ABSTRACT

We take an opportunity to present this project report on “2d to 3d modelling” and put before readers some useful information regarding our project.the basic purpose of this project is to model the 3d view of 2d images taken by rotating a solid image through 360 degree. Along with hardware like microcontroller, camera and servo motors, matlab software has also been used extensively for generating and displaying the final 3d model. The software allows the visualization of the virtual model in a three-dimensional environment controlled by the user; besides, the data can be exported in different formats.

We have made sincere attempts and taken care to present this matter in precise and compact form, the language being as simple as possible. We are sure that the information contained in this volume would certainly prove useful for better insight in the scope and the dimension of this project in its true perspective. The task of completion of the project though being difficult was made a bit simple interesting and successful due to deep involvement and complete dedication of our group members. The project includes the design and development of a three-dimensional scanner, which uses in its data acquisition system using webcam, canny edge detection technique using matlab, a microcontroller and a display unit as personal computer. The analog part of project includes a rotating platform on which the solid object whose three-dimensional model is to be created is rotated by using servomotor and is controlled by using microcontroller. The software allows the visualization of the virtual model in a three-dimensional environment controlled by the user; besides, the data can be exported in different formats. The mathematical modeling, the design, implementation and the results are discussed.

In our project we intend to capture images of an object that needs to be modeled, from all directions with the help of a camera. The images are then processed i.e. The 2d images that are captured are processed using needle plotting which determines depth of each pixel and then this image is converted into surface image. The processed images are then placed on the visual platform thus giving us a 3d model of the object that was captured by the camera. The alternate methods are laser circuit oriented which is highly expensive or based on different complex algorithms hence we have opted for the above mentioned method as it is simple and cost efficient.

CHAPTER 1: INTRODUCTION

3D images provide a depth and perspective that we cannot achieve from 2D images. 3D Modeling is not a novel concept in itself as it has been attempted before and considerable research has been extended in that field. Most of these methods have some inherent shortcomings and flaws. The problem with some of these technologies is that it is very expensive and can only be afforded by huge corporations. We will be trying to make our technique relatively inexpensive and portable to a large extent.

The first step of the project is image acquisition with a webcam, the object to be scanned it placed on a rotating surface and the surface will be rotated using a servo motor, motion of which will be controlled by Microcontroller Atmega8 which will also take care of synchronisation. The object has to rotated to obtain the images of the object from all angles.

BRIEF HISTORY OF 3D MODELING:

The advent of 3D images was with the introduction of holography. Holograms were the most primitive form of this technology.Holographyis a technique that allows the light scattered from an object to be recorded and later reconstructed so that it appears as if the object is in the same position relative to the recording medium as it was when recorded. The image changes as the position and orientation of the viewing system changes in exactly the same way as if the object were still present, thus making the recorded image (hologram) appears three dimensional.

The next widely used technique is called Stereoscopy. Stereoscopy is the enhancement of the illusion of depth in a photograph, movie, or other two-dimensional image by presenting a slightly different image to each eye. The drawback with this method is that the 3rd dimension is only an illusion and it cannot be used for technical detailed analysis of the object. Thus its use is restricted to the entertainment industry only.

One approach to providing detailed 3D images consists of taking a series of pictures of the object to be modeled and then using various image processing techniques to reconstruct a 3D image from the 2D images. Tremendous research has been carried based on this simple approach and many algorithms have been developed. A developer has to first ascertain his priorities while

developing such an algorithm because most of these methods are a trade-off between the resolution and detail in the image and the time taken to create a satisfactory model. The 'Surface Model Reconstruction of 3D Objects From Multiple Views' by Vincenzo Lippiello and Fabio Ruggiero is one such example where the details are compromised to increase the speed.

OBJECTIVE:

This project aims at providing a 3D image using nothing more than a simple image capturing device [a high resolution webcam] and a rotating platform on which the object is mounted. The various images captured are then reconstructed into a 3D whole with the help of Matlab functions. The proposed project is an attempt at providing a low cost portable alternative to the existing techniques that provide similar results. Special attention has been paid to reduce the tradeoff between details and the time required thus satisfying a wide spectrum of needs.

The first step of the project is image acquisition with a webcam, the object to be scanned it placed on a rotating surface and the surface will be rotated using a servo motor, motion of which will be controlled by Microcontroller Atmega8 which will also take care of synchronisation. The object has to rotated to obtain the images of the object from all angles. To obtain a 3D model we had initially attempted to first obtain the edges in the object using the predefined functions in MATLAB which is very easy and gives satisfactory output.

CHAPTER 2: LITREATURE SURVEY

A. IMAGE ACQUISITION

This is the first process of our project where the methods to acquire raw images is discussed.

1. LASER SCANNING

Laser scanners are widely used for 3D scanning, they operate on two different principles for obtaining the co-ordinates of the surface of the object.

1) Time of flight measurement
2) Triangulation

Time of flight measurement :

The time-of-flight 3D laser scanner uses laser light to probe the subject. At the heart of this type of scanner is a time-of-flight laser rangefinder. The laser rangefinder finds the distance of a surface by timing the round-trip time of a pulse of light. A laser is used to emit a pulse of light and the amount of time before the reflected light is seen by a detector is timed. Since the speed of light c is a known, the round-trip time determines the travel distance of the light, which is twice the distance between the scanner and the surface. If t is the round-trip time, then distance is equal (c*t/2). The accuracy of a time-of-flight 3D laser scanner depends on how precisely we can measure the t time: 3.3 picoseconds (approx.) is the time taken for light to travel 1 millimeter.

The advantage of time-of-flight range finders is that they are capable of operating over very long distances, on the order of kilometers. These scanners are thus suitable for scanning large structures like buildings or geographic features.

The disadvantage of time-of-flight range finders is their accuracy. Due to the high speed of light, timing the round-trip time is difficult and the accuracy of the distance measurement is relatively low, on the order of millimeters.

Triangulation :

The triangulation 3D laser scanner also uses laser light to probe the environment. With respect to time-of-flight 3D laser scanner the triangulation laser shines a laser on the subject and exploits a camera to look for the location of the laser dot. Depending on how far away the laser strikes a surface, the laser dot appears at different places in the camera's field of view. This technique is called triangulation because the laser dot, the camera and the laser emitter form a triangle. The length of one side of the triangle, the distance between the camera and the laser emitter is known. The angle of the laser emitter corner is also known. The angle of the camera corner can be determined by looking at the location of the laser dot in the camera's field of view. These three pieces of information fully determine the shape and size of the triangle and gives the location of the laser dot corner of the triangle. In most cases a laser stripe, instead of a single laser dot, is swept across the object to speed up the acquisition process.

The triangulation process is explained in Fig2.1 and Fig 2.2

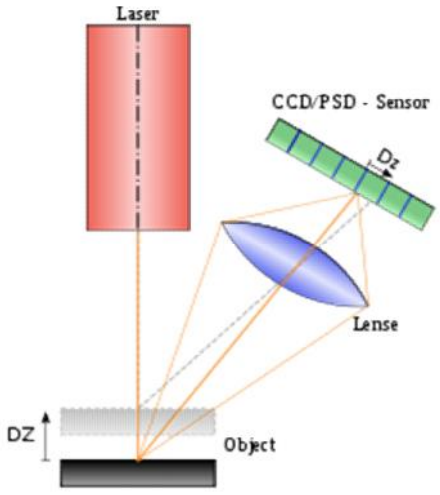


Fig 2.1 Triangulation setup

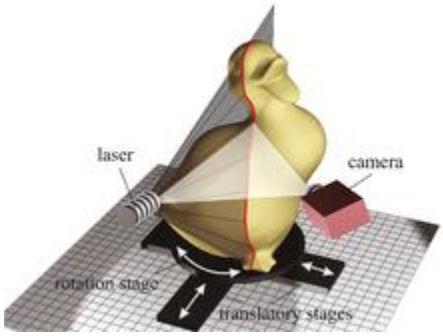


Fig 2.2 Line scanning

The calculation of distance by triangulation method is done in the following way as explained in Fig 2.3

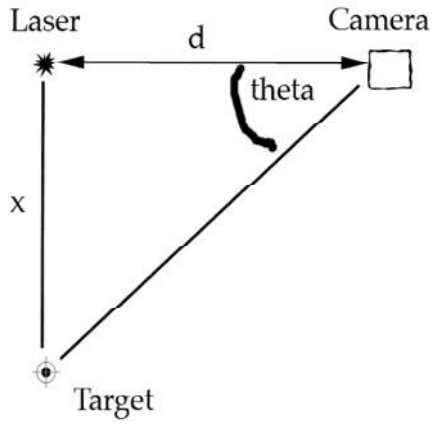


Fig 2.3Triangulation distance measurement logic

The distance(x) of the object from laser is given by the formula

$$x = d \tan \theta$$

Triangulation range finders have a limited range of some meters, but their accuracy is relatively high. The accuracy of triangulation range finders is on the order of tens of micrometers

2. CONTACT SCANNER

Contact 3D scanners probe the subject through physical touch. A CMM (coordinate measuring machine) is an example of a contact 3D scanner. It is used mostly in manufacturing and can be very precise. The disadvantage of CMMs though, is that it requires contact with the object being scanned. Thus, the act of scanning the object might modify or damage it. This fact is very significant when scanning delicate or valuable objects such as historical artifacts. The other disadvantage of CMMs is that they are relatively slow compared to the other scanning methods. Physically moving the arm that the probe is mounted on can be very slow and the fastest CMMs can only operate on a few hundred hertz. In contrast, an optical system like a laser scanner can operate from 10 to 500 kHz.

3. IMAGE ACQUISITION USING WEBCAM

The Image Acquisition Toolbox in Matlab (Windows version) allows one to interface Matlab with a Webcam. This is available from R2007a (not sure about earlier versions). Similar to the audio recording object created earlier, here we create a **videoinput** object. But before that is done, Matlab needs to find out what are the webcam devices that are connected to your computer.

Firstly, a imaqhwinfo gives information about the existing adaptors for your webcam device. You can get more information on each adapter, by using imaqhwinfo('winvideo') where *winvideo* is one of the adaptors. In this, (if you have a device connected) you shall get a Device IDs attached to your webcam device. Further information pertaining to the device can be obtained by imaqhwinfo('winvideo',1) where 1 is the Device ID you saw earlier.

This gives you much needed information regarding the capture device. The resolution (800×600, 1024×768, 1600×1200, etc.), format (RGB, YUV, etc.) which needs to be selected when creating a video object.

Armed with all this imaqhwinfo (**im**age **acq**uisition **h**ard**w**are **info**rmation) you are ready to create your own video object.

```
vidobj = videoinput('winvideo',1,'RGB_1024x768');
```

'RGB_1024x768′ was just the format that I selected. You should use one of those that were available in your device info query. The most important command now would be to start your video objectstart(vidobj). It is at this point, or during the creation of video object, that the light (if any) on your webcam would start glowing indicating capture.

You can obtain snapshots of capture by using the frame = getsnapshot(vidobj); or view the continuous stream of frames by saying preview(vidobj);.

A safe closure (unlocking of the video handles) of the video object is extremely important so that it can be started again easily. Astop(vidobj) followed by delete(vidobj) is the best way to follow.

Another point to note is that all external capture devices, are locked by software which try to access it. Thus, you would get errors like Device not ready, or Device already in use in case you are already viewing the capture stream in any other software. So its recommended that you cleanly stop that software first and then let Matlab take over.

There are a variety of options that are not discussed here for lack of purpose like an automatic trigger (after a defined interval). All the options can be seen by imaqhelp(videoinput).

B. IMAGE PROCESSING USING MATLAB:

In our project, the image processing includes basically two steps :

1)Edge detection
2)Edge Linking

EDGE DETECTION :

Edge detection is a fundamental tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction, which aim at identifying points in a digital image at which the image brightness changes sharply or more formally has discontinuities.

EDGE DETECTION TYPES: There are many ways to perform edge detection. However, the most may be grouped into two categories, Gradient and Laplacian. The Gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find edges.

i) Laplacian Edge Detection:

We wish to build a morphing algorithm which operates on features automatically extracted from target images. A good beginning is to find the edges in the target images.

Steps to implement a Laplacian Edge Detector:

Step 1: Start with an image

Step 2: Blur the image prior to edge detection by convolving the image with a Gaussian (A Gaussian is used because it is "smooth"; a general low pass filter has ripples, and ripples show up as edges).

Step 3: Perform the Laplacian on this blurred image. We observe the edges of the test image, but we also get many false edges due to ripple and texture in the image. To remove these false edges, we add a step to our algorithm. When we find a zero crossing of the Laplacian, we must also compute an estimate of the local variance of the test image, since a true edge corresponds to a significant change in intensity of the original image. If this variance is low, then our zero crossing must have been caused by ripple.

Step 4: Find the zero crossings of the Laplacian and compare the local variance at this point to a threshold. If the threshold is exceeded, we declare an edge.

Step 5: Median Filter the image. We apply a Median Filter because it removes the spot noise while preserving the edges.

ii) Sobel Edge Detection:

Based on this one-dimensional analysis, the theory can be carried over to two-dimensions as long as there is an accurate approximation to calculate the derivative of a two-dimensional image. The Sobel operator performs a 2D spatial gradient measurement on an image. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. The Sobel edge detector uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows). A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time

The magnitude of the gradient is then calculated using the formula:

$$G = \sqrt{GX^2 + GY^2}$$

An approximate magnitude can be calculated using:

$$|G| = |Gx| + |Gy|$$

iii) Compass Edge Detection:

Compass Edge Detection is an alternative approach to the differential gradient edge detection . The operation usually outputs two images, one estimating the local edge gradient magnitude and one estimating the edge orientation of the input image.

When using compass edge detection the image is convolved with a set of (in general *8*) convolution kernels, each of which is sensitive to edges in a different orientation. For each pixel the local edge gradient *magnitude* is estimated with the maximum response of all *8* kernels at this pixel location:

$|G| = \max(|G_i|:i=1 \text{ to } n)$

where G_i is the response of the kernel *i* at the particular pixel position and *n* is the number of convolution kernels. The local edge *orientation* is estimated with the orientation of the kernel that yields the maximum response.

The whole set of *8* kernels is produced by taking one of the kernels and rotating its coefficients circularly. Each of the resulting kernels is sensitive to an edge orientation ranging from *0°* to *315°* in steps of *45°*, where *0°* corresponds to a vertical edge.

The maximum response |*G*| for each pixel is the value of the corresponding pixel in the output magnitude image. The values for the output orientation image lie between *1* and *8*, depending on which of the *8* kernels produced the maximum response.

This edge detection method is also called *edge template matching*, because a set of edge templates is matched to the image, each representing an edge in a certain orientation. The edge magnitude and orientation of a pixel is then determined by the template that matches the local area of the pixel the best.

The compass edge detector is an appropriate way to estimate the magnitude *and* orientation of an edge. Although differential gradient edge detection needs a rather time-consuming calculation to estimate the orientation from the magnitudes in the x- and y-directions, the compass edge detection obtains the orientation directly from the kernel with the maximum response. The compass operator is limited to (here) *8* possible orientations; however experience shows that most direct orientation estimates are not much more accurate. On the other hand, the compass operator needs *8* convolutions for each pixel, whereas the gradient operator needs only *2*, one kernel being sensitive to edges in the vertical direction and one to the horizontal direction.

The result for the edge magnitude image is very similar with both methods, provided the same convolving kernel is used.

iv) Canny Edge Detection:

The Canny operator was designed to be an optimal edge detector (according to particular criteria --- there are other detectors around that also claim to be optimal with respect to slightly different criteria). It takes as input a gray scale image, and produces as output an image showing the positions of tracked intensity discontinuities.

The Canny operator works in a multi-stage process. First of all the image is smoothed by Gaussian convolution. Then a simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothed image to highlight regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as *non-maximal suppression*. The tracking process exhibits hysteresis controlled by two thresholds: *T1* and *T2*, with $T1 > T2$. Tracking can only begin at a point on a ridge higher than *T1*. Tracking then continues in both directions out from that point until the height of the ridge falls below *T2*. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

The effect of the Canny operator is determined by three parameters --- the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds used by the

tracker. Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The localization error in the detected edges also increases slightly as the Gaussian width is increased.

Usually, the upper tracking threshold can be set quite high, and the lower threshold quite low for good results. Setting the lower threshold too high will cause noisy edges to break up. Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output.

One problem with the basic Canny operator is to do with Y-junctions *i.e.* places where three ridges meet in the gradient magnitude image. Such junctions can occur where an edge is partially occluded by another object. The tracker will treat two of the ridges as a single line segment, and the third one as a line that approaches, but doesn't quite connect to, that line segment.

EDGE LINKING:

The edges have to be linked appropriately to reconstruct the image form the edges, this process is called as edge linking.

1. LOCAL EDGE LINKING METHODS:

(i) Edge linking algorithms assemble edge pixels into meaningful edges and or region boundaries.

Local approaches only require information about edge points in local neighbourhoods, typically 3x3 neighbourhoods. Regional approaches require information about all the points on the boundary of a region. We say that two edge pixels (x, y) and (s, t) are similar in magnitude if the absolute value of the difference of their gradient magnitudes is less than a threshold.

$$|M(x,y) - M(s,t)| \leq E$$

We say that two edge pixels (x, y) and (s, t) are similar in angle if the absolute value of the difference of their gradient direction is less than a threshold.

$$|\alpha(x,y) - \alpha(s,t)| \leq A$$

We traverse all the pixels of the image. We link each edge pixel (x, y) with all edge pixels in its neighbourhood that are similar to it in both magnitude and angle. A record must be kept of linked points as the centre of the neighbourhood is moved from pixel to pixel.

(ii) A simplification of the algorithm can be particularly useful, especially when we are interested on few edge directions only.

1. Compute the gradient magnitude and angle arrays, $M(x, y)$ and $\alpha(x, y)$ of the input image. We can do this using, for example, the Sobel masks.

2. Form a binary image g, whose value at any pair of coordinates (x, y) is given by:

$$g(x,y) = \begin{cases} 1 & \text{if } M(x,y) > T_M \quad \text{AND} \quad \alpha(x,y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

Where $T_M$is a threshold used for edge detection, A is a specified angle direction, and $\pm T_A$ defines the band of acceptable directions about A.

3. Scan the rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length K.

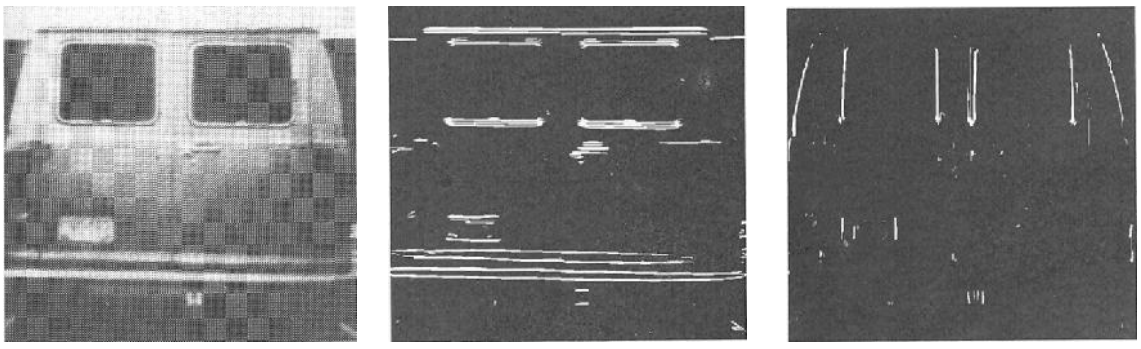
   By definition, a gap is bounded at both ends by one or more 1s.

   Step 3 of the algorithm links edges in the horizontal direction.

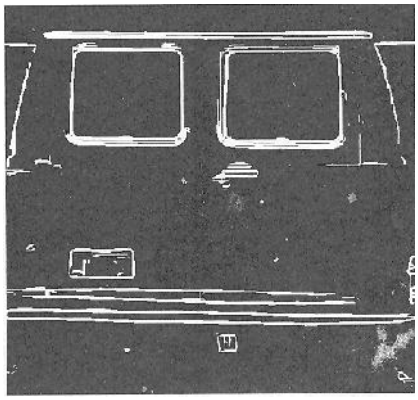
4. To link edges in any other direction θ, we rotate g by $\theta$and apply Steps 2-3. We then rotate the result back by $-\theta$.

5. The simplified edge linking algorithm can be used in applications where we are interested in the horizontal and vertical directions only.

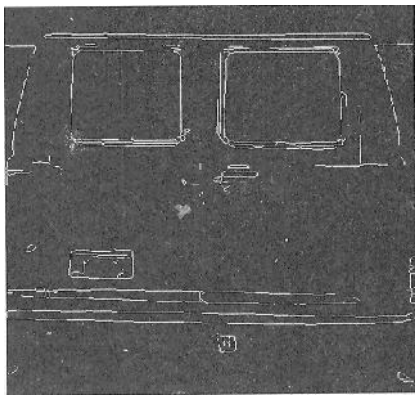
6. For example, finding rectangular regions in an image that are good candidates for corresponding to licence plates.



(a) Gradient magnitude **(b)** Horizontal edge linking **(c)** Vertical edge linking



(d) The logical OR of the horizontal and vertical edge linking



(e) The final result after morphological image-post processing

Fig 2.4 Local edge linking

For the horizontal edge linking, $T_M$was set at 30% of the maximum gradient value, A=90° because the gradient direction is vertical to the edge, T_A=45°. K=25, that is, we fill gaps of 25 or fewer pixels.

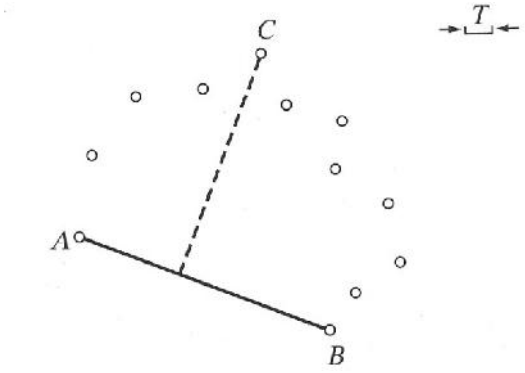
2. REGIONAL EDGE LINKING:

The regional edge linking algorithm assumes that the edge points on the region's boundary are known and ordered, e.g. in a clockwise or counter-clockwise direction.

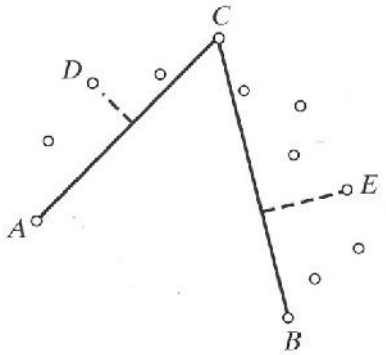
The algorithm computes a polygon approximating the boundary. The polygon is defined by the set of its vertices, which is a subset of the input edge points.

The algorithm starts with two edge points and iteratively adds new points until the polygonal approximation is accurate enough.

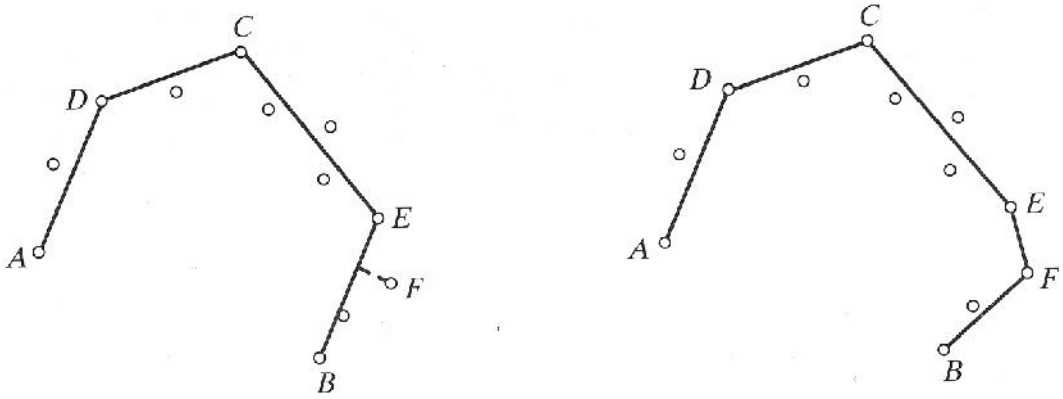
Example:



(a) Iteration No. 1



(b) Iteration No. 2



(c) Iteration No. 3 (**d)**Iteration No. 4

Fig2.5 Regional Edge Linking

Starting with two vertices *A* and *B*, we measure the distances of the other points to the line *AB*.

If the largest distance is above a threshold *T* than we add the furthest point *C* to the set of vertices and obtain the new open polygon *ACB*.

We continue iteratively, measuring distances to *AC* and then *CB*.

The algorithm stops when no other points can be added.

Before starting the algorithm we need to know if the input points represent an open or a closed curve.

A large distance between two consecutive points *A*, *B* relative to the other distances between consecutive points indicates an open curve with *A*, *B* as end points.

The algorithm can be described as follows

1. Specify two starting points *A* and *B*. Specify a threshold *T* and two empty stacks, OPEN and CLOSED.
2. If we deal with a closed curve, put *A* into OPEN and *B* into OPEN and into CLOSED. If we deal with a open curve put *A* into OPEN and *B* into CLOSED.
3. Compute the line passing from the last point in CLOSED and the last point in OPEN.
4. Compute the distance from that line to all the points in *P* whose sequence place them between the two points defining the line. Select the point V_{max} with the maximum distance D_{max}.
5. If $D_{max}>T$, place V_{max}at the end of the OPEN stack. Go to Step 3.
6. Else, remove the last vertex from OPEN and insert it as the last vertex of CLOSED.
7. If OPEN is not empty, go to Step 3.
8. Exit. The vertices in CLOSED are the vertices of the polygon.

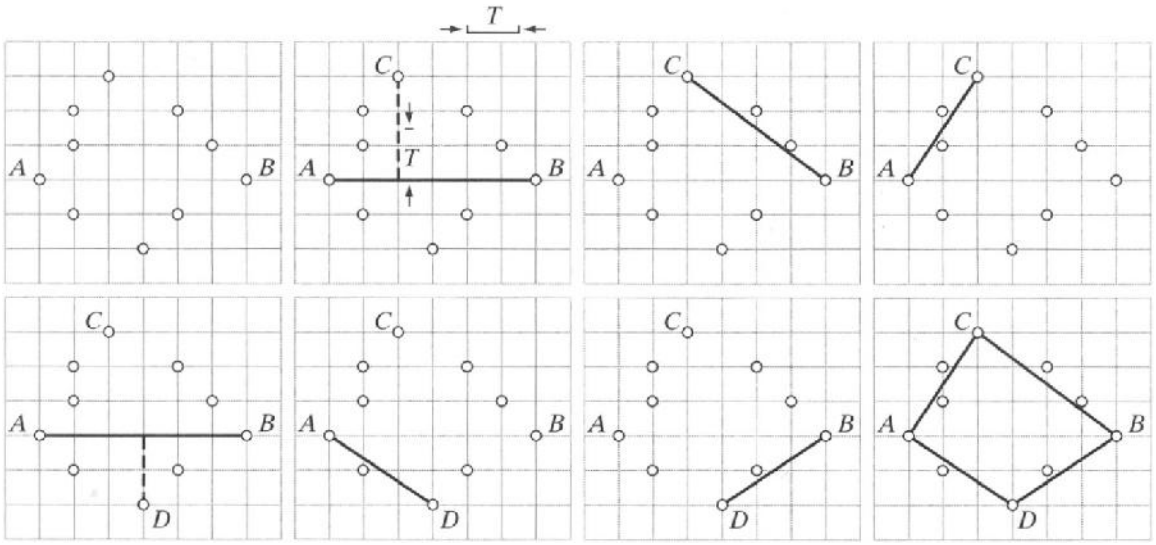


Fig 2.6 Step by step reconstruction of outline

CLOSED	OPEN	Curve segment processed	Vertex generated
B	B,A	-	A,B
B	B,A	(BA)	C
B	B,A,C	(BC)	-
B,C	B,A	(CA)	-
B,C,A	B	(AB)	D
B,C,A	B,D	(AD)	-
B,C,A,D	B	(DB)	-
B,C,A,D,B	Empty	-	-

Table2.1 Table showing the step by step reconstruction of the object

SURFACE GENERATION BY SHAPELETS

What are Shapelets?

The term ‘shapelet’ was first coined by Refregier for his orthonormal basis set consisting of weighted Hermite polynomials. Here we use the term to describe any basis function of finite support used for representing shape.

There are a number of computer vision techniques that attempt to determine the surface normals within a scene. These include shape from shading and its extension, photometric stereo, shape from texture, and slant and tilt from differential invariants. Often the surface normals obtained from these methods are ill-conditioned and subject to noise. But the surface shape of an object can be deduced given just the surface normals. The traditional approach to reconstructing a surface from its surface normals is via integration. The difficulty with this approach is that it can be very sensitive to noise and there is the problem of finding appropriate regularization techniques to impose the requirement that the surface gradient be integrable.

Constraints on the choice of Shapelet

Potentially there are many basis functions that could be used as shapelets. In designing a bank of shapelet filters we start by noting that correlating the gradient of one shapelet filter with the gradient of the signal will correspond to extracting a band of frequencies from the signal gradient. The need to reconstruct a surface from correlations between surface normals and shapelet gradients imposes some constraints on the shapelet function. These are:

• The gradient of the shaplet function must satisfy the admissibility condition of zero mean.

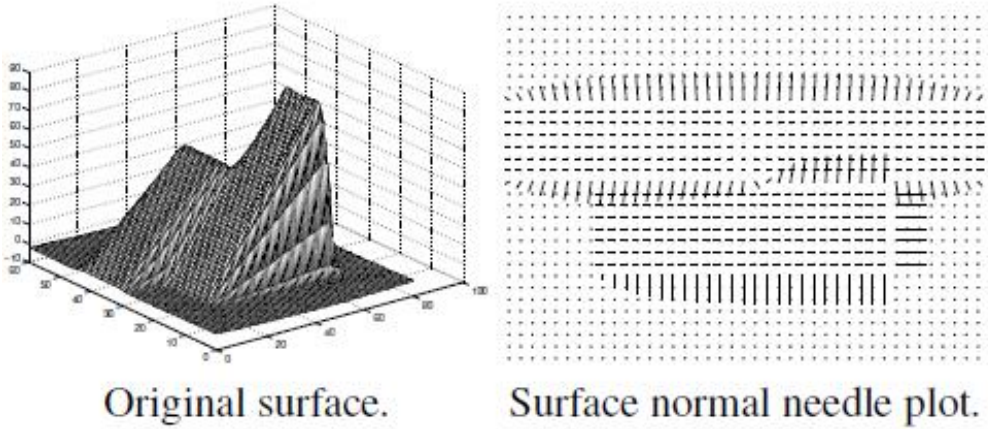
• The shapelet must have minimal ambiguity of shape with respect to its gradient.

• The shapelet function must allow preservation of phase information in the signal.

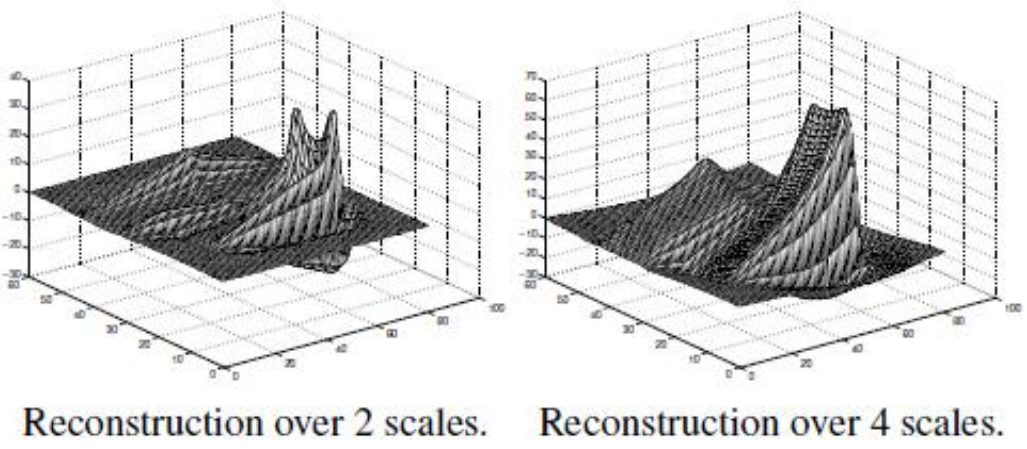
• The bank of shapelet filters should, ideally, provide uniform coverage of the signal spectrum so that it is faithfully reconstructed.

To achieve a shapelet with minimal ambiguity of shape with respect to its gradient, it should be simple and ideally take the form of a single, symmetric peak so that the gradient

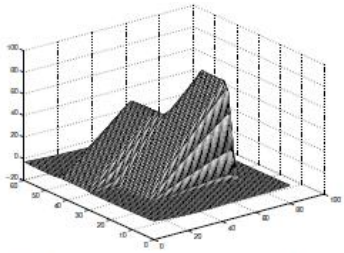
function will have a single positive peak and a single negative peak. With the assumption that the shapelet is non-negative and symmetric, say a Gaussian, the gradient function will be odd-symmetric, resulting in a frequency domain transfer function that is complex and odd symmetric. Changing the scale of the shapelet will have the effect of changing the separation of the peaks of the gradient function producing a transfer function that responds to a different band of frequencies. Thus, if several scales of shapelets are used one can achieve fairly complete coverage of the spectrum. As shown in Fig 2.7



(a) (b)



(c) (d)



Reconstruction over 6 scales.

(e)

Fig 2.7 Shapelet reconstruction over different scales

An important constraint is that the transfer function of the shapelet gradient should not corrupt phase information in the gradient of the signal. This implies that the transfer function of the shapelet gradient should be nonnegative for positive frequencies and non-positive for negative frequencies. Achieving this with a simple shape of finite support limits the shapelet function to a Gaussian, or near Gaussian, shape.

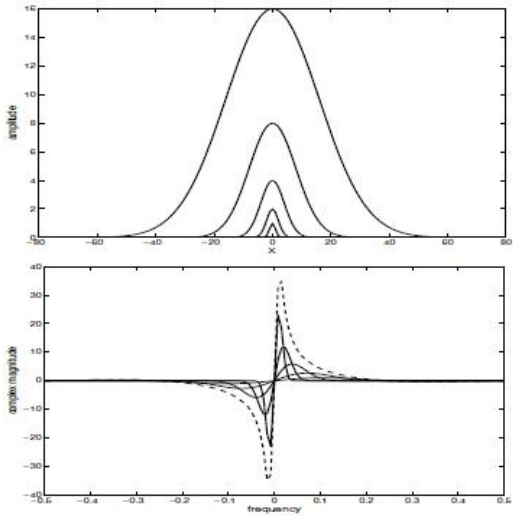


Figure 2.8 A shapelet bank formed by five Gaussians (top),and transfer functions of their gradients (bottom).

The dashed line in Fig2.8 shows the sum of the transfer functions, ideally this should form a curve that is inversely proportional to frequency.

The sum of all the transfer functions of the shapelet gradients will represent the total energy extracted from the signal gradient spectrum. The signal gradient spectrum differs from the original signal spectrum in that its Fourier components are phase shifted by $\pi/2$ and the

amplitudes are scaled by the frequency. Accordingly the sum of the shapelet gradient transfer functions should attempt to form a curve that is inversely proportional to frequency to counteract this scaling of the gradient spectrum. Ideally the net result is that we obtain uniform coverage of the original signal spectrum. This ensures all frequency components of the signal are represented equally and we obtain a faithful reconstruction, up to a scale factor.

3.2.1.2.3 Reconstruction Of 3D Surface

Correlation of the surface and Shapelet slants is done in terms of the magnitude of gradient (▫) which is given by the tangent of the slant (σ) i.e.|▫| = tan(σ) The gradient correlation ($C_{▫i}$) is then formed as $C_{▫i} = |▫_f| \star |▫_{si}|$

where $|▫_f|$ and $|▫_{si}|$ denote the gradients of the surface and shapelet at scale i respectively.
With no tilt information, this correlation matches positive and negative gradients equally because we only have access to the gradient magnitude. Thus a mound in the surface will correlate equally with a similarly shaped depression in the surface. To be able to make a distinction between positive and negative shapes tilt information must be used. If at some point the surface and shapelet gradient magnitudes match and the tilt directions also match then the component of the shapelet at this point must be positive. If the tilts of the surface and shapelet are in opposite directions then the shapelet component must be negative. If the tilts are orthogonal then there is no correlation, positive or negative, between the surface and shapelet. Thus the gradient correlation must be multiplied by a tilt correlation measure that varies between 1, when tilts are aligned, to -1 when the tilts are in opposite directions. An obvious measure that satisfies this requirement is the cosine of the tilt angle difference.

To form the tilt correlation between a shapelet at scale i and the surface we sum the cosine of the tilt differences between points on the surface and shapelet, and use the standard trigonometric difference equation to overcome any angle wraparound problems at the origin. Thus,

$C_{\tau i} = \cos(\tau_f) \star \cos(\tau_{si}) + \sin(\tau_f) \star \sin(\tau_{si})$

where τ_f and τ_{si} denote the tilts of the surface and shapelet at scale i respectively.

The overall correlation measure between surface and shapelet at scale i is obtained by the point-wise product of the gradient and tilt correlations

$C_i = C_{\square i}.\ C_{\tau i}$

$= | \square_f | \star | \square_{si} |.\ [\cos(\tau_f) \star \cos(\tau_{si}) + \sin(\tau_f) \star \sin(\tau_{si})]$

$= [\ |\square_f|.\cos(\tau_f)\ ] \star [\ |\square_{si}|.\cos(\tau_{si})\ ] + [\ |\square_f|.\ \sin(\tau_f)\ ] \star [\ |\square_{si}|.\ \sin(\tau_{si})\ ]$

where .denotes point-wise multiplication. This process is performed over multiple shapelet scales and the results summed to form the reconstruction (R)

$R = \sum_i Ci$

Reconstruction of surfaces from their surface normals via shapelets provides a reconstruction method that is robust to noise. The use of basis functions implicitly imposes a continuity constraint in the reconstruction, yet at the same time allows sharp transitions to be

represented in the surface via the finer scales of shapelets. There is no need to infer the locations of discontinuities in the surface and/or apply special conditions at these points. The correlation process treats slant and tilt separately and makes the different roles of slant and tilt explicit in the reconstruction process. This permits data with tilt ambiguity to be considered and allows slant and tilt data to be considered in isolation. These options would be impossible to consider using a reconstruction process based on integration of gradients. It is envisaged that the flexibility and robustness of the shapelet reconstruction approach will create opportunities for new shape from texture and shape from shading algorithms. For example, where tilt information is uncertain one may be able to adopt an iterative approach. First, a surface would be hypothesized using slant information and, if available, any estimates of tilt. The surface would then be verified by comparing an estimate of the appearance of the reconstructed surface against the original image, and on the basis of this the tilt estimates would be updated. This would be repeated until convergence occurs.

C. ROTATION OF PLATFORM :

There are two methods for rotating the platform which is controlled by microcontroller.

1. By using Stepper motor

A stepper motor's shaft has permanet magnets attached to it. Around the body of the motor is a series of coils that create a magnetic field that interacts with the permanet magnets. When these coils are turned on and off the magnetic field causes the rotor to move. As the coils are turned on and off in sequence the motor will rotate forward or reverse. This sequence is called the phase pattern and there are several types of patterns that will cause the motor to turn. Common types are full-double phase, full-single phase, and half step.

To make a stepper motor rotate, you must constantly turn on and off the coils. If you simply energize one coil the motor will just jump to that position and stay there resisting change. This energized coil pulls full current even though the motor is not turning. The stepper motor will generate a lot of heat at standstill. The ability to stay put at one position rigidly is often an advantage of stepper motors. The torque at standstill is called the holding torque.

Because steppers can be controlled by turning coils on and off, they are easy to control using digital circuitry and microcontroller chips. The controller simply energizes the coils in a certain pattern and the motor will move accordingly. At any given time the computer will know the position of the motor since the number of steps given can be tracked. This is true only if some outside force of greater strength than the motor has not interfered with the motion.

An optical encoder could be attached to the motor to verify its position but steppers are usually used open-loop (without feedback). Most stepper motor control systems will have a home switch associated with each motor that will allow the software to determine the starting or reference "home" position.

2. By using Servomotor

There are several types of servo motors but I'll just deal with a simple DC type here. If you take a normal DC motor that can be bought at Radio Shack it has one coil (2 wires). If you attach a battery to those wires the motor will spin. See, very different from a stepper already!. Reversing the polarity will reverse the direction. Attach that motor to the wheel of a robot and watch the robot move noting the speed. Now add a heavier payload to the robot, what happens? The robot

will slow down due to the increased load. The computer inside of the robot would not know this happened unless there was an encoder on the motor keeping track of its position.

So, in a DC motor, the speed and current draw is a affected by the load. For applications that the exact position of the motor must be known, a feedback device like an encoder MUST be used (not optional like a stepper).

The control circuitry to perform good servoing of a DC motor is MUCH more complex than the circuitry that controls a stepper motor.

3. RC Servos:

Often when talking about robots the word "servo" really means an RC (remote control) servo motor. This is a small box designed for use in hobby airplanes and cars. Inside this box is a complete servo system including: motor, gearbox, feedback device (pot), servo control circuitry, and drive circuit. It's really amazing that they can stick all of that in such a small package.

RC servos normally have 3 wires: +v, ground, control. The control signal is a pulse that occurs at about 50hz. The width of the pulse determines the position of the servo motors output. As you can see, this would be pretty easy to control with a digital controller such as a Basic Stamp. Most will run on 5-6 volts and draw 100-500ma depending on size.

CHAPTER 3: PROBLEM DEFINITION

METHODOLOGY

A. HARDWARE:

There is a web camera where we use cct sensor and captures in YCbCr range whose default size is 240*160. Then we are using a blackbox inside which we have attached one black round wood type material below which we have a srvomotor inside. Servomotor is basically a DC motor with a feedback.

Having a feedback control circuit inside the motor. It can rotate anywhere between 0degree and 180 degree. For which object which you see you will find out edges from this side and rotate it exactly 180 degree you will get all the edges of the object and this servomotor requires only three connections from the controller side , center one is VCC, side one is ground and orange pin is your PWM input. It works on PWM.

When PWM is 1ms, then servo is 90 degree. Whem PWM goes below 1ms, it moves towards 0 degree to -180 degree. When it goes towards 2 ms, it goes towards +180 deg. That is the basic logic of servo, it works on PWM.

We are using ATMEGA 8 microcontroller which is named as 8 because it is an 8bit microcontroller and it has 8kb of internal memory. The 10k resistor is used for pull-up between pin no1 and pin no 7. Because pin 1 is reset pin and it should not be connnected to ground or left open because IC will not work. So for that it is merged that you pull up the pin. Another pin goes to reset bar and it will be always high. That means your IC will not reset ever. We are using communication with the computer and for that we are using a board called FTDI and for any communication the baud rate should be fixed to the communication standard with the device. Instead of using internal RC oscillator we are using a external crystal oscillator of 16 Mhz..

Because crystal is temperature invariant, so if temperature changes ,crystal frequency will not change. Which is not the case with the RC oscillator, the next is how the track are connected, you can connect two sensors one is attached at pin no. 16, both are PWM pins & these are with according to programming pin no.9 & 10 this is because your programming pin

start with zero.now this is our complete circuit,so you cannot have physical pin 0 on your IC's so as well as your first pin goes to reset then pin no. 2 & 3 are TXD & RXD respectively.Pin no.2 goes to RXD of the controller & pin no. 3 goes to TXD of controller resp. so in our program we are transmitting from the controller, what is the angle set on your servomotor, it should be minimum, so i.e. pin no.0 is a physical pin 2, then VCC as pin no.7 ground pin no.8, pin no.9 as crystal-2 & programming pins are lagging behind your actual physical pins. so you can see there is arduino map is on the internet, arduino map will map the all physical pins with the programming pins & we will find out that pin no.9 & 10 are the actual pin connected to the servo than we are using 7805 regulator because my controller require exact 5V supply & it must have a regulated power supply but there is another issue that servo require 800 mA of current & your 7805 can give only upto 500 mA ,so it was very necessary to put a bypass from your 7805 to the servo so actually servo is getting directly connected with your VCC & servo is rating it 6V, this 7805 will work 0.6V drop considering silicon, anything voltage above 5.6V so it becomes very necessary that you connect only 6V to circuit.

There is no other supply require to connect, because circuit 0 supply range only goes between 5.6V to 6V because 5.6V is the minimum requirement of the regulator & 6V is maximum supported by servo.the one is the positive of supply, so even if you are not using battery you should connect it to 6V voltage supply & below one is ground, so this two are the VCC & ground pins. Just remember VCC goes to pin no. 1 of 7805, & ground goes to center pin & third pin is the output pin, then TXD from the controller will going to RXD of FT232 IC, it a SMD, so we did not solder it & purchased it directly & FT232 is a FTDI chip which converts your normal USB port to RS232 standard port & the two resistor, the capacitor is a charge holding capacitor, it works on normal USB supply so there USB B-type cable A/D converter, on the other hand we have normal A-type cable.

So it's A to D converter & this FDDI record port is easily available in the market with following pin connections we just require ground, because ground logic should be w.r.t. some voltage, so ground of both circuit shoud be shorted & RXD of FDDI will be connected to TXD of controller, so that you will keep on getting angles from the controller to the computer so finally you use to the computer & then we will explain about computer programming, so that was with the hardware,we have brightness control knob at the webcam, which we have to rotate while focusing.

B. SOFTWARE

This is the matlab explanation for 2D to 3D object converter. In the first line all are right now commented because we are doing some testing but these all are actual lines we will be using for "i" equal to one in steps of 1 upto 20 images [for i=1:1:20], because we are taking 20 images. [s=serial('COM20','Baud rate',9600)] This is the same baud rate which we sent into your Arduino program i.e. for ATMEGA8, so this two baud rate should match, then 's' is equal to inputBuffersize(s,'InputBufferSize',1) as 1 because we are dealing just first bit i.e. send by the controller. Then it will keep on if it is sending 15 degree, it will through only 1 & then it will read only 5, something like that, then if you want to store higher degree i.e. upto 180 degree, you have to change the buffersize '1' to '3'. Then "Set(s,'FlowControl','none')", so we don't require any flowcontrol. "set(s,'Parity','none')" so you just hve to check serial port configuration on internet or check the parity bits it will be odd or even, how the flow control takes place, this all comes under theory, then databits are '8'[set(s,'databits',8)], because we are sending the 8-bit data, one stop bit we require[set(s,'stopbit',1)] & timeout is 100sec, also if you don't receive anything then the port will be timed out[set(s,'Timeout',100)]. Then "fopen(s)" will open this start serial port, then "[i = fscanf(s)]", it will read fromwhatever coming from serial port, "fscanf" is the command to read a serial port, then "[j = str2num(i)]" because you will read this "i" as a string, so you have to convert the string to a number & store it in "j". Then "[disp(j)]" is display "j" so you will come to know that '1' had come, '2' had come, '3' had come. We are sending 3,6,9,12,15,…… like that it goes, after 20 reads atleast you should save upto all 10.jpg, because first angle will go as 3,6,9,….. then it will save as 3.jpg,6.jpg,9.jpg respectively. So next is 12- so it will save it as 1.jpg, for 15- so it will save it as 1.jpg, because inputBufferSize is '1'. If you want to store 12.jpg, then make inputBufferSize as '2'. Then we go ahead, after you display that no. "[disp(j)]", you close "[fclose(s)]" , delete "[delete(s)]", clear"[clear s]", so your now value is stored in "j" that much we require only from the serial communication.

Next we move on to the camera communication-Now for camera communication, we first create a object name "vid" [vid=videoinput('winvideo',1)], so if your laptop is having 2 cameras, then you have to make it "2" instead of "1" i.e. "[vid = videoinput('winvideo',2)]" or default camera,if setup hen you have to make it 3, so once you do that 'winvideo' or 'matroax', these two types of cameras are available in the market. Rightnow we have purchased is 'winvideo'

type camera is there [set(vid,'TriggerRepeat',Inf)], because we require infinite triggering, camera should be continuously ON, so triggerrepeat 10 is send to infinity. "[vid.FrameGrabInterval=2]", so after every 2m, we keep on rubbing the frames from the camera & we can change it to 1,3,5,7,…. Whatever we want, that basically controls the speed. Then "vid_src=getselectedsource(vid)" so this vid is passed as a source to vidsource, then "set(vid_src,'Tag','2D to 3D modelling')" i.e. the name given to the project. Then "start(vid)", this will start up the video, 'while(vid.FramesAcquired < = 5)', so we are starting the video acquiring 5 frames as soon as it hits 5 frames, 'data = getdata(vid,2)', now actual image is stored in the vid object second matrix,because vid object is 3-dimensional matrix, i.e. layer-1,layer-2 & layer-3.The matrix at the backside is image, so that we will expect 2 is stored as 'diff_im', whose data is "(:,:,:,1)", so this is your video, so video is a 4-dimensional object. It has x,y,z as coordinates of picture has & it also has time information, so we have captured & that is stored in "diff_image" & then end so we will actually finish with the video camera communication.

Once you end this, your image is stored in "diff_im", so rightnow we had acquired 20 j's, this all are inside the loop & for all 20, we are actually capturing videos of 5frames each. Now camera works at 15 frames/sec, so around 1.333sec is required to capture these many frames,so we are running the loops 20 times , so each 20 loops will require 5 frames each , so total 100 frames we are capturing, we are using 20 frames only.to capture 100 frames at 15 FPI will take somewhere around 6 to 7 sec only & we are trying to keep as minimum time as possible then we " stop(vid)","delete(vid)".Now we will convert g as num2 string "(g=num2str(j))" , this is exactly opposite to string 2 now.Now the number is 'g' which we are making it string."path=stract(g,'.jpg'), we have to store the images, before we came out of the loop , because if loop ones get completed before string the image then this image data will be overwritten.so we guess make it as g.jpg, so when controller sends '1', it will become 1.jpg, it will send '2' will become 2.jpg, it will send '3' then 3.jpg & so on.Basically the main concept is to make the file with file name as 'angle.jpg', whatever angle it is taking , then we are writting 'diff-image', which cintains your actual captured image & path is where this image will be stored "(imwrite(diff-im-path))" & then 'end' which will indicate the ending of For loop, means it will run 20 times, in which we will get at least 10 images.If you are not getting 10 images,you will increment the loop to 20,30,...,etc.

Then "I=ycbor2rgb(diff-im)", all will be taken by the camera in ycbcr, so ycbcr to rgb will convert this image. Next time is "figure()", "imshow(I), so the images which you had captured will be seen here. Then "b=rgb2gray(I)", because we want the colour of the object & so on, so we convert the object from RGB to gray. we stored it in b, then we take a backup in A"(a=b)". We have resize a to size "240*320"(=imresize(a,(240.320)))", so this is the actual image size now because your camera is supporting that much of resolution but you can obviously scale to "600*800", information don't increase because after scaling also your pixel just repeat , may be what type of zooming you are using then "a=edge(a,'canny',0.4)", in this 0.4 factor is passed & it is different for different type of edge detections like canny, filtering,normal,etc. & then what is specific about canny edge detection is all algorithms will come here . Now "a=bwareopen(a'20)" it is an image opening , it is a part of "Morophology Chapters" used for opening the image.Now "a=bwareopen(a,20)' means part of the image below 20 pixels will be black out & only the image will be retained then 'figure(35)' is used for showing that fig.as fig.(35) then "m,n)=size(a)" for k=1, then we are finding all the 1's in the image then by following For loop follows :

```
for i=1:1:m

for j=1:1:n

if (a(i,j)=1)

x(k)=i;

y(k)=j;

k=k+1;

end
```

This means we are scanning & where '1' gets hitted , its 'x' & 'y' matrix , so this is how you read all white pixels in the image.

This is the software explanation for 2D to 3D controller coding. First line includes “<servo.h>”, i.e. the standard servo library available on the internet & we just include it to our program to rotate our servo. Then we create one object called “Myservo”, i.e. servo name is “Myservo” , that object is created, then in potpin was not required also, so we put comment overhere, so “int val”, so we define it is a C-Programming, so we define one int as “val”, then we

go to set up. Now setup is the part of program,which happens only once. Its like a initialization,so in that we say “Serial.begin(9600)” that means your baud rate which ever we seen in the matlab9600 this should be seen, then “Myservo.attach(9)” as it was already explained in the hardware, pin no.9 in the programming means pin no.15 on hardware,so this is where your hardware is attach .

The “void ()” loop , all embedded system works like this only, after this initialization loop will work, so loop is the part of the program which runs again & again. Here we have a for

loop, val going from 1 (val=1), till val is less than 180 (val<180),val is equal to val plus three (val=val+3). If you want to increase this angle, you can put 5,6,10,30,….etc. in place of three. Then “Myservowrite(val)”, so it will directly write that value on servo, means it will write 0,3,6,9 & so on. Then “delay(1500)” i.e. delay of 1500ms to capture the photo that means after 1.5sec, it will rotate by three degree. Then “Serial.println(val)”, so we are printing the value on to your this FTDI chip, which will get the value from micro-controller, whatever you have written for servo & this loop will continue till it reaches 180 degree, after becoming 180 degree, it will automatically reset on “val=1” , because the bigger loop will run, so it will be continue. So you can see your servo will go on rotating at 180 degree & when your servo reaches 180 degree completely, immediately it will come to 1(val=1) . In this way the complete coding works.

CHAPTER 4: IMPLEMENTATION

A. HARDWARE:

The hardware requirements are:

1. Computer
2. Microcontroller (Atmega 8)
3. Stepper motor and motor controller (ULN2003)
4. RS232 cable
5. Webcam

The block diagram of the hardware setup is given below in Fig 4.1

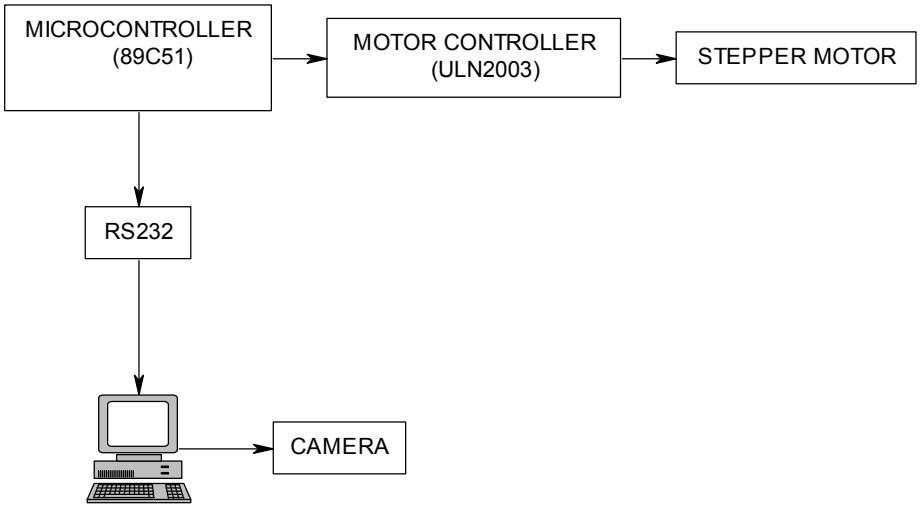
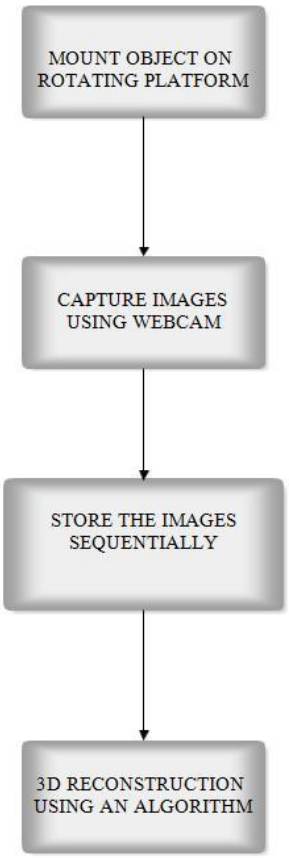
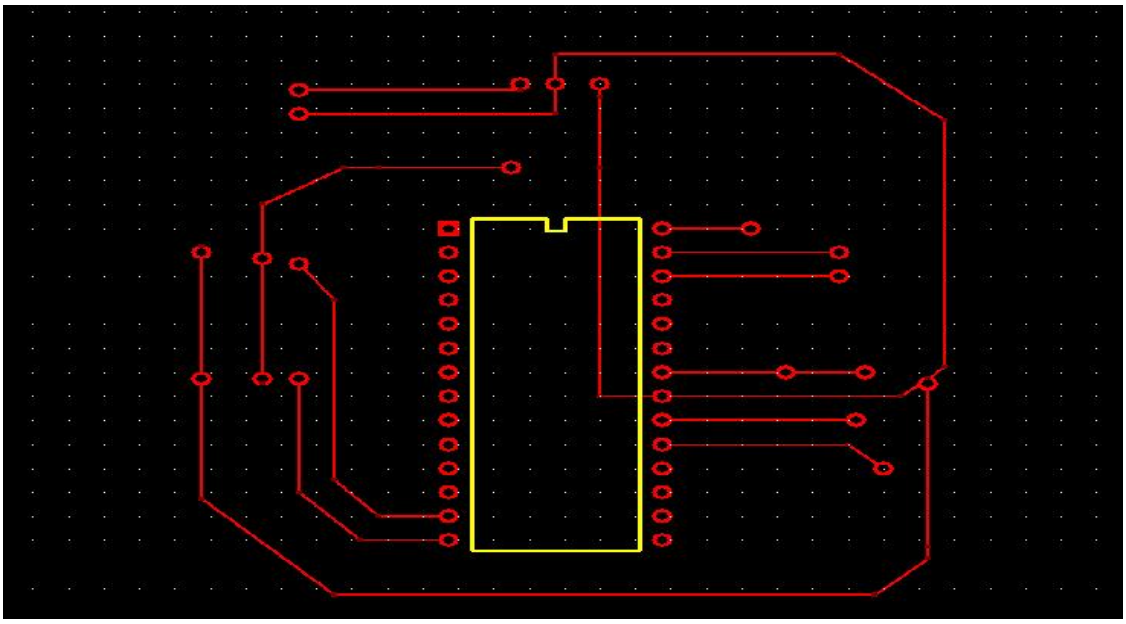


Fig 4.1

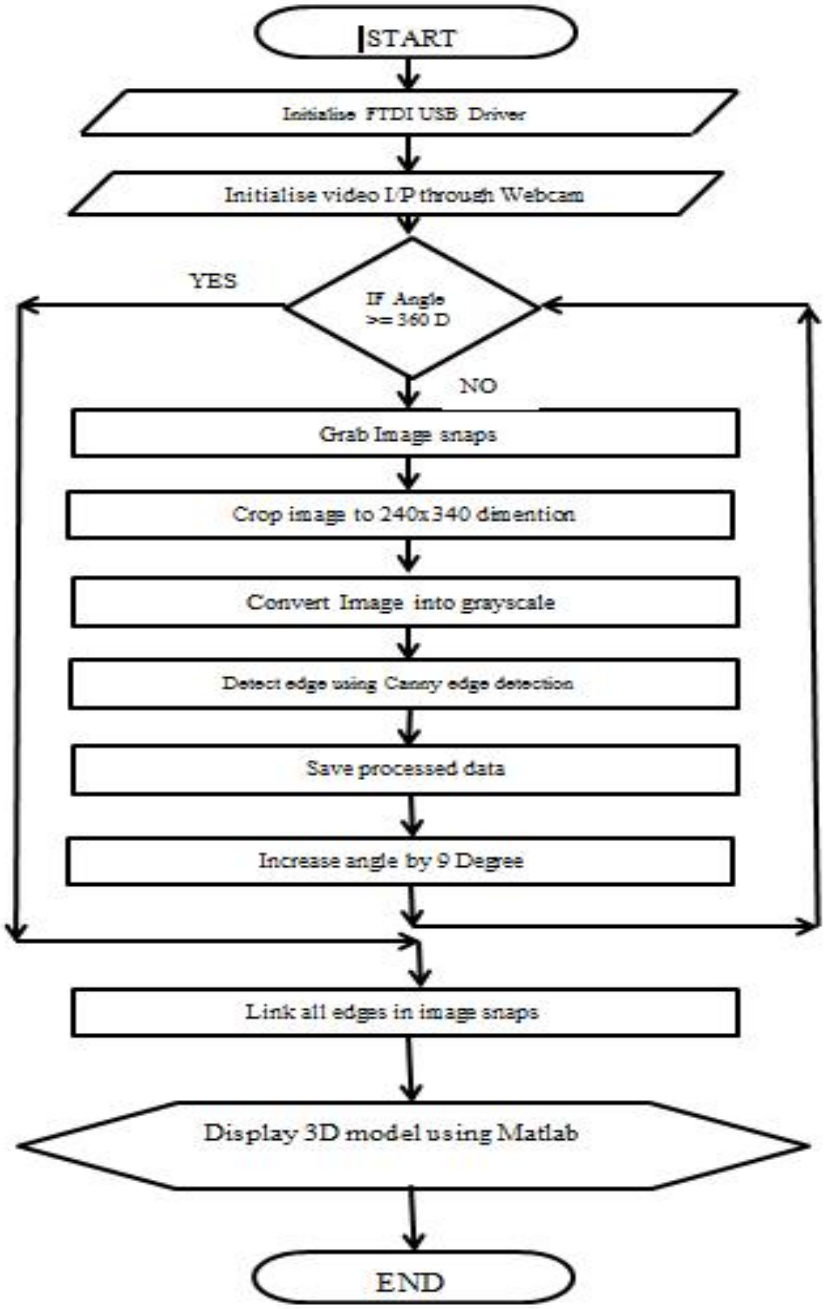
FLOWCHART:



LAYOUT:



ALGORITHM :



HARDWARE DESCRIPTION:

1. CAMERA :

The Camera used for this project is **Tech-Com SSD-350**
Night Vision PC Camera with microphone, 40 Mega Pixels,Driver Free Plug & Play.

Specification:

- Image Sensor: 1/7" CMOS sensor
- Image Resolution: 1280x960, 1024x1280, 1600x1200, 4032x2034
- Frame Rate: Up to 30fps
- Image control: Brightness, Contrast, Hue, Saturation, Gamma, White Balance
- Image Flip: Horizontal, Vertical
- Monitor type: CRT, LCD
- Environment: Indoor, Outdoor
- Focus distance: 4Cm~ infinity
- Lens View angle: 54 Degree
- I/O Interface: USB 1.1, 2.0
- Image Format: RGB24, 1420
- Power Consumption: 160MW Typical
- Operating System: Windows/ 2000/ ME/XP /Vista

Features:

- 300k pixels resolution, frame rate upto to 30 fps (40 Megapixels by Software)
- Excellent Quality & fashionable styles
- True Plug & play easy USB Interface
- High-quality CMOS sensor
- Clear and sharp still-picture & motion video capturing
- Ideally designed to work well with both Laptops & Desktop computers
- Adjustable lens for accurate image shooting
- Includes variety of image control
- Auto white balance & exposure

2. MICROCONTROLLER

We are using microcontroller Atmega 8

Description

The Atmel ®AVR® ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By

executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

The Atmel ®AVR® core combines a rich instruction set with 32 general purpose working registers.

All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8 provides the following features: 8 Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1 Kbyte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port and five software selectable power saving modes. The Idle mode stops the CPU while allowing The SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping.

The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

The main advantages of Atmega8 are as follows:-

- High-performance, Low-power Atmel ®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
- High Endurance Non-volatile Memory segments
  – 8Kbytes of In-System Self-programmable Flash program memory
  – 512Bytes EEPROM
  – 1Kbyte Internal SRAM
  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  – Data retention: 20 years at 85°C/100 years at 25°C(1)
  – Optional Boot Code Section with Independent Lock Bits
  – In-System Programming by On-chip Boot Program
  – True Read-While-Write Operation
  – Programming Lock for Software Security

C. Servo Motors

A **servomotor** is a rotary actuator that allows for precise control of angular position. It consists of a motor coupled to a sensor for position feedback, through a reduction gearbox. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

FEATURES

- Small Compact Size
- 2 kg.cm torque
- Rotation angle: 180 degrees
- 50gm weight
- Same size motor available in various rpm

SERVOMOTORS VS. STEPPER MOTORS

Servomotors are generally used as a high performance alternative to the stepper motor. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate. This lack of feedback though limits their performance, as the stepper motor can only drive a load that is well

within its capacity, otherwise missed steps under load may lead to positioning errors. The encoder and controller of a servomotor are an additional cost, but they optimise the performance of the overall system (for all of speed, power and accuracy) relative to the capacity of the basic motor. With larger systems, where a powerful motor represents an increasing proportion of the system cost, servomotors have the advantage.

D. FTDI : USB TO SERIAL CONVERTER

FEATURES

Single chip USB to asynchronous serial data transfer interface.

• Entire USB protocol handled on the chip - No USB-specific firmware programming required.

• UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd/even/mark/space/no parity.

• Fully assisted hardware or X-On / X-Off software handshaking.

• Data transfer rates from 300 baud to 3 Mega baud (RS422 / RS485 and at TTL levels)

• 256 byte receive buffer and 128 byte transmit buffer for high data throughput.

• FTDI's royalty-free VCP and D2XX drivers eliminate the requirement for USB driver development in most cases.

• In-built support for event characters and line break condition.

• New USB FTDIChip -ID™ feature.

• New configurable CBUS I/O pins.

• Auto transmits buffer control for RS485 applications.

• Transmit and receive LED drive signals.

• New 48MHz, 24MHz, 12MHz, and 6MHz clock output signal Options

• FIFO receives and transmits buffers for high data throughput.

• Adjustable receive buffer timeout.

• Synchronous and asynchronous bit bang mode interface options with RD# and WR# strobes.

• New CBUS bit bang mode option.

• Integrated 1024 Bit internal EEPROM for storing USB VID, PID and serial number

• Device supplied preprogrammed with unique USB serial number.

• Support for USB suspend and resume.

• Support for bus powered, self-powered, and high-power bus powered USB configurations.

• Integrated 3.3V level converter for USB I/O.

• Integrated level converter on UART and CBUS for interfacing to 5V - 1.8V Logic.

• True 5V / 3.3V / 2.8V / 1.8V CMOS drive output and TTL input.

• High I/O pin output drive option.

• Integrated USB resistors.

• Integrated power-on-reset circuit.

• Fully integrated clock - no external crystal, oscillator, or resonator required.

• Fully integrated AVCC supply filtering

• UART signal inversion option.

• USB bulk transfer mode.

• 3.3V to 5.25V Single Supply Operation.

• Low operating and USB suspend current.

• Low USB bandwidth consumption.

• UHCI / OHCI / EHCI host controller compatible

• USB 2.0 Full Speed compatible.

• -40°C to 85°C extended operating temperature range.

• Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS complian

E. REGULATOR

FEATURES

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

TO-220

D-PAK

1. Input 2. GND 3. Output

HARDWARE COMPONENT COST:

1) ATmega8 Microcontroller Rs.65/-

2) 16 MHz crystal Rs.5/-

3) 10K resistor Rs.1/-

4) USB to RS232 Converter FTDI breakout board Rs.350/-

5) Servo motors Rs. 900/-(Each)

6) 7805 Regulator Rs. 10/-

7) Bug strip Male Rs.25/-

9) USB B-Type Cable Rs.60/-

10) One way Jumperwire 1 point relimate Rs.3/-(Each)

11) 100uF/16V capacitor Rs. 2/-

12) USB 8MPixel camera with light Rs.800/-

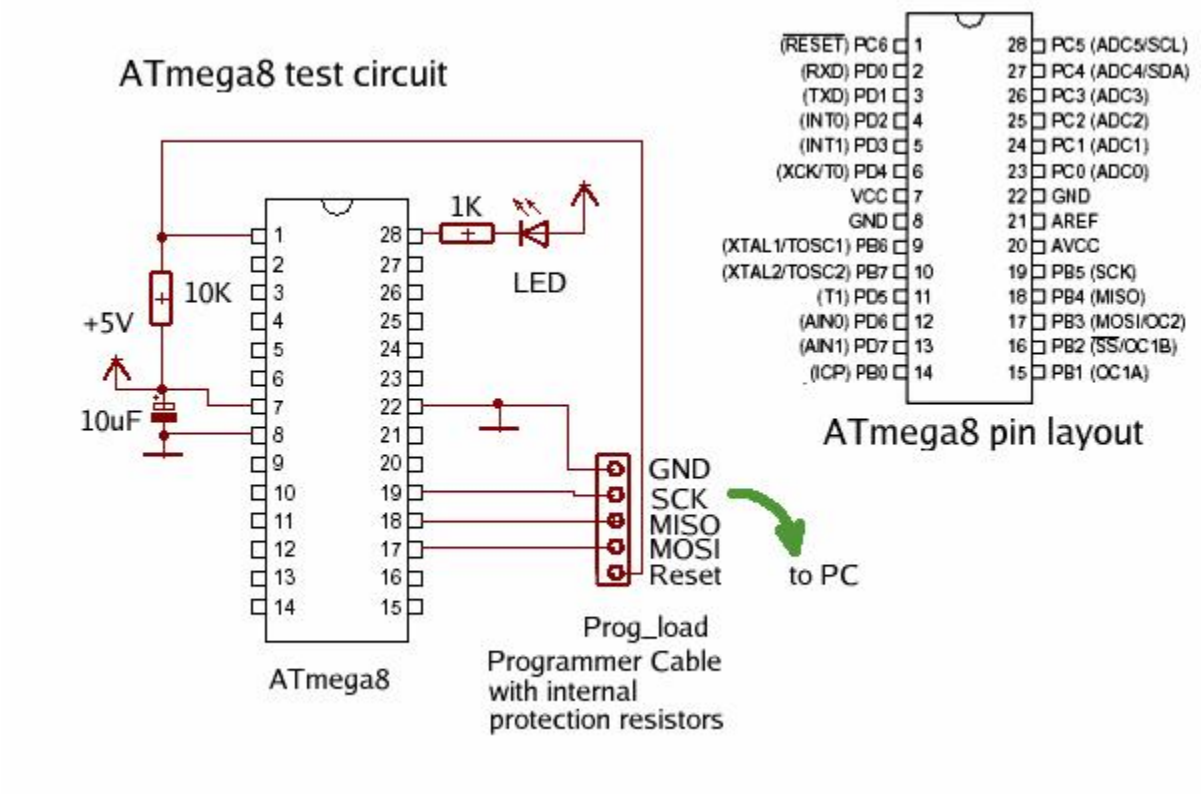
13) 28pin IC base Rs.3/-

14) Copper Clad Rs.20/-

15) Photo Paper Rs.14/-

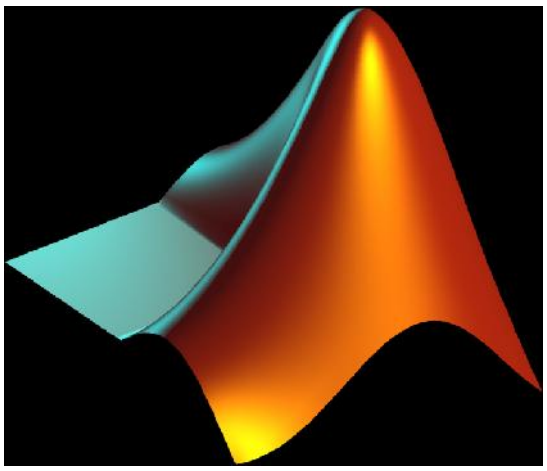
16) Reset Switch PCB mountable Rs.10/-

CIRCUIT DIAGRAM:



B. SOFTWARE:

1. MATLAB



MATLAB (**mat**rix **lab**oratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing

capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

Functions used in Matlab code:

- **VIDEOINPUT**: Create video input object.

obj = videoinput(adaptorname,deviceID) constructs

a video input object obj, where deviceID is a numeric scalar value that identifies a particular device available through the specified adaptor, adaptorname. Use the imaqhwinfo(adaptorname) syntax to determine the devices available through the specified adaptor. If deviceID is not specified, the first available device ID is used. As a convenience, a device's name can be used in place of the deviceID. If multiple devices have the same name, the first available device is used.

- **BWAREAOPEN**: Morphologically open binary image (remove small objects).

BW2 = bwareaopen(BW,P) removes from a binary image all connected components (objects) that have fewer than P pixels, producing another binary image, BW2. The default connectivity is 8 for two dimensions, 26 for three dimensions, and conndef(ndims(BW),'maximal') for higher dimensions.

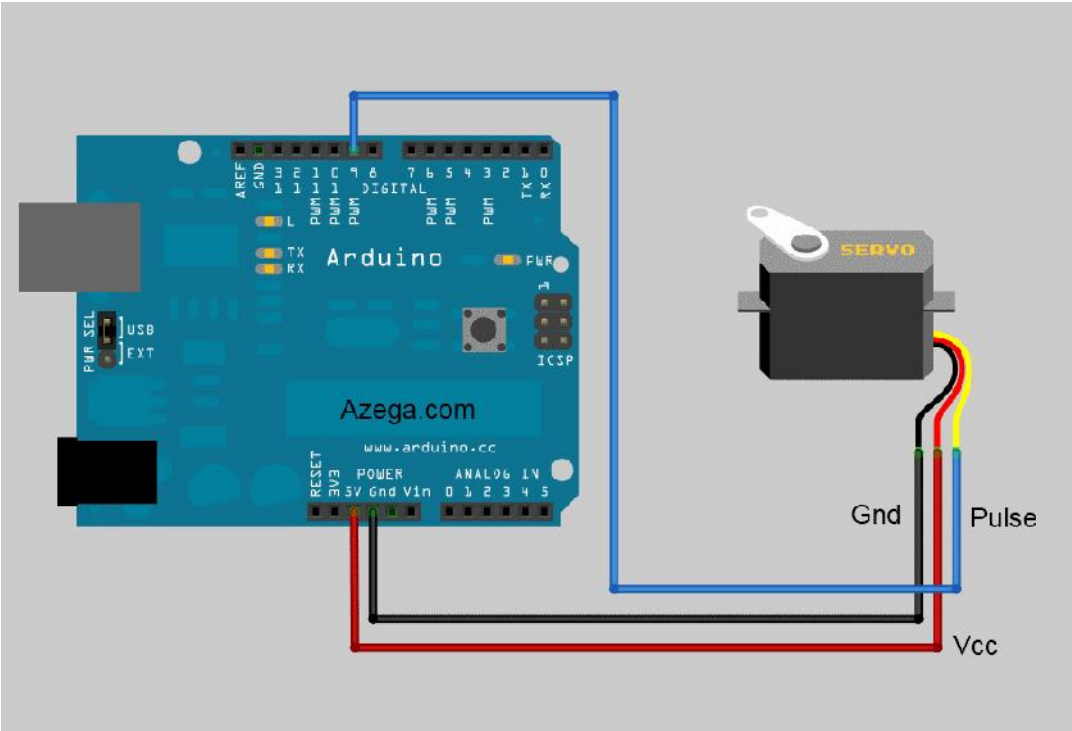
- **EDGE**: Find edges in intensity image.

BW = edge(I,'canny',thresh) specifies sensitivity thresholds for the Canny method. thresh is a two-element vector in which the first element is the low threshold, and the second element is the high threshold. If you specify a scalar for thresh, this value is used for the high threshold and 0.4*thresh is used for the low threshold. If you do not specify thresh, or if thresh is empty ([]), edge chooses low and high values automatically.

- **IMCROP**: Crop image.

I = IMCROP creates an interactive image cropping tool, associated with the image displayed in the current figure, called the target image. The tool is a moveable, resizable rectangle that is interactively placed and manipulated using the mouse. After positioning the tool, the user crops the target image by either double clicking on the tool or choosing 'Crop Image' from the tool's context menu. The cropped image, I, is returned. The cropping tool can be deleted by pressing backspace, escape, or delete, or via the 'Cancel' option from the context menu. If the tool is deleted, all return values are set to empty.

2. ARDUINO



Arduino is a open-source single-board microcontroller, descendant of the open-source Wiring platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board input/output support. The software consists of a standard programming language compiler and the boot loader that runs on the board. Arduino hardware is programmed using a Wiring-based language (syntax and libraries), similar to C++ with some slight simplifications and modifications, and a Processing-based integrated development environment.

Current versions can be purchased pre-assembled; hardware design information is available for those who would like to assemble an Arduino by hand. Additionally, variations of the Italian-made Arduino—with varying levels of compatibility—have been released by third parties; some of them are programmed using the Arduino software. The Arduino project received an honorary mention in the Digital Communities category at the 2006 Prix Ars Electronica

An Arduino board consists of an 8-bit Atmel AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. (When used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.)

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs. These pins are on the top of the board, via female 0.1 inch headers. Several plug-in application shields are also commercially available.

The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board to be plugged into solderless breadboards.

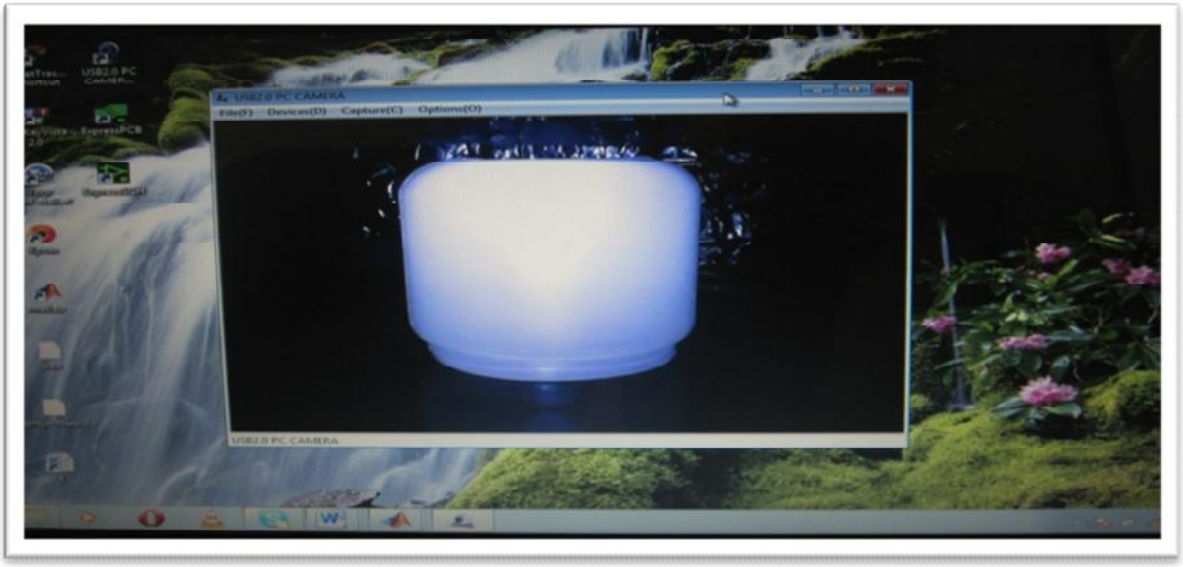
CHAPTER 5: RESULT

STEP 1: Complete setup

We have implemented complete setup of our project as shown in figure below.we have interface camera and FTDI USB with computer..we have connected microcontroller to computer through FTDI USB to serial converter.servo motor is connected to microcontroller using regulator which regulates power supply to servo motor through external 6V power supply.

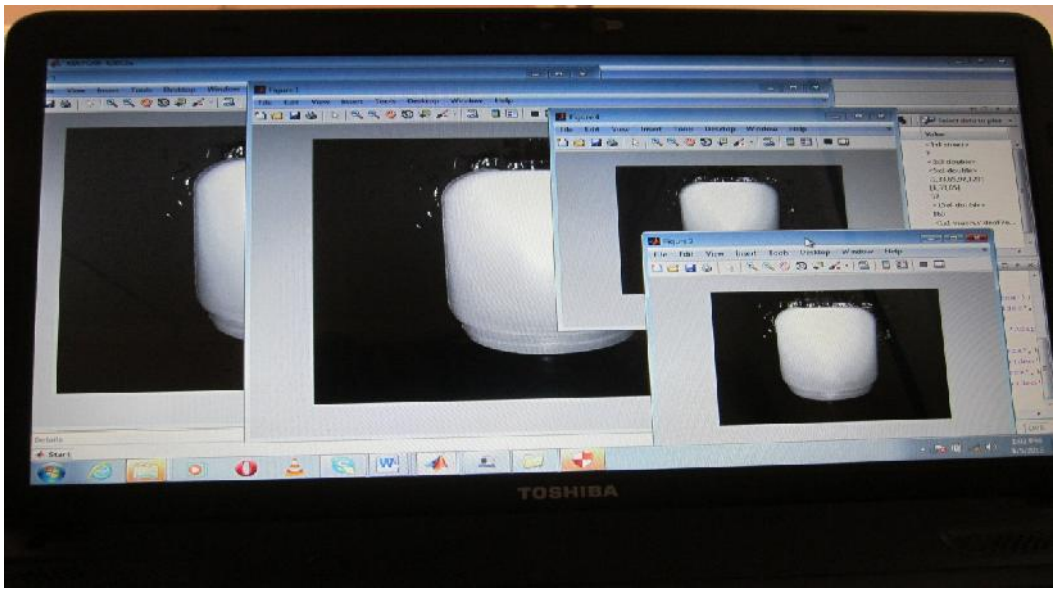
STEP 2:Initialisation of Webcam

We have initialised camera and as shown in figure an output of camera is displayed on monitor. Matlab program will capture the images of the objects which we are trying to convert it into 3D at different angles.



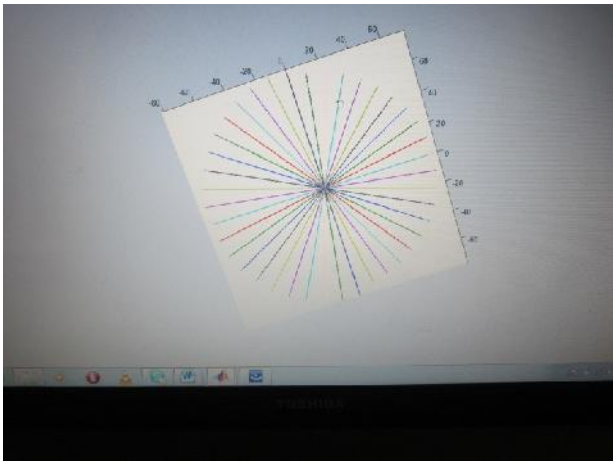
STEP 3: Processing of program in Matlab

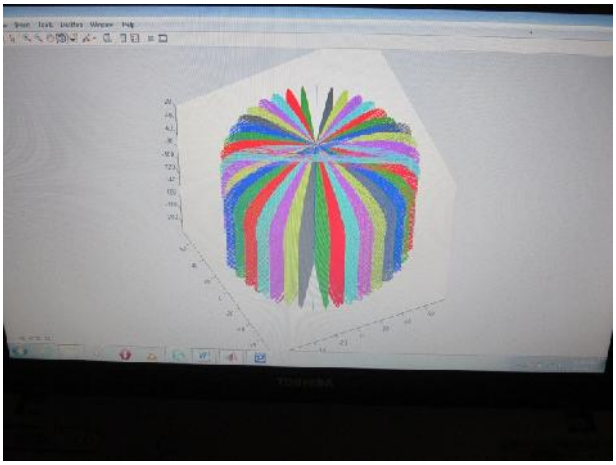
The images of object at different angles are as shown in figure below while processing of Matlab program. The number of captured images depends on the angle we have considered.

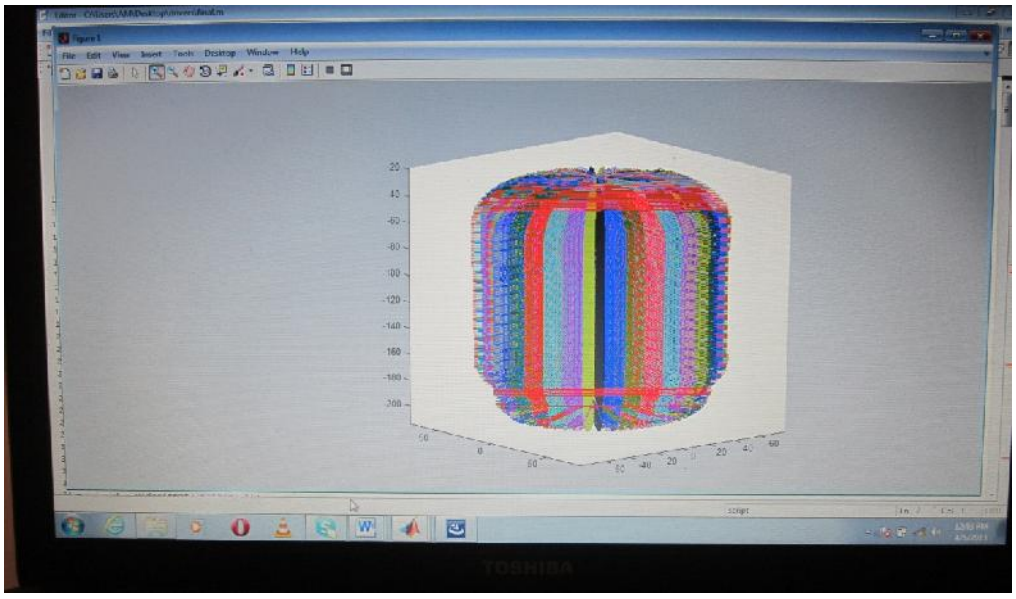


STEP 4: Final Output

The figures shown below displays our final 3D output.



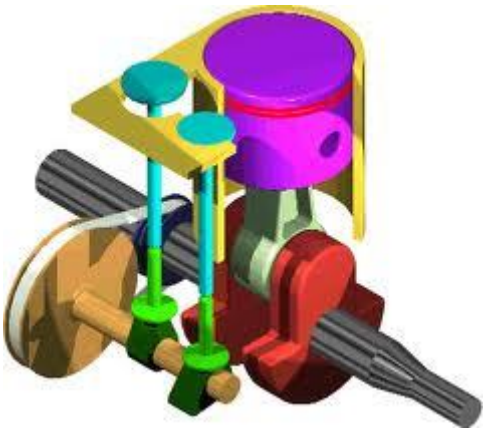




CHAPTER 6: ADVANTAGES AND FUTURE SCOPE

1. BENEFITS:

- Faster product design (roughly 45% faster on average)
- Beat your competition to market
- Automatic flattening of sheet metal parts (with bend allowance)
- More effective communication with suppliers/customers
- Visualize more 'what-if' scenarios during the design process
- The ability to create renderings and animations for design proposals or reviews
- More effective internal design reviews



- Generation of virtual prototypes allows non-CAD people to participate in the process

- Easily incorporate late design changes
- Test and validate your designs to reduce costs from quality problems, errors, ECO's
- Reduce the need and cost of physical prototypes
- Automatic Bills of Materials
- Data management to organize and manage your design data
- Helps to standardize on detailing and drafting practices
- Automate your design process and increase speed and accuracy of output and response to customers
- Allow non-technical personnel such as sales department (and even customers) to quote, specify and configure product whilst maintaining your design & engineering integrity
- *Flexibility,* ability to change angles or animate images with quicker rendering of the changes;
- *Ease of rendering,* automatic calculation and rendering photorealistic effects rather than mentally visualizing or estimating;
- *Accurate photorealism,* less chance of human error in misplacing, overdoing, or forgetting to include a visual effect.
- Providing a low cost alternative to the existing techniques
- simple and portable
- 3D scanners are used by the entertainment industry to create digital 3D models for both movies and video games. In cases where a real-world equivalent of a model exists, it is much faster to scan the real-world object than to manually create a model using 3D modeling software. Frequently, artists sculpt physical models of what they want and scan them into digital form rather than directly creating digital models on a computer.

2. LIMITATIONS :

Existing System :

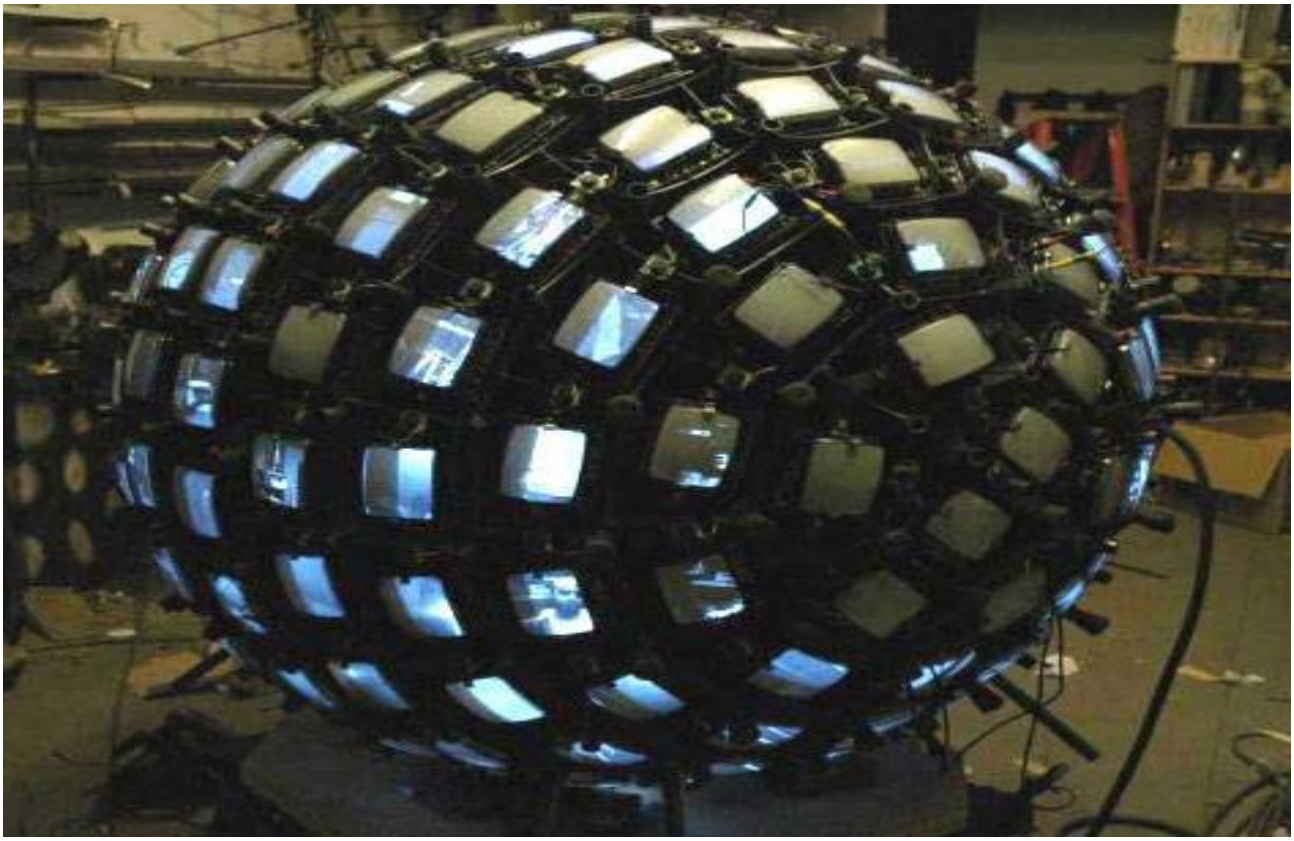


Fig. 6.3

This is an example of the current existing system.this type of systems are used by microsoft ,different channels,offices.

Problem Formulation :

- This system is having different cameras at difrent angles.they put the object inside the sphere.they take picture of that object at different angles with the help of cameras.then they extract that image & combine them to form the final 3d image.
- This process is very lengthy and costy too.degrees till 360 degrees.
- The range of laser triangulation 3D scanner is very small. It can only be used to scan small objects. Larger structures like buildings and geographic structures can't be scanned using this method.

3. APPLICATIONS:

A. MEDICAL INDUSTRY

3D image conversion technology can greatly enhance the quality and efficiency of current endoscopic surgical procedures, which rely on 2D images.

3D ultrasound is a medical ultrasound technique, often used in obstetric ultrasonography (during pregnancy), providing three dimensional images of the fetus.

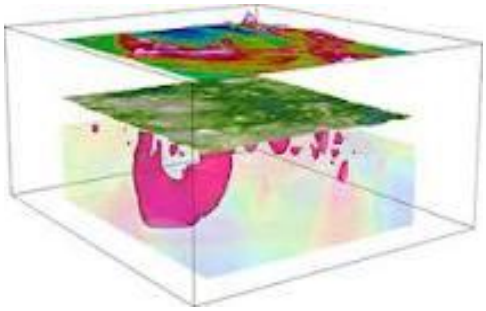
There are several different scanning modes in medical and obstetric ultrasound. The standard common obstetric diagnostic mode is 2D scanning. In 3D fetal scanning, however, instead of the sound waves being sent straight down and reflected back, they are sent at different angles. The returning echoes are processed by a sophisticated computer program resulting in a reconstructed three dimensional volume image of fetus's surface or internal organs, in much the same way as a CT scan machine constructs a CT scan image from multiple x-rays. 3D ultrasounds allow one to see width, height and depth of images in much the same way as 3D movies but no movement is shown.

B. VIDEO GAME INDUSTRY:

3D scanners are used by the entertainment industry to create digital 3D models for both movies and video games. In cases where a real-world equivalent of a model exists, it is much faster to scan the real-world object than to manually create a model using 3D modeling software. Frequently, artists sculpt physical models of what they want and scan them into digital form rather than directly creating digital models on a computer.

C. ARCHITECTURE INDUSTRY AND EARTH MORPHOLOGY:

With the help of this concept we can imagine 3D model of building , bridges , malls, etc. It gives us rough idea about the space requirement for the total construction. It can also be used for perfect morphing of our earth for various scientific studies and research.



4. FUTURE WORK

- The accuracy can be improvised by using high sensitive cameras.
- An infrared camera can also be added in order to calculate the depth and also to avoid reflection from shinny objects which cases noise in optical cameras
- The code can be further improved to map the actual image on to the 3d model, thus giving it a realistic look.

The current proposed model is merely a prototype and is not very user friendly. Therefore its use will be restricted to engineers and other technicians. To increase the scope of its marketability the whole unit can be integrated under one central control panel just connected to the pc via a USB port. The control of the rotating platform and the camera settings can be integrated at one user friendly interface.

The proposed algorithm does not account for surface irregularities. Therefore objects with notches and other asymmetric irregularities are not successfully captured. Advanced image processing algorithms which can figure out the dent on the surface of a 2D image by the shadows formed and the light pattern can be integrated into the current algorithm. Thus a more accurate model can be created reducing the limitations on the general shape of the object that can be modeled using this project.

The micro controller used can be changed to one with a re writeable flash card. Therefore the resolution settings [degree of rotation in one step] can be easily changed as required. This makes the unit more versatile as compared to the one with a read only micro controller in which the settings cannot be changed once it is burned onto the memory.

To keep the computational requirements minimum the current algorithm reconstructs a 3D image from a relatively fewer number of images. This is a tradeoff between the resolution and the time required to build the image. If a suitable algorithm can be developed which reduces the number of calculations required, more images can be used for reconstruction therefore increasing the accuracy of the 3D model in the same time span.

CONCLUSION

In our project we are using single camera & rotating platform. In existing system, they are using 360 cameras, each camera at every angle starting from 0 to 360 degrees. Thus we are providing 3D output, thereby reducing the cost of existing system almost by 360%.

As we are using single camera & rotating platform, thus we are nothing but using Cylindrical system. Our system is definitely better than Cartesian system. In existing system, they are using 360 cameras & we are using single camera thus our 3D output is not as good as Spherical system.

Speed of our project depends upon the servo motor speed & frames per second of the camera. In existing system as they are using 360 cameras thus the speed of 3D output of existing is faster than our system.

Our system is simple & portable whereas existing system is complex & bulky.

REFERENCES

[1] Vicenzo Lippiello and Fabio Ruggiero. Surface Model Reconstruction of 3D Objects from Multiple Views.

[2] P. D. Kovesi. MATLAB functions for computer vision and image analysis. School of Computer Science & Software Engineering, The University of Western Australia. Shapelets Correlated with Surface Normals Produce Surfaces.

[3] Cesar Branco and Joao Costeria. A 3D Image Mosaicing System Using the Factorization Method.

[4] C.Daul, W.P.C.M. Blondel, A. Ben-Hamadou, R. Miranda-Luna, C. Soussen, D. Wolf, F. Guillemin From 2D towards 3D cartography of hollow organs.

[5] http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/index.html.

[6] http://www.acuitylaser.com/resources/principles-measurement.shtml

[7] http://en.wikipedia.org/wiki/3D_scanner

[8] http://courses.cit.cornell.edu/ee476/FinalProjects/s2009/dat38/Website/index.htm

[9] http://www.8051projects.net/stepper-motor-interfacing/stepper-motor-

[10]http://arduino.cc/en/Tutorial/HomePage

[11]http://www.math.ufl.edu/help/matlab-tutorial/

[12]http://en.wikipedia.org/wiki/3D_scanner

[13]www.alldatasheets.com

APPENDIX

ATMEGA8 MICROCONTROLLER Datasheet:

Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 8Kbytes of In-System Self-programmable Flash program memory
  - 512Bytes EEPROM
  - 1Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Six Channels 10-bit Accuracy
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V (ATmega8L)
  - 4.5V - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8MHz (ATmega8L)
  - 0 - 16MHz (ATmega8)
- Power Consumption at 4Mhz, 3V, 25°C
  - Active: 3.6mA
  - Idle Mode: 1.0mA
  - Power-down Mode: 0.5µA

ATMEL

8-bit AVR® with 8KBytes In-System Programmable Flash

ATmega8
ATmega8L

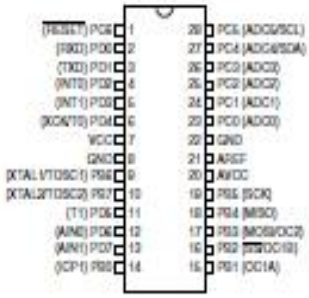
Summary

Rev. 2486ZS–AVR–02/11

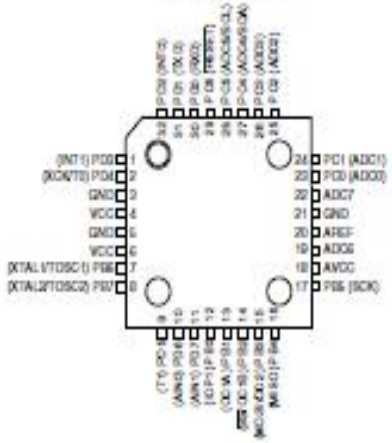
ATMEL

Pin Configurations

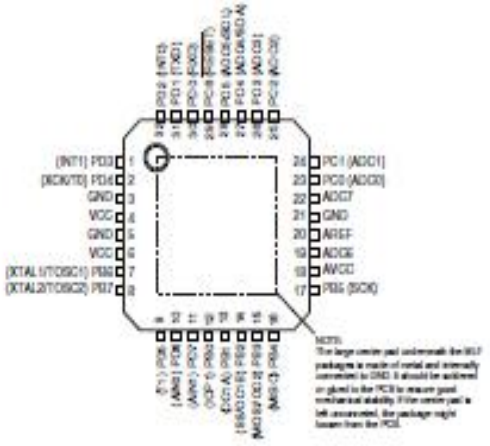
PDIP



TQFP Top View



MLF Top View

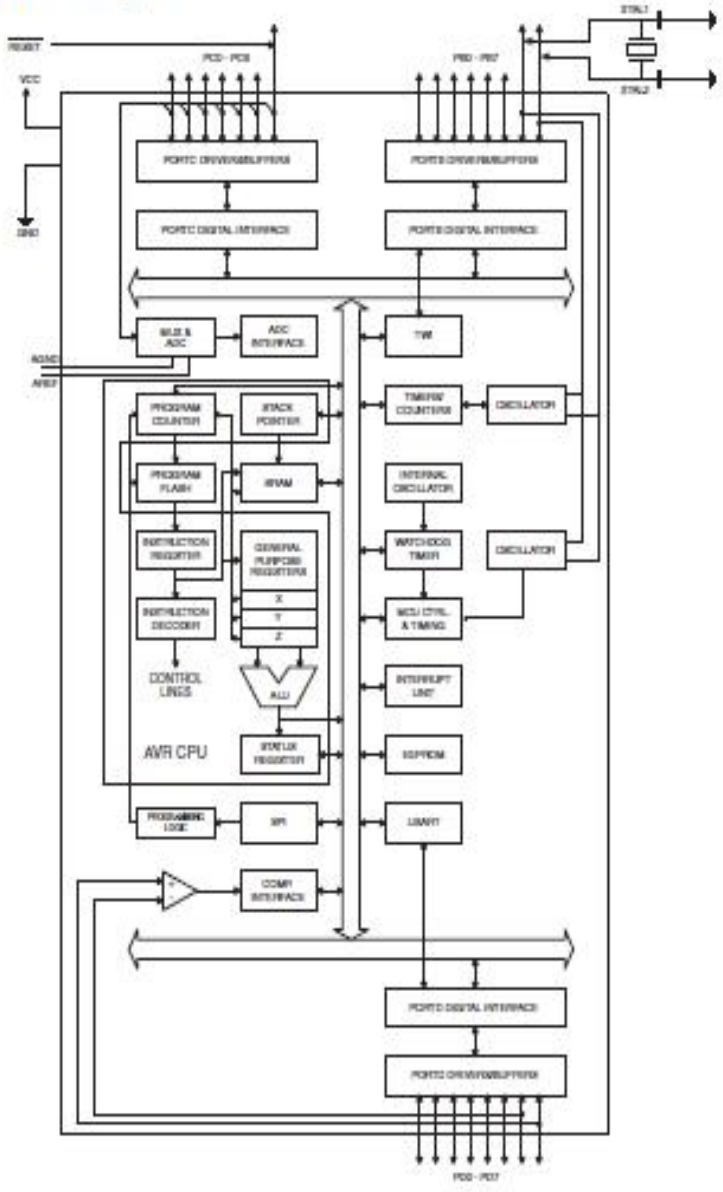


Overview

The ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. Block Diagram



Pin Descriptions

VCC — Digital supply voltage.

GND — Ground.

Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/TOSC2 — Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 58 and "System Clock and Clock Options" on page 25.

Port C (PC5..PC0) — Port C is an 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/$\overline{\text{RESET}}$ — If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated on page 61.

Port D (PD7..PD0) — Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega8 as listed on page 63.

$\overline{\text{RESET}}$ — Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 38. Shorter pulses are not guaranteed to generate a reset.

VOLTAGE REGULATOR DATASHEET

FAIRCHILD SEMICONDUCTOR®

www.fairchildsemi.com

KA78XX/KA78XXA

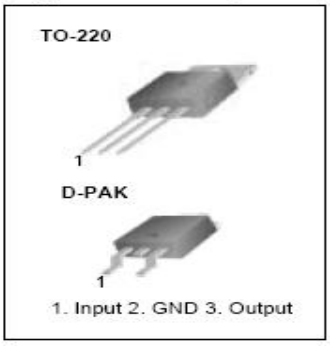
3-Terminal 1A Positive Voltage Regulator

Features

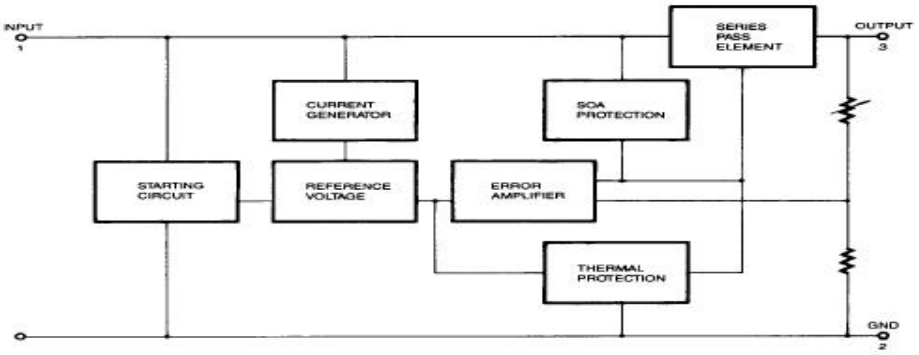
- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The KA78XX/KA78XXA series of three-terminal positive regulator are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



Internal Block Digram



Rev. 1.0.0

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range (KA78XX/A/R)	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (KA7805/KA7805R)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I =10V, C_I= 0.33µF, C_O=0.1µF, unless otherwise specified)

Parameter	Symbol	Conditions		KA7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ I_O ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	µV/V_O
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty is used.

FTDI USB USART Datasheet

FTDI Chip

Future Technology Devices International Ltd.

FT232R USB UART IC

The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receive and transmit buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

1 Typical Applications

- USB to RS232/RS422/RS485 Converters
- Upgrading Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU/PLD/FPGA based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA to USB data transfer
- USB Smart Card Readers
- USB Instrumentation
- USB Industrial Control
- USB MP3 Player Interface
- USB FLASH Card Reader and Writers
- Set Top Box PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
- USB Bar Code Readers
- USB Software and Hardware Encryption Dongles

1.1 Driver Support

Royalty free VIRTUAL COM PORT (VCP) DRIVERS for...

- Windows 98, 98SE, ME, 2000, Server 2003, XP and Server 2008
- Windows 7 32,64-bit
- Windows XP and XP 64-bit
- Windows Vista and Vista 64-bit
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Mac OS 8/9, OS-X
- Linux 2.4 and greater

Royalty free D2XX *Direct* Drivers (USB Drivers + DLL S/W Interface)

- Windows 98, 98SE, ME, 2000, Server 2003, XP and Server 2008
- Windows 7 32,64-bit
- Windows XP and XP 64-bit
- Windows Vista and Vista 64-bit
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Linux 2.4 and greater

The drivers listed above are all available to download for free from FTDI website (www.ftdichip.com). Various 3rd party drivers are also available for other operating systems - see FTDI website (www.ftdichip.com) for details.

For driver installation, please refer to http://www.ftdichip.com/Documents/InstallGuides.htm

1.2 Part Numbers

Part Number	Package
FT232RQ-xxxx	32 Pin QFN
FT232RL-xxxx	28 Pin SSOP

Note: Packing codes for xxxx is:

- Reel: Taped and Reel, (SSOP is 2,000pcs per reel, QFN is 6,000pcs per reel).

- Tube: Tube packing, 47pcs per tube (SSOP only)

- Tray: Tray packing, 490pcs per tray (QFN only)

For example: FT232RQ-Reel is 6,000pcs taped and reel packing

3 Device Pin Out and Signal Description

3.1 28-LD SSOP Package

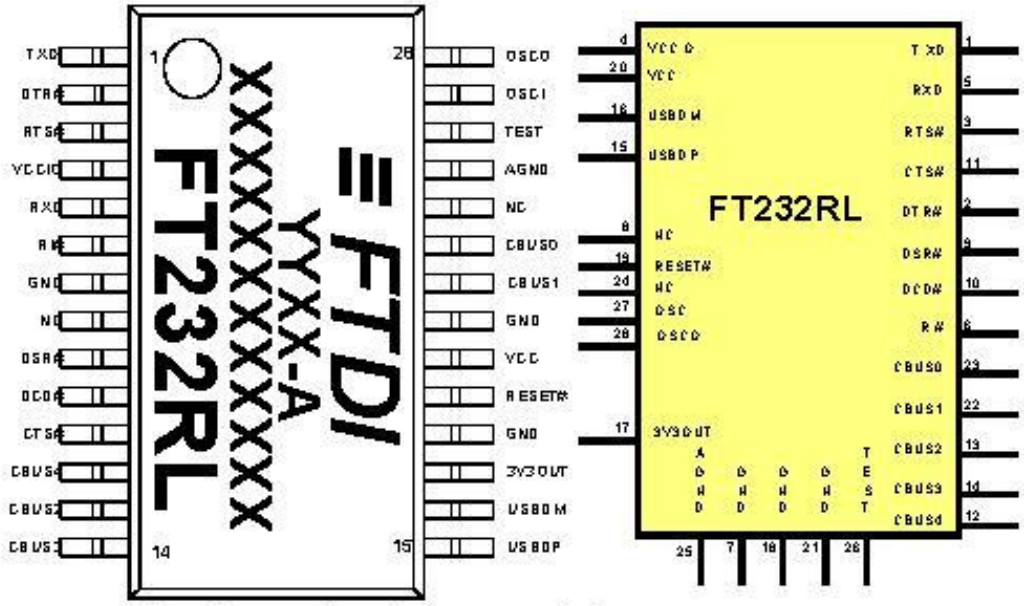


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

3.2 SSOP Package Pin Out Description

Note: The convention used throughout this document for active low signals is the signal name followed by a #

Pin No.	Name	Type	Description
15	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V.
16	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.1 USB Interface Group

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.

3.3 QFN-32 Package

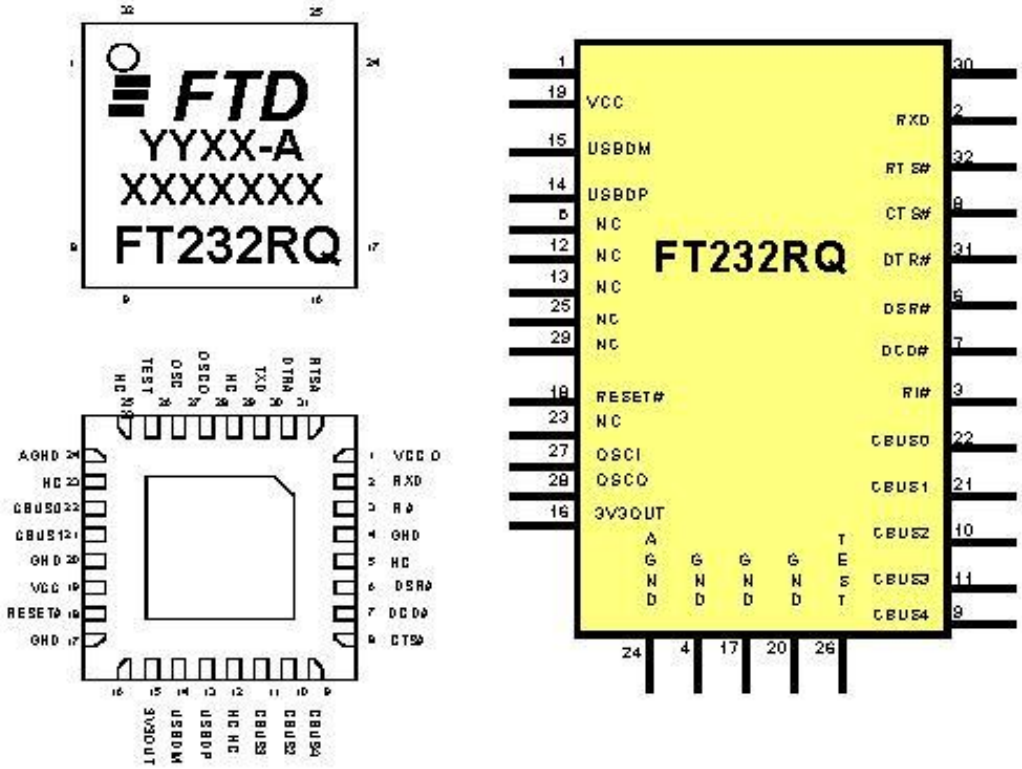


Figure 3.2 QFN-32 Package Pin Out and schematic symbol

3.4 QFN-32 Package Signal Description

Pin No.	Name	Type	Description
14	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to +3.3V.
15	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.5 USB Interface Group

Pin No.	Name	Type	Description
1	VCCIO	PWR	+1.8V to +5.25V supply for the UART Interface and CBUS group pins (2, 3, 6, 7, 8, 9, 10 11, 21, 22, 30, 31, 32). In USB bus powered designs connect this pin to 3V3OUT to drive out at +3.3V levels, or connect to VCC to drive out at +5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive out at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.

SERVO MOTOR DATASHEET

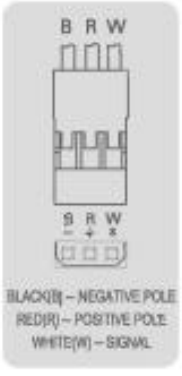
SPRINGRC®
www.springrc.com

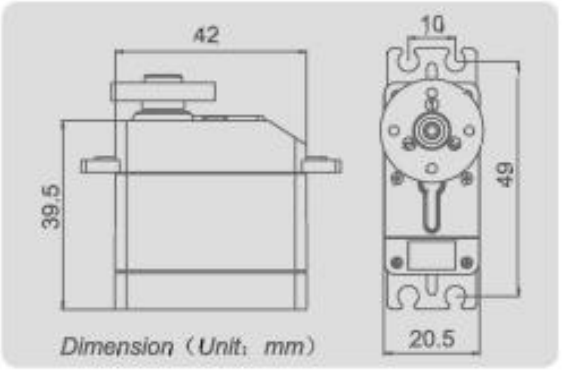
43R Servo(360° Rotation) Specification

Thank you for choosing Spring Model's product

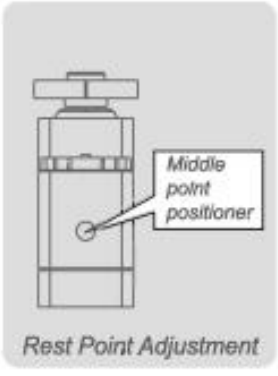
MODEL	TYPE	WEIGHT		4.8V			6V			DESCRIPTION	
		g	oz	SPEED	TORQUE		SPEED	TORQUE		GEAR	BEARING
				r/min	kg.cm	oz.in	r/min	kg.cm	oz.in		
SM-S4303R	Analog	44	1.55	60	3.3	45.8	70	4.8	66.7	1Metal Gear+ 4Plastic Gear	2
SM-S4306R		44	1.55	60	5.0	69.4	50	6.2	86.1	1Metal Gear+ 4Plastic Gear	2
SM-S4309R		60	2.12	58	7.9	109.7	49	8.7	120.8	Metal Gear	2
SM-S4315R		60	2.12	62	14.5	201.4	53	15.4	213.9	Metal Gear	2

▲ 43R Robot series servo controled via analog signal(PWM),stopped via middle point positiner.
▲ Standard interface(like JR)with 30cm wire.
▲ Rotation and Rest Point Adjustment:when analog signal inputs,servo chooses orientation according to impulse width.when intermediatevalue of impluse width is above 1.5ms, servo is clockwise rotation,conversely,anticlockwise.Rest point need use slotted screwdriver to adjust the positioner carefully.Servo stopped rotation when the input signal is equivalent to impluse width.
▲ Please choose correct model for your application.
Caution: Torque over-loaded will damage the servo's mechanism.
▲ Keep the servo clean and away from dust, corrosive gas and humid air.
▲ Without further notification when some parameters slightly amend for improving quality.





Dimension（Unit：mm）



Rest Point Adjustment