# Poisson Image Solver

C. Albert Thompson

Department of Computer Science
The University of British Columbia

## Abstract

This paper presents the Jacobi method to solve Poisson's equation and regenerate images from the divergence of an original image. There has been a lot of related work that implement Poisson solvers, which explains the steps necessary to generate an image using a linear solver. Poisson's equation can be simplified to be solved by a linear equation, and this paper uses the Jacobi method approach. The program calculates divergence of the original image and the Jacobi method is able to approximate an image similar to the original. Regenerated images have an error of 0.01%with the iteration count of 50000 . The project also implements a basic watermark feature that adds watermarks to an image. Future work of the project can further implement different types of image manipulation, such as copy paste, image restoration and other linear solvers. This paper demonstrates one approach to solve Poisson's equation for image regeneration.

## 1 Introduction

The goal of this project is to understand how image reconstruction is done using Poisson's equation, and then implement a simple reconstruction algorithm on an image.

Restoring dated or printed images becomes more prominent as older images become affected by time. Using chemicals to print photos is a good way to have them last a long time. A long time is not enough, photos have been around for two hundred years, and photos will inevitably get damaged either by light, or any other of the many ways to damage printed photos. Figure 1 shows a photo that is about twenty years old and it is facing noticeable damage 1. As you can see here are vertical lines running across the photo and there is noticeable fading of the whole photo.

Restoration of images has been around for many years and early work used Fourier-transformation to restore images [Richardson 1972]. His work was later build upon by future authors such as [Bertalmio et al. 2000], and [Pérez et al. 2003]. They were able to use the old well known Poisson's equation to solve for images and repair damage or images with missing pieces. These authors used Poisson's equation to restore images in more discrete methods.

In our project takes a further look at solving Poisson's equation to regenerate images, and leaves restoration of images for future work. Initial investigation of related work shows that most authors do manipulations on gradient vectors and then restore the image using divergence. The reason Poisson's equation is so useful is that it is a partial differential equation, and allows computation an image from it's divergence.

When Poisson's equation is setup properly a linear solver may be used to retrieve an image from the divergence. The Jacobi method is the linear solver we use to solve Poisson's equation. Our implementation of the Jacobi method reconstructs an image with an error of only 0.01%in 50000 number of iterations. With these initial results show the importance of linear solvers and Poisson's equation.

In this paper shows how to calculate the gradient of an image. The first step is to change the image into a scalar function. The scalar function is used to calculate gradient field of an image. The gradient of the image allows us to make minor changes to an image, and even make estimations where images contain artifacts. After the gradient is used to calculate the divergence of an image and restore it using Poisson's equation.

This process shows how we can add images together to get a completely new image. In this paper we show results of using a Jacobi solver to regenerate an image from its divergence. We also look to manipulate an image divergence by merging two images together to add a simple watermark to an image. The results of this paper are promising and give us many paths to take to improve our methods.

The goal this project is to understand how image reconstruction is done using Poisson's equation, and then implement a simple algorithm to regenerate an image from the divergence a vector.

The sections are broken down as follows. In section 2 I discuss current research in Poisson image restoration. Section 3 is dedicated to introducing explaining Poisson's equation, gradients and a way to solve Poisson's equation for image restoration. Section 4 describes artifacts and proposed methods to remove them. In section 5 we discuss future work. Section 6 is the conclusion.

## 2 Related work

Using Poisson's equation to reconstruct and repair images is a popular because images can be transformed into vectors and completing Poisson's equation can be solved to smooth an image. From an initial analysis we have found that only

Figure 1: An old picture of C. Albert's family.

recently [Bertalmio et al. 2000] has Poisson's equation used to do image image reconstruction. The current work includes many different ways of doing image reconstruction. One field looks at taking current images that are damaged or missing parts and using gradient and Poisson to reconstruct the image. Another area looks at dragging and dropping images on top of each other then treat the resulting image as a damaged image and do the repair using Poisson.

Bertalmio et al. introduces their paper by making a goal that they want to repair damaged photos. These photos may have inpainting, meaning words written on top of the picture and remove part of the image to add the text. The authors also include photos that are damaged or missing parts of the image. Their contribution is different from other approaches as they do not require user input to define the texture of the spot being restored. They create a tool that uses partial differential equations to determine the area that is defined to be filled in. Once they fill in the area they use a smoothness function to improve the image. [Bertalmio et al. 2000] made a major contribution by allowing restoration of images to be done automatically.

Image completion is bad in many cases because it lacks gradient of an image and thus cannot fill in missing parts of an image without having significant artifacts. To overcome this problem they introduce three steps in the process. First they introduce a gradient-patch filling to make up gradient that should be in the image. Then second they use an update strategy to gradient-patches by measure the distance between the source patch and a target in the gradient domain. Last to complete the image by using the Poisson equation on the gradient map generated by the tool.[Shen et al. 2005] [Ballester et al. 2001]

Image reconstruction has also been extended into restoring documents that have folds or other geometric distortions. [Sun et al. 2005] combines 3D images with restoration of documentation. The two initial challenges present themselves in using 3D images are shading, and geometric correction. The paper first solves this problem using previous work on the related subjects. Once the image has been restored to its 2D format then they solve Poisson equation and restore the image. This is work is useful for restoring documents without touching them or that may be fragile.

Mixing images is a difficult task to accomplish because the two image have different gradients. The gradients are important because it is what allows an image to seemingly mesh together. [Pérez et al. 2003] offered a solution to fixing the problem of images not being able to be place together without causing major clashing. His idea was to mix the gradients of the images together to soft of weave the two images into one another. They also made a contribution selecting only part of an image that one might want to reconstruct and then running Poisson's partial differential equation on that section. These tools that he added helped move image drag and drop along.

Drag and drop images as previously mentioned simply selected an image and dropped it onto the destination image. By doing this the destination image may contain undesired artifacts.[Jia et al. 2006] offers a way to avoid having these artifacts by making automatic edits to the image that is being dropped into a destination image. The boundary of the image that is being dropped does not fit within the gradient of an image. This method that they use changes the boundary and makes the image fit nicer into the destination image.

## 3 Poisson

The related work gives insight to using Poisson's equation to solve for an image. Understanding Poisson's equation is necessary learn how to reconstruction an image. The next step is to retrieve a gradient field of an image. With the gradients a linear method is used to solve Poisson's equation restore an image from the divergence from the gradient field. This project shows how to setup Poisson's equation, manipulate an image and restore the image from a gradient vector.

### 3.1 Poisson Equation

Figure 2 taken from [Pérez et al. 2003] illustrates notations used in applying Poisson's equation to our image editing problem. S is a domain for the function $f$ which is a closed subset of $\mathbb{R}^2$. Also in the image above $\Omega$ is a closed subset of S with the function $f^*$. $\mathbf{v}$ is the vector field of image $\Omega$.

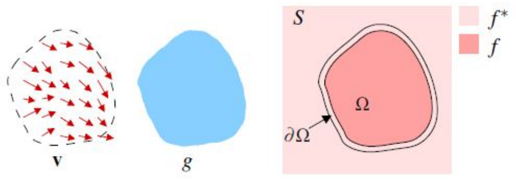

Figure 2: **Guided interpolation notations**. Unknown function $f$ interpolates in domain $\Delta$ the destination function $f^*$, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function $g$

Poisson's equation is a partial differential equation that is expressed as follows:

$$\Delta\varphi = f \quad (1)$$

$\Delta$ is the Laplace operator, which is a differential operator, and $\varphi(x, y)$ and $f(x, y)$ are functions, in $\mathbb{R}^2$ domain. In this project Poisson's equation is used to regenerate 2D images.

We Poisson's equation and solve the for $\Omega$ to get an image. This can be accomplished using satisfying the associated Euler-Lagrange equation $\Delta f = 0$ over $\Omega$, $\Delta$ in our equation is represented as $\frac{\delta^2}{\delta x^2} + \frac{\delta^2}{\delta y^2}$. We now transform this problem into a minimization problem to solve for our image.

For the minimization problem we define it as the simplest interpolant of $f$ of $f^2$ over $\Omega$ and represent it by the following equation:

$$\mathbf{min}_f \iint_{\Omega} |\nabla f|^2 \quad (2)$$

If we include the vector field $\mathbf{v}$ we have a version of the the minimization problem 2 but this now applies directly to the vector field $\mathbf{v}$.:

$$\mathbf{min}_f \iint_{\Omega} |\nabla f - \mathbf{v}|^2 \quad (3)$$

there problem has a unique solution with Dirichlet boundary conditions and is expressed:

$$\Delta f = div\mathbf{v} \text{ over } \Omega \quad (4)$$

We use this equation for each of the three colors in the RGB space to solve for the colored image $f$. All the results in this paper are reported with solving equation 3 each color red blue and green of an image independently. In the next section we discuss how to calculate the gradient of an image.

## 3.2 Gradient and Divergence

The gradient of an image is important when solving Poisson's equation. The divergence of the vector field describe in equation 4 for our image solver is the calculation of the divergence of the gradient field. To calculate the gradient we need 1) represent the image as a scalar, 2) have an equation to calculate the gradient, 3) take care of boundary conditions, 4) calculate the divergence and 5) normalize numbers to represent results as an image.

In order to calculate a gradient we need a scalar function.

$$f(x, y) \quad (5)$$

From this scalar function we can then take the partial derivative of $x$ and $y$ to get a gradient field. The gradient field is the rate of change in the $x$ and $y$ direction and is represented by the following equation:

$$\nabla f = \left(\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}\right) \quad (6)$$

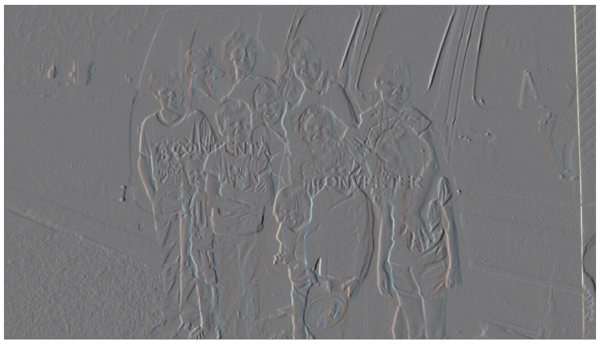

Figure 3: $f_x$

To get an image gradient field we need to be able to represent the image as a scalar function. An easy way to represent the image as a scalar is to represent each pixel as a red green blue (RGB) value. The way we represent that in this paper is to have image $f$ and the variables $x$, and $y$ are the x y coordinates of the pixel in the image. The return value of this function is RGB value for each pair $(x, y)$. This is how we represent our image as a scalar function.

With the scalar function we can find the gradient field of our image. This can be calculated by taking the difference between the current and its neighbors. This is calculated with the follow equation, where $f_x$ is the is the gradient in the x direction.

$$f_x(i, j) = f(i)(j + 1) - f(i)(j - 1) \quad (7)$$

$$f_y(i, j) = f(i + 1)(j) - f(i - 1)(j) \quad (8)$$

Using these equations we get the rate of change that happens at every pixel.

If we were use only these equations to calculate the gradient, we end up with a picture that is one pixel smaller than the original image. One of the problems that occur when using equations 8 and 7 is we cannot use these equations to calculate the boundary conditions. If 7 come to a boundary pixel it cannot use the equation calculate the gradient because there is no $j + 1$ pixel. We over come this boundary problem by duplicating every boundary pixel in both the x and y axis. Duplicating the pixels allows us to keep the original image size and still get the gradient of the image.

With the gradient field of the image we can now get the divergence which is the right hand side of the Poisson equation 4. An example of the divergence is seen in figure 5, this image demonstrates the divergence of figures 4 and 3. The divergence is the second derivative of equations 8 and 7 added together that gives us:

$$divf = \left(\frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}\right) \quad (9)$$

When calculating divergence we also need to take into count the border conditions that happened with calculating the gra-

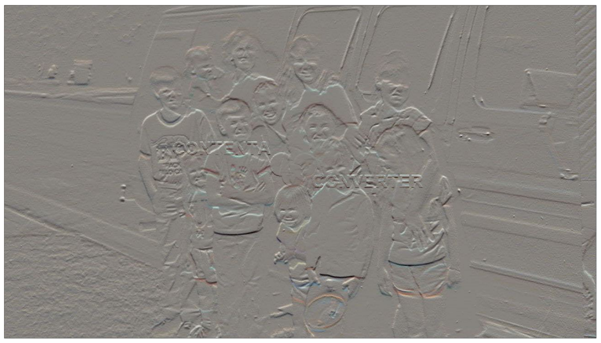

Figure 4: $f_y$

Figure 5: $(f_{xx} + f_{yy})$ Which is the divergence of $f_x$ and $f_y$ normalized to display in an image.

dient. We solve the boarder problem the same way we did for gradients and duplicate each border pixel.

The calculation of the gradient gives us an array that can have the values -255 up to 255. But negative values cannot be represented in an image that has regular RGB values. In order to make the correction to display these values in a file and on this paper we normalize the array. To normalize the array we use the following function:

$$I(x, y) = \lfloor \frac{I(x, y) - I_{min}}{I_{max} - I_{min}} \times 255 \rfloor \qquad (10)$$

Normalizing the array we are able to display the gradient in the X direction with figure 3 and the gradient in the y direction with figure 4.

## 3.3 Poisson Solver

In the Poisson's equation discussed in section 3.1 we deal with one matrix $\Delta$ and two vectors $f$, and div $\mathbf{v}$. We know the divergence vector, but we still have two unknowns the Laplacian $\Delta$ and $f$, which is the image we are trying to compute. We can compute $f$ in a discrete way by expressing the Laplacian and divergence as discrete filters. But this expression is difficult to solve for large vectors so we use an iterative Jacobi Poisson solver to approximate the solution. Using the Iterative Jacobi solver we are able to regenerate an image from the divergence of that same image.

$$\left[\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & & & -1 & & & & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & & & & \\
 & -1 & 4 & -1 & & & -1 & & & & & & & & & \\
 & & -1 & 4 & & & & -1 & & & & & & & & \\
\hline
-1 & & & & 4 & -1 & & & -1 & & & & & & & \\
 & -1 & & & -1 & 4 & -1 & & & -1 & & & & & & \\
 & & -1 & & & -1 & 4 & -1 & & & -1 & & & & & \\
 & & & -1 & & & -1 & 4 & & & & -1 & & & & \\
\hline
 & & & & -1 & & & & 4 & -1 & & & -1 & & & \\
 & & & & & -1 & & & -1 & 4 & -1 & & & -1 & & \\
 & & & & & & -1 & & & -1 & 4 & -1 & & & -1 & \\
 & & & & & & & -1 & & & -1 & 4 & & & & -1 \\
\hline
 & & & & & & & & -1 & & & & 4 & -1 & & \\
 & & & & & & & & & -1 & & & -1 & 4 & -1 & \\
 & & & & & & & & & & -1 & & & -1 & 4 & -1 \\
 & & & & & & & & & & & -1 & & & -1 & 4
\end{array}\right] * \begin{bmatrix} U(1,1) \\ U(2,1) \\ U(3,1) \\ U(4,1) \\ U(1,2) \\ U(2,2) \\ U(3,2) \\ U(4,2) \\ U(1,3) \\ U(2,3) \\ U(3,3) \\ U(4,3) \\ U(1,4) \\ U(2,4) \\ U(3,4) \\ U(4,4) \end{bmatrix} = \begin{bmatrix} b(1,1) \\ b(2,1) \\ b(3,1) \\ b(4,1) \\ b(1,2) \\ b(2,2) \\ b(3,2) \\ b(4,2) \\ b(1,3) \\ b(2,3) \\ b(3,3) \\ b(4,3) \\ b(1,4) \\ b(2,4) \\ b(3,4) \\ b(4,4) \end{bmatrix}$$

Figure 6: $I_y$

To solve Poisson's equation 4 we need to express $\Delta$ and the divergence as discrete filters. If we were to solve for on a single pixel grid the discretized Laplacian would look as follow:

$$\Delta \rightarrow \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad (11)$$

The divergence of the image needs to be setup to be a vector. To do this we stack the columns of the image on top of each other. The image that we need to reconstruct is also a vector of the same dimension as the divergence vector. The final representation of this equation for an image that is 4-by-4 pixels would looks like Figure **??**. This problem is now reduced to a linear system but does not scale very well as the image increases in size the size of the matrix increases by $N^2$.

**Jacobi Method** We use a Jacobi solver to approximate the image from the divergence, because direct solving of Poisson's equation 4 is not possible for larger images. The algorithm for the Jacobi method is as follows:

```
initialize all of f'(x,y) to 0
do
    for i to n
            for j to m
            f(i,j) = (f'(i-1,j)+ f'(i+1,j)+ f'(i,j-1)+ f'(i,j+1)
                        + div(i,j))/4
f'(i,j) = f(i,j)
while (image has not converged)
return f(i,j)
```

This algorithm needs only one input which is the divergence vector **div**. It initializes the approximation of the image to zero. Then the program goes to every pixel and takes it's four neighbors and adds them and the **div** of the pixel and then divides by four. This algorithm loops until the image converges or for a defined number of iterations. Our Jacobi approximation method that minimizes the differences between $\nabla f$ and $div\mathbf{v}$.

**Applying Mask** Using a Poisson solver we can add divergences of different images together to create a transparent watermark in our resulting image. To do this we use two images, one that is our original image and another that is our watermark. Both images are the same size and the watermark has a complete white background, except for the location of the watermark. The white background allows us to locate the watermark and add together only the divergence the watermark and the picture.

## 4 Artifacts

We use the Jacobi method on a number of images and compare them to the original image. There are a few artifacts in the resulting image. The watermark functionality also can be improved. Our final images had an average of 0.01%error when compared to the original image, when doing 50000 iterations.

**Regenerating image** After running Jacobi method figure 7 is an example of the result. This image compared to the the original 1 it has good coloring in the middle of the picture. One of the problems in the picture are that the boarders are white and do not have the same darker coloring of the original image. The artifacts occur due to my method of dealing with the boarder of the image, using the Jacobi method.

We tried two different methods when dealing with boarder pixels using the Jacobi method. The first way we dealt with the board pixel was to not calculate that pixel and leave it at zero. The result of that method is shown in figure 7. The other method we tried was similar to calculating the gradients. When we were at a boarder pixel we duplicated the pixel and used the duplicated pixel instead of adjacent non existing pixels. The result of that image is show in figure 8, it produced an image that was too dark. We stayed with our original method of calculating boarder pixels in the Jacobi method and left all boarder pixels at zero.

**Watermark** An example of the watermark feature of our application is show in figure 9. The image produced by the watermark has more artifacts than only regenerating the image. We found that simply adding divergence of the watermark and the image sometimes caused the image to not have equal levels over the entire image. One of the ways that we can fix this may be to use only a small ratio of the divergence of the watermark. We did when the divergence of the watermark was small it would allow the image to produce a result much closer to that seen in when only regenerating an image.

## 5 Future Work

Poisson's equation allows regeneration of images that have changed gradient vectors, or seamlessly integration of two images as discussed in section 2. Watermark feature of the project produces initial results on seamless integration of images, and is to how drag and dropping discussed by [Jia et al. 2006] works. Further work includes adding boundary calculations to improve the boarder of the image that is being

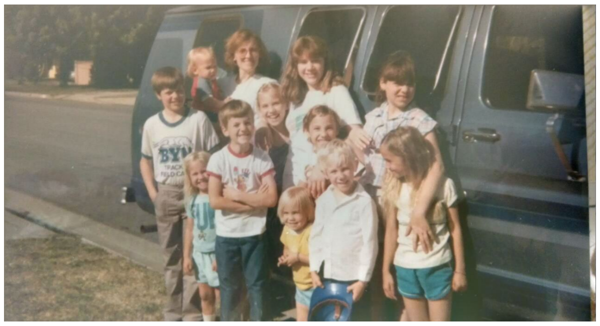

Figure 7: An approximation using Jacobi method to solve for the original image. With boarder pixels ignored.

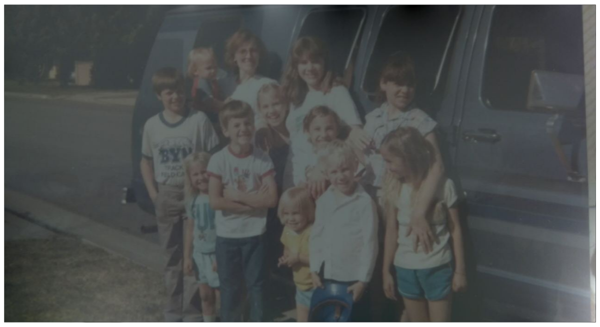

Figure 8: An approximation using Jacobi method to solve for the original image. With boarder pixels duplicated.

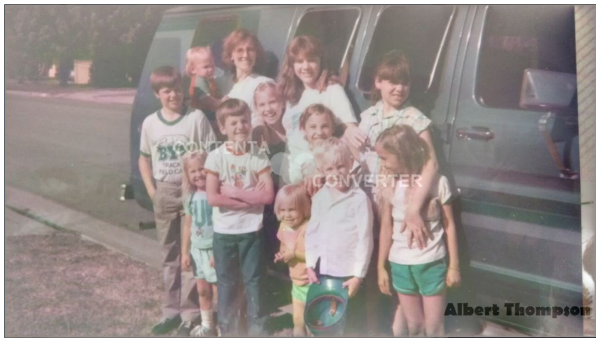

Figure 9: An example watermark on an image.

pasted on. Adding or removing the transparency of an integrated image can also help improve the final image.

Further exploration is necessary to implement an image restoration feature to this project. In order to accomplish this divergences need to be estimated for the mission sections of an image, and is discussed by [Pérez et al. 2003]. A mask may be used to identify what parts of the image needs restoration. With the mask estimations of missing sections are calculated and an linear solver can reproduce an image to restore missing, or damaged sections.

There are many different methods that can compute the linear equation (4) and minimize the difference in the original image and desired image. The Jacobi method used solve Poisson's equation is slow. A different linear solver can be implemented to improve the speed of the program. The Jacobi method in calculates linear problems in $N^2$ time. An improvement to the project would be to use the the conjugate gradient implementation which cuts the calculation time to $N^(3/2)$, or Fast Fourier Transform calculates in only $N$ time. Further research allows us to change from using the Jacobi method and implement other linear solvers.

## 6 Conclusion

We show how Poisson's equation is used to regenerate an image from the divergence of gradient fields. We calculate gradients of an image and also take in to account the boarder conditions. An implementation of the Jacobi method solves Poisson's equation using only a divergence. With an error of 0.01%, our program obtains images that looked similar to our original image.

The Jacobi method provided us with an initial experience on methods of solving Poisson's equation. An implementation of a water mark feature demonstrates how image divergences can be layer to produce a new image. Implementing our watermark method allowed us to see how images react when two divergences are added together. Further work includes implementing further gradient manipulations and refining our calculation of boundary conditions.

## References

BALLESTER, C., BERTALMIO, M., CASELLES, V., SAPIRO, G., AND VERDERA, J. 2001. Filling-in by joint interpolation of vector fields and gray levels. *Image Processing, IEEE Transactions on 10*, 8 (aug), 1200 –1211.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 417–424.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2006. Drag-and-drop pasting. *ACM Transactions on Graphics (SIGGRAPH)*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22* (July), 313–318.

RICHARDSON, W. H. 1972. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am. 62*, 1 (Jan), 55–59.

SHEN, J., JIN, X., AND ZHOU, C. 2005. Gradient based image completion by solving poisson equation. In *Advances in Multimedia Information Processing - PCM 2005*, Y.-S. Ho and H. Kim, Eds., vol. 3767 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 257–268.

SUN, M., YANG, R., YUN, L., LANDON, G., SEALES, B., AND BROWN, M. 2005. Geometric and photometric restoration of distorted documents. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, 1117 –1123 Vol. 2.