T U M

I N S T I T U T F Ü R I N F O R M A T I K

Robust Tracking of Humans by Intelligent Multi-modal Fusion of Visual Cues using Machine Learning

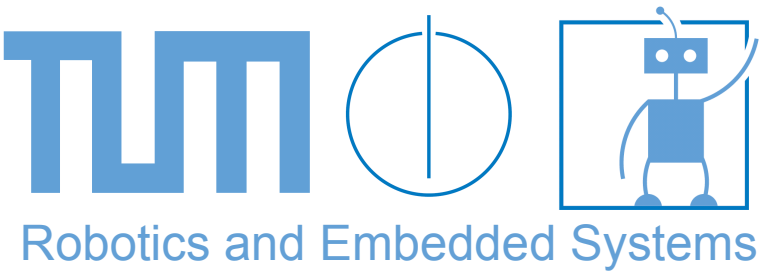
Suraj Nair and Alois Knoll

TUM-I1023
November 10

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-11-I1023-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München



Technische Universität München
Fakultät für Informatik

Technical Report

Robust Tracking of Humans by Intelligent Multi-modal Fusion of Visual Cues using Machine Learning

Suraj Nair, Alois Knoll
nair,knoll@in.tum.de

November 17, 2010

Abstract

This document formulates a technical report proposing a methodology to improve the robustness of vision based human tracking systems in real-world scenarios. We propose a system consisting of 2 stages, 1. a vision based human tracking system using multiple visual cues, 2. an intelligent multi-modal fusion module, using machine learning techniques, to determine the right weights for each visual modality for the operating environment of the system. The function of the second stage is to perform on line analysis of parameters in the current scene that influences the performance of the tracker. Depending on this analysis, optimal weights are generated for each visual modality, indicating its contribution in the current scene. With such a fusion module we intend to boost the robustness of the system.

Contents

Abstract iii

1 Introduction 1

2 Prior Art 3

3 The Human Tracking System 5

4 Experiments in Real-World Scenarios 11

5 Proposed Intelligent Multi-Modal Fusion using Machine Learning 15

6 Expected Improvement in Performance 21

7 Conclusions and Future Work 23

1 Introduction

Vision based human tracking systems are gaining importance in many real-world applications. Their primary objective is to localize and track humans in real-time under different scenarios. A variety of human tracking systems exist depending on the type of application and operating environments. Visual servoing, surveillance, human robot interaction, etc. are some applications where these systems are deployed. Although such systems already exist, they are generally tuned to their specific application scenarios in order that they perform within their specifications. Hence it is a challenge to develop a generic system which adapts itself to a variety of applications under dynamic tracking environments, while maintaining a high degree of robustness.

Performance of a vision based tracking system depends on different factors such as the tracking environment, the visual modalities used, sensor parameters, etc. For example, systems which perform well under controlled lighting conditions suffer when lighting conditions change or fluctuate. Hence, relying on a single visual modality is not always sufficient. Using a fusion of multiple visual modality is an option but also has its disadvantages. Some modalities are suitable under certain conditions. However, they reduce the robustness of the system in less favorable conditions. Multi-modal fusion works well under static conditions but could affect the robustness when the tracking environment changes.

In this report we introduce a vision based human tracking system which is capable of localizing and tracking multiple people in 3D using a fusion of visual modalities. Under the framework of our system we propose a machine learning based module to perform intelligent multi-modal fusion of the visual modalities. With such a system we tend to improve the robustness of the existing tracking system in terms of self adaptability to changing tracking conditions. It can be viewed as a general purpose human tracking system suitable for any application, operational under a large range of environments, both indoors and outdoors. The fusion module has a primary task of analyzing the tracking scene and selecting the most suitable visual modalities in the right proportion.

The following sections of this report will discuss the tracking system in detail. We will discuss our experimental setup and results obtained. This will be followed by the proposed intelligent fusion system where we shall discuss the architecture of the system and how it shall be integrated into the already existing human tracking system. Finally, we will discuss some initial tests conducted and how the intelligent fusion module will boost the performance of the system.

2 Prior Art

The literature concerning single person or multiple people tracking in video surveillance, mobile robotics and related fields, already counts several well-known examples, that we briefly review here. Although implementations of similar vision systems do exist in the research and scientific domain, each system is tuned to its specific application, creating a void for a self adapting general purpose tracker.

Multiple people trackers [1, 2, 3], have the common requirement of using a very little and generic off-line information concerning the person's shape and appearance, while building and refining more precise models (color, edges, background) during the on-line tracking task; this unavoidable limitation is due to the more general context with respect to single-target tracking, for which instead specific models can be built off-line.

Many popular systems for single-target tracking are based on color histogram statistics [4, 5, 6, 7] and employ a pre-defined shape and appearance model throughout the whole task.

In particular, [6] uses a standard particle filter with color histogram likelihood with respect to a reference image of the target, while [5] improves this method by adapting the model on-line to light variations, which however may introduce drift problems in presence of partial occlusions; the same color likelihood is used by the well-known *mean-shift* kernel tracker [7].

The person tracking system [4] employs a complex model of shape and appearance, where color and shape blobs are modeled by multiple Gaussian distributions, with articulated degrees of freedom, thus requiring a complex modeling phase, as well as several parameters specification.

The work presented in [8], uses a template based approach. This method uses about 4,500 templates to match pedestrians in images. The Chamfer distance measure is used for similarity measure.

As mentioned earlier, these systems satisfy their performance specification but only under controlled tracking conditions. Change in tracking conditions can affect the performance of such systems. We propose a multi-modal fusion module under the framework of our human tracking system in order to allow the system to adapt to changing tracking conditions while maintaining the robustness over a wide range of applications.

3 The Human Tracking System

The human tracking system has the primary goal of localizing and tracking humans in real-time within a desired working area. The system constantly monitors the tracking area and automatically detects a human target when he/she enters the tracking area. Once the target is localized the system tracks the target in real-time while the target occupies the tracking area. When a new target enters the tracking area the system identifies and adds him/her to a target list to be tracked. If a target leaves the tracking area, it is erased from the list of targets being tracked.

The system tracks the targets in $3D$ space. For this purpose, it requires multiple cameras sharing a common view on the tracking area. In our setup we use 4 USB cameras mounted on the ceiling, sharing a common viewing region. They are calibrated for both intrinsic and extrinsic parameters with respect to a global origin. The cameras stream images of the tracking area at a rate of $25 - 35$ fps. The target is represented as a $3D$ rectangular box satisfying the dimensions of a human. The tracker holds a state-space representation of the $3D$ model pose, given by a translation (x, y, z) of the box model in the image plane of each camera view in the stereo setup.

The tracker uses a bank of sampling-importance-resampling particle filters [9] working on a $3D$ pose and color modality represented by joint probability color histograms. The camera images undergo an initial step of background segmentation. Each target is associated with a unique particle filter. We choose a particle filter for the tracker over the more conventional Kalman Filtering [10] techniques because the tracker needs to be highly robust in dealing with multi-modal likelihoods due to a high probability of having a cluttered background. The particle filter provides the sequential prediction and update of the respective $3D$ *state* $s = (x, y, z)$.

Particle filters are usually computationally intensive due to the likelihood computation for each hypothesis. We use a bank of particle filters, which mean increased computation with every new target entering the tracking area. In order to achieve real-time tracking we hold a global particle set and distribute it evenly among the bank of particle filters. Hence, if a new target enters the tracking area, the system instantiates a new particle filter but at the same time redistributes the global particle set evenly among the new number of particle filters. This reduces the number of particles for each target. this is possible because, when the number of targets increase in the target area their mobility reduces, the number of particles needed to track a target can be reduced.

Tracking multiple targets requires handling of occlusion between targets in camera views. This is important as when one target occludes another target in any camera view, that particular camera should be excluded during the likelihood computation for the targets which are occluded,

since in the observation the region sampled will contain measurement data only for the target which occludes the other targets. However for successfully obtaining the $3D$ *pose* of each target, it is necessary that the features of this target be visible in at least 2 or more camera views. Our system handles occlusions between targets in real-time using a occlusion query module. It is also used during the target detection phase, which is important because when the system models the target the appearance information should be sampled only from the camera views in which the target is visible. This module will be discussed in greater detail in the sections to come.

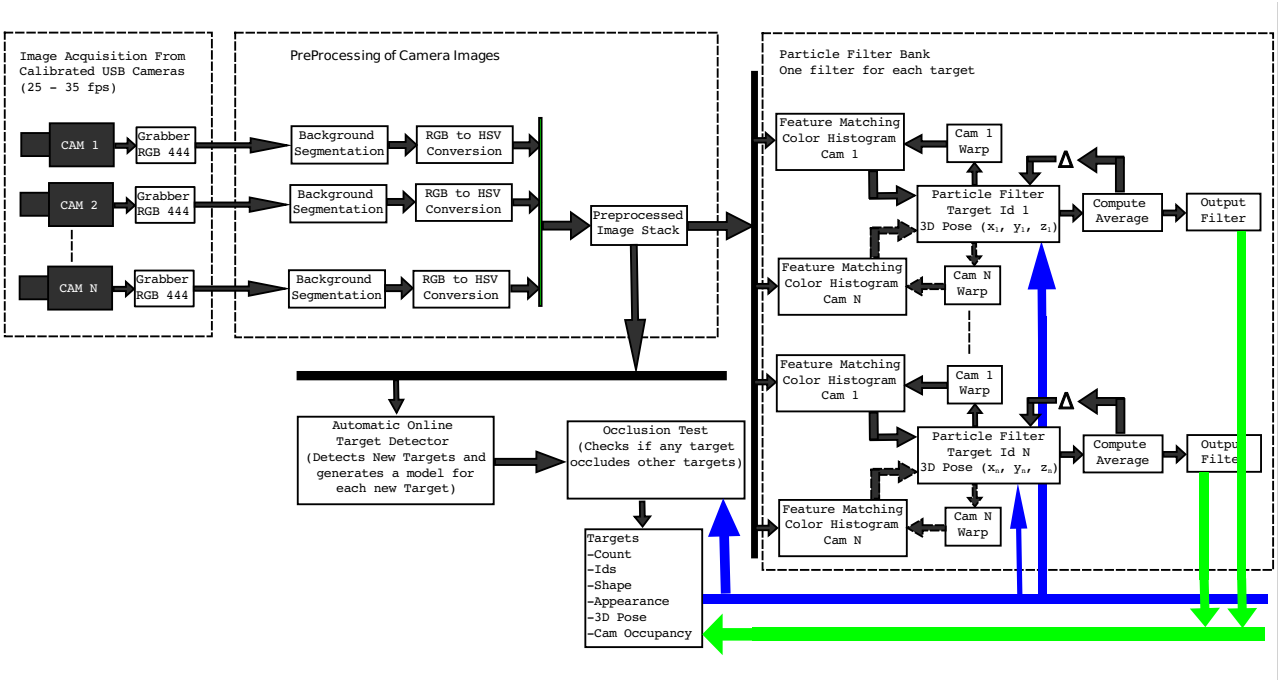


Figure 3.1: The figure illustrates the block diagram of the tracking system. It consists of the image acquisition module, target detection module, occlusion test module and the particle filter bank.

Fig. 3.1 describes the complete pipeline of the tracking system. The system consists of different modules interacting with each other and the hardware components. Each module is discussed in detail in the subsections below.

Image Acquisition

The sensor used are 4 USB cameras capable of streaming images of resolution (752×480) at the rate of $25 - 35 fps$. The cameras are arranged such that they observe a common tracking area with good overlap. Each camera is calibrated for its intrinsic parameters as well as extrinsic parameters with respect to a global origin fixed at a point on the floor of the tracking area. The image obtained from the cameras are in raw RGB 444 format. In our setup all 4 cameras are connected to a single PC with 2 dedicated USB controllers. The cameras operate in streaming

mode where images are written into the memory continuously. When a request arrives from the tracker for an image update, the latest image from the camera buffer is returned to the tracking system.

Pre-processing

The sensor images obtained from the image acquisition system undergo a two stage pre-processing. In the first step a background segmentation is performed on each camera image using a static background model. The background segmented image from each camera is then converted from RGB to HSV z_i^{col} $(i = 1...M)$ for the color-based likelihood, where the index i corresponds to the USB camera index and M equals to the total number of cameras. The pre-processed images are available at both stages since the online target detection module only requires the background segmented image while the tracker requires the pre-processed image resulting after both stages are performed.

Online Target Detection

This module automatically detects targets when they enter the tracking area. This is achieved by performing a scan along the tracking floor area using a $3D$ rectangular cubic target model. At each location in the scan the probability of a possible target detection is computed using the background segmented image. The number of foreground pixels are computed within the $2D$ region obtained by warping the $3D$ *pose* of the target model on the respective camera images. Regions occupied by existing targets are not considered. In this manner, an occupancy map of the tracking area is generated. If foreground occupancy of at least 70% is observed in each camera view then a target is registered with an initial $3D$ *pose* of the particular scan location. A occlusion test if performed at the target location in order to identify the cameras in which the target is completely visible. Using this information the shape and appearance model of the target is generated. The shape consist of a $3D$ rectangular cube with dimension of a normal sized human $(0.2m \times 0.3m \times 1.8m)$. The appearance model consists of $2D$ histograms of the targets in the HSV color space. Only cameras in which the target is visible are selected. If the target is occluded in a camera view, the appearance model in that view is suspended until the target is visible in that camera view. In order to register a target, we require that it be visible in at least 2 camera views. Finally, when a target is registered, it consists of:

- a unique $Target\ ID$
- initial $3D$ *pose*
- shape data
- appearance data
- occupancy information depending on occlusion test

Occlusion Testing

This module determines if a target is occluded by any other targets. This is very important during target detection and tracking since we use $2D$ regions in the camera views obtained by warping the $3D$ *pose* of the hypothesis under consideration. When a target occludes other targets in a camera view, the warped $2D$ regions are overlapped making the appearance data visible only for the target which occludes the other target. In such situations, the information from these $2D$ regions should not be sampled for the targets which are occluded. The occlusion test provides information of such occlusions.

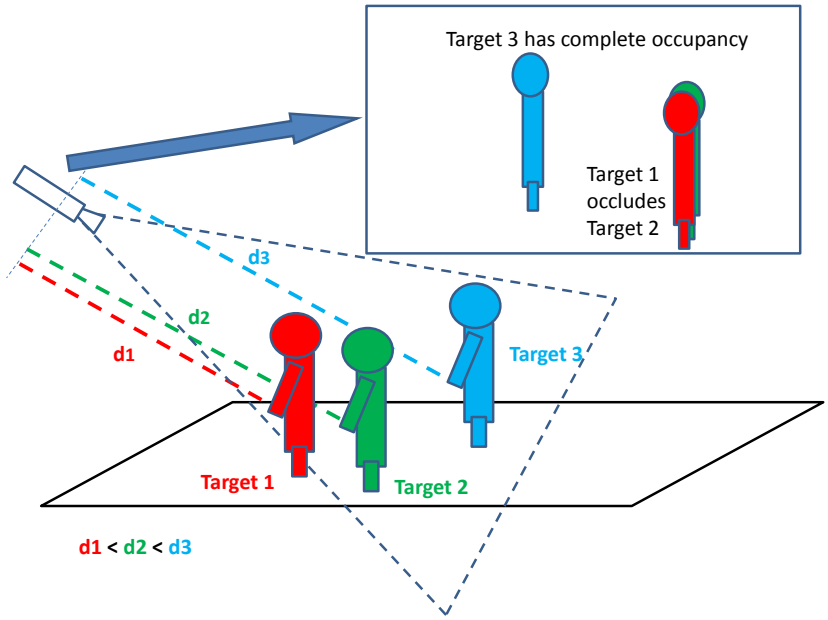


Figure 3.2: The figure illustrates the occlusion test system. The left part of the figure shows a scene in a camera view with 3 targets, where target 1 occludes target 2. The right part of the figure shows how the occlusion is detected by rendering the targets with respect to their distance from the camera The target closes to the camera is rendered first.

Fig. 3.1 illustrates the occlusion test system. This system considers all the targets and computes their occupancies in each camera image. For each camera view the euclidean distance from the camera to each target is computed. The target which is farthest from the camera is rendered first on the camera image. This is followed by the remaining targets, where the closest target is rendered last. Once all targets are rendered an overlap test is conducted. If a target occludes another target it will overlap the rendered region of that target. For a target, if more than 70% of the rendered region remains non-overlapped, then its considered to have a good occupancy in the current view of this particular camera. Thereby, while tracking the filter associated with a particular target considers only the camera views in which the target is not occluded. For a target to be tracked it is required to be visible in at least 2 or more camera views.

Tracker

The tracker has the primary goal of keeping a track of all the targets in real-time, once they have been registered by the target detection system. In order to do this, the tracker uses a bank of Sampling-Importance-Resampling based Particle filters[9]. Each target is associated with its own particle filter. Each filter uses a Brownian motion model and a $3D$ translation state. The visual modality used is $2D$ color histograms and the likelihood estimation is performed by computing a distance measure between the histogram sampled from the current hypothesis and the reference histograms. For each hypothesis, the likelihood is computed for each camera view, in turn computing an average likelihood. Camera views in which targets are not visible are dropped during the likelihood computation for the respective targets.

Particle filters are computationally expensive and hence in order to obtain real-time performance from a bank of particle filters, we maintain a common global particle count which is distributed evenly among the filters. This distribution depends on the number of targets. When a target is added or removed from the target list, the number of particles allocated to each filter is updated. Hence, if N_p is the global particle count and n_p is the number of particles allocated to each filter, then

$$n_p = \frac{N_p}{N} \tag{3.1}$$

where, N is the number of targets. This approach is well suited since as the number of targets increase within the tracking area, their mobility reduces and hence the number of particles needed to track them can be reduced. The following subsections provide detailed explanation of the functioning of the particle filter.

- **Tracker prediction** The particle filter generates several prior state hypotheses s_t^i from the previous distribution $(s^i, w^i)_{t-1}$ through a Brownian motion model.

$$s_t^i = s_{t-1}^i + v_t^i \tag{3.2}$$

  with v a zero-mean Gaussian white noise of pre-defined covariance in the (x, y, z) state variables. Deterministic resampling strategy over the previous weights w_{t-1}^i is also employed.

  For each generated hypothesis, the tracker asks for a computation of the likelihood values $P(z^{col}|s^i)_n$ after projecting every hypothesis on to each camera image.

- **Color likelihood** The object model defining the targets shape is projected onto the HSV image of each camera image at the predicted hypothesis s_t^i using the intrinsic and extrinsic parameters of the respective cameras. The underlying H and S color pixels are collected in the respective 2D histogram $q\left(s_t^i\right)$, that is compared with the reference one q^* through the Bhattacharyya coefficient [6]

$$B_m\left(q_i\left(s\right), q_i^*\right) = \left[1 - \sum_N \sqrt{q_i^*\left(n\right) q_i\left(s, n\right)}\right]^{\frac{1}{2}} \tag{3.3}$$

where the sum is performed over the $(bin \times bin)$ histogram bins (in the current implementation, $bin = 10$). The computation is done for each camera where c represents the camera id $(c = 1...M)$.

The color likelihood is then evaluated under a Gaussian model in the overall residual

$$P(z^{col}|s_t^i) \propto exp(-\sum_M \log\left(B_i^2/\lambda\right)) \tag{3.4}$$

with given covariance λ.

- **Computing the estimated state**

  The average state $\overline{s}_t$

  $$\overline{s}_t = \frac{1}{N}\sum_i w_t^i s_t^i \tag{3.5}$$

  is computed and the three components $(\overline{x}, \overline{y}, \overline{z})$ are returned. In order to reduce the jitter in the output, the average pose is smoothed using an exponential filter.

Graphical User Interface

The tracking system can be effortlessly controlled by the user using an intuitive graphical user interface as shown in fig. 3.3. Although the system can be operated automatically, the GUI provides useful functions such as start, stop, background training, etc. The complete tracking scene from all the camera can be visualized by the GUI. The scene rendering is done using widgets with capability of rendering into OpenGL contexts.

Figure 3.3: Graphical user interface for controlling the system and visualizing the tracking results

4 Experiments in Real-World Scenarios

The system was tested and evaluated on a real-world application. We setup an application for multi-target visual servoing in operational space using an industrial robot arm. It is a distributed system consisting of the visual tracking system, the robot controller and real-time $3D$ visualization of the complete scene. A fire-wire camera with a wide angle lens is mounted on the robot arm. The objective is to control the robot arm using the tracking results such that the robot camera observes all the targets in the tracking area with a predefined perspective while they are in motion. It is also possible to servo selected targets.

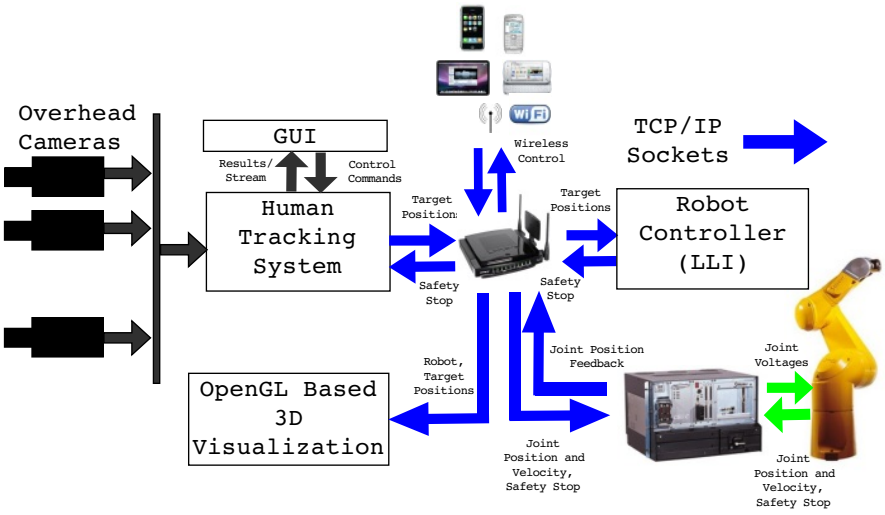


Figure 4.1: The figure provides the overview of the tracking system used in a real-world application for visual servoing of multiple targets in operational space using an industrial robot arm.

Fig. 4.1 illustrates the multi-target visual servoing system in operational space using an industrial robot arm. Each module is discussed in detail hereunder.

Tracking System

It tracks humans in the tracking area and returns the $3D$ position of each target in operational space. The tracking is real-time between $15 - 25$ fps, which is sufficient as the robot controller request for a the target positions every 100 ms. The tracker sends information of all the targets to be servoyed by the robot. If selected targets need to be servoyed, then information regarding

only those targets are sent. The tracker is connected to it GUI and the robot controller through TCP/IP sockets over a Ethernet link. The GUI is used to control the tracker parameters and also to visualize the tracking results. The software runs on Desktop PC with an Intel Core i7 processor running 64 *bit Linux OS*.

Robot System

The robot system consists of a Stäubli TX-90 industrial arm with a CS-8 controller. We use an open architecture of the Stäubli robot with a low level interface which allows us to set joint positions/ trajectories directly without using the kinematics on board the CS-8 controller. The robot controller generates position data for the robot every 4 ms although it requests the tracker for data every 100 ms. The position data is generated using the target position in operational space obtained from the tracking system. The position is set such that the robot camera observes all the targets with a desired perspective. For this purpose, we use information from all the targets to compute their average $3D$ position and generate the joint position for the 6 joints of the robot along with the desired zoom to keep all targets in the filed of view of the robot camera.

The robot controller has a safety stop in case any target is closer than the safety limit of 2 $meters$. When such a situation occurs, the robot controller sends a safety stop signal to the CS-8 controller and the tracking system in order to shut down the systems.

The robot controller runs on a desktop PC with a Intel Dual Core processor running *Linux OS* with real time extension. It communicates with the tracking system and the CS-8 controller using TCP/IP sockets over Ethernet. It also communicates with the $3D$ visualization system by sending position information of the robot and targets to the virtual world.

3D Visualization

The complete tracking environment is modeled in $3D$. Real-time visualization of the tracking scene is done by rendering the virtual world with respect to the robot and target positions. The visualization module is implemented in OpenGL using Coin3D. It is connected to the robot controller through TCP/IP sockets and receives position data of the robot and all targets which are in turn updated in the $3D$ virtual world.

Network and Wireless Control

As described above, each system communicates using TC/IP socket over Ethernet. For this purpose, we setup a local network consisting of a router to which the individual systems are connected. Each system run on an individual machine with a unique IP address and uses a uniques port. In this way all individual system modules can communicate with each other. The router used is a wireless router enabled with WiFi. This allows the complete system to be controlled by wireless devices such as tablet PCs, iPhones, smart phones etc.

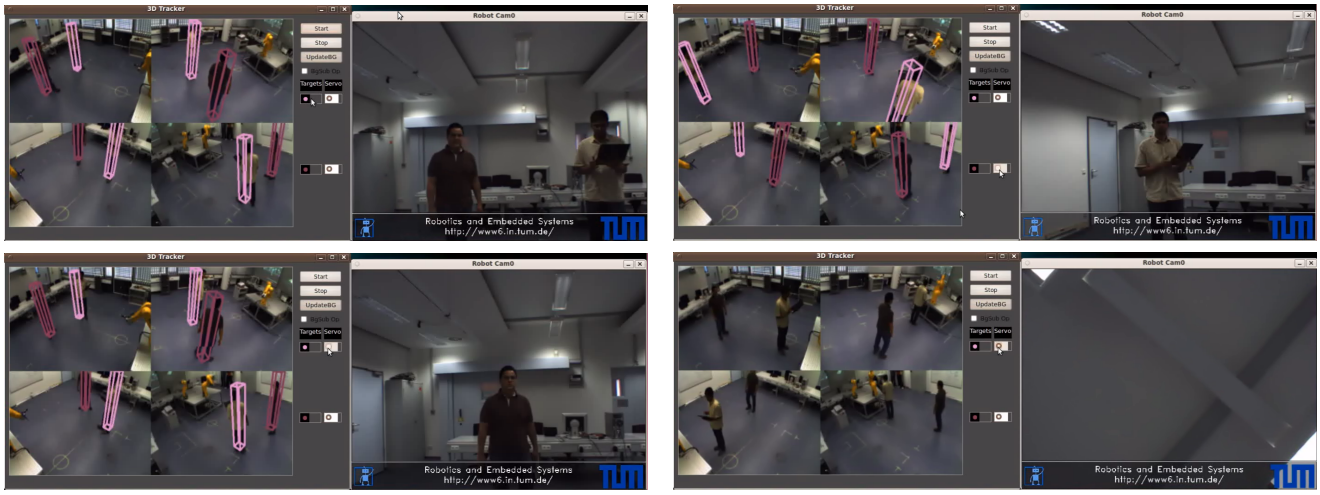


Figure 4.2: The figure illustrates the test results. The 4 clustered images represent the tracking system results along with an additional ronbot mounted camera output. **Top Left:** Two targets are tracked and servoyed by the robot. **Top Right:** Only target 1 is enabled to be servoyed by the robot. **Bottom Left:** Only target 2 is enabled to be servoyed by the robot. **Bottom Right:** Target 1 gets closer than the safety limit of the robot and robot goes to park position and all systems are shutdown.

Fig. 4.2 demonstrates the results obtained. The tracker tracks 2 targets simultaneously in real-time and the robot arm servos both targets. Later target 2 is disabled such that the robot arms servoyes only target 1, followed by target 1 being disabled and target 2 being enabled. Later both targets are enabled and it is observed that target 1 gets closer than the safety limit of the robot which is detected by the tracking system and a signal is sent to the robot controller so that it goes to a safe parking position and thereafter all systems are shutdown.

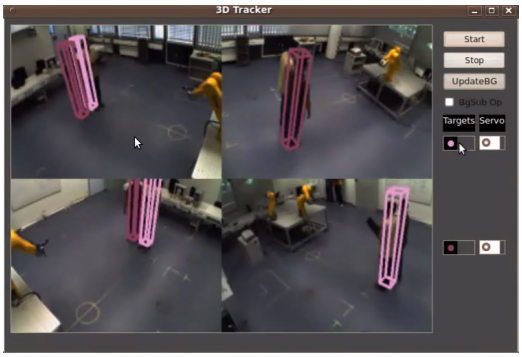


Figure 4.3: The figure illustrates how the system detects occlusion between targets.

Fig. 4.3 shows how the system handles occlusion of targets by other targets. It can be seen that in camera 1 (top right), target 1 is occluded by target 2. Hence, during the likelihood computation for the filter associated to target 1, camera 1 is not considered. Similarly, in camera 3 (bottom right), target 2 is occluded by target 1 and therefore camera 3 is not considered in

the likelihood computation for the filter associated to target 2.

5 Proposed Intelligent Multi-Modal Fusion using Machine Learning

The vision based tracking system performs within its specifications as illustrated earlier. The system has been successfully integrated into a real-world visual servoing application involving industrial robot systems. The conditions under which our system was tested were fairly static although not highly controlled for lighting conditions and other factors affecting the performance of vision systems.

Although our system performs with the desired robustness, we intend to extend its capabilities to be suited for all kinds of tracking environment. The system should be generic and usable in a variety of application with minimal tuning requirements. The system should automatically adapt to changing tracking environment without compromising its robustness. Such a system is highly desirable in many industrial as well as social environments.

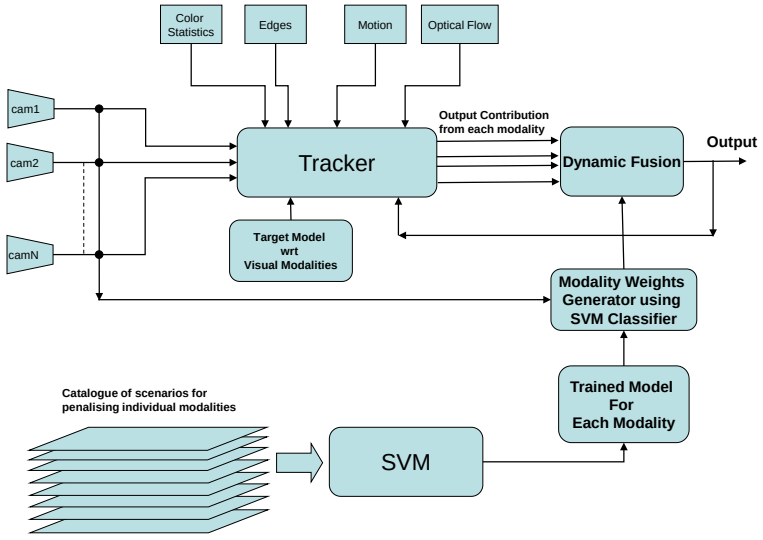


Figure 5.1: The figure illustrates the block diagram of the proposed intelligent multi-modal fusion module.

In order to build a vision based tracking system which automatically adapts to its environment, it is necessary to have a good understanding of the environment. To overcome this challenge, we propose an intelligent multi-modal fusion module which selects the right combination of visual cues in right proportions using a model which is trained to detect changes in the tracking

environments. Using such a model, the system can decide on the fly, which visual modalities are well suited for the current situation in order to maintain the desired robustness.

Fig. 5.1 illustrates the proposed intelligent fusion module. Each sub module is discussed in detail in the following subsections.

Visual Modalities Considered

As discussed before, in order to keep a robust track of targets it is essential to use a combination of visual modalities instead of relying on a single modality. The modalities we intend to use in our system are:

- Color Histograms
- Edges
- Motion History
- Optical Flow

These modalities provide important visual cues during tracking although information obtained from each modality varies depending on the scenario. Each modality has its own strength and weakness depending on changing tracking condition. For example the color histogram modality performs well under controlled lighting conditions, but suffers when lighting conditions change. Hence it is necessary to sample information from modalities which are suited for a certain situation.

Training Model with respect to Visual Modalities

We propose to generate a training model for each visual modality using a large data set of scenarios. Each modality has its own training set. The training set consists of scenarios representing feasibility of using the visual modality. The feasibility range is divided into 5 sections as listed below.

- Very Bad
- Bad
- OK
- Good
- Very Good

This classification is drawn depending on the extent to which the use of the modality is desired and vice versa, under a certain scenario. Scenarios in which the modality suffers to perform are weighted lower than the scenarios which are well suited for the particular modality. Preferable, the tagging of the training set with respect to the above mentioned groups is performed manually. The tagged training set is fed into a multi-class Support Vector Machine [11, 12] in

order to generate a model which can be used to classify scenarios on line into the 5 groups. Depending on the classification results decisions can be made on the usage of the modality.

The quality of the model generated for each modality depends on the diversity and volume of the training samples used. Larger the training set, better is the quality of the model generated. For example, in order to train the color histogram modality, we propose to use a large set of tracking scene images with different lighting conditions ranging from uniform and controlled lighting to completely non-uniform and uncontrolled conditions. Each image is converted to the HSV color space and a histogram of the intensity channel is computed. Each histogram is tagged using the 5 classes, with respect to its shape. This data set is fed as an input to the multi-class support vector machine in order to generate the trained model for this particular modality.

On line Classification and Modality Weight Generator

This module will have the primary goal of classifying the current scene with respect to the training model of each modality. Depending on the classification it will decided if a modality is fit to be used in the current scene. If a modality qualifies to be used, a suitable weight is selected for that modality with respect to the class identified for it by the classifier. If a modality falls under the $VeryBad$ or Bad class, it can be considered to be dropped in the current scene. All the qualified modalities with their respective weights are supplied to the fusion module within the particle filter. In this way an intelligent multi-modal fusion, in order to boost the robustness of the tracker will be possible.

We propose to use this on line classification and fusion module after a fixed number of frames since its not necessary to monitor ever frame as the tracking conditions generally tend to change gradually. The interval in which this supervision will be performed can be set according to the conditions in which the system will be used, since a general prior idea of these conditions is available in most cases.

Training Lighting Conditions for Color Histogram Modality

We performed tests of our proposed approach on one of the modalities. We generated a training model for the color histogram modality using a training set consisting of different lighting conditions. The training set of lighting conditions satisfying the 5 classes mentioned before.

Figs. 5.2 and 5.3 shows a broad classification of the lighting conditions based on the intensity channel histogram. In Fig. 5.2 the histogram is evenly distributed with the maximum intensity pixels concentrated around the pixel with value 128 representing the center of the pixel values distribution $(0 - to - 255)$. On the other hand, fig 5.3 represents histograms concentrated at the extreme ends representing very dark and very bright lighting respectively which infers to bad lighting conditions. Using these observations as a reference we generate a more refined training set with the previously stated 5 classes. This training set was used to train a multi-class support vector machine in order generate a model representing the 5 classes of lighting

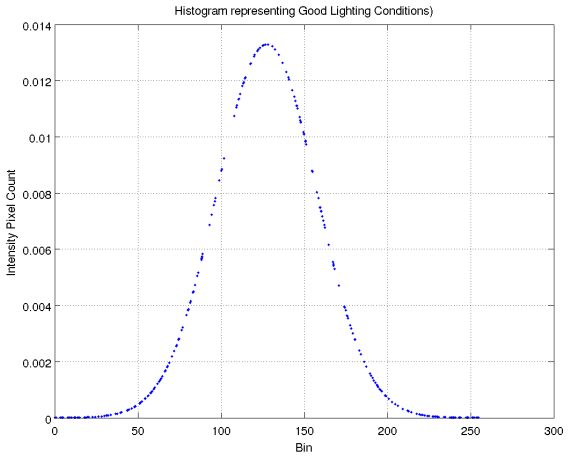


Figure 5.2: The figure illustrates good lighting conditions by observing the histograms of the intensity channel.

conditions. Using this model a on line classification of the lighting conditions during tracking was obtained.

Fig. 5.4 illustrates the test conducted for the color histogram modality by online classification of lighting conditions. For this purpose we trained a *svm* with a relatively small sample set consisting of images. For our initial experiment we used 120 images representing the different lighting conditions. Although this sample set is small we found it sufficient to provide a proof of concept. A trained model was generated using this sample set. The model was used to classify lighting condition. As shown in fig. 5.4 the model is able to classify and associate the current lighting conditions in the camera views to their respective classes. The first image shows a very dark view in each of the camera which is successfully recognized by the classifier as bad lighting. The second image shows an improved set of camera images in terms of lighting conditions, however there exists some specular reflections creating bright stops. The classifier identifies this as OK lighting conditions. The third image shows better lighting with respect to the other images and is classified as Good lighting conditions. The results show the ability of the classifier to successfully classify lighting conditions into their respective classes even with a initial small training set. With a large training set it is possible to achieve better classification covering all 5 classes. The first column in each image represents images from each camera, the second column shows the intensity distribution, the third column illustrates the intensity histogram and the fourth column provides the classification results from the *svm* classifier.

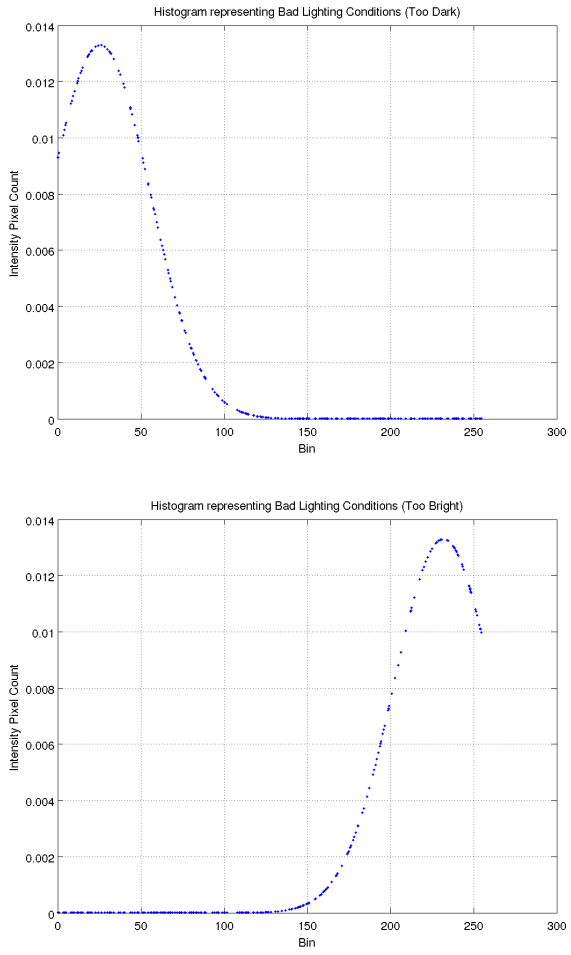


Figure 5.3: The figure illustrates bad lighting conditions by observing the histograms of the intensity channel.

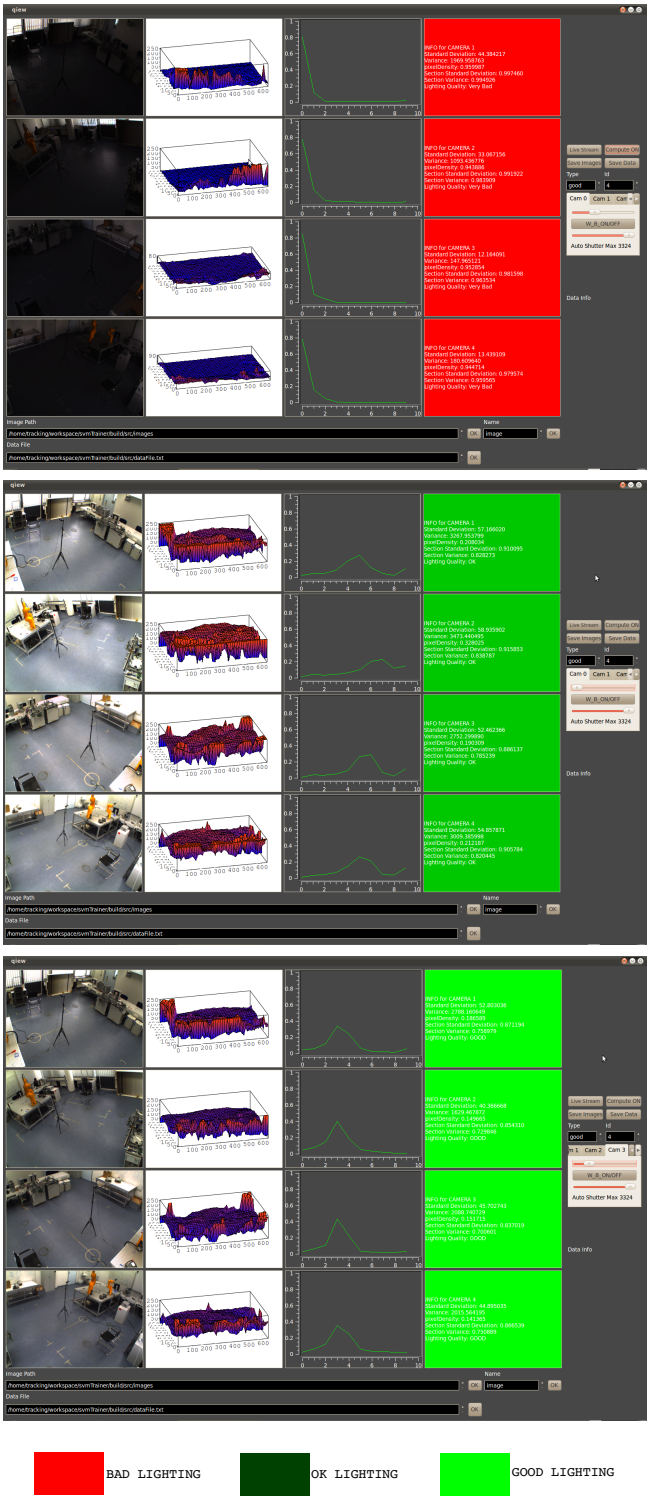


Figure 5.4: The figure illustrates the the test conducted on the color histogram modality. Online classification of the lighting condition is obtained for each camera view using a trained model.

6 Expected Improvement in Performance

With our proposed approach we anticipate a significant improvement in the robustness of the system. With the intelligent multi-modal fusion of visual modalities, we intend to achieve superior performance in a wide range of tracking scenarios and environments. This in turn will allow the system to be used as a general purpose human tracking system for a variety of applications without the need for customization due to the self adaptability of the proposed system.

The system will react to changes in different aspects of the tracking environment and adapt itself by selecting the most suitable modality in the right proportions to maintain the desired robustness desired from the system.

To summarize, we highlight the following improvements

- Acclimatization to changing tracking scenarios
- Flexibility towards wide application range
- Robustness to abrupt changes in tracking conditions
- General purpose system with minimal tuning

A system which intends to be used in real-world applications, should satisfy the above mentioned points. This compliments our proposal toward researching and implementing an intelligent multi-modal fusion module.

7 Conclusions and Future Work

We have introduced a vision based multi-target human tracking system in detail. From the experiments performed in a real-world application setup, we have validated the performance of the system under controlled tracking conditions. From the results obtained, we can conclude that under these conditions, our system performs with the desired robustness.

In order to extend the usability of our system to a wide range of real-world applications under controlled and non-controlled conditions, we proposed a intelligent multi-modal fusion system. With this proposal we intend to provide a general purpose tracking system which is adaptive, flexible and easy to use. From the positive results obtained by the test conducted on the color histogram modality we intend to extend a similar procedure for all the other visual modalities under consideration. This will be followed by implementation of the on line classification engine which will classify the scenes and select the desired modalities and associate the right weights to them. This module will be interfaced with the fusion block in the particle filter.

Bibliography

[1] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: A real time system for detecting and tracking people," in *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 1998, p. 962.

[2] N. T. Siebel and S. J. Maybank, "Fusion of multiple tracking algorithms for robust people tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*. London, UK: Springer-Verlag, 2002, pp. 373–387.

[3] M. Isard and J. MacCormick, "Bramble: A bayesian multiple-blob tracker," in *ICCV*, 2001, pp. 34–41.

[4] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[5] K. Nummiaro, E. Koller-Meier, and L. J. V. Gool, "An adaptive color-based particle filter," *Image Vision Comput.*, vol. 21, no. 1, pp. 99–110, 2003.

[6] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*. London, UK: Springer-Verlag, 2002, pp. 661–675.

[7] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

[8] D. M. Gavrila, "Pedestrian detection from a moving vehicle," in *Proc. of European Conference on Computer Vision*, Dublin, Ireland, 2000, pp. 37–49.

[9] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision (IJCV)*, vol. 29, no. 1, pp. 5–28, 1998.

[10] G. Welch and G. Bishop, "An introduction to the kalman filter," Tech. Rep., 2004.

[11] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class svms," in *JMLR*, 2001.

[12] T. Joachims, B. Schölkopf, C. Burges, and A. Smola, "Making large-scale svm learning practical. advances in kernel methods - support vector learning," in *MIT Press*, 1999.