Abstract – Robotic researches can contribute to the restoration of some functions lost by handicapped people. The over-cost generated by the additive potentialities must be affordable and related to the value of the usual product. In most cases, autonomous functions are direct transpositions of solutions applied in industrial robotics. If we consider that in addition with cost, security is a supplementary constraint of rehabilitation robotics, an important research effort is needed to propose technological components. The aim of this paper is to introduce a functionality within the context of the following plan: “technical achievement and psychological analysis of a master/slave robot assistance for the invalid person” (HTSC: Human/Technology Complex System). This application allows a severely handicapped person to handle at the same time a movable platform and a Manus® arm. In this paper, we will present an “automatic tracking” functionality we developed in order to keep the Manus® arm automatically close to the patient by using two omnidirectional vision sensors.

Index Terms - Wheelchair, Manus® Arm, Tracking, Automatic follow, CAMShift, Stereoscopic Omnidirectional Vision, Assistive technology.

I. INTRODUCTION

The assistive robot proposes solutions that are more and more adapted to severe handicapped. Robotics, automatics and information technologies have shown a great improvement in order to make up for the lack of the invalid person. Robotics can provide technological solutions for completing medical assistance [20][22]. The objective is to give some hours of independence to persons without the presence of a third party. Manipulation is at the center of some main life functions such as carrying, picking up, moving objects on shelves or on the ground. Rehabilitation robotics aims at partially restore user’s manipulative function by interpose a robot arm between the user and the environment. The Manus® [8] and the Master® [12] propose interesting solutions which are put into practice and improved [18].

Robotic assistance system can be divided into three main configurations. The first one consist in a table-mounted manipulator which operate in a known environment. Another way consists of mounting a manipulator on a powered wheelchair [3]. Several demonstrations in real situations, notably with Manus® arm, show the adaptability of the approach which adds to indoor outdoor operations. The last configuration is the most complex but the most versatile one. Moreover, it could be used by severely handicapped people: bedridden or quadriplegic people for example. In this case a manipulator arm is mounted on a mobile platform. However, contrary to the first two configurations, this one is not marketed and stays at a research level. It is due principally to the difficulty to control a complex robot in an unknown or partially unknown environment and, of course, the cost of an autonomous robot. In fact, even if machines get improved day by day, their use represents a real ergonomic problem.

Several research have demonstrate that handicapped people doesn’t accept an assistive arm onto the wheelchair. In fact, they feel like being assimilated to a robot. In this way, we have chose to place a Manus® arm onto a mobile robot as in the ARPH project [6]. The first functionality implemented is to keep the assistant robot close to the wheelchair autonomously. Then, when the handicapped person desire to use the Manus® arm, he takes the control of the mobile robot and move the platform wherever he wants and can also use the manipulator.

After presenting briefly our assistance device architecture in section II, we will introduce the omnidirectional vision in section III. Section IV will present a new functionality which consists in an automatic visual tracking and follow of a wheelchair by using a stereoscopic omnidirectional vision sensor. We will see how we use the omnidirectional vision in two different ways. Firstly, to estimate the free space around the mobile robot and secondly, to identify and track a wheelchair in the environment. Some experimental results are shown in section V. Finally, we will present some conclusions about our work and discuss about the future improvements.

II. GLOBAL DESCRIPTION

Our goal is to create an application that lies within the scope of a plan aiming at separating the Manus® arm from the patient’s wheelchair, in order to make him accept the robotic arm more easily. Figure 2 shows a diagram of the application. It is made of two parts. In one hand, the wheelchair equipped
with a standard PC and an interface screen and a joystick for control without anymore equipment or assistive technology unlike many other projects [17][21]. In the other hand, we have the mobile robot equipped with two omnidirectional vision sensors and a Manus® arm.

This robotic assistance will be used in:

- **Automatic mode**: track and follow a wheelchair autonomously without holding up the wheelchair progression (with the arm tucked up).
- **Remote mode**: Standing close to the wheelchair in order to be manually controlled. It is in this mode the arm can be used.

There are many advantages of using omnidirectional vision. Firstly, in one acquisition, we obtain a full view of the environment with no mechanical system. Secondly, even if the interpretation of omnidirectional picture is difficult for novices, we can provide with some little process a “classical perspective view” of the scene. Finally, providing a picture in a chosen direction is instantaneous.

The omnidirectional vision system we use is made of a digital color video camera and a hyperbolic mirror. Figure 3 shows an omnidirectional view of an environment with a wheelchair in the field of view.

### III. THE OMNIDIRECTIONAL VISION

Main vision applications in mobile robotics use the classical pinhole camera model. Thus according to the lens used, the field of view is limited. Nevertheless, it is possible to enlarge the field of view by using cameras mounted in several directions [14], but the information flow is very important and time consuming. Other applications [6] use only one camera, with a rotation motion, in order to sweep a large space. The disadvantage of such system is that the camera’s movement takes time; and what’s more, a mechanical looseness can appear in the course of time. To get wide-angle pictures another possibility exists: omnidirectional vision. These kinds of sensors allow acquiring scenes with 360° field of view [19]. There are two major classes of omnidirectional vision systems. First of all, systems made of a mirror and a camera are called “catadioptric systems” [3][15]. The second one is composed of a classical camera with a fish-eye lens; such mountings are called “dioptric systems” [4]. We focus on the first class.

There are many advantages of using omnidirectional vision. Firstly, in one acquisition, we obtain a full view of the environment with no mechanical system. Secondly, even if the interpretation of omnidirectional picture is difficult for novices, we can provide with some little process a “classical perspective view” of the scene. Finally, providing a picture in a chosen direction is instantaneous.

The omnidirectional vision system we use is made of a digital color video camera and a hyperbolic mirror. Figure 3 shows an omnidirectional view of an environment with a wheelchair in the field of view.

### IV. THE AUTOMATIC TRACKING

One of the functionalities we want to provide for a severe handicapped person is the automatic tracking of a wheelchair. The main goal is to enable the mobile robot to track a wheelchair with no mark-up and no particular equipment onto the wheelchair. Our method is completely based upon the stereoscopic omnidirectional vision system.

In fact, the omnidirectional vision is used in two different ways. Firstly, we match the two omnidirectional pictures in order to build an instantaneous map of free space around the mobile robot. This 2D map is used to allow or not the move of the robot. Secondly, we use the omnidirectional vision to identify and track a wheelchair by using an algorithm called CAMShift (Continuously Adaptive Mean Shift) [2].

#### A. The free space map

In this part, the two omnidirectional sensors are used as goniometric sensors rather than a vision system. In fact, what we want to compute are distances of the different verticals parts called natural landmarks observed around the mobile robot. In this way, we can build an instantaneous map that represents the actual free space around the mobile robot as in...
The following down part describe the method we developed to carry out this task.

First of all, on each omnidirectional picture, we define a signal which represents the mean RGB color from a ring localized around the horizon in the field of view. In fact, what we want to detect are the natural vertical beacons of the environment. Omnidirectional vision system project those vertical parts of the environment according to radial straight lines onto the image. During this computation, it is very important that the rings are centered onto the projection of the revolution axis of the mirror. Otherwise, we will not compute the mean RGB color according to the projection of the vertical elements of the environment. This centering task is automatically done with a circular Hough transform [1]. In fact, we look for a circle corresponding to the projection of the black needle situated onto the top of the hyperbolic mirror (see Figure 4) which is situated onto the center of the mirror. It gives us the center’s coordinates $x_c$ and $y_c$.

Then, the two 1D mean RGB signals are computed from the ring (Figure 5) and matched together according to a rule of visibility.

In fact, if an object is detected from one omnidirectional sensor, it will be visible in a certain area of the other one, according to the distance between the object and the mobile robot. Figure 6 shows the correspondences between the angle of the left sensor and the angle of the right sensor according to different distances. The distance between the left and right sensors is 40 centimeters. Actually, the more the object is close to the mobile robot, the more the two angles are different.

\[
Y = \left( \sum Y \right)/\left( \sum Y^2 \right) - \left( \sum X \right)/\left( \sum X^2 \right)
\]

(1)

Figure 7: The two extracted mean color signals from omnidirectional pictures of Figure 5 and the matching between the left sensor (upper) and the right sensor (bottom) before the filtering process. The matching has only been done from 0 to 180 degree because the direction of the movement was forward.

Figure 8: The two extracted mean color signals from omnidirectional pictures of Figure 5 and the matching between the left sensor (upper) and the right sensor (bottom) after the filtering process.
The search area is defined with the visibility rule (Figure 6). Figure 9 shows the two extracted signals for the first gradient detection of the Figure 8.

Then we have to filter the matching because some incoherencies could appear. For example, some matched angles could be crossed. If this case appears, it would mean that we can see behind an object, it is impossible. So, after the matching process, we have a filter process. The filtering consist in a two stages process:

1. thinning filter: as you can note on the Figure 7, a matching could be close to another one. So, we only keep the most significant matching according to the correlation value,
2. crossing filter: as you can note on the Figure 7, a matching could cross another one. In practice, it is impossible except in one particular case: the object is situated in front of the robot and between the two sensors. Otherwise, it means that an object could be seen through another one. Once again, the filter consist to keep the most significant matching according to the correlation value.

After this filtering process we obtain a set of corresponding angles between the left and right sensors. The last thing we have to do is to compute distances of the object by triangulation. It gives us a free space map around the mobile robot as shown on Figure 10.

Pictures of Figure 5 was acquired during a motion through a corridor. In this example, we can see a disadvantage of vision. In fact, we can see through the window and match objects seen behind it (left part of Figure 8). This is visible in the map of free space built with these two acquisitions on Figure 10.

B. The tracking

As the wheelchair is not equipped with any particular mark-up, we have to track it as it is. In this way, we use the CAMShift algorithm which perform a tracking by using an image of the object to track. The Continuously Adaptive Mean Shift (CAMShift) algorithm [2], is based on the mean shift algorithm [7], a robust non-parametric iterative technique for finding the mode of probability distributions.

a. Principle

Given a color image and a color histogram, the image produced from the original color image by using the histogram as a look-up table is called back-projection image. If the histogram is a model density distribution, then the back projection image is a probability distribution of the model in the color image. CAMShift detects the mode in the probability distribution image (PDF) by applying mean shift. In a single image, the process is iterated until convergence (or until an upper bound on the number of iterations is reached). CAMShift’s main difference towards mean shift is that it can fit in size the tracked object. In our omnidirectional images, the tracked object can quickly scaled. ( The wheelchair is two times smaller when seen at 2 meters than at 1 meter. ).

The PDF may be determined using any method that associates a pixel value with a probability that the given pixel belongs to the target. A common method is known as Histogram Back-Projection. In order to generate the PDF, an initial histogram is computed as explain in the following section.

The histogram is quantized into bins, which reduces the computational and space complexity and allows similar color values to be clustered together. The histogram bins are then scaled between the minimum and maximum probability image intensities (Equation 3).

Histogram back-projection is a primitive operation that associates the pixel values in the image with the value of the corresponding histogram bin. The back-projection of the target histogram with any consecutive frame generates a probability image where the value of each pixel characterizes probability that the input pixel belongs to the histogram that was used. Given that \( m \)-bin histograms are used, we define the \( n \) image pixel locations \( \{x_i\}_{i=1}^n \) and the histogram \( \{\hat{q}_h\}_{h=1}^m \).

We also define a function \( c: \mathbb{R}^2 \rightarrow \{1...m\} \) that associates to the pixel at location \( x_i \) the histogram bin index \( c(x_i) \). The histogram is computed as follow:

\[
\hat{q}_h = \sum_{i=1}^n \delta[c(x_i) - h] 
\]
In all cases the histogram bin values are scaled to be within the discrete pixel range of the 2D probability distribution image using

\[
\hat{p}_u = \min\left( \frac{255}{\max(q)} q, 255 \right)
\]

That is, the histogram bin values are rescaled from \([0, \max(q)]\) to the new range \([0, 255]\), where pixels with the highest probability of being in the sample histogram will map as visible intensities in the 2D histogram back-projection image.

b. Initialization of the CAMShift.

In order to init the CAMShift, we need images of the wheelchair. To do this, we capture the scene’s background. Then, the user presents the wheelchair in four different directions, and for each side we compute the subtraction between the background and the current image. To get better results, we use a 8-connectivity filter. The most significant group extracted is the wheelchair. In our future work we will deal with the problem of shadows extracted in the image.

The model to track is recorded as an image that represents the object to track. In our case, we chose to have four possible models, a front view of the wheelchair, a back view and the left and right side views (Figure 11). All the time, the user can redefine the models by presenting the four sides of the object to track. In this way, we want our robot to track any wheelchair with no additional beacons.

c. Target localization.

The CAMShift algorithm can be summarized in the following steps:

1. Calculate a color probability distribution of the region centered at the Mean Shift search area.
2. Iterate Mean Shift algorithm to find the center of the probability image. Store the zero\(^{th}\) moment (distribution area) and center location.
3. For the following frame, center the search area at the mean location found in step 2 and set the new size of the search area.

The mean location (center) within the search area of the discrete probability image computed in step 2 is found using moments ([13], [2]). Given that \(I(x, y)\) is the intensity of the discrete probability image at \((x, y)\) within the search area.

\[
M_{00} = \sum_x \sum_y I(x, y) \quad (4)
\]

2) Find the first moment for \(x\) and \(y\)

\[
M_{10} = \sum_x \sum_y xI(x, y) \quad (5)
\]

\[
M_{01} = \sum_x \sum_y yI(x, y) \quad (6)
\]

3) Compute the mean search area location

\[
x_c = \frac{M_{10}}{M_{00}} \quad \text{and} \quad y_c = \frac{M_{01}}{M_{00}} \quad (7)
\]

The Mean Shift component of the algorithm is implemented by continually recalculate new values of \((x_c, y_c)\) for the area position computed in the previous frame until there is no significant shift in position. The maximum number of Mean Shift iterations is usually taken to be 10-20 iterations. Since sub-pixel accuracy cannot be visually observed, a minimum shift of one pixel in either of the \(x\) and \(y\) directions is selected as the convergence criteria. Furthermore, the algorithm must terminate in the case where \(M_{00}\) is zero, which corresponds to an area consisting entirely of zero intensity.

The use of moments to determine the scale and orientation of a distribution in robot computer vision is describe in [13] and has been used for vision in computer games in [11] and for head and face orientation and tracking in [2].

Defining the first and second moments for \(x\) and \(y\)

\[
M_{20} = \sum_x \sum_y x^2 I(x, y) \quad (8)
\]

\[
M_{02} = \sum_x \sum_y y^2 I(x, y) \quad (9)
\]

\[
M_{11} = \sum_x \sum_y xy I(x, y) \quad (10)
\]

The first two eigenvalues (the length and width of the probability distribution) are calculated in closed form as follows, from the intermediate variables \(a, b\) and \(c\)
\[ a = \frac{M_{20}}{M_{00}} - x_c^2 \]  
\[ b = 2 \left( \frac{M_{11}}{M_{00}} - x_c y_c \right) \]  
\[ c = \frac{M_{02}}{M_{00}} - y_c^2 \]  

We find the orientation:

\[ \theta = \frac{1}{2} \arctan \left( \frac{b}{a - c} \right) \]  

The distances \( l_1 \) and \( l_2 \) from the distribution center are given by

\[ l_1 = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \]  
\[ l_2 = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \]  

### d. OmniCAMShift

CAMShift is useful when using images given by planar camera. When using omnidirectional images, the use of CAMShift needs a prior step: un-warping the image. In our application we tried to use CAMShift on un-warped images, but un-warping the image is at least 3 times more time consuming than the CAMShift algorithm itself. Even when trying to un-warped the very area needed.

Because of this waste of time we choose to create the OmniCAMShift in order to work directly on omnidirectional images.

In the CAMShift algorithm, the tracking area is represented by a rectangle whereas in OmniCAMShift the tracking area is set by a part of a ring defined by four criteria:

1. a min ray (\( r_{\min} \)),
2. a max ray (\( r_{\max} \)),
3. a min angle (\( \text{angle}_{\min} \))
4. a max angle (\( \text{angle}_{\max} \)).

We create a back image according to the ring criteria. \( x_s, y_s \) are the center’s coordinates of the sensor in the image. \( B(x, y) \) is the set of pixel of the back image.

\[ P(x,y) = \begin{cases} I(x,y) & r_{\min} < \sqrt{(x - x_c)^2 + (y - y_c)^2} < r_{\max} \\ 0 & \text{otherwise} \end{cases} \]  

Then we compute the center of the track area using center of mass. The \((x_c, y_c)\) coordinates are used to find the new parameter of the track area.

\[ \text{angle}_{\text{average}} = \arctan \left( \frac{y_c - y_s}{x_c - x_s} \right) \]  
\[ \text{angle}_{\min} = \text{angle}_{\text{average}} - \frac{(\text{angle}_{\max} - \text{angle}_{\min})}{2} \]  
\[ \text{angle}_{\max} = \text{angle}_{\text{average}} + \frac{(\text{angle}_{\max} - \text{angle}_{\min})}{2} \]  

Finally, when the OmniCAMShift applied onto the two omnidirectional images gives us a direction for the left sensor and another for the right sensor, we could compute the distance of the wheelchair and the mean direction. In this way, we are able to control the mobile platform to move to this goal according to the free space map. If an object or an obstacle come through the path between the wheelchair and the robot, for the moment we just stop the robot until the obstacle (person) goes away from the path, no efficient obstacle avoidance strategy is implemented yet, it is plan in our future works.

e. Avoiding lost of tracking.

CAMShift and so OmniCAMShift tracking is a real-time algorithm that endeavours to maximize the correlation between two statistical distributions. The correlation, or similarity between two distributions is expressed as a measurement derived from the Bhattacharyya coefficient [7].

Properties of the Bhattacharyya coefficient such as its relation to the Fisher measure of information, quality of the sample estimate, and explicit forms for various distributions are discussed in [9][16].

The derivation of the Bhattacharyya coefficient from sample data involves the estimation of the densities \( p \) and \( q \), for which we employ the histogram formulation. The discrete density \( \hat{q} = \{ \hat{q}_u \}_{u=1}^{m} \) (with \( \sum_{u=1}^{m} \hat{q}_u = 1 \)) is estimated from the \( m \)-bin histogram of the target model, while \( \hat{p}(y) = \{ \hat{p}_u(y) \}_{u=1}^{m} \) (with \( \sum_{u=1}^{m} \hat{p}_u = 1 \)) is estimated at a given location \( y \) from the \( m \)-bin histogram of the target candidate.

Therefore, the sample estimate of the Bhattacharyya coefficient is given by

\[ \hat{\rho}(y) \equiv \rho(\hat{p}(y), \hat{q}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y) \hat{q}_u} \]  

Based on equation (4) we define the distance between two distributions as

\[ d(y) = \sqrt{1 - \rho(\hat{p}(y), \hat{q})} \]
The statistical measure (22) is a metric valid for arbitrary distributions, being nearly optimal (due to its link to the Bayes error [16]) and invariant to the scale of the target.

The similarity measure reflects the correspondence between the original model and the current tracked model (candidate). If the coefficient’s value is under 0.6, we lost the tracked object. As we track the same object with two different cameras, we have two different results. So, if one of the OmniCAMShift is lost (Table 1), it can be re-initialized with the data of the other camera’s OmniCAMShift.

<table>
<thead>
<tr>
<th>Similarity measure’s values</th>
<th>State of the tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>val. &lt;=0.6</td>
<td>Do not correspond to the original model</td>
</tr>
<tr>
<td>0.6 &lt; val. &lt;0.8</td>
<td>Uncertainty interval.</td>
</tr>
<tr>
<td>0.8 &lt;= val.</td>
<td>Correctly correspond to the original model</td>
</tr>
</tbody>
</table>

Table 1: Use of the similarity measure criteria.

As the two camera are closed one to each other (40 centimeters), the position of the tracked object are quite similar in both pictures. We redefine the coordinates of the CAMShift and it corrects the slight difference between the two coordinates by re-centering itself.

Furthermore we use the similarity measure coefficient to detect which side of the wheelchair we are currently tracking. At each frame, we test the four different models on the current area. The model which provide the higher criteria define which side we are observing.

V. RESULTS

Our application needs to respond quickly to its environment. We are using 2 cameras providing images in 1024x768x24bpp at 4 frames per second.

![Diagram representing the CPU consumption for each step.](image)

Figure 12: Diagram representing the CPU consumption for each step.

The Figure 12 represents the CPU consumption for the different steps. Building free space map and path planning have constant CPU consumption whereas OmniCAMShift depends on the area of pixel to compute. If the object is small in term of pixel in the image, the time for OmniCAMShift will be about 70ms. But this task can reach 190ms. If so, we can not maintain the 4 fps.

By now, we have tested our automatic controlled tracking in our laboratory. Results are very promising since the platform has done its missions during relatively long distances.

The interested reader is invite to consult videos demonstration at

The OmniCAMShift is very robust. As we use an RGB space, quantized in 32x32x32 bins, we can also use this method in quite dark indoor environments. Anyway, most of the time wheelchairs appears to be black, only new models are using colors. The Figure 13 shows a sequence of tracking.

![Tracking wheelchair using OmniCAMShift.](image)

Figure 13: Tracking wheelchair using OmniCAMShift.
For each image, the back image is shown. To improve computing time, we only create the back image for the area within and around the track zone. In this back image we can see the well extracted wheelchair in white. On the last image of Figure 13, you can note that the mobile robot has rotate. Since we don’t have any odometry information, we compute the rotation motion by emphasis the omnidirectional image at time t and t-1. Results obtained during our different experimental tests are relatively accurate. In fact, the orientation between the mobile robot and the wheelchair is computed with an error lower than one degree. In term of experimental tests are relatively accurate. In fact, the time

Since we don’t have any odometer information, we compute

Within and around the track zone. In this back image we can

see the well extracted wheelchair in white. On the last image

within a given area, regardless the spatial color's organization. So, if an object has the same color probability the OmniCAMShift easily “jump” on it too. To avoid those problems we wish to create an algorithm which analyze the target in order to create multiple CAMShift for one tracked object. Those OmniCAMShift would be linked together with spatial rules.

VI. CONCLUSIONS AND PERSPECTIVES

In this paper, we have seen the use of omnidirectional vision to estimate the free space around the mobile robot and then to track and follow a wheelchair in an unknown environment. In the first part of this paper, we have shown how we match two 1D signals computed from omnidirectional pictures in order to compute a depth map corresponding to the free space around the mobile robot. The algorithm we developed is robust but the tests we have made showed us that using an instantaneous free space map is not sufficient. Our robot navigates in an unknown environment, so we are working on a Simultaneous Localization And Mapping (SLAM) algorithm [10] to build a more complete map during the process. This would allow us to implement some obstacle avoidance procedures. In this way, we are currently adding a set of Infra-Red range distance sensor to avoid problems such as seeing through windows as shown in Figure 10.

The tracking using OmniCAMShift is very robust. The only annoying point is that if one color takes a major part of a model, the tracking may “jump” to an object despite of the presence of other color in the model. Also OmniCAMShift works on probability of colors within a given area, regardless the spatial color's organization. If an object has the same color probability the OmniCAMShift easily “jump” on it too. To avoid those problems we wish to create an algorithm which analyze the target in order to create multiple CAMShift for one tracked object. Those OmniCAMShift would be linked together with spatial rules.

ACKNOWLEDGMENT

REFERENCES


