



FAKULTÄT FÜR INFORMATIK  
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

MASTER THESIS IN INFORMATIK

# Evaluation of the impact of super resolution on tracking accuracy

Florian Liegsalz





FAKULTÄT FÜR INFORMATIK  
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

MASTER THESIS IN INFORMATIK

# Evaluation of the impact of super resolution on tracking accuracy

## Evaluierung der Auswirkungen von Super Resolution auf die Tracking-Genauigkeit

Processor: Florian Liegsalz

Supervisor: Prof. Gudrun Klinker, Ph. D.

Advisor: Frieder Pankratz

Submission Date: 12.11.2012



I assure the single handed composition of this master's thesis, only supported by declared resources.

Ottobrunn, 12th November 2012

---

## **Abstract**

---

Super resolution is a technique to generate a high resolution image from one or more low resolution pictures. This technique allows the usage of normal cost cameras or even webcams to receive data that would normally need better hardware. One of the purposes of this paper is to determine whether this data is usable for tracking as well as receiving more precise and still accurate sub pixel positions in the enhanced images. To be able to make a test of the impact of super resolution on tracking accuracy the first task to accomplish is to create a simple and fast to use super resolution program. After that an analysis can be done with several tests.

## **Inhalt**

---

Super Resolution ist eine Technik, mittels derer aus einem oder mehreren Bildern ein höher auflösendes Bild errechnet wird. Dies würde die Möglichkeit eröffnen, mit einer günstigen Kamera oder sogar einer Webcam Daten zu erhalten, für die normalerweise eine bessere Hardware nötig wäre. Eine der Aufgaben dieser Arbeit ist festzustellen, ob diese Daten für Tracking, sowie für das Erhalten genauerer und dennoch korrekter subpixel Positionen in den verbesserten Bildern geeignet sind. Um die Auswirkung von Super Resolution auf die Tracking-Genauigkeit zu messen, muss als erstes ein einfach und schnell zu benutzendes Super Resolution Programm erstellt werden. Anschließend kann mittels einer Reihe von Test eine Auswertung erfolgen.

---

## Contents

---

Abstract	I
Contents	II
1 Recognizing “the world” with a computer	1
1.1 The camera sensor	2
1.2 The structure of a computer image	3
1.3 Different coordinate systems	5
1.4 Camera parameters	5
1.5 Marker tracking	6
1.6 Ubitrack	9
1.7 The limitations of marker tracking	13
2 Super resolution – the basic idea	14
2.1 One image algorithms	15
2.2 Multi image algorithms	15
3 Super resolution algorithm of this paper	16
3.1 Step 1: rotation and shift estimation	16
3.2 Step 2: combining the images	20
4 Similarity to matlab tool	21
5 Empirical determination of usability	23

---

## **Abstract**

---

Super resolution is a technique to generate a high resolution image from one or more low resolution pictures. This technique allows the usage of normal cost cameras or even webcams to receive data that would normally need better hardware. One of the purposes of this paper is to determine whether this data is usable for tracking as well as receiving more precise and still accurate sub pixel positions in the enhanced images. To be able to make a test of the impact of super resolution on tracking accuracy the first task to accomplish is to create a simple and fast to use super resolution program. After that an analysis can be done with several tests.

## **Inhalt**

---

Super Resolution ist eine Technik, mittels derer aus einem oder mehreren Bildern ein höher auflösendes Bild errechnet wird. Dies würde die Möglichkeit eröffnen, mit einer günstigen Kamera oder sogar einer Webcam Daten zu erhalten, für die normalerweise eine bessere Hardware nötig wäre. Eine der Aufgaben dieser Arbeit ist festzustellen, ob diese Daten für Tracking, sowie für das Erhalten genauerer und dennoch korrekter subpixel Positionen in den verbesserten Bildern geeignet sind. Um die Auswirkung von Super Resolution auf die Tracking-Genauigkeit zu messen, muss als erstes ein einfach und schnell zu benutzendes Super Resolution Programm erstellt werden. Anschließend kann mittels einer Reihe von Test eine Auswertung erfolgen.

---

## Contents

---

Abstract	I
Contents	II
1 Recognizing “the world” with a computer	1
1.1 The camera sensor	2
1.2 The structure of a computer image	3
1.3 Different coordinate systems	5
1.4 Camera parameters	5
1.5 Marker tracking	7
1.6 Ubitrack	10
1.7 The limitations of marker tracking	13
2 Super resolution – the basic idea	14
2.1 One image algorithms	15
2.2 Multi image algorithms	15
3 Super resolution algorithm of this paper	16
3.1 Step 1: rotation and shift estimation	16
3.2 Step 2: combining the images	20
4 Similarity to matlab tool	21
5 Empirical determination of usability	23

---

5.1 Euclidian distance between marker positions in SR image of generated image sequence with high resolution generated image	24
5.2 Pose distance between two printed markers	25
5.3 Different positions in Image	31
5.4 Determining the better way of taking the images: using a tripod or using a hand	32
5.5 Can SR enhance images shot with a reflex camera	36
5.6 The influence of different lighting conditions on the efficiency of SR	37
6 Conclusion	39
7 Possible ways to improve the results	40
8 Literature	A
9 Figures	C



## 1 Recognizing “the world” with a computer

---

To enable a computer to recognize the environment, it needs a device for recognition. There are a lot of sensors like thermometer, radio sensors and gyroscopes, but one of the most important sensors is the camera. To say “the” camera is a little bit difficult because there are a lot different versions. There are cameras that work with mirrors, cameras that just take one line instead of a whole image and there are standard video cameras that can be found for example in webcams and mobile phones. All of these cameras have in common that light shines somewhere in the camera, some sort of optic bundles the light and a sensor converts the light rays in a discrete signal representable in a computer. Depending on the sensor, the size of the image a camera can take is fixed. But sometimes a bigger image with a higher resolution than the used camera can create is needed. A simple resize of an image leads to a bigger image but the resolution stays the same. Most times these resizing algorithms result in producing so-called artifacts. These are washy areas or big squares that do not belong to the original image. Super resolution (SR) is an approach to increase the size of the image as well as its resolution.

In addition to the device that is needed for recognizing “the world”, algorithms are needed which are able to evaluate the data the sensors gain. One of these algorithms is the so-called marker detection and tracking algorithms. They can recognize predefined objects and calculate their position and orientation. In order to guarantee the best recognition it is essential that the images of the objects are as fine as possible. If an object is far away from the camera or if the sensors don't record high quality data the gain of position and orientation values

might be bad or might not even be possible. SR could be the solution to improve these inferior source data to even get accurate evaluation data.

Knowing the opportunity SR could provide, it is worth to examine and understand this domain closely. However the first thing necessary is to understand the functionality of the sensor in a camera, the representation of images in the computer as well as basic mathematical conditions to fully understand SR.

### 1.1 The camera sensor

---

There are mainly two different types of camera sensors, the so-called CCD<sup>1</sup> and CMOS<sup>2</sup> sensors. These are two different kinds of how to measure the light and how to store the data. The exact functionality can be found in several publications regarding this subject. To put it very briefly, both have in common a two-dimensional array of sensors that can recognize and measure the light shining on them. Every sensor corresponds to one pixel in the later image. To determine which color which pixel holds the sensors are exposed to a bundle of light rays. These rays cannot be separated individually. Therefore every single sensor cannot get one single ray but the light “floods” all sensors. Every sensor then gets its individual charge by accumulating all light he “sees”. These charges will then be read out and saved as the color values of each pixel of the image [1] [2]. Unfortunately these color values contain noise. This noise can have several sources. One of these sources is that the sensors, because of being “flood” with light, will always get slightly different values. Using the CCD sensor can even lead to a blooming effect. That means that one sensor gets

---

<sup>1</sup> CCD = Charged-coupled Device

<sup>2</sup> CMOS = Complementary Metal Oxide Semiconductor

charged to the maximum and further charges of the same sensor spread to the surrounding sensors. All these inaccuracies lead to a not exact duplicate of the world. The image is just a copy of the world with narrow capacities.

## 1.2 The structure of a computer image

Having shot an image with a camera or having created a virtual image, this needs to be represented and saved on a computer. A computer can only store discrete values. As a result of this limitation there are always gaps between two values for which no data is available. As mentioned before, an image is stored in pixels. A pixel is the smallest entity in an image. But a pixel can be comprised of up to four channels which determine the absolute color value of the pixel. There are a lot of ways to describe a color which gives every channel another meaning. The following table shows the most important techniques [3]:

Technique	channels	Short description
RGB	3	The channels represent the color components <b>R</b> ed, <b>G</b> reen and <b>B</b> lue. Each channel stores one value between 0 and 255. The three channels are blended additive. It is also possible to use values between 0 and 1.
ARGB	4	This is nearly the same technique as RGB but an <b>A</b> lpha component is added. The alpha channel stores the visibility of the pixel. It can also store values between 0 and 255. The minimal values mean “totally transparent” and the maximum “no transparency”.

HSV	3	The color value is represented by its <b>Hue</b> , <b>Saturation</b> and <b>Value</b> . The values of hue are defined within a range between 0 and 1 where 1 corresponds to 360°. The hue values are representing a color circle with 0° corresponding to red, 120° corresponding to green and 240° corresponding to blue. The saturation and value are stored between 0 and 1 with 1 being the pure color or full brightness.
Grayscale	1	The image consists of one single channel which represents a number between 0 and 255. 0 means black and 255 white.

Whatever technique is used, the problem remains the same. By just storing discrete values a continuous signal can just be approximated but not reproduced. That means if a photo is shot, color gradients

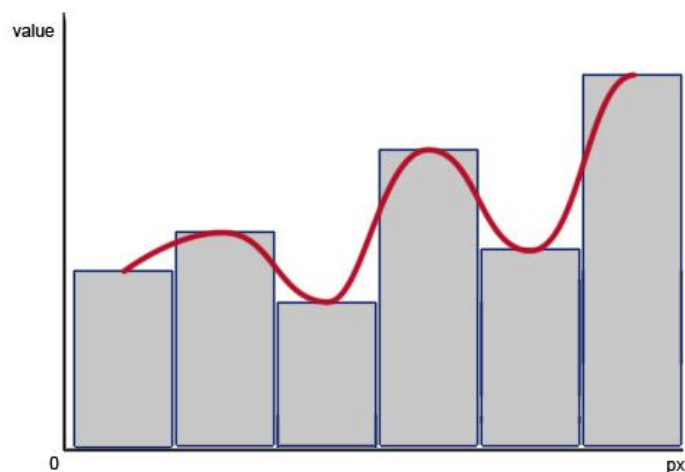


Figure 1 - "stairway" effect

in this image can just be approximated. This can lead to a sort of a "stairway" effect in the image. Figure 1 shows a patch of six pixels displayed in a diagram where the x-axis corresponds to the pixel position and the y-axis corresponds to the pixel value. The red line shows the actual color values in the real world. It

can be seen that the continuous signal can just be approximated. This will always lead to inaccuracies.

### 1.3 Different coordinate systems

---

It must be differentiated between several spaces. These are for example the image, camera and world space. These are all different coordinate systems. The image space is always 2D with the center point lying in the lower left corner of the image. The camera and world space are 3D coordinate systems with different center points and orientations. The camera space is defined by the camera position with the camera as origin and the orientation defined by the camera orientation. The origin of the world space is defined by the system or the needs of the user. Every 3D space can be converted to every other 3D space if the relationship between the spaces is known.

### 1.4 Camera parameters

---

As explained before, a camera can take an image of the world. There are cases it needs to be known where a specific point in the world will be projected in the picture or the other way round. Therefore the projection from 3D to 2D space needs to be calculated. This projection can be described by its so-called projective matrix  $M$ . This is assembled by the intrinsic matrix  $A$  and the extrinsic matrix  $[R|T]$ .

$$\mathbf{M} = \mathbf{A}[\mathbf{R|T}]$$

$$\text{with } \mathbf{A} = \begin{bmatrix} f_x & s & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix}$$

The intrinsic matrix contains the values specifying a specific camera, the focal length parameter  $f_x$  and  $f_y$ , the sheer value  $s$  and the camera center point  $u_x$  and  $u_y$ . The

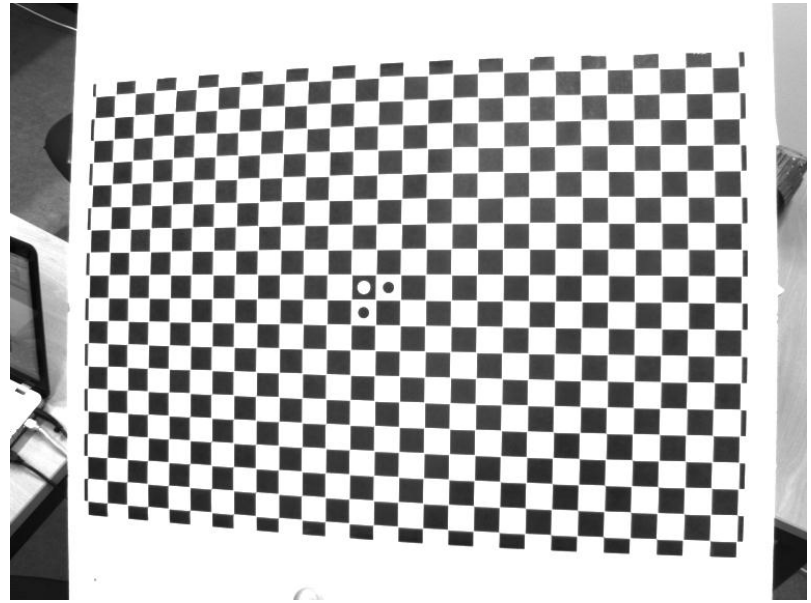


Figure 2 – example of a chessboard calibration pattern

extrinsic parameters denote the coordinate system transformation from 3D world coordinates to 3D camera coordinates [4]. To reveal the intrinsic values a camera needs to be calibrated. Actually not the camera itself is changed in any way but the user “learns” the values according to the camera. Camera calibration is normally done with a well-known chessboard pattern (see e.g. Figure 2) by which the computer can recalculate the distortion of the camera as well as its other parameters like focal length, camera center und sheer factor.

The actual procedure requires several steps. First the user takes multiple pictures of the well-known chessboard from different angles and distances. It is important that photos are shot so that the chessboard is in the corners of the images. It is exactly in the image borders that the camera lenses lead to a distortion of the image that needs to be calculated. The next step is to mark all corner points in the images, the actual outer border corners as well as every point of intersection within the chess board. Having gained these data the camera pose can be calculated relative to the chessboard as well as the image distortion resulting through the camera lens. Doing this with all images the

general camera parameters can be computed. These can be used to calculate where objects in the world will be displayed in the image.

### 1.5 Marker tracking

Generally speaking tracking means all steps needed to pursue objects over a period of time. Mostly these objects are moving. Marker tracking specifies this common definition on specific icons that are predefined and easily recognizable with a computer. There are two branches of marker tracking, the classic marker tracking and the marker less tracking. Both have

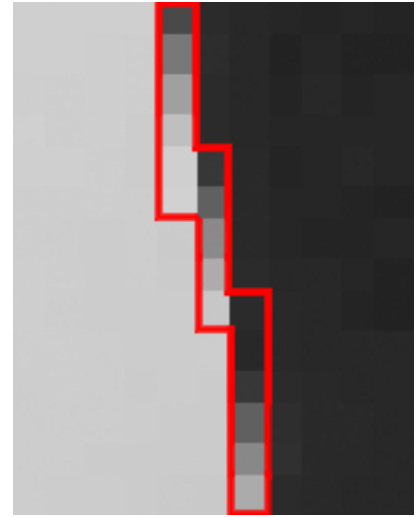


Figure 3 - Zoom: marker border

in common that a predefined object needs to be found in a picture and later the position and orientation of this object in camera space is tried to be evaluated. In marker less tracking these objects can be everything, a cover of a book, an image on a wall or the poster of an event agency. The tracker then analyzes every picture it gets for similarities to the known sources and if it finds correspondences it can return the desired position and orientation, the so-called pose, of the spot found in the picture.

This paper uses tracking with markers. To understand the purpose of this paper it is essential to understand the basic steps done by a marker tracker. These steps start with the taken image and lead to the resulting marker pose in camera space. There are a lot of different approaches but they all have some similarities. Normally the first step is creating a black and white copy of the source image. If the source image is a colored image, it should first be

converted into a gray value image. Within the black and white picture all shapes with exactly four corners can be easily detected. Every single one of these shapes is a potential marker. Unfortunately the coordinates of the corner points of these shapes are just on pixel accuracy which is not fine enough for really precise marker poses. Normally lines as well as edges in images are not clearly defined. The center point of corners and edges can lie between two pixels. This will result in a blurred color between the surrounding pixels. Figure 3 demonstrates that the marker border is not clearly recognizable but somewhere inside the read area. That is the reason why the so far detected corner coordinates need to be refined. This happens by analyzing a certain number of spots on the lines between the found corners in the original image. Every single one of these spots is analyzed in small strips, like for example three on seven pixels, where the stripes are rotated perpendicular to the borderline. Figure 4 shows the found markers in the image and illustrates these with yellow boxes. The little ochre lines on each border of the marker are the spots that are analyzed to refine the border position. At these small patches the derivative of the color values are build. This can be done by applying a so-called Sobel filter. This is a three by three matrix that converts the image to the image derivative.

After converting for example the three on seven pixel patches they have a size of one on five. In this line the maximum value can be found. This maximum conforms to the greatest slope on the patch therefore must be the boarder around this pixel. To determine the exact

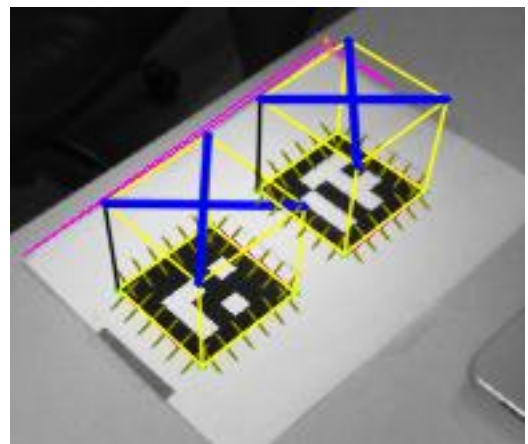


Figure 4 - debug image marker tracker



spot a cubic approximation of the maximum point and its two surrounding pixels is calculated. The turning point in the approximated function then defines the point of the edge in sub pixel accuracy. Having computed all edge points on all borders these values can be used to approximate the border lines. The intersection points of the borders then result in the actual marker corners. Knowing these and the camera parameters, its equivalents in camera space can be computed. In addition the inner configuration of the marker should be analyzed to determine whether the found square is a desired marker or just any square. This analysis needs specific algorithms for every type of marker. Some markers have special images or signs as inner context which would need image comparison algorithms to determine similarities. Other markers like the ones in Figure 4 are defined by bit code values similar to barcodes. These bit code structures can be easily checked by copying the black and white version of the found square to a small image or matrix of the size of the expected marker. Every pixel or field in the matrix would then represent one bit of the marker. This would allow simply reading the value of the square. If this value is the value of the desired marker it is found. Otherwise it is something else. It could be another marker or even just a four-sided square like the pink framed area in Figure 4. By copying the found square including the black border surrounding the inner context of the marker it can even be checked whether the area has a continuous frame which a marker must have. This would be an additional checkup to verify that the found object really is a marker [5].

## 1.6 Ubitrack

The computer aided medical procedures and augmented reality chair of the Technische Universität München has developed a framework for tracking, calibration and sensor fusion, the so-called Ubitrack. For demonstration purposes several vision components have been added including a very high sophisticated

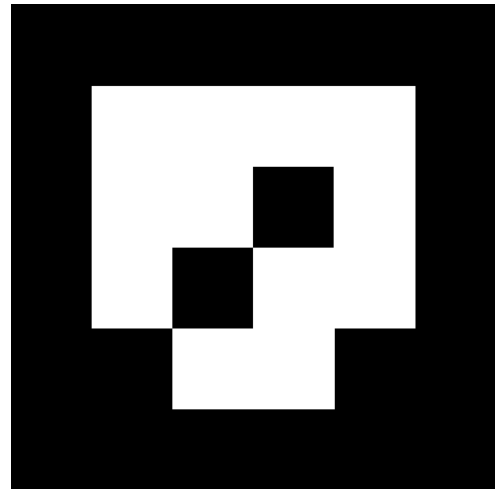


Figure 5 - example Ubitrack marker

tracking algorithm. It does not only use the outer square of the markers for getting the sub pixel accuracy, but - depending on the inner marker structure - it can also use its inner lines and corners. Ubitrack uses OpenCV as internal base for accessing images and camera streams. OpenCV is a framework that provides functions and libraries for working with images and videos [6]. It is a very useful cross platform collection used by about 47,000 people (October 2012). Ubitrack is platform independent and licensed under the LGPL license.

Using Ubitrack as a base platform made it unnecessary to implement a lot of basic methods needed for marker tracking. In addition it was guaranteed that these basics are working correctly. All fundamentals like accessing the camera, undistorting used images, evaluating results and saving images to the harddisk can easily be done without writing one line of code.

Ubitrack has a graphical component in order to create so-called “spatial relationship graphs” (SRGs) which are the programmatical description of complex mathematical relationships. This component is called “Trackman” and provides a graphical interface to combine all available functions just by drag and drop. As the name “spatial relationship graph” suggests it is a graph structure consisting of nodes and edges. In this case the nodes represent objects and the edges are measurements. Figure 6 shows a screenshot of trackman with an SRG of a small program that uses a camera to shoot an image on pressing a button as well as computing the SR image corresponding to the one image. Afterwards both images are saved.

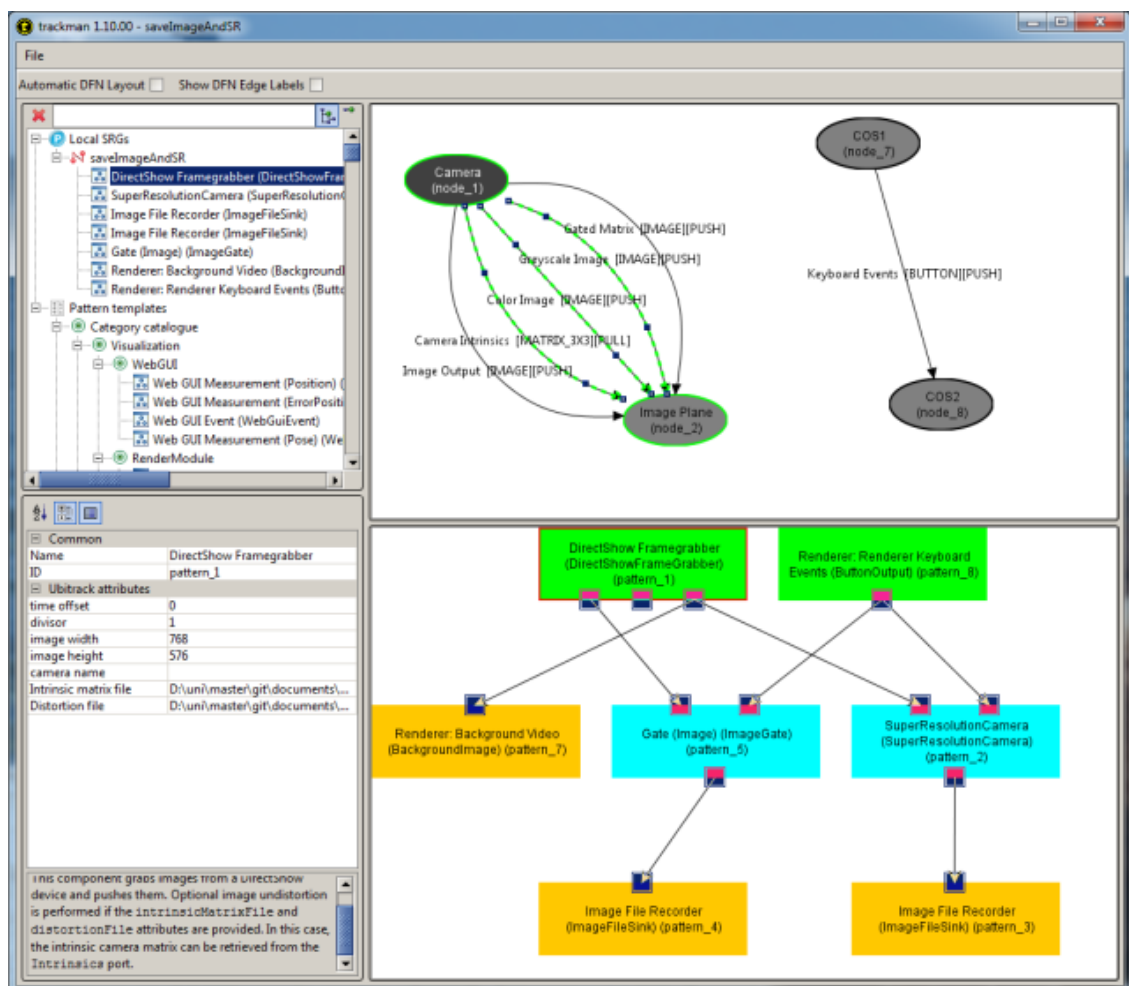


Figure 6 - Trackman: SRG of saving images as well as its SR equivalent

The upper right part of the program contains the above mentioned tree structure. For example between the camera and the image plane there are several edges containing e.g. the intrinsic parameter determining the camera and the image delivered by the camera. The second branch represents the button event that needs to be raised for taking an image. The lower part is an alternative structure of the upper tree. In the upper tree several Ubitrack features are logically combined. For example the camera and the super resolution camera are both represented by the same node because both represent a camera in the real world. The lower tree shows that the component “SuperResolutionCamera” is fed with the data of the actual camera - the “DirectShow Framegrabber” - and delivers its computed image to an “Image File Recorder” which saves the received files to the harddisk. On the lower left side there is a setting browser which allows setting all required preferences like filenames. On the upper left side all available components for creating an Ubitrack program are selectable and can be added to the SRG by using drag and drop. By clicking with the right mouse button in the upper right side the actual program can be started. In addition it is possible to start the SRG after exporting, using the command line tool of the operation system as well as combining Ubitrack solutions with totally independent software.

Another advantage of Ubitrack is the fact that all measurements and results are computed in meters. Therefore all evaluations are simplified and easily understandable. Converting the SR algorithm of this paper from a standalone C program to an Ubitrack component a very simple task with minimal effort was due to the intensive pattern structure. In addition a second component was built to calculate the reprojection error (see 5.4)

## 1.7 The limitations of marker tracking

Marker trackers work on images and as already mentioned before these are just discrete approximations of the real world. Continuous signals can only be displayed in a sampled way. Straight lines, that are not exactly oriented along the axes of an image, always have a sort of a stairway effect (see Figure 3). In addition as mentioned before the accuracy of a marker tracker massively depends on the precision of the detected edges. The more precise the values representing the edges are, the more precise the evaluated marker pose will be. For that reason it would be good, if not just the tracking algorithm were on a high standard but if there was an additional way to increase the precision of the corner positions. A possible solution would be to use a camera with a higher resolution, but that would only delay the problem to a later time. Surely it would solve the problems for specific domains but a more general solution would be handy. In some cases the hardware cannot be changed so that a software solution is needed. A possible solution could be resizing the image. If the image is bigger, the marker is bigger and the algorithms should work better. The problem is that a simple resize of an image does not change the resolution of the image. An algorithm is needed that does not only resize the image but that also enhances the resolution and so enhances the border precision. The desired set of algorithms could be the techniques around super resolution. Under the condition that SR is used and e.g. the resolution of an



Figure 7 - edge refinement

image is doubled, the sample rate of the image on the underlying continuous real world would double as well. Taking this into account the polygonal approximation on the edges in the marker tracker could use more precise data and the calculated turning point could be even more precise. The upper image in Figure 7 is a small patch of an unedited source image. The border has a width of four pixels with two different color values. The lower image is a converted image with doubled size as well as refined resolution. The border now has a width of six pixels with six different color values. The increased amount of data available can clearly be recognized. An enhancement of the accuracy should be detectable.

## **2 Super resolution – the basic idea**

---

The goal of SR is to generate an image that is larger than the original source. But if someone tries to stretch an image for example to its doubled size, gaps arise which have no source data. There are a lot of methods to fill these data. The simplest method is to use the nearest data point. This method is called “nearest neighbor” algorithm. There are a lot of other techniques that are all using some sort of interpolation. But all these algorithms can only calculate values from the data already existing in the sources. No additional information can be added. SR on the contrary does not only want to resize the image, it wants to fill the gaps with additional data, too so that the resolution of the image increases. SR has a large range of techniques that can work one or more source images. There are several different approaches with sometimes totally different approaches. Basically it can be differentiated between algorithms that

work only on one image and others that work on a stream of nearly similar pictures, like in a video stream.

## 2.1 One image algorithms

---

To generate a higher resolution image with just one source image, this image is split into small patches like four on four pixels. These patches will then be resized individually and merged again. The big question is however where the information can come from what the resized patches should look like. Because none additional information than the one in the image can be accessed, some algorithms have been developed to compensate this problem. Very briefly described it can be said that one way to determine the larger version of the patches is to compare the small patches with samples of a precompiled database which contains low and high resolution pairs of patches [7]. The problem with this SR approach is that information is used which is not given within the image but from other sources. This can lead to some sort of “hallucinating” pixels in the generated image. Using SR for tracking it must be guaranteed that not false information is used because it could falsify the result of the tracking algorithms.

## 2.2 Multi image algorithms

---

If false data may not be used, solely information of the actual picture must be taken. However with just one picture no improvement can be achieved. That is why there are algorithms which use a stream of pictures with minimal delay and movement between the different recordings. For creating a SR image of an image stream, a possible algorithm could use the first image of the image stream as a sort of source image which could then be enhanced with the data of

the other pictures. Even if a tripod for the camera is used and two pictures are shot right one after another, both images will never be the same because of noise in the sensor of the camera or other factors like e.g. illumination conditions. When the camera is hold by a person the shaking of the palm will lead to a minimal shift of the objects in the picture. With these minimally different images the missing data for enhancing the source image can be computed [8].

### **3 Super resolution algorithm of this paper**

---

In the internet there are several theses of how to compute really good SR images [9] [10]. Most of these papers include complex math. Because of these complex calculations a lot of these algorithms cannot be run in real-time. There is a matlab<sup>3</sup> distribution of SR algorithms available, which will serve as a source for the algorithms of this paper [11] [12]. All of the used algorithms in the distribution work in a two-step approach. The first step is to determine the shift and rotation values between the taken images and the second step is to combine the images to a high resolution image.

#### **3.1 Step 1: rotation and shift estimation**

---

To determine which of the provided estimation algorithms is the most accurate one, some simple test with generated images have been made which were rotation and/or shifted by known values. The algorithm which detected these known parameters most exactly is called “A Frequency Domain Approach to Registration of Aliased Images with Application to Super-Resolution” [13]. It uses a frequency domain algorithm to estimate the desired parameters. The

---

<sup>3</sup> Matlab is a software for numeric calculation, visualization and programming [21]



algorithm allows to determine horizontal and vertical shifts  $(\Delta x_1, \Delta x_2)$  as well as planar rotation  $(\varphi)$ .

The determination is based on the declarations of a reference signal  $f_1(\mathbf{x})$  and its shifted and rotated version  $f_2(\mathbf{x})$ :

$$f_2(\mathbf{x}) = f_1(\mathbf{R}(\mathbf{x} + \Delta\mathbf{x})),$$

$$\text{with } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \Delta\mathbf{x} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \mathbf{R} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix}$$

Using a Fourier domain, this can be expressed as

$$\begin{aligned} F_2(\mathbf{u}) &= \iint_{\mathbf{x}} f_2(\mathbf{x}) e^{-j2\pi\mathbf{u}^T \mathbf{x}} d\mathbf{x} \\ &= \iint_{\mathbf{x}} f_1(\mathbf{R}(\mathbf{x} + \Delta\mathbf{x})) e^{-j2\pi\mathbf{u}^T \mathbf{x}} d\mathbf{x} \\ &= e^{j2\pi\mathbf{u}^T \Delta\mathbf{x}} \iint_{\mathbf{x}'} f_1(\mathbf{R}\mathbf{x}') e^{-j2\pi\mathbf{u}^T \mathbf{x}'} d\mathbf{x}' \end{aligned}$$

With  $F_2(\mathbf{u})$  being the Fourier transform of  $f_2(\mathbf{x})$  and the coordinate transformation  $\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}$ . Setting  $\mathbf{x}'' = \mathbf{R}\mathbf{x}'$  the relation between the amplitudes of the Fourier transform can be computed as

$$\begin{aligned} |F_2(\mathbf{u})| &= \left| e^{j2\pi\mathbf{u}^T \Delta\mathbf{x}} \iint_{\mathbf{x}'} f_1(\mathbf{R}\mathbf{x}') e^{-j2\pi\mathbf{u}^T \mathbf{x}'} d\mathbf{x}' \right| \\ &= \left| \iint_{\mathbf{x}'} f_1(\mathbf{R}\mathbf{x}') e^{-j2\pi\mathbf{u}^T \mathbf{x}'} d\mathbf{x}' \right| \\ &= \left| \iint_{\mathbf{x}''} f_1(\mathbf{x}'') e^{-j2\pi\mathbf{u}^T (\mathbf{R}^T \mathbf{x}'')} d\mathbf{x}'' \right| \\ &= \left| \iint_{\mathbf{x}''} f_1(\mathbf{x}'') e^{-j2\pi(\mathbf{R}\mathbf{u})^T \mathbf{x}''} d\mathbf{x}'' \right| \\ &= |F_1(\mathbf{R}\mathbf{u})| \end{aligned}$$

$|F_2(\mathbf{u})|$  can be seen as a rotated version of  $|F_1(\mathbf{u})|$  over the same angle  $\varphi$  as the spatial domain rotation.  $|F_1(\mathbf{u})|$  and  $|F_2(\mathbf{u})|$  do not depend on the shift values  $\Delta\mathbf{x}$  because the spatial domain shifts only affect the phase values of the Fourier

transform. Therefore the rotation and shift estimation can be computed completely independently [13]. The determination of the special shift is also explained in the so far explained paper, but to simplify the determination of the shift values the function `phaseCorrelation` of the OpenCV framework is used. The functionality of this function is explained on the OpenCV Website [6].

Sometimes the introduced algorithm does not generate the desired result. There are cases where these frequency domain based algorithms compute totally wrong shift and rotation values. Therefore alternatives are needed that can be used if the results do not match the actual input. Other computer vision tasks like e.g. image stitching need shift values between images, too. This is the reason the invention of algorithms which are quite approved. Some of these methods use feature based approaches. A feature based image shift analysis first detects unique spots in every image, the so-called feature points. There are several different approaches to generate unique scale-, rotation- and translation-invariant feature points. The algorithm for this paper uses the so-called "Good features to track" detector [14]. Afterwards the detected feature points are compared to find similar sites in the compared pictures. Because of outliers, correspondences detected by the algorithm which do not belong together, a further step is needed to get rid of these points. One possibility to detect these outliers is to use a RANSAC algorithm [15]. RANSAC first takes at least four point correspondences randomly of the set of all pairs and computes a homography. Afterwards it uses this homography to convert the first set of points, which should now contain the same points as the second set. To determine whether the sample points were not outliers, so-called inliers, the Euclidian distance for every calculated point with its correspondent point is

calculated and compared with a defined threshold. If too many points are discarded the algorithm starts again and uses a different set of random points.

For ensuring that there is always a method to calculate the desired values a third algorithm is added that is implemented based on the function `cvCalcOpticalFlowPyrLK` of the OpenCV framework. Information on this algorithm can be found in the OpenCV documentation [6].

The user of the application of this paper can decide which of the presented algorithm he wants to use. It is strongly recommended to choose the respective algorithm appropriate to the respective image. Images e.g. with a very low texture should not be analyzed using the feature detection algorithm because

the algorithm would have problems to find clear feature points that can then be compared. Usually the easiest way to find the appropriate method would probably be to create the SR image with one of the techniques and if a result is not satisfying to use a different technique. The first image in Figure 8 was created with the Vandewalle algorithm which could not find the correct rotation parameters between the



Figure 8 - result of a wrong shift technique

source images. The second image used the exact same source images as well as the same recombining approach but determined the shift and rotation values using the feature detection algorithm.

### 3.2 Step 2: combining the images

---

The image recreation process is done according to the paper presented by “A. Zomet and partners” [16]. Their approach is an iterative process in which the source image is converged step by step into the desired SR image. In every iteration step a list of difference images between the approximation at the specific moment and all source images is calculated. For every single pixel the median value of the difference images is computed afterwards. This median value is then scaled by 0.05 and added to the approximation. Thus the SR image is created. The loop ends when the normalized difference between two steps of the SR image is smaller than a specific lambda. This paper uses a lambda of 0.0001. Zomet and his colleagues revealed that the median creates better approximations than the mean because extraordinary measurements in the sources will not lead to corruptions of the output image.

The matlab distribution used as source for the programming calculates the rotation and shift values before creating the SR image. It then shifts the approximation in every step according to these values to align it with the source images. This leads to two rotation and shift operations of the image for every source image every iteration step. Using five images and needing 20 iterations this leads to 200 calculations that can be skipped by pre rotating and pre shifting the source images.

#### 4 Similarity to matlab tool

As a starting point for the software of this paper the referenced matlab tool was used. To demonstrate the similarity of the two programs tests on generated images were done.

As a first test a 3D scene was generated in Autodesk's 3ds Max<sup>4</sup>. This scene contains a simple marker with a plain background (see Figure 9). It was used to create a high resolution image with 640x480 pixel as well as five images with

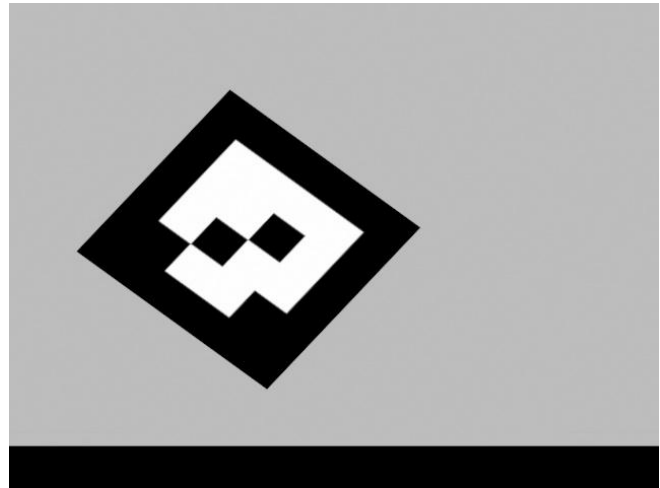


Figure 9 - first generated image

a marginal camera shift and a resolution of 320x240. The small images were used to create several SR images, one using the matlab tool and one for every shift analyzing method used by the program of this paper. In addition two images were created by bicubic interpolation and using the nearest neighbor algorithm.

Every single one of these images was used to calculate a histogram. A histogram is the presentation of all color values in the image in a diagram by color value. In this case the histogram was built with the difference values of the particular image and the 640x480 pixel sized image. To have a better comparable value the average of the absolute values in the histograms were taken. The results are shown in the following table:

<sup>4</sup> 3ds max is one of the leading soft wares for 3d modeling, animation and rendering [22]

Image	Average difference
Matlab SR-image	2,113
Vandewalle algorithm	1,342
Optical flow pyramid LK algorithm	1,394
Feature detection algorithm	1,344
Nearest neighbor	1,692
Bicubic interpolation	1,492

The results show that all algorithms of this paper are giving a fairly good approximation to the high resolution image. They are even more efficient than the implementation in matlab.



Figure 10 - second generated image

To exclude that the results depend on coincidence because

of the used image which is quite straightforward with low texture and mostly straight lines, a second identical test with a completely different image was done. A free 3D model from the internet was used as image source [17]. The image displays a boat in a sunset on a wavy sea (see Figure 10). Again all images described like in the test before were created and the histograms were calculated. The mean difference values are displayed in the table below:

---

<b>Image</b>	<b>Average difference</b>
<b>Matlab SR-image</b>	7,766
<b>Vandewalle algorithm</b>	7,369
<b>Optical flow pyramid LK algorithm</b>	7,360
<b>Feature detection algorithm</b>	7,387
<b>Nearest neighbor</b>	6,214
<b>Bicubic interpolation</b>	6,441

This test can confirm that the implementation of this paper works fine and generates usable results. Unfortunately the results for the simple resize methods are even better than the SR images. Comparing the difference images of the “nearest neighbor” approach and the “optical flow pyramid LK algorithm” it could be recognized that the sun reflection in the sea was resized very differently between the algorithms. The “nearest neighbor” approach was closer to the original image. It seems that very turbulent images might not be suited best for SR.

## **5 Empirical determination of usability**

---

Knowing that the algorithm itself is working it needs to be solved whether it helps in tracking problems. The following test cases try to give data for possible use cases and the usability of SR in these situations. Every used camera was calibrated.

## 5.1 Euclidian distance between marker positions in SR image of generated image sequence with high resolution generated image

A very interesting question is whether SR improves the accuracy of tracking. To determine this, a generated 3D scene containing a marker on a wall in an empty room (see Figure 11) is rendered in a high resolution image (640x480 pixel) and five low resolution images (320x280 pixel). The small images have a very little shift in the camera position.

A generated 3D image was used as a source because several images with different resolution could be taken without having the problem of being fixed to a camera resolution. In addition the calibration of the virtual camera

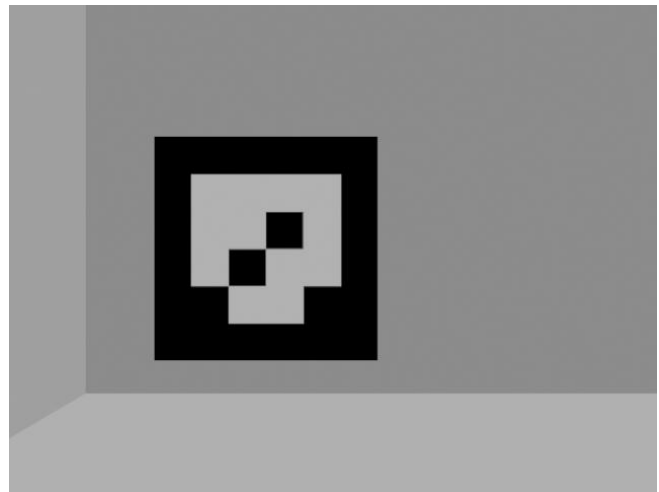


Figure 11 - generated perspective marker

was relatively simple because 3Ds max allows setting camera parameters in its program. Therefore the intrinsic parameters could just be applied and didn't have to be calibrated.

The first step is to determine the position of the marker in the big source image and to save this position to a file. The second step is to determine the same marker in one of the low resolution images as well as the generated SR images. Ideally the marker position should be the same. To have a comparable value for the position difference the Euclidean distance between the discovered marker positions are calculated based on the values of the high resolution image. To



determine which method for calculating the shift in an SR image is the best one, one image based on every method is tested. The following table shows the results:

<b>Image</b>	<b>Distance</b>
<b>low resolution image</b>	0.00857
<b>SR – method: Vandewalle</b>	0.00179
<b>SR – method: feature detector</b>	0.00109
<b>SR – method: flow pyramid LK</b>	0.00394

All methods attain significant improvements against the low resolution image. For the generated image with no noise, the feature detection algorithm brings the best results.

## 5.2 Pose distance between two printed markers

In this test a sheet of paper with two markers was printed (see Figure 12). Both markers have a total size of eight to eight centimeters. Because of being printed the markers will always remain in the exact same distance to each other. To gain

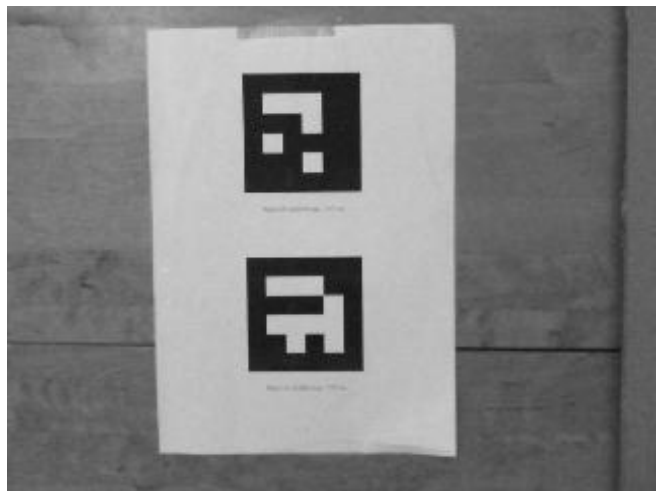

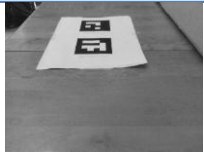





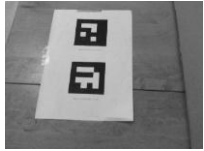
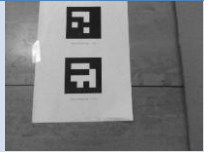

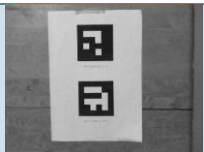
Figure 12 - the marker sheet

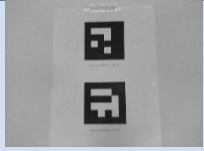
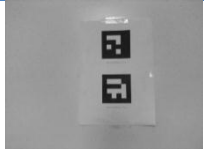

these distance photos of the marker sheet were taken with a reflex camera. These photos have then been analyzed with the marker tracking algorithm of




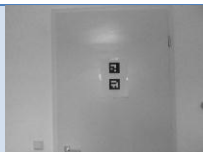
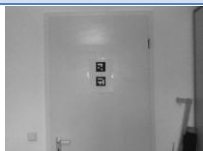
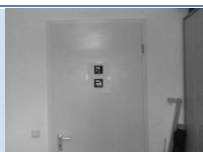
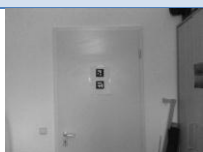
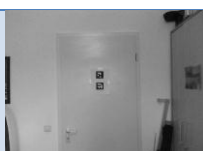

Ubitrack and the distance was saved in a file. This file is set as a base for all tests with the marker sheet.


Having the distance of the markers on the sheet a Microsoft LifeCam HD-3000 [18] was used to take another set of images with a resolution of 320x240. These images vary in distance and angle to the sheet. All images shot for the angle test are taken from a distance of about 0.6m. The question is whether SR improves the maximum angle or distance the markers on the sheet are recognizable. Having recognized the two markers on the low resolution or SR images, the distance difference to the reflex camera distance is calculated. For better comprehensibility the Euclidian distance is formed. The SR images are computed using the Vandewalle algorithm. The following table shows the results as well as the deviation of the marker pose of one of the two markers:

<b>Test: angle</b>	<b>Source image</b>	<b>Source deviation</b>	<b>SR image</b>	<b>SR deviation</b>	<b>Source image</b>
<b>ca. 10°</b>	not recognized		0.00179	0,03991	
<b>ca. 20°</b>	0.00273	0,03780	0.00220	0,03783	
<b>ca. 30°</b>	0.00417	0,03444	0.00191	0,03452	
<b>ca. 40°</b>	0.00252	0,02331	0.00618	0,02376	

<b>ca. 50°</b>	0.00279	0,01648	0.00450	0,01660	
<b>ca. 60°</b>	0.00205	0,00934	0.00263	0,00943	
<b>ca. 70°</b>	0.00180	0,00303	0.00406	0,00328	
<b>ca. 80°</b>	0.00266	0,00209	0.00165	0,00198	
<b>ca. 90°</b>	0.00416	0,00082	0.00464	0,00119	

<b>Test: distance</b>	<b>Source image</b>	<b>Source deviation</b>	<b>SR image</b>	<b>SR deviation</b>	<b>Source image</b>
<b>ca. 0.4 m</b>	0,00279	0,00104	0,00277	0,00115	
<b>ca. 0.6 m</b>	0,00699	0,00065	0,00599	0,00062	
<b>ca. 0.8 m</b>	0,01291	0,00021	0,01249	0,00046	

<b>ca. 1.0 m</b>	0,00381	0,00250	0,00234	0,00354	
<b>ca. 1.2 m</b>	0,02568	0,00148	0,02520	0,00340	
<b>ca. 1.4 m</b>	0,00616	0,00180	0,00506	0,00787	
<b>ca. 1.6 m</b>	not recognized		0,00317	0,00092	
<b>ca. 1.8 m</b>	not recognized		0,00961	0,00062	
<b>ca. 2.0 m</b>	not recognized		0,01642	0,01516	
<b>ca. 2.2 m</b>	not recognized		0,02003	0,02749	
<b>ca. 2.4 m</b>	not recognized		0,04117	0,00931	
<b>ca. 2.6 m</b>	not recognized		0,04258	0,02127	

ca. 2.8 m	not recognized		0,01396	0,00148	
-----------	-------------------	--	---------	---------	---

The test changing the angles shows that the SR extends the range the markers can be recognized, certainly depending

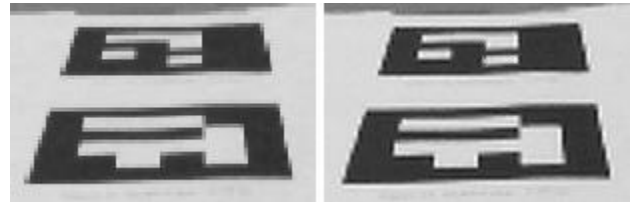


Figure 13 – comparison: markers from an angle of 10°

on the higher amount of pixels available for the markers. Figure 14 shows the difference in the marker quality. The left image is the original image with a quite rough structure whereas the right SR image has an obviously refined structure. Unfortunately the accuracy for determining the distance between the markers stays in the same range.

The test changing the distance shows that the marker distance difference increases, the greater the distance to the marker gets (see Figure 13).

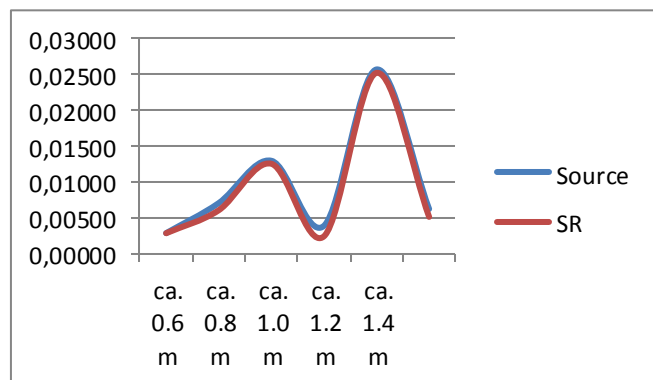


Figure 14 - increasing marker distance difference

There are outliers to the data.

These outliers occur if images have an extraordinarily sharp image. This can happen in good lighting conditions. Very bright reflections like in the 0.8 meter test image seem to disturb the camera sensor. Most of the modern cameras use automatic modes for determining the white balance. The “whitest” value determines the white in the image. If a specific part is extremely bright, the total time of light exposed to the image will be shortened. This induces an image with

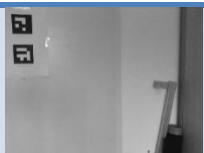
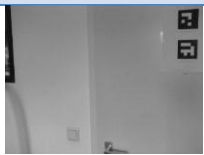
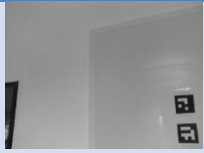

less contrast. A lower contrast complicates the recognition of the border for the marker detection algorithm. Unfortunately the usage of SR images for the marker detection does not increase the quality of the calculated values very much. There is a refinement but this is really low, although the distance out of which the marker can be recognized at least doubles. This is a remarkable advantage what must not be ignored.

Another interesting fact the distance test has shown, is that the measurement for a small distance between normal and SR images is equivalent. This can explain why the results for the distance difference in the angle test which were shot from 0.6 m are quite similar. This suggests the assumption that the results for the SR images in the angle test were better in comparison to the source image results if the images would be taken from a greater distance. If the images were taken from a greater distance the advantages of SR would take effect.

The deviation value shows the error value regarding the recognition of the marker pose. For small camera distances the deviation value has nearly no difference between the source and SR images. The greater the camera distance gets, the more difference accrues between the deviations. Paradoxically the bigger inaccuracy occurs with the SR images. This is probably owed to the calculation of the marker pose. For calculating this pose the intrinsic matrix of the camera is used. Because of the SR images being twice as big as the source images, the camera intrinsics can be doubled, too. This leads to a doubled focal length which therefore increases the inaccuracy in the marker recognition.

### 5.3 Different positions in Image

This test should determine whether the position of the marker in the image has any influence on the quality of the SR enhancement. The setup is very simple. The sheet of paper with the printed markers of the test before is reused as well as the measurements of the reflex camera. A difference is however that four sets of pictures are taken from about 1.0 m of the sheet. The markers are recorded in different positions in the image. It is avoided to have the markers in the center of the image. Again like in the last test the Euclidean distance is calculated. The SR images are computed using the Vandewalle algorithm. The following table shows the results:

marker sheet position	Source image	Source deviation	SR image	SR deviation	Source image
<b>Upper left</b>	0,02429	0,02186	0,02688	0,02878	
<b>Upper right</b>	0,01757	0,02936	0,04517	0,02710	
<b>Lower right</b>	0,05606	0,02572	0,04379	0,01563	
<b>Lower left</b>	0,02803	0,01447	0,03729	0,02057	

Cameras are using lenses to bundle the incoming light. However these lenses cannot bundle the light consistently but generate distortion in the image. This distortion can be removed through image undistortion methods which use the parameters of the camera calibration. If this undistortion is not done, the results of a marker detection algorithm may not be accurate but can

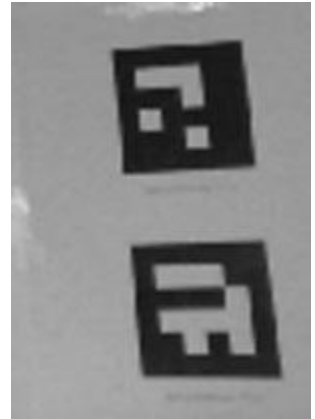


Figure 15 - zoom: distorted marker

be slightly false. As shown in Figure 15 the lower marker stretches unnaturally wide to the lower right corner. This is due to the mentioned distortion. This source image will be undistorted before the marker detection algorithm and can then be detected correctly.

However, the results for the marker difference are considerably worse for the equivalent distance from the distance test before. Despite the undistortion the results of the center of an image should be preferred to the results of the image borders.

The deviations are more volatile than before despite a constant distance because of the distortion. The bigger the image gets with the same distortion values, the greater will be the change in the deviation. Again this is a reason why the points of interest in an image should be located in the image center.

#### 5.4 Determining the better way of taking the images: using a tripod or using a hand

This test should determine whether a SR image calculated with source images that are taken with a tripod might bring better results than taken with the camera in the hand of the user. To resolve this question a constant test environment is



needed. A possible test arrangement could be tracking the marker and the camera with an external tracking system. An external tracking system is a construct which can detect certain objects that are marked in a special way and returns its coordinates in the external tracking system space. Therefore it could return the poses of the camera and the marker. Having these poses the relative pose to each other could be computed. This would allow calculating the accuracy of the specific tracking methods relatively to the detected external poses. Unfortunately the accuracy of these external systems is not as precise as needed for creating really meaningful results.

Another way of solving this issue is to use a multimarker board and to compute its total reprojection error. The reprojection error is the

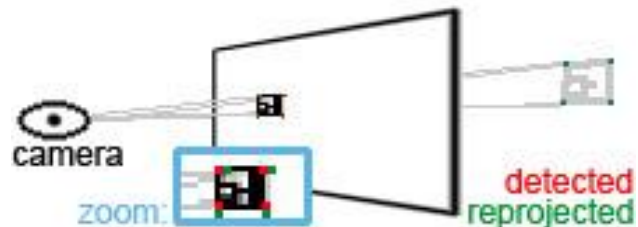


Figure 16 - reprojection error

distance (in pixel) between the actually detected corners of a marker (in 2D) to the points which are calculated by projecting the actual 3D pose onto the image plane (see Figure 16). If just one marker is used the reprojection error should be zero because its 2D corner points are the source to calculate the 3D pose using the camera intrinsics.

To be enabled to use a multimarker board for calculation the overall reprojection error, the marker board must first be known to the detection system. Only if the marker poses are all available, a reprojection error between the known 2D corner points and the calculated 3D poses can be computed. This means that the actual test is a two-step approach. First the marker board will be analyzed and stored and after that, the overall reprojection error will be calculated. As the reprojection error depends strongly on the underlying multimarker board

configuration and the accuracy of its known marker poses, the effective error value will become smaller, the more precisely the initialization of the data is done. Therefore, the multimarker poses registration will be done with

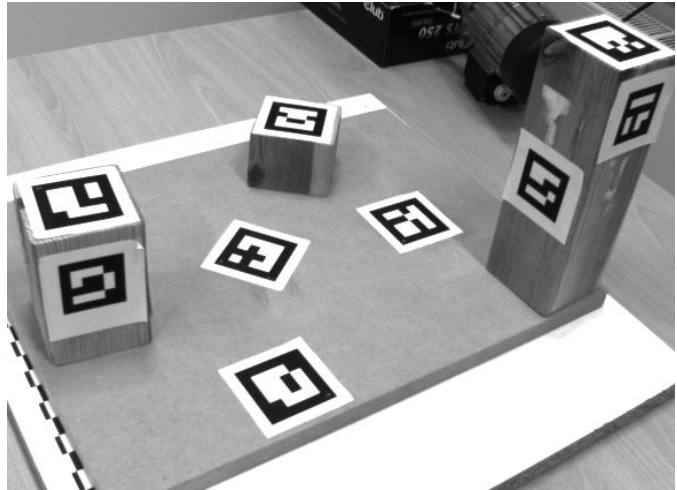


Figure 17 - multimarker board

different sources and the actual calculation of the reprojection errors will use the same set of images in every evaluation. Only if the evaluation set is always the same, the effective error value can be compared.

The multimarker board should provide several different markers with as many different poses as possible. If the marker poses are varying very strongly, it will be easier to calculate useful results, even if only a few of the markers are detected in one image. Figure 17 gives an example of what a multimarker board could look like and what is used in the actual test. The markers have a size of 4.65 cm. In total there are twelve markers applied.

For this test a better camera than a standard webcam is used. More accurate data can be computed if the source images have a higher resolution. This time an uEye cam [19] was chosen. It has a native resolution of 768x576 pixels which conforms to a DVD movie resolution. The SR images have the doubled size and are computed using the feature detection algorithm.

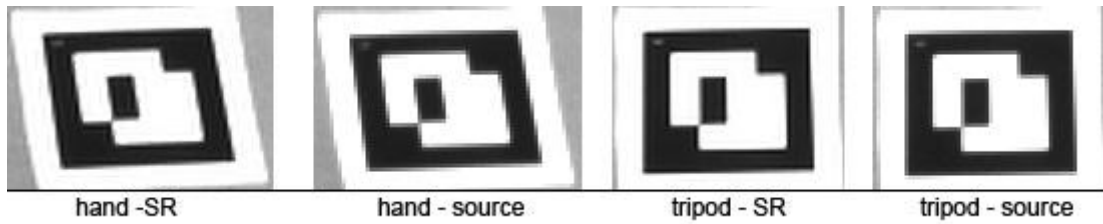
For the test three sets of images were shot. One set of 18 images for initialization using the hand, one set of 19 images for initialization using a tripod

and a set of five images for evaluation. For the two initialization sets the corresponding SR images were calculated and suited sets of images were chosen. If the wrong images for initialization are used, the reprojection error can get enormous values over 10000. Of course the sets of source and SR images always contained the equivalent pictures. After the registration of the multimarker board the reprojection error for every single image of the evaluation set was calculated and the mean value was built. The following table shows the results:

<b>Image recording method</b>	<b>Source image</b>	<b>SR image</b>
<b>Using hand</b>	19.36826	15.57125
<b>Using tripod</b>	9.78149	7.85015

The results suggest that the usage of a tripod can enhance the results quite a lot. The initial idea of SR was to use several images with a minimal shift to have additional information on every image point to enhance the image. With a tripod there is absolutely no shift between the images, but as explained in 1.1 every sensor generates noise. If several images are merged together, this noise contains the desired information for enhancing the image which leads to more accurate marker detection. Furthermore images shot with the use of a tripod have no motion blur which can be generated by the shaking of the hand e.g. or the breathing of the person holding the camera. With no blurring the image is sharper and this leads to a more accurate calculation.

Figure 18 shows examples of one marker of the different initialization sources. It can be recognized that the quality of the corners in the image behaves



**Figure 18 - comparison markers**

equivalent to the calculated values above. The tripod source image has significantly clearer edges than both markers of the “hand” images.

This test can show clearly that SR can minimize the error values by nearly 20%.

### 5.5 Can SR enhance images shot with a reflex camera

As seen so far SR can contribute its part to enhancing marker detection and tracking. All the tests before were made with standard cameras. The most accurate cameras are reflex cameras which leads to the question whether SR can improve their images and hence their results.

To gain data on this question the test in 5.4 is repeated but images of a reflex camera and its SR equivalents are used as source images for initializing the marker board configuration. The images of the reflex camera have a size of 3872x2592 pixels. The SR images have a doubled size and are calculated with the feature detection algorithm as in the test before. Therefore everything stays constant except for the camera. The following table shows the results:




	<b>Source image</b>	<b>SR image</b>
<b>Reflex camera</b>	1,10930	1,09813






Despite the fact that the total reprojection error is already extremely low for the unenhanced source images, SR is still capable of improving its quality. The actual improvement is very low but still a little bit more accurate. The question

remains whether about 1% enhancement is worth generating the SR images and working on these really big images. Despite the computational speed of the algorithm of this paper it takes about one minute to generate a SR image of reflex camera sources. In addition the analyzing of 7744x5184 takes its time, too. But this is a question that must be solved regarding the needs of the specific problem.

### 5.6 The influence of different lighting conditions on the efficiency of SR

This test is supposed to determine whether SR images can lead to an improvement under bad lighting conditions. Several sets of pictures are taken from a distance of about 0.6 m. Again the marker sheet is used and the distance difference is calculated. The camera and the marker sheet are fixed. The brightness is measured roughly with a Voltcraft MS-1300 [20]. The distance difference as well as the deviation of one of the two markers is shown in the table below:

<b>Test: lighting</b>	<b>Source deviation</b>	<b>SR image</b>	<b>SR deviation</b>	<b>Source image</b>	<b>Source deviation</b>
<b>ca. 1 Lux</b>	0,00375	0,02087	0,00458	0,02082	
<b>ca. 2 Lux</b>	0,00354	0,02091	0,00361	0,02081	
<b>ca. 5 Lux</b>	0,00280	0,02096	0,00275	0,02093	

<b>ca. 18 Lux</b>	0,00465	0,02102	0,00286	0,02115	
<b>ca. 170 Lux</b>	0,00413	0,02096	0,00293	0,02105	
<b>ca. 200 Lux</b>	0,00351	0,02096	0,00332	0,02096	
<b>ca. 285 Lux</b>	0,00318	0,02100	0,00298	0,02103	
<b>ca. 350 Lux</b>	0,00385	0,02097	0,00268	0,02095	

Unfortunately there is no improvement in the results. The used SR algorithm does no additional illumination. The algorithm just uses several similar values und forms the ideal values. What the algorithm can do however, is that if one image of the source set contains bad lighting conditions, this will be ignored because of the used median value in the computation of the SR algorithm. But if the mean value of all source images has bad lighting conditions, the resulting SR image will have them, too.

Thanks to the constant distance between the camera and the marker sheet the deviation in these image sets are nearly constant.

## 6 Conclusion

---

Analyzing the tests leads to several conclusions. SR can resize the image with enhancing the actual resolution. Additional information is added that is not available with simple resize methods. SR does not bring any false shift in the image, that means everything remains at its original position considering the stretch factor (cf. Similarity to matlab tool). SR does improve the accuracy of detected marker poses (cf. Determining the better way of taking the images: using a tripod or using a hand; Can SR enhance images shot with a reflex camera) but sometimes the enhancement is minimal and it needs to be decided whether the improvement is worth the computational effort. SR can definitely widen the area for which cameras are useful. Markers can be detected from a greater distance as well as a smaller angle of vision (cf. Euclidian distance between marker positions in SR image of generated image sequence with high resolution generated image). SR cannot improve the lighting conditions because it can only use the data available in each source image (cf. The influence of different lighting).

Summing up these results SR seems an algorithm that is definitely useful for tracking purposes. Considering the computational effort and the fact that more than one image is needed for one SR image, there are cases SR should or could not be used. Analyzing objects on a fast conveyer belt e.g. could be difficult because of the probably high shift between the images. Sometimes it will be most efficient to work with standard images and have the option to enhance chosen images with SR to improve accuracy.

## 7 Possible ways to improve the results

---

The SR method used in the application of this paper depends strongly on the method to align the source images. None of the used methods can be declared as “the one” universal method that always works the best. If an algorithm could be found that would always work fine, SR would become even better.

Another shortcoming is the fact that only planar shifts and rotations in the image plane are allowed. Otherwise, the changes between two images could not be detected. Even if they could be found out, the SR algorithm would rather corrupt the image than bring any enhancement because the source images do not display the same information. Shooting images of a scene from different poses would lead to totally different views with strongly varying objects in the scene. A possible approach in order to solve this problem could be using a marker in the source images and aligning all images according to its pose. That could lead to a corruption of the area in the image around the marker but it itself could possibly be enhanced because the image would be protectively transformed to have the marker at exactly the same position. Whether this is an approach worth evaluating must be investigated separately.

In chapter 1 the assertion was made that SR could improve recognizing “the world” because it should be capable of enhancing the sensor data, in this case the images taken with a camera. Altogether the tests of this paper suggest the assumption that this statement can be confirmed with some minor constraints as summarized in chapter 6.



## 8 Literature

---

- [1] D. Barbe, "Imaging devices using the charge-coupled concept," in *Proceedings of the IEEE*, New York, 1975.
- [2] D. Göhring, "Digitalkameratechnologien: Eine vergleichende Betrachtung CCD kontra CMOS," 2002. [Online]. Available: [http://www.drgoehring.de/uni/papers/CCD-CMOS\\_08122002.pdf](http://www.drgoehring.de/uni/papers/CCD-CMOS_08122002.pdf).
- [3] S. Mühlke, Adobe Photoshop CS 5 - Das Proxisbuch zum Lernen und Nachschlagen, Bonn: Galileo Press, 2011.
- [4] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," *Cambridge University Press*, pp. 155-157, 2003.
- [5] K. Hirokazu and M. Billinghamurst, "ARToolKit Documentation (Computer Vision Algorithm)," 24 08 2005. [Online]. Available: <http://www.hitl.washington.edu/artoolkit/documentation/vision.htm>. [Accessed 30 10 2012].
- [6] OpenCV, "OpenCV | OpenCV," Itseez, 07 04 2012. [Online]. Available: <http://www.opencv.org>. [Accessed 25 10 2012].
- [7] E. Freeman, T. Jones and E. Pasztor, "Examplebased super-resolution," in *Comp. Graph. Appl.*, 2002.
- [8] S. Farsiu, M. Robinson, M. Elad and P. Milanfar, "Fast and robust multiframe super resolution," in *T-IP*, 2004.
- [9] D. Capel, Image Mosaicing and Super-Resolution, Springer-Verlag, 2004.
- [10] M. Irani and S. Peleg, "Improving resolution by image registration," in *CVGIP*, 1991.
- [11] A. C. L. -. LCAV, "Super-Resolution | LCAV," ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 04 06 2012. [Online]. Available: <http://lcav.epfl.ch/software/superresolution/index.html>. [Accessed 25 10 2012].
- [12] ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, "Welcome to Reproducible Research Repository - Reproducible Research Repository," EPrints 3, [Online]. Available: <http://rr.epfl.ch>. [Accessed 25 10 2012].
- [13] P. Vandewalle, S. Süsstrunk and M. Vetterli, "A Frequency Domain Approach to Registration of Aliased Images with Application to Super-Resolution," *EURASIP Journal on Applied Signal Processing (special*

*issue on Super-resolution*), p. 14, 2006.

- [14] J. Shi and C. Thomasi, "Good Features to Track," in *CVPR94*, 1993.
- [15] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," 1981.
- [16] A. Zomet, A. Rav-Acha and S. Peleg, "Robust Super-Resolution," in *Proceedings international conference on computer vision and pattern recognition (CVPR)*, 2001.
- [17] "[http://artist-3d.com/free\\_3d\\_models/dnm/model\\_disp.php?uid=2478&count=count](http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=2478&count=count)," [Online].
- [18] Microsoft, "Microsoft-Webcam: LifeCam HD 3000 | Microsoft Hardware," Microsoft, [Online]. Available: <http://www.microsoft.com/hardware/de-de/p/lifecam-hd-3000/T3H-00003>. [Accessed 25 10 2012].
- [19] IDS IMAGING DEVELOPMENT SYSTEMS GMBH, "USB 2 uEye SE," IDS IMAGING DEVELOPMENT SYSTEMS GMBH, [Online]. Available: <http://www.ids-imaging.de/frontend/products.php?interface=usb&family=SE&tp=0>. [Accessed 25 10 2012].
- [20] Voltcraft, "Voltcraft Digitales Luxmeter MS-1300," [Online]. Available: [http://www.produktinfo.conrad.com/datenblaetter/100000-124999/101148-an-01-ml-LUXMETER\\_MS\\_1300\\_de\\_en\\_fr\\_nl.pdf](http://www.produktinfo.conrad.com/datenblaetter/100000-124999/101148-an-01-ml-LUXMETER_MS_1300_de_en_fr_nl.pdf). [Accessed 25 10 2012].
- [21] artist-3d.com, "Island ship seascape sailboat sunset, (.max) 3ds max, Nature Objects, by Captain Nemo," Artists-3d, 03 03 2010. [Online]. Available: [http://artist-3d.com/free\\_3d\\_models/dnm/model\\_disp.php?uid=2478&count=count](http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=2478&count=count). [Accessed 25 10 2012].
- [22] MathWorks, "MathWorks Deutschland - MATLAB - The Language of Technical Computing," MathWorks, 25 10 2012. [Online]. Available: <http://www.mathworks.de/products/matlab/>. [Accessed 15 10 2012].
- [23] Autodesk, Inc., "3ds Max - Software für 3D-Medellierung, Animation und Rendering - Autodesk," Autodesk, Inc., [Online]. Available: <http://www.autodesk.de/adsk/servlet/pc/index?id=14642267&siteID=403786>. [Accessed 25 10 2012].

---

**9 Figures**

---

Figure 1 - "stairway" effect	4
Figure 2 – example of a chessboard calibration pattern	6
Figure 3 - Zoom: marker border	7
Figure 4 - debug image marker tracker	8
Figure 5 - example Ubitrack marker	10
Figure 6 - Trackman: SRG of saving images as well as its SR equivalent	11
Figure 7 - edge refinement	13
Figure 8 - result of a wrong shift technique	19
Figure 9 - first generated image	21
Figure 10 - second generated image	22
Figure 11 - generated perspective marker	24
Figure 12 - the marker sheet	25
Figure 14 – comparison: markers from an angle of $10^\circ$	29
Figure 13 - increasing marker distance difference	29
Figure 15 - zoom: distorted marker	32
Figure 16 - reprojection error	33
Figure 17 - multimarker board	34
Figure 18 - comparison markers	36