14] L. F. Tammero, M. H. Dickinson *The influence of visual landscape on the free flight behavior of the fruit fly Drosophila melanogaster*

[15] L. F. Tammero, M. H. Dickinson *Collision-avoidance and landing responses are mediated by separate pathways in the fruit fly, Drosophila melanogaster*

[16] M. H. Dickinson *The Initiation and Control of Rapid Flight Maneuvers in Fruit Flies*

[17] S. N. Fry, R. Sayaman, M. H. Dickinson *The Aerodynamics of Free-Flight Maneuvers in Drosophila*

[18] R. Hengstenberg *Gaze control in the blowfly Calliphora: a multisensory, two-stage integration process*

[19] G. Nalbach, R. Hengstenberg *The halteres of the blowfly Calliphora II*

[20] Stanislav N. Gorb *The design of the fly adhesive pad: distal tenent setae are adapted to the delivery of an adhesive secretion*

[21] Vicon, *Go Further with Vicon MX T-Series*

[22] PCO, *pco.camera User's Manual for pco.1200hs*

[23] PCO, *pco.camera SDK-Description*

[24] PCO, LUT_API *PCO Lookup Table Application Interface*

A Annexes

A.1 Vicon system calibration

Here is the advised procedure for system preparation and camera calibration (originally written for the lab's wiki).

To setup the Vicon system to be able to track movements, the suggested procedure is basically to : first position everything and do the focus for each camera, then configure and save the setup for tracking, and then realize the calibration (usually made at lower frequency than the calibration)

1. Put the subject in the space where he will be tracked later and position each camera so that it has a good angle of vision. Then adjust the focus of each camera. You can use the fine focus as well as the back focus (unscrew the three screws and turn the ring at the base of the lense for the back focus). The closest from the subject one can get with the camera using only the fine focus is around 25cm. Using the back focus to the maximum allows a minimal distance of about 10cm.
   At this stage, it is quite efficient to use the preview mode of the cameras (B&W images). To use the preview mode, the Frame Rate of the cameras has to be between 30 and 100Hz otherwise it bugs. To activate the preview mode: right-click on a camera then "Enable preview mode". It is now easier to adjust the focus.

2. Make sure the subject is focused everywhere in the space it will be moving in because the focus musn't be changed after calibration (since the calibration takes into account the deformation of the image due to the particular lenses positions). Be aware that you will maybe open the aperture later, to get more light at higher frequencies, and that this will reduce your depth of field.

3. Configure now the settings for the frequency you want to track the subject at (for example 500Hz or 1,000Hz). Even if the calibration is usually done at lower frequencies and needs different settings, it is wise to start by preparing the high-speed tracking, because it is still easy to change the camera positions (as opposed to after the calibration, where it implies to recalibrate everything). For example, one can remark that a camera has to be tilted because the field of view was reduced and the subject not visible anymore because of the windowing happening at high frequencies.
   You'll mainly have to adjust these four settings for each camera : the aperture, the strobe intensity, the threshold, and the gain, so that you see clearly the markers, and not too many reflections... Here are the initial values I used, but they have to be adapted to every setup: strobe intensity to 1 (at high speed you want as much light as possible), the threshold quite low (0.5? to start with, then has to be tuned) and the gain to x2 at 500Hz and x4 at 1,000Hz. The gain simply multiplies each pixel's value to brighten the image (color resolution is lost, but it is not very

important). Then, the aperture of the camera (use ring on lense) is usually around f/4 and f/8 (keep in mind that a smaller aperture implies a larger depth of field, so use an aperture as close as possible to f/16 if your capture volume is big). Then use a trial and error process to optimize the detection of the markers by varying mainly the aperture and the threshold. It helps a bit the visualization to set the "Grayscale mode" to "All".
To keep in mind :

- If too many bright spot are detected, the camera can get to a point where there's too much data to send, and where only part of the data is sent. It therefore can happen that only a fraction of the image is shown. It can also arrive to a point where nothing is sent anymore. "Therefore, not seeing anything in the camera view can mean that the image is too dark, and sometimes too bright".
- You can mask the static reflections in the software, but no marker will be detected when going in this region. That is why it is advised to mask as much as possible the reflections physically (with black felt sheets for example).
- The windowing is done automatically in function of the frequency. There's no way to visualize it but to move an object in the field of view of the camera an see where it disappears.

4. Once the setup is well configured to track the subject at the good frequency, tighten all lenses screws (focus and aperture), and save a profile for your camera configurations (right under the "System" tab, click on the down-looking-arrow then "Save as"). Now create a new profile for the calibration. We were advised not to realize the calibration with a too high frame rate. Till now, 100Hz has been used. Try to make the markers of your subject appear in the camera views at this new frequency by reducing the strobe intensity and the gain (if the frequency is lowered, the exposure time is increased and more light is captured by the cameras, this is why the images have to be darkened by lowering the strobe intensity or the gain).

5. To realize the calibration, remove the subject of the capture volume, and mask any bright spot detected by the cameras by using the "Create mask" function in the "system preparation" panel. Then, a "T frame" with 5 markers has to be used. If you use a new T frame, or whatever new shape, for calibration, you have to create the corresponding file in `C:/ProgramFiles/Vicon/Nexus/CalibrationObjects`. Make sure the "Wand" and "L-Frame" choices are right in Nexus and correspond to the T frame you use for calibration. Make also sure the calibration type is set to "Full calibration". Realize the calibration by moving the "T frame" in the space where the subject will move.
   Once the calibration process is done, don't move or bump cameras...

6. To check the quality of the calibration, you can get an idea by looking at the "image error". All numbers should be around 0.2-0.4px, slightly higher for the

TX160 cameras. The "image error" isn't sufficient to check the quality of the calibration. It happened that cameras with small "image errors" where out of calibration.
You can check in the 3D perspective view which cameras are used to compute the position of each marker (clicking on a marker will show lines coming from the cameras to the marker). Strange behaviors may be detected here, for example if a camera is never used, or used only for markers that are just next to the valid ones, etc. Also, in "camera view", the "3D overlay" function is useful to see if the reconstructed markers are aligned to the blobs detected by each camera.

7. You can then set the origin : "Set Volume Origin" then "start" then "Set origin" (the T Frame has to be in the field of view). It is also practical to reorder the cameras : left click on "MX Cameras" (left column) then "Reorder".

8. You may want to save the settings (strobe intensities, thresholds, frame rate, etc...) you used during calibration, if you want to calibrate again later ("Save" button of the left column, when the "System" tab is selected). Now you can load the profile previously saved with the settings for the high-speed capture.

A.2 Subject tracking with Vicon

Here are a few instructions on how to insert, recognize and track a subject with Vicon Nexus (originally written for the lab's wiki).

Once the Vicon cameras are well set up, we can see in the "3D perspective" view the reconstructed markers that are present in the field of view. But we can also ask the software to recognize objects (or subjects).

To do so, a subject model has to be created. It will contain the description of each marker (name, color, ...), the type of link between groups of markers (segments), and the size information of the subject.

The first time a subject is tracked, a skeleton is created (a *.vsk* file, for Vicon skeleton). But to accelerate the process each time a subject of the same kind is tracked (= same number and same approximate position of the markers), it is advised to create a template (a *.vst* file, for Vicon skeleton template). This will allow for each new slightly different subject (like a new wing, but still with the markers approximately at the same positions), to just have to recalibrate the template, instead of creating a new subject from zero.

So in a new session, there are 3 options to have a subject : either create a new subject, either use an existing template (*.vst*) and calibrate it, either use an existing skeleton (*.vsk*). The last option implies that exactly the same subject was calibrated in a previous session.

Notes:

- When a subject is created, it will be present for all trials of the session, but not in a new session.
- The subjects of the current session (it is advised to have only one subject per session) can be seen in the "Subject" tab.

Create a new subject

To do when the subject is tracked for the first time, or if the marker configuration on the subject has changed a lot (different number of markers, or complete change of position).

1. A blank subject has to be added in the subject tab of the left panel (resources)
2. Then use the subject section of the Tools panel
3. A quick capture of the subject (which has to be placed in the field of view of the cameras) has to be done, then a reconstruction has to be processed (Pipeline Reconstruct), so that the markers are visible in the 3D view.
4. Segments forming the subject can now be created, by first giving a name and clicking on each marker (for a subject without flexible parts, one segment is enough).
5. It is advised to rename the new markers and changing the color, in the Subjects tab of the left panel (because you will have to know later which marker is which, to fill gaps, or to analyze the data).
6. You can create a template for this subject with a left click on the subject and selecting "Save Model as Template...". This will make quicker the process the next time a similar subject will be used in a different session.

Calibrating a subject from a template To do when a new subject is tracked, but when it is quite the same and has the same configuration of markers as a previous one.

1. A template has to be loaded in the subject tab of the left panel (resources)
2. Then use the subject section of the Tools panel
3. A quick capture of the subject (which has to be placed in the field of view of the cameras) has to be done, then a reconstruction and labeling has to be processed (Pipeline "Reconstruct and Label"). The subject should have been recognized (labeled). If not, go to the Label/Edit section of the left panel to label the right markers.
4. The pipeline "Static Subject calibration" has then to be run, to have the new skeleton matching exactly the subject.

Use an existing subject To do when you create a new session, and you want to use the same subject as in a previous session, because your subject didn't change.

Note : Vicon advises to recalibrate the cameras every day. They say the same for the the subject calibration, because even if you use the same subject the markers may have slightly moved from one day to the other.

1. Just load an existing skeleton (.vsk file) from a previous session foler, or from the CalibrationObjects folder

Now that your subject is active in your session, it is possible to track it in real time. To do so, click on “Local Vicon System” under the Resources panel, and set the processing level to “Label” or “Kinematic Fit”. If you click on “Show advanced options”, there are a few parameters you can try to play with to improve the detection of markers or the labeling (read the help). For example if the ”Ray intersection Factor” is too low, some markers may not be detected, and therefore your subject will be ill-labeled. Also, use a low number for the ”Minimum Recon separation” option.

A.3 Monitoring software configuration for automatic captures

To capture automatically flights, the monitoring software presented in chapter 2.3.2 has to be used. Here are the points that have to be achieved in order to have the automatic system running :

1. Vicon Nexus is in ”Live” mode and the processing level is set to ”Label”. When the fly is in the capture volume, the fly suit has to appear in the 3D perspective view and the markers should be labeled.

2. The custom software is launched, and connected to the PCO camera, to the COM port and to the Vicon real-time server thanks to the different ”Connect” buttons (the PCO is automatically connected at startup). The buttons ”Start autocapture” and ”Start capture” have to be clickable, and the value in the ”Vicon TS” field should be incrementing.

3. The folder, file name and id for the videos have been set so that they correspond to the capture saved in Nexus.

4. The ”continuous mode” option of the monitoring software is checked, and the signals provided by the Giganet are configured as described in Figure 16.

5. The button ”Start Autocapture” of the monitoring software is pressed, and ”Capturing” appears just below. If ”Waiting for capture” appears, the signals from the Giganet to the PCO may not be well configured, or sometimes Nexus simply has to be restarted.

6. The autocapture settings in Nexus are set as in Figure 18, with the ”Capture before start” value equal to the ”start before” value of the monitoring software, and the ”Arm” button pressed. To test the remote start/stop from the monitoring software, the button ”Start Capture” can be pressed. It should perform a capture both in Vicon and of a video of the length specified in the ”duration” field plus the ”capture before” duration. If the capture in Nexus is not triggered, the cables going in the ”Remote start/stop” inputs of the Giganet may not be connected correctly.

7. Finally, the option "S/S Auto" can be checked to enable the automatic captures.

Note: the options allowing to change the region of interest of the camera, or the thresholds used to trigger automatically the captures are hard-coded since they are not changed frequently. Therefore, to changed them they have to be modified in the code and the program compiled again.

A.4 Description of the *TrackingReplay* files

Here are the descriptions of the file part of the *TrackingReplay* program (see chapter 4.1.1):

- *TrackingReplay_start.m*
  This is the file that has to be run to launch the program. It will ask for a c3d file (containing the markers' positions), for a video file (in AVI format), and for a configuration file. Only the c3d file is mandatory. If the c3d file was already loaded before in the GUI, and that a video or a configuration file was already associated to it, these will be automatically loaded.

- *TrackingReplay.fig*
  This can be opened with GUIde, the Matlab GUI editor, to modify the element positions (buttons, labels, etc) of the TrackingReplay window.

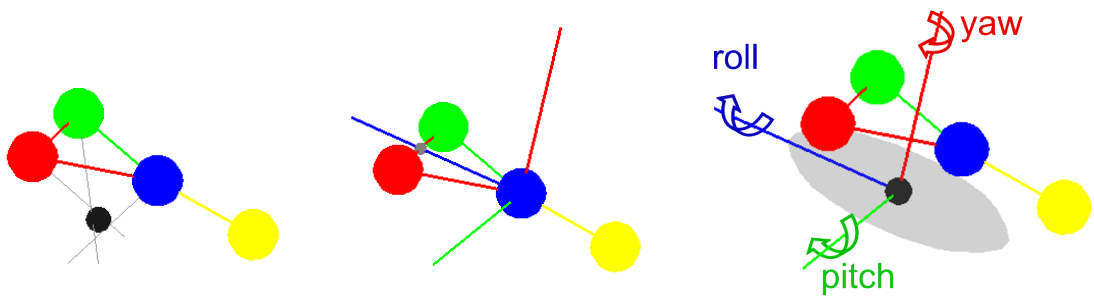
- *TrackingReplay.m*
  Partially automatically generated file, where the events coming from user-actions on the GUI are treated.

- *TrackingReplay_config_test.m* or *TrackingReplay_config_xxx.m*
  Configuration scripts, specific to a subject. It configures the display of the markers (which one to display, the colors, legends, trajectories). It permits to add graphs when the 'Other' option is selected in the drop-down menu. Additional graphs can be any data you calculate from the markers, such as angles, forces, etc... Different configuration files can be loaded at the start of the application (when TrackingReplay_start.m is run), so that the views are easily adapted to the subject.

- *TrackingReplay_init.m*
  Function that is called each time the TrackingReplay program is launched. It initializes a few elements, runs the configuration file, initializes the figures, etc...

- *TrackingReplay_update.m*
  Function called at every update of the figures. It is called continuously when the GUI is "playing". This is where the plots are drawn, some elements of the GUI updated, etc...

The first time a new set of data is opened after Matlab is launched, the file TrackingReplay_start.m has to be run. After that, simply entering TrackingReplay in the command window will launch the program, without reloading in Matlab again all the data.



(a) (b) (c)

Figure 47: (a) and (b) Illustration of the computation of the COM position and body orientation from the three front markers. (c) construction of an ellipsoid representing the fly's body, placed at the COM.

A.5 Dynamics calculations

Calculations done for the study of the fly dynamics are performed in the *Dyn_compute.m* file. From, the marker positions, the center of mass (COM) of the fly is determined, as well as its speed and acceleration. Also, the vectors forming the body axis are computed, in order to extract the roll, pitch and yaw angles. This finally allows to compute the linear and angular speed and acceleration in the body axis. Only the three front markers of the flysuit, forming a triangle, are used in the computations. They determine entirely the position and orientation of the fly, and are supposed to be precise enough not to need a complicated fusion of the position of the four makers. The fourth marker is principally used for a correct tracking by the Vicon system, and to help the reconstruction and trajectory editing in Nexus (the tail marker trajectory is sometimes used to fill gaps present in other trajectories).

In order to determine the center of mass position, one fly wearing a fly suit was used just after death. It was hung to a thin wire attached consecutively to three of the fly suit's markers. Since the fly was supported only by one marker, the center of mass is along the vertical line passing by this marker. Then, the position of the fly suit was recorded in Vicon in each of the three positions. This allows to build three segments leaving from each marker, and intersecting at the center of mass position with a relatively good precision (see Figure 47(a)).

The body axis are determined by the three unit vectors showed in Figure 47(b). The components of the unit vectors form a rotation matrix $R_b = [u_x\ u_y\ u_z]$, from which the yaw ϕ, pitch θ and roll ψ angles can be easily extracted: $[\phi\ \theta\ \psi] = (R_b)_{CosineMatrix \to EulerAngles}$. To compute the angular speed, the rotation of the body axis from one frame to the other is calculated by taking the body axis at time t expressed in the axis at time $t-1$: $|\Omega| = R_{b,t-1}^{-1} \cdot R_{b,t}$. The angular velocities around the pitch, yaw and roll axis are then easily computed: $[\omega_\phi\ \omega_\theta\ \omega_\psi] = (|\Omega|)_{CosineMatrix \to EulerAngles} \cdot \frac{1}{\Delta t}$.

Performance	T160	T40	T20	Notes
Camera maximum frame rate at full resolution	120 fps	370 fps	500 fps	
Camera maximum frame rate at partial scan	2,000 fps	2,000 fps	2,000 fps	
Camera frame rates	30-2,000 fps	30-2,000 fps	30-2,000 fps	
Sensor Specification				
Sensor Type	CMOS	CMOS	CMOS	
Sensor	AVALON-16 (Custom Vicon Sensor)	VEGAS-4 (Custom Vicon Sensor)	VEGAS-2 (Custom Vicon Sensor)	
Sensor Resolution	4704 X 3456	2352 x 1728	1600 x 1280	
Number of Pixels	16,257,024	4,064,256	2,048,000	
Physical Sensor Size	18.35mm(H); 13.48mm(V); 22.77mm(Diagonal)	16.46mm(H); 12.10mm(V); 20.43mm(Diagonal)	11.20mm(H); 8.96mm(V); 14.34mm(Diagonal)	
Optical Format	>1 inch	>1 inch	1 inch	
Shutter Type	V-Shutter	Electronic freeze frame shutter	Electronic freeze frame shutter	
On Camera Processing				
256 Shades and Grayscale Processing	Yes	Yes	Yes	Note 1
Grayscale Depth	10 bit	10 bit	10 bit	
Sub pixel resolution	1,200,000 x 880,000 (1/256 pixel resolution)	600000 x 440000 (1/256 pixel resolution)	410000 x 325000 (1/256 pixel resolution)	
On-Board Processors	3 processors	3 processors	3 processors	Note 2
On-Camera Masking	Yes	Yes	Yes	Note 3
In-Camera Dynamic Large Blob Eliminator	Yes	Yes	Yes	Note 4
Software Masking	Yes	Yes	Yes	
Auto Masking	Yes	Yes	Yes	
On-Camera Thresholding	Yes	Yes	Yes	
2D Tracking	Yes	Yes	Yes	Note 5
Supersampling	Yes	Yes	Yes	
Camera Output Modes	5	5	5	Note 6
Full Frame Preview Output	Yes. See Preview mode above	Yes. See Preview mode above	Yes. See Preview mode above	Note 7
Back Focus Mode	Yes, using Preview mode	Yes, using Preview mode	Yes, using Preview mode	
Strobe/Ringlight Specification				
Strobe Types Available	Infrared (850nm), Near Infrared (780nm), Visible Red (623nm)	Infrared (850nm), Near Infrared (780nm), Visible Red (623nm)	Infrared (850nm), Near Infrared (780nm), Visible Red (623nm)	Note 8
Number of LEDs	320 (Visible); 252 (NIR + IR)	320 (Visible); 252 (NIR + IR)	320 (Visible); 252 (NIR + IR)	
Cover Types Available	Not required	Not required	Not required	
Strobe Electronics	Integrated, software reprogrammable and controlled	Integrated, software reprogrammable and controlled	Integrated, software reprogrammable and controlled	Note 9
Adjustable Illumination	Yes	Yes	Yes	
Adjustable Levels	1,000 (software controlled)	1,000 (software controlled)	1,000 (software controlled)	
Physical				
Camera Housing	Complex mold custom die-cast aluminum	Complex mold custom die-cast aluminum	Complex mold custom die-cast aluminum	
Camera Body Dimensions	207mm(H) X 130mm(W) X 75mm(D)	207mm(H) X 130mm(W) X 75mm(D)	207mm(H) X 130mm(W) X 75mm(D)	
Weight	1.8 kg including strobe, excluding lens	1.8 kg including strobe, excluding lens	1.8 kg including strobe, excluding lens	
Number of Camera Mount Points	1 (Universal optional with Cage)	1 (Universal optional with Cage)	1 (Universal optional with Cage)	Note 10
Camera Architecture				
Software and Firmware upgradeable	Yes	Yes	Yes	
Upgrade Methods	Any standard transmission method including FTP, email, CD, USB stick etc.	Any standard transmission method including FTP, email, CD, USB stick etc.	Any standard transmission method including FTP, email, CD, USB stick etc.	
Cabling	Cat 5e	Cat 5e	Cat 5e	Note 11

Camera Architecture	T160	T40	T20	Notes
Connectors	A single connection between Camera and Giganet. Single connector to strobe. Two serial ports in a single Auxiliary connector			Note 12
Power Supply	All power to the Cameras from the power supply within the Giganet			Note 13
Max. No. of Cameras Supported with Each Hub/Net	Up to 10 cameras per Giganet	Up to 10 cameras per Giganet	Up to 10 cameras per Giganet	
Lens Type Supported	Vicon Lens / SLR	C-Mount / Vicon Lens / SLR	C-Mount	
Zoom Lens Supported	Yes	Yes	Yes	
Lenses Available (C-Mount)	None	Yes	Yes	
Lenses Available (35mm SLR)	Canon / Sigma (Canon EF Mount)	Canon / Sigma (Canon EF Mount)	None	
Motorized optics	With Canon / Sigma Lens	With Canon / Sigma Lens	No	Note 14
Plug and Play Compatibility	Yes	Yes	Yes	
Mixed Camera System Compatibility	Interoperable with MX-F40, MX-F20, MX-40+, MX-20+, MX-13+, MX-3+, MX-40, MX-13, MX-3			
System Connectivity/Communication	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet	
Maximum Number of Cameras in System	Unlimited	Unlimited	Unlimited	
Custom Control Interface	Yes	Yes	Yes	
Communication Status Indicators	Yes	Yes	Yes	
Integrated Camera Display Panel	Optional	Optional	Optional	
Camera Number Indicator	Optional	Optional	Optional	
Camera Status Indicators	On camera and in software	On camera and in software	On camera and in software	
IP Addressable	Yes	Yes	Yes	
IP Reconfigurable	Yes	Yes	Yes	
Genlock to External Video Source	Yes	Yes	Yes	
Synchronize to General External Signal	Yes	Yes	Yes	
External Sync output	Yes	Yes	Yes	
External Sync box needed	No	No	No	
External VGA output	Not required	Not required	Not required	
External A/D Sync + Clock	Yes	Yes	Yes	Note 15
Camera Diagnostic Interface	Yes	Yes	Yes	
Cooling	Advanced thermal design	Advanced thermal design	Advanced thermal design	
Camera protection level	IP 61	IP 61	IP 61	

Notes

T160/T40/T20

1 Full marker grayscale. Marker centers are calculated based on every pixel of grayscale available for the marker, not just the detected marker edges. An on-camera circularity test ensures merged or partially occluded markers which need high-level processing are sent in full grayscale to the PC.

2 1. Scalable multi-processor extracts grayscale markers from sensor image. Although one physical device, it provides parallel processing of datastream. 2. Digital Signal Processor (DSP) locates markers in 2D and calculates centers and radii. 3. High performance and highly configurable soft-processor streams grayscale and marker center data over Ethernet to host PC. Also handles housekeeping and configuration tasks.

3 On-camera masking removes areas of the sensor where undesirable static light sources are recorded, for example strobes from other cameras.

4 Camera firmware automatically removes undesirable image data including both large blobs (e.g. sunlight reflections) and/or an unusually large number of blobs.

5 In combination with the Supersampling, the 2D tracking algorithms allow for the markers' movement to be tracked from frame to frame on the camera. This allows even fast motion of close markers to be easily processed.

6 Automatic (centers for circular markers, grayscale for overlapping/partially occluded markers), Centers Only, Grayscale Only, Centers / Grayscale, and Preview (the entire sensor image).

7 Ten times faster than previous generations of Vicon cameras and taking full advantage of the Gigabit Ethernet connection to the PC.

8 Infrared (850nm), Near Infrared (780nm) and Visible Red (623nm) surface mount LED strobes.

No requirement for secondary optics or strobe covers. This new generation of strobe is twice as bright as previous generations making setup in awkward environments easier.

9 With built-in temperature monitor.

10 Optionally, the Vicon Cage allows accessories to be attached to the camera without additional mounting hardware.

11 Gigabit Ethernet with power and sync over Ethernet.

12 Using only connectors of the highest quality, guaranteeing reliable use for years to come.

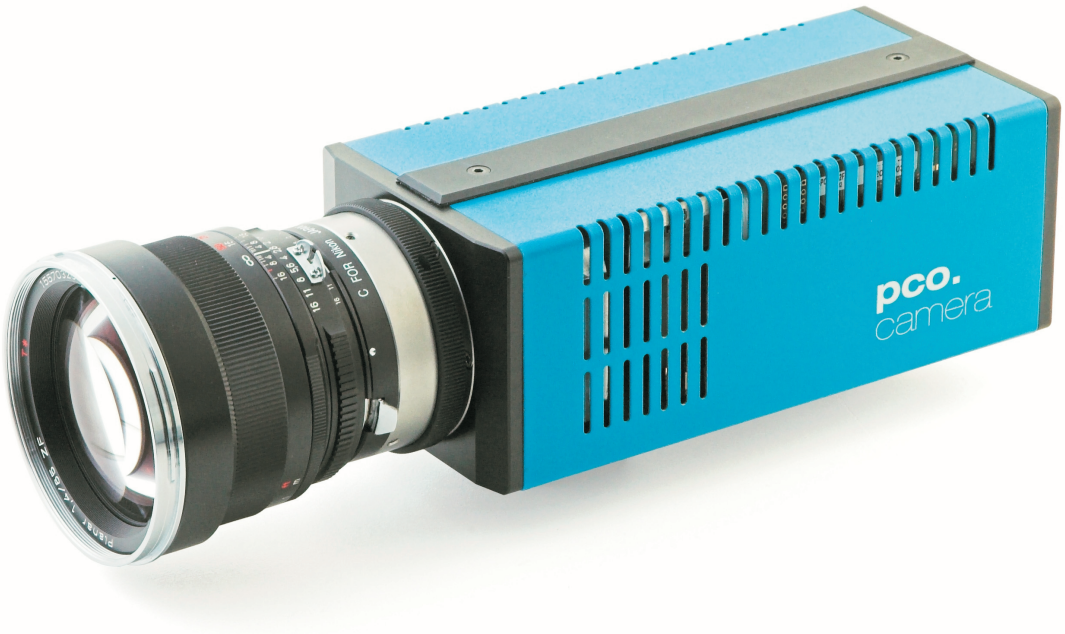
13 750 Watts per Giganet.

14 18-55 mm Sigma lens available, others on request.

15 Internal A/D is synchronized to the cameras, but an external synchronization is programmable for third party equipment.

pco.1200 hs / pco.1200 s
digital high speed 10bit CMOS camera system

- 636 fps at full resolution (1357 fps at VGA resolution)
- high resolution (1280 x 1024 pixel)
- exposure time range 50ns - 5s
- image memory in camera (camRAM up to 4 GB)
- standard interfaces (IEEE 1394, camera link)
- mobile solution with Tablet PC LE1600



pco.
imaging

1288
EMVA Standard Compliant
data available on
www.pco.de

pco.1200 hs / pco.1200 s

This high speed 10bit CMOS camera system comprises advanced CMOS and electronics technology. With the new approach to integrate the image memory into the camera itself (camRAM up to 4GB), it enables unmatched fast image recording with 1GB/s (hs) / 820MB/s (s). The system features an excellent resolution (1280 x 1024 pixel) and low noise. It consists of a compact camera with an external intelligent power supply. The image data are transferred via customer selectable standard data interfaces to a computer (IEEE 1394a Firewire, Camera Link). The available exposure times range from 1µs (50ns opt.) to 5s (hs) / 1µs to 1s (s). This digital camera system is perfectly suited for high speed camera applications such as material testing, fast inspection, external crash tests or super slow motion image recordings for video clips and advertising.

technical data

	unit	setpoint	pco.1200 hs	pco.1200 s
resolution (hor x ver)[1]	pixel		1280x1024	1280x1024
pixel size (hor x ver)	µm²		12.0 x 12.0	12.0 x 12.0
sensor format/ diagonal	mm²/ mm		15.36x12.29/ 19.67	15.36x12.29/ 19.67
peak quantum efficiency	%	@ 520nm typical	25	25
full well capacity	e⁻		63 000	63 000
image sensor			MT9M413	MT9M413
dynamic range	dB	@ camera	59.6	59.6
dynamic range A/D[2]	bit		10	10
readout noise	e⁻ rms	@ 66 MHz @ 67.7 MHz	41	 41
imaging freq., frame rate	fps	@ full frame @ ROI VGA	636 1357	501 1068
pixel scan rate	MHz	hs: dual speed	66 / 86	67.7
A/D conversion factor	e⁻/count		25	25
spectral range	nm		290..1100	290..1100
exposure time	s	hs: 50ns opt.	1µs..5s	1µs..1s
anti-blooming factor		typical	no blooming	no blooming
smear	%		no smear	no smear
binning horiz.	pixel		1, 2	1
binning vert.	pixel		1, 2	1
dark current	e⁻/pixel·s	@25 °C typical	5900	5900
region of interest (ROI)	pixel	horizontal vertical	steps of 10 steps of 1	steps of 10 steps of 1
interframing time (PIV mode)	ns	@ FWHM[3] and 100% fullwell signal	70	not available

pco.
imaging

technical data

	unit	setpoint	pco.1200 hs	pco.1200 s
non linearity	%	full temperature	< 2	< 2
uniformity darkness DSNU[4]	e^- rms	@ 90% center zone	< 700	< 700
uniformity brightness PRNU[5]	%	typical	0.6	0.6
trigger auxiliary signals		internal external	software TTL level	software TTL level
power consumption	W	typical maximum	25 40	25 40
power supply	VAC		90..260 (12VDC opt.)	90..260 (12VDC opt.)
mechanical dim. camera (w x h x l)	mm^3		84 x 66 x175	84 x 66 x 175
mechanical dim. power supply (w x h x l)	mm^3		135 x 51 x 195	135 x 51 x 195
weight	kg		1	1
operating temp. range	°C		+5..+40	+5..+40
operating humidity range	%		10..90	10..90
storage temp. range	°C		-20..+70	-20..+70
optical input			Nikon f-mount, c-mount	Nikon f-mount, c-mount
data interface			IEEE1394a, camera link	IEEE1394a, camera link
CE certified			yes	yes

[1] horizontal versus vertical
[2] Analog-to-Digital-converter
[3] full width half maximum
[4] dark signal non-uniformity
[5] photo reponse non-uniformity

pco.
imaging

software:	Camware software for camera control, image acquisition and archiving of images in various file formats, WindowsXP and later, 32bit-dynamic link library (DLL) is available for user customisation and integration on PC platforms (software development kit - SDK), software is operational in either single mode or with built-in recorder functions, drivers for popular third party software packages are available (see website)
options:	CMOS image sensor in color version custom-made versions camRAM available in: 1 GB, 2 GB, and 4 GB

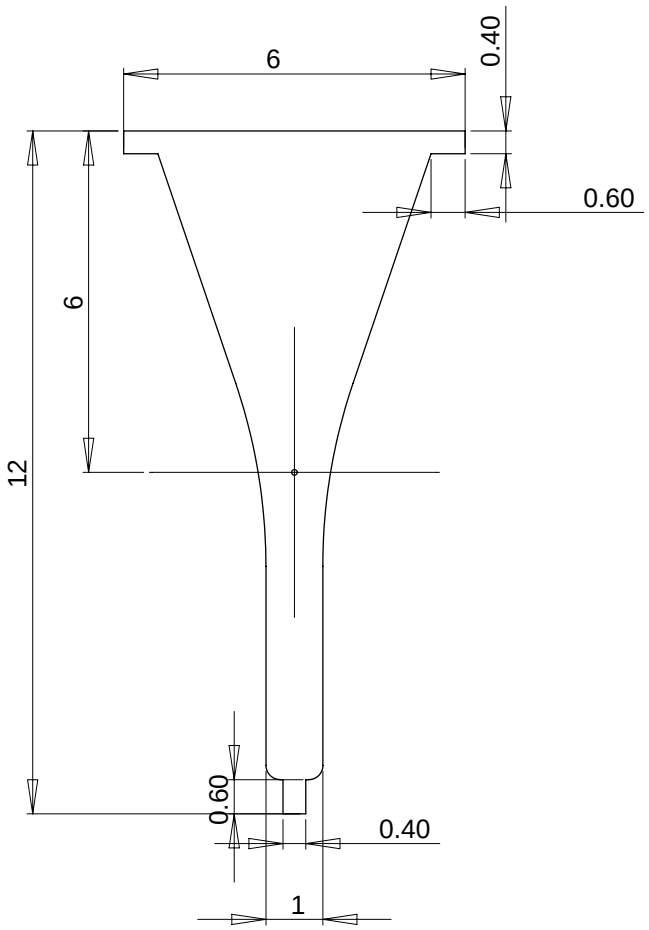
pco.1200 hs frame rate table [frames per second][1]

pixelclock exposure time	66 MHz 1/fps / <1/fps	86 MHz 1/fps / < 1/fps
1280x1024 pixel (full frame)	488 / 486	636 / 634
1280x512 pixel	977 / 969	1272 / 1263
1280x256 pixel	1953 / 1923	2545 / 2506
1280x128 pixel	3906 / 3788	5090 / 4936
1280x64 pixel	7813 / 7353	10180 / 9581
1280x32 pixel	15625 / 13889	20360 / 18098
1280x16 pixel	31250 / 25000	40720 / 32576

pco.1200 s frame rate table [frames per second][1]

pixelclock exposure time	67.7 MHz 1/fps / <1/fps
1280x1024 pixel (full frame)	501 / 499
1280x512 pixel	1002 / 994
1280x256 pixel	2003 / 1972
1280x128 pixel	4006 / 3883
1280x64 pixel	8011 / 7533
1280x32 pixel	16019 / 14216
1280x16 pixel	32023 / 25545

[1] The given resolutions are selected for the frame rate calculations in the tables only, they are not mandatory. For ROIs see "technical data" table on page 2.


6
0.40
0.60
6
12
0.60
0.40
1