Augmented Reality

SIGB Assignment 4

Christoffer Stougaard Pedersen, Morten Roed Frederiksen, Sigurt Bladt Dinesen

{mrof,sidi,cstp}@itu.dk

IT-University of Copenhagen
SIGB, F2013
Dan Witzner Hansen & Diako Mardanbeigi
May 13, 2013

Contents

1 Introduction

This is the second half of assignment three in the course SIGB on the IT-University. It will cover the use of Phong illumination together with flat shading and Phong shading, respectively.

2 Phong Illumination

Illumination modeling is an important part of realistic rendering. It makes colors look more realistic, and enhances the perception of perspective in the final image.

The Phong illumination model aggregates three elements:

- Ambient lighting
- Diffuse lighting
- Specular lighting

We denote these so that

$$I_{ambient} + I_{diffuse} + I_{specular} = I_{phong}$$

2.1 Ambient lighting

Ambient lighting is the odd one out here. As a fixed-intensity, fixed-color light that applies to all everything in the model, it corresponds to a positionless (or omni-present) light source. It has no real-world pendant, but provides a basic intensity value for all objects. Its only variable is the object material, denoted $K_a(x)$. The ambient light value for a point x is found by the formula

$$I_{ambient}(x) = I_a K_a(x)$$

Where I_a represents the 'light source' intensity, and $K_a(x)$ an intensity weight for the object material at the point x.

2.2 Diffuse lighting

When calculating the diffuse lighting in a single point in the phong shading process, we estimate that the intensity of the light that is reflected does not vary depending on the position from which the point is viewed. This is also referred to as a lambertian reflectance. As our light source is of the type 'point'

we use the following formula to calculate the reflection of light in a single point on the surface:

$$I_{diffuse}(x) = Li(x)Kd(x)max(n(x)\Delta l(x), 0)$$

Li is the intensity of the light source. The light intensity decreases as we move farther away from the source. This is calculated through

$$Idiffuse(x) = Il(x)kd(x)max(n(x)\Delta l(x), 0)$$

In our model we assume that the light intensity does not decrease. Kd is the properties of the Lambertian surface. As some surfaces reflects light more than others (ranging from perfect mirrors to total absorption of the light), this property dictates how much of the light is reflected.

The $max(n(x)\Delta l(x), 0)$ part finds the dot product between the light vector and the normal in the point. If this value is less than zero, the angle between the two is large enough for the light source to be behind the surface. As a result the lighting value is set to zero (as all previous values are multiplied by this factor). This makes sense, as as surface lit from behind would not receive any direct illumination. The technique used is similar to the one used in back face culling, where we also measure the normal. The difference between the two being that in culling we used the vector from the camera to the surface, and here we use the vector from the light source to the surface-point.

2.3 Specular lighting

The specular lighting element uses the angles between a surface's normal $\vec{N}$ and the direction to the light source $\vec{l}$, and to the point of view $\vec{V}$ respectively, to determine an intensity weight based the amount of light reflected in the direction of the viewer. Intuitively, the result should be inversely proportional to α.

The specular element is found by

$$I_{specular} = I_s K_s(\vec{R} \cdot \vec{V})^e$$

where $\vec{R}$ is $\vec{l}$ mirrored around $\vec{N}$, and e is a measure of the specularity sharpness.

3 Shading

Shading is a way of giving the perception of depth in 3D illustrations. This is achieved by depicting levels of darkness with respect to a given light source. This section will attempt to describe to kinds of shading used in computer graphics, flat and Phong shading. Each section will consist of a short description of the overall theory behind the shading method followed by a quick mention of our implementation.

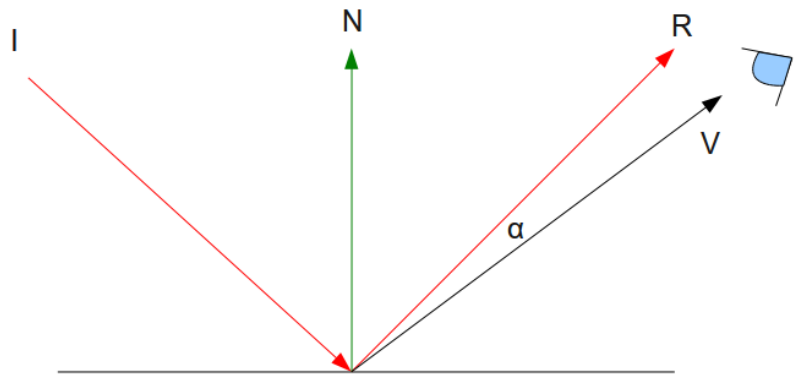

Figure 1:

3.1 The ShadeFace method

The assignment provides us with a lot of boilerplate code used in shading of the cube faces. This section will attempt to clarify what is being done in the provided code

The code is segregated in multiple sections separated by a comment line of dots. The first section is declaration of variables. One important variable is the shadeRes, which gives the resolution and therefore the dimensionality of the shade. The next section projects the homogenious points from real world coordinates to 2D coordinates. Next section defines a square that covers the desired face (meaning from 0 to shadeRes in all dimension. Next section is defines a homography from the texture to the projected points, meaning that we can go from real world to 2D. In the next section three matrices for red, green and blue channels respectively is set by our CalculateShadeMatrix function. Next section applies the intensity values as a texture on top of the image using the cv2 method warpperspective and the homography estimated earlier. Next section simply converts the image from BGR 2 RGB and extracts each colour channel with the cv2 split method. The next section first creates a copy of one of the channel images. Next it set's them all to 0 to create a mask that will be affected only by the points within the face. This is done by taking our previously projected points and fill the polygon they make up with the value 255. This is done with the cv2 method fillconvexpoly, for speed most likely. The next uses this mask of the face to first index each channel and perform a map on these values. The mapping function multiplies the value of the original colour channels with the overlay channels and applies normalization in the range of 0 to 255. The next and final section stitches the channels back together to form

a complete image and transforms it back to BGR.

3.2 Flat shading

Flat shading is a computationally cheap way of achieving a shading effect of an object in a 3D image. The theory behind is that each polygon gets the same intensity value added to every pixel. Provided enough rasterization, it can give a decent result, but the main advantage of flat shading compared to more advanced methods like Phong and Gouraud shading is the smaller faster computations.

With flat shading, one surface normal is assumed for each polygon, usually the normal at the center of the surface, or an average of the vertex normals. This one normal is then used in the lighting model, and the resulting intensity applied to the entire surface.

The result of flat shading is dependant on the position of the light source with respect to the object, but the surface is assumed to be diffuse reflector, and as such the entire polygon will get the same amount of intensity added. The intensity itself is calculated with the phong illumination model calculated for the center of each face. The flat shading assumes some facts about the setup, namely that both viewer and light source is far enough away from the object so that the illumination can be spread evenly (or constant) througout the entire face.

3.3 Phong shading

Flat shading is a little brutal, especially with respect to specularity. If a surface has large specular areas they will either be ignored, or define the intensity of the entire surface. Phong illumination corrects this problem.

By interpolating bilinearly 3.3)

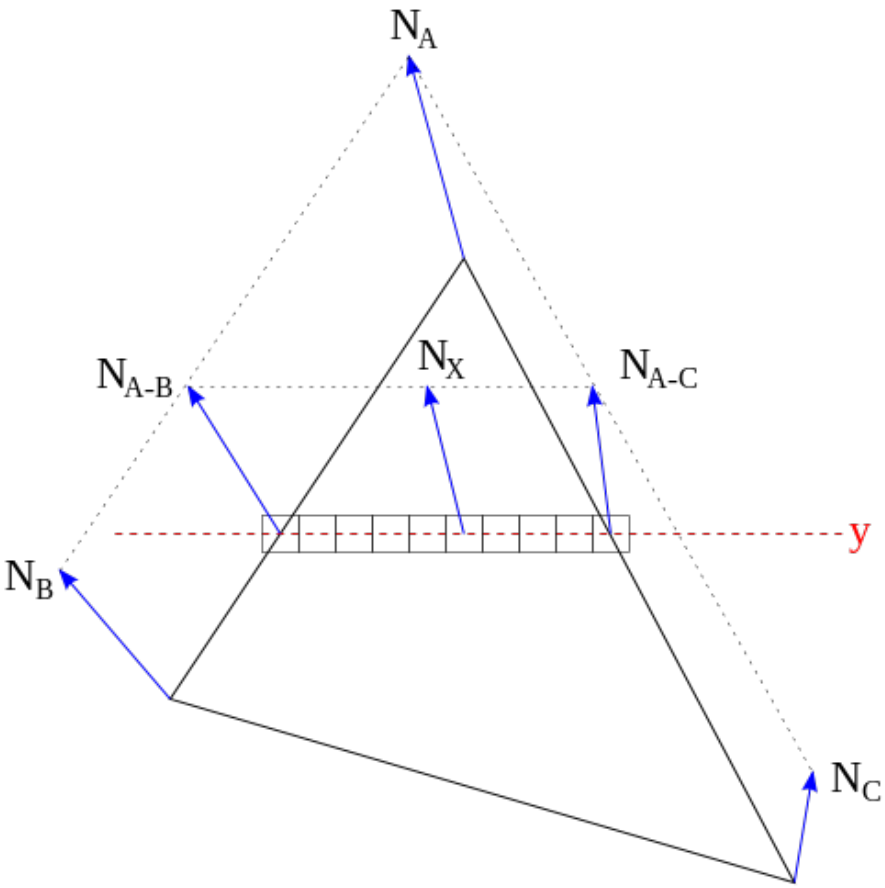

Figure 2: Normal interpolation in Phong shading

3.4 Estimating the point of impact

Estimating a point of impact. One of the main obstacles we faced while programming the Phong shading model, was determining the exact points in world coordinates on which the light hit the box surface. This point is needed for correct visualization of both the diffuse and the specular lighting. As a bad estimate we started out just by using the center point for each surface, but the results gained from this method were not impressive. Instead we tried to utilize that we had some known points on each surface, trying to estimate some points in between to use in a more realistic lighting model. Instead of iterating through the pixels of each surface, we iterate through the 10x10 shadeRes matrix. Each point in 2d here corresponds to a 3d point on the box surface. (Note that we

also experienced a bit with using a homography from the box surface in 3d to a 2d coordinate system, in effect doing an inverse projection, but instead we decided to estimate it the other way around). As our 2d coordinate system in this case is 10x10 each surface determines the vector from the center to one of the corner points. This vector is divided by 5 to get the increment factor of each axis. This factor times ten (each axis x,y,z fully incremented) gives us the opposite corner point. Each time we increment either of the axis (x or y) in our 2d coordinate system, we increment each axis accordingly in our 3d world coordinate system adding the value to our start/corner coordinate, thus allowing us to iterate through each surface on the box. The new light vector is then calculated from the light source to the surface point in world coordinates.

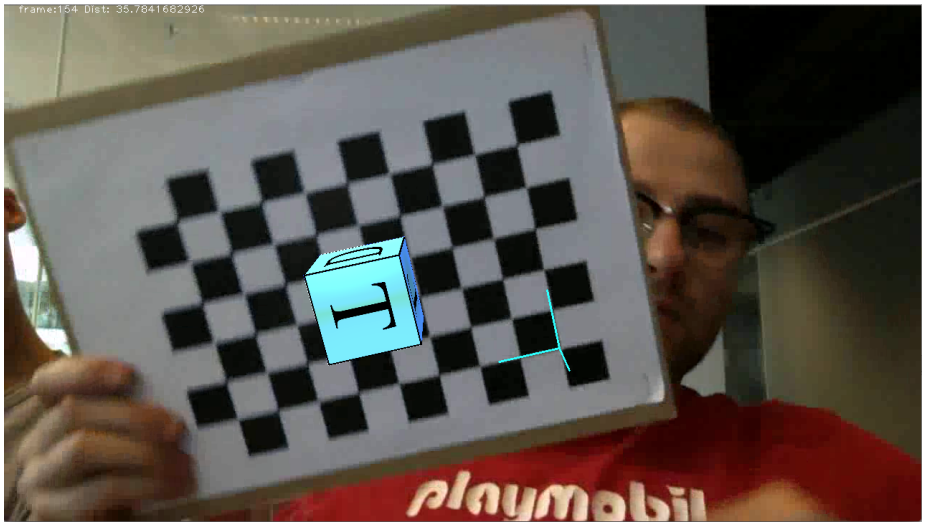

Figure 3: **Using the surface center as the impact point of the light for all surface pixels.** Using the box center as the impact point is evidently producing a less satisfying result - not showing much of the specular (white) reflections. This means, that the only effect we get from the added specular lighting to our Phong illumination model is some color changes. The color of the reflectiongs from an object can easily be changed by changing the parameters of the lightsource, but in this case we think its a bug. The color object should ideally not be changed as much as we see in the picture compared to the diffuse-only picture (see below)

4 Our findings

In figure 4 we see the first steps towards augmented reality. It is kind of where we left off from assignment two, only with a new method. We also implemented

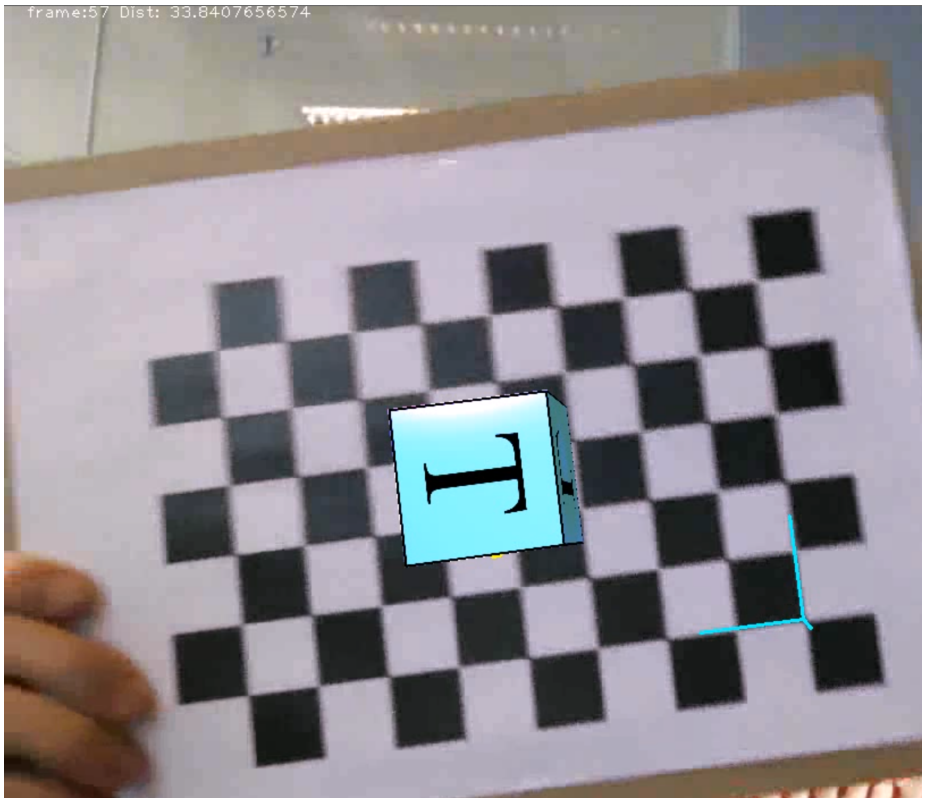

Figure 4: **Using our impact point estimation.** The result is better, but far from perfect using our estimated point-of-impact. Getting the correct point in the world coordinates has a huge impact of the realism on the specular lighting, and we're still far from that result. It is an improvement though, over using the surface center as seen in the image above. Some of the specular reflection highlight are shown in with this method, and furthermore shown in the area pointing towards our lightsource. More highlight shine through on surfaces pointing away from our lightsource which is not meant to happen, but we contribute that to two factors: • **The estimation contains errors on each surface** • **The light source is placed at the camera center, the placement of which is also estimated and not completly reliable.** All in all a result that leave us with a lot of room for improvements.

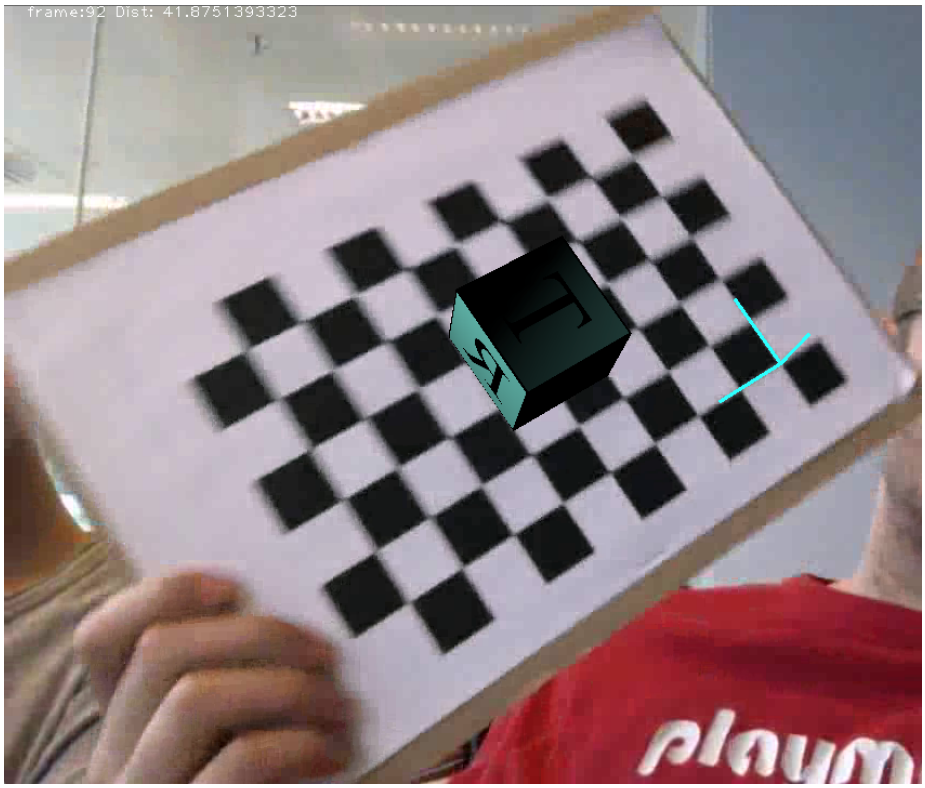

Figure 5: **Using phong light model and interpolated shading, but showing diffuse surface light only.** This image shows the color degrading in the other images as the result of the added specular lighting. Note that the result here using only the diffuse light, seems more fitting as it matches the original lighting in the video a bit better.

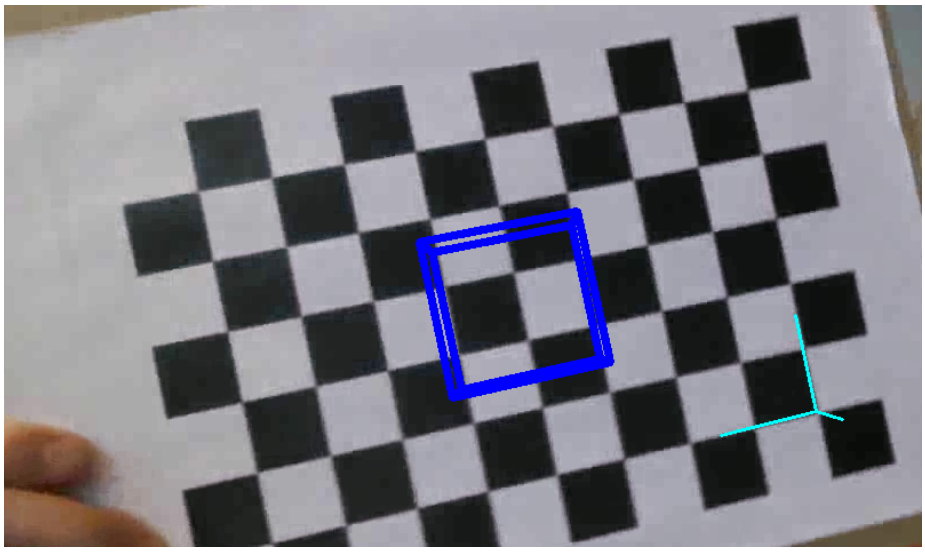

Figure 6: Wireframe of the augmented cube

the same method as was used in assignment 2. This can be seen in figure **??**. This method give us a skewed result. Most likely because of a bad homography.

The difference between these two method is also clearly seen in 4 and 4 respectively.

Next up was adding textures to the cube to make it look cooler. The naive result of this is seen in figure 4

Obviously the approach used in figure 4 is far from perfect because all sides are visible. Even those that would be occluded by the cube itself. This is fixed in figure 4 where back-face culling is applied to correctly hide the occluded faces

The next part of the assignment focused on shading. The approach taken was based on the Phong illumination model and manipulates the RGB channels of the image on the cube face to give a shading effect. The first method implemented was flat shading. A successful result of this is seen in figure 4.

Here can be seen that the diffuse lighting hit's the top face more than the outward facing face. Thereby leaving the outward facing darker. For illustrative purposes figure **??** shows an example where the ambient lighting has a much higher red value. What is seen is that the shadowy face (outwards facing) is more red than the face where the diffuse lighting falls.

The next shading method used is phong shading, which is more advanced and consequently more difficult to implement. An example of this is seen in 4. What can be observed is that the lighting falls more realisticly. Mainly because the

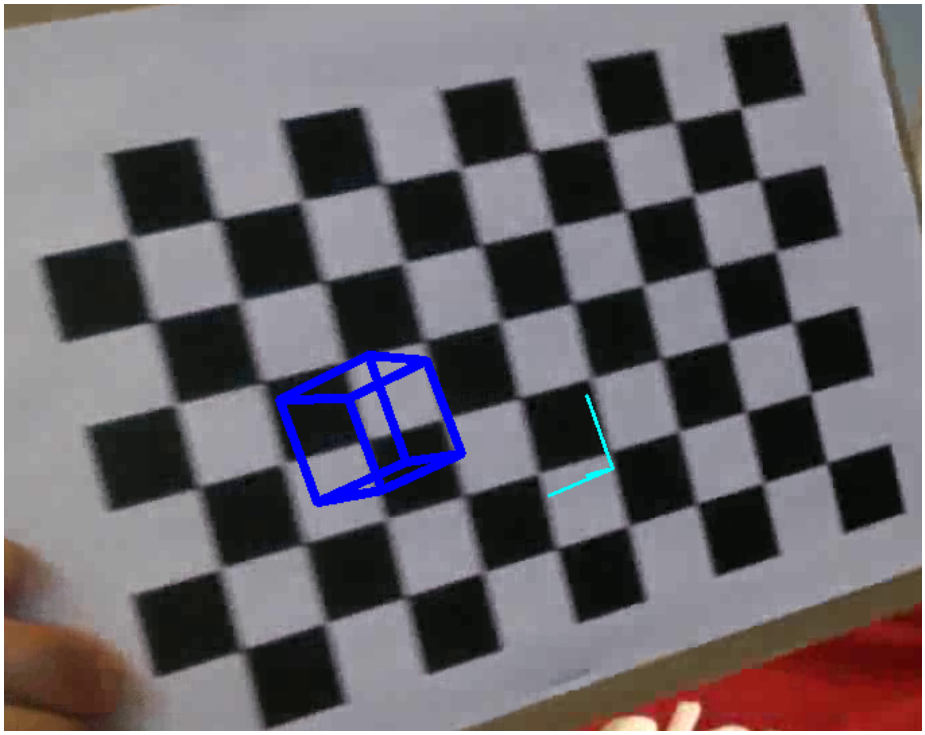

Figure 7: Wireframe of the augmented cube with old method

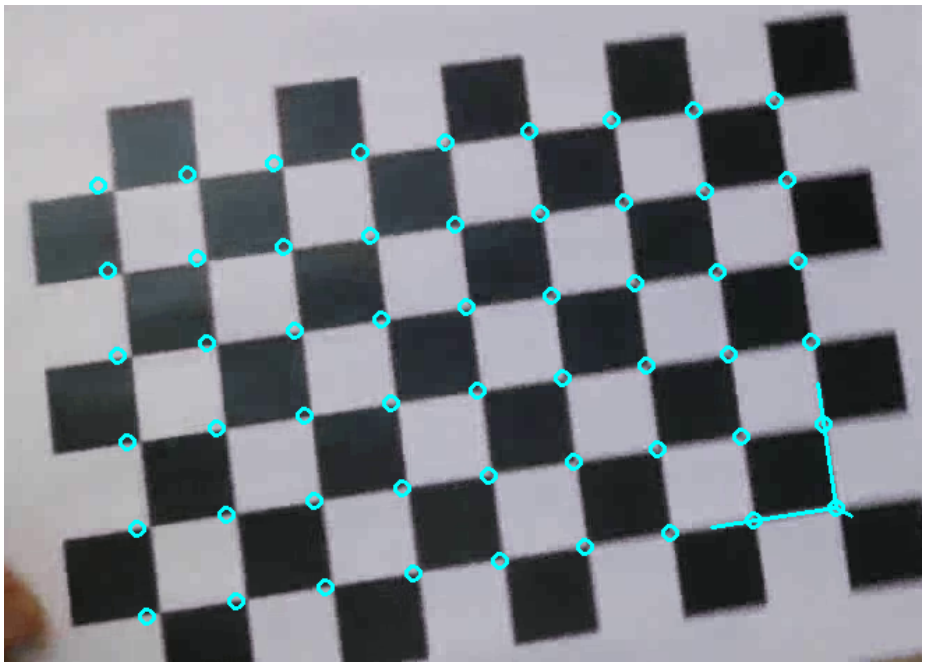

Figure 8: The augmented grid

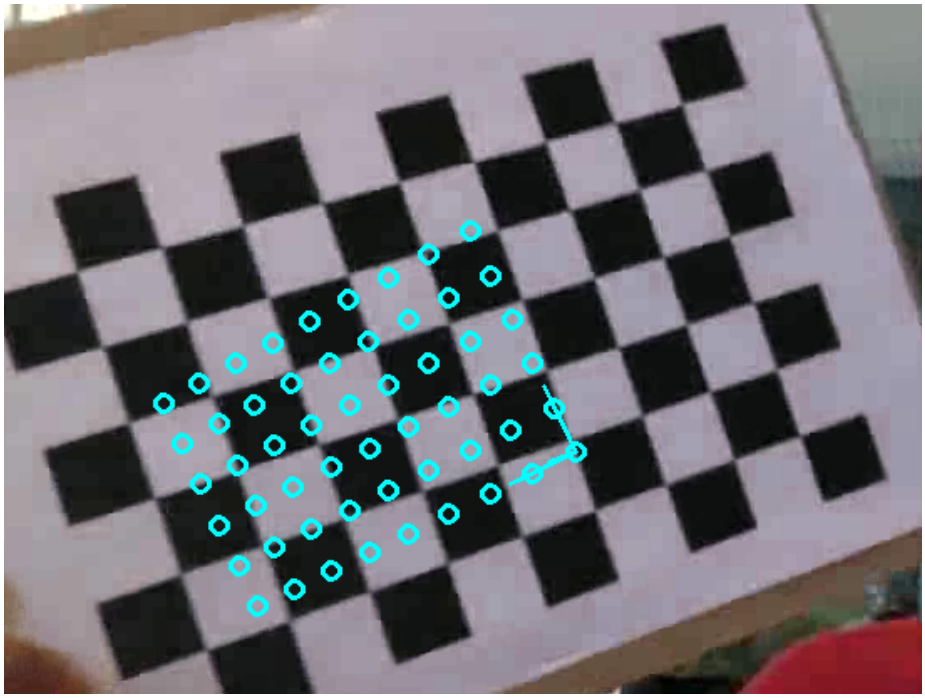

Figure 9: The augmented grid with the old method

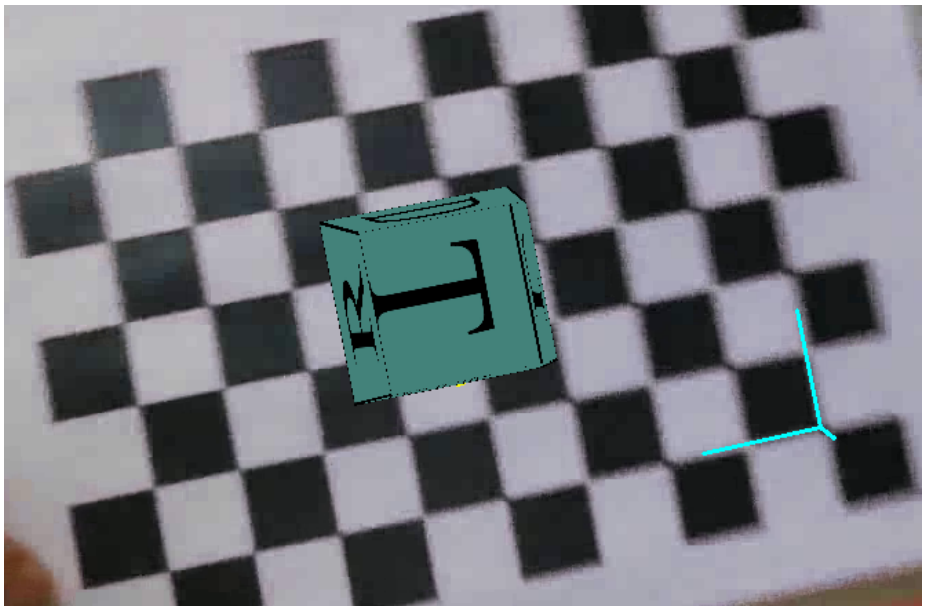

Figure 10: Naive texture mapping on cube

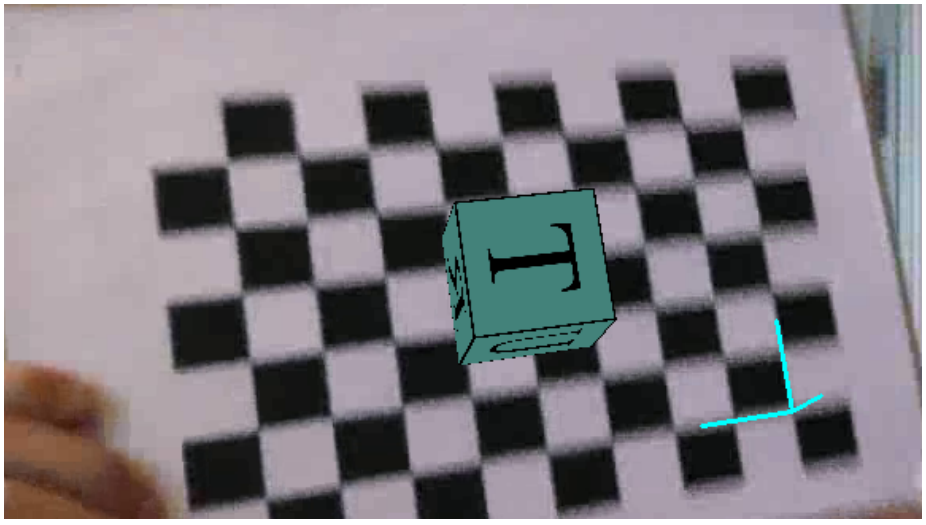

Figure 11: Texture mapping with backface culling on cube

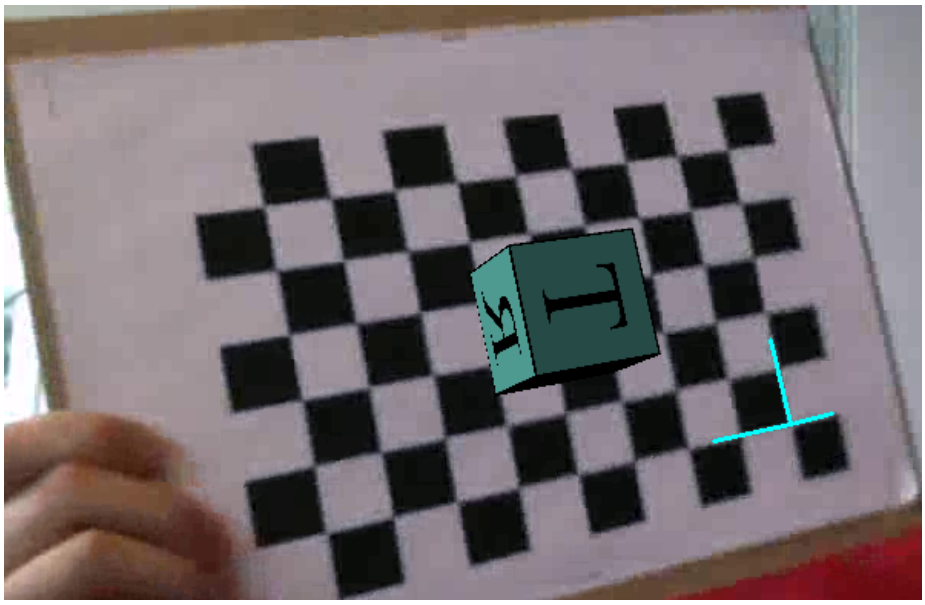

Figure 12: flat shading on the figure

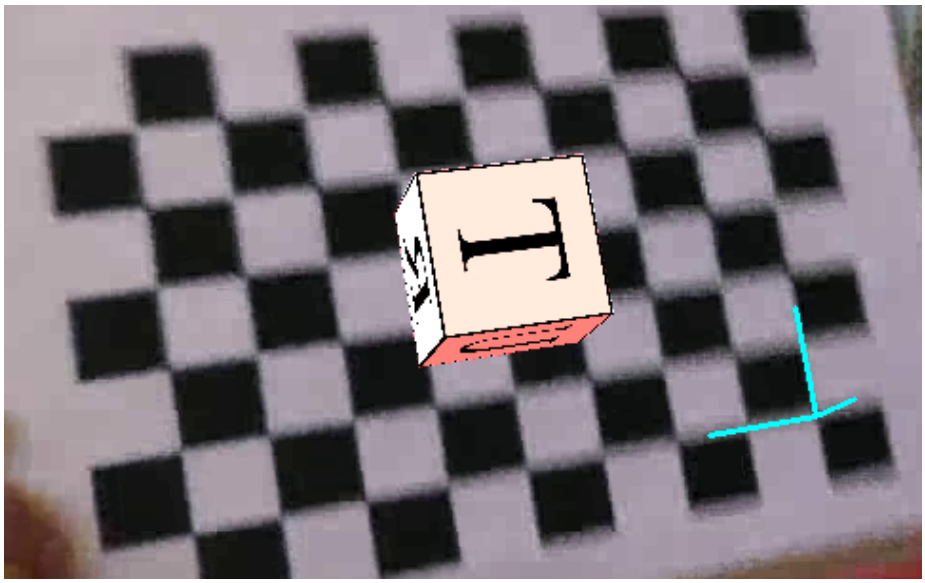

Figure 13: flat where the ambient value has it's red channel manipulated

light falls differently spread over the face. What can also be observed is that specular lighting has a big effect on the quality of the results in figure **??**.

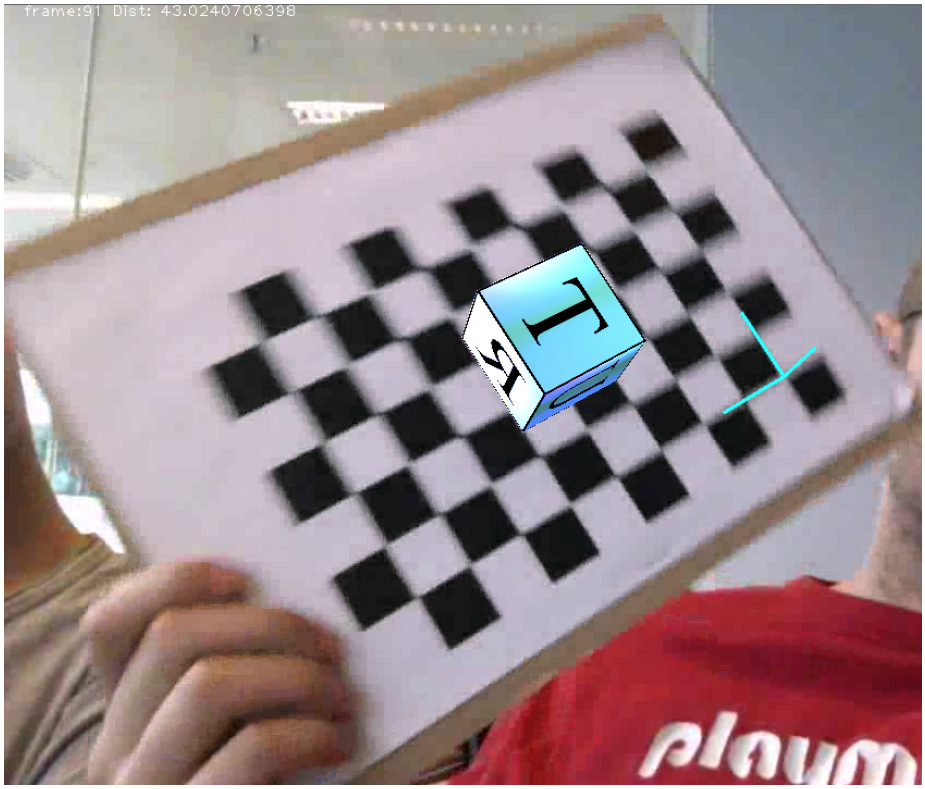

Figure 14: an example of Phong shading

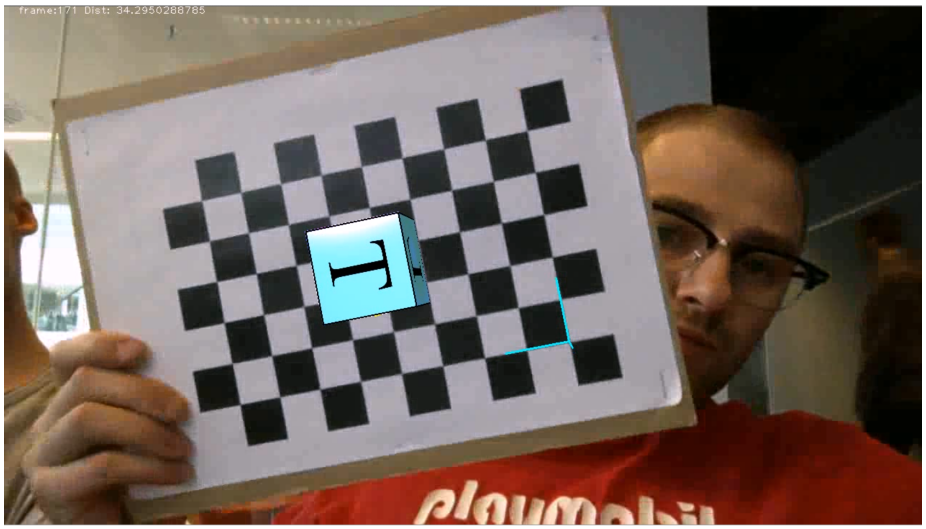

Figure 15: an example of Phong shading with the specular lighting having an impact