Making Faces

2005/06/08

R93922063 陳坤毅　R93922087 莊曜誠　R93922105 王博民

3D acquisition for faces:

如何取得臉的 3D model?

方法 1:（經費足夠時）

使用 Cyberware scanner.（可對臉部 scan 亦可對全身 scan）

方法如下圖所示:

face & head scanner

whole body scanner

缺點:

人必須保持靜止。無法取得動態的 3D model，且成本較高。

優點:

比較能得到精確的 Mode。

方法 2:（由照片取得 3D model.）

方法概念與 mosaic 類似，先給一個人臉形狀的 model（積木），用 warping、morphing、或是調整一些參數，使得由各個角度去看時，feature points 能夠 match 變型後的人臉。

好處: 人可以動，不須需靜止。

Step1: 用五台 camera 拍攝人臉

Step2: User 必須給定幾十個 feature point，指出各個部位的位置。

Step3: 估計 feature point 3D 中座標以及 Camera 的參數。(亦即 Structure from motion)

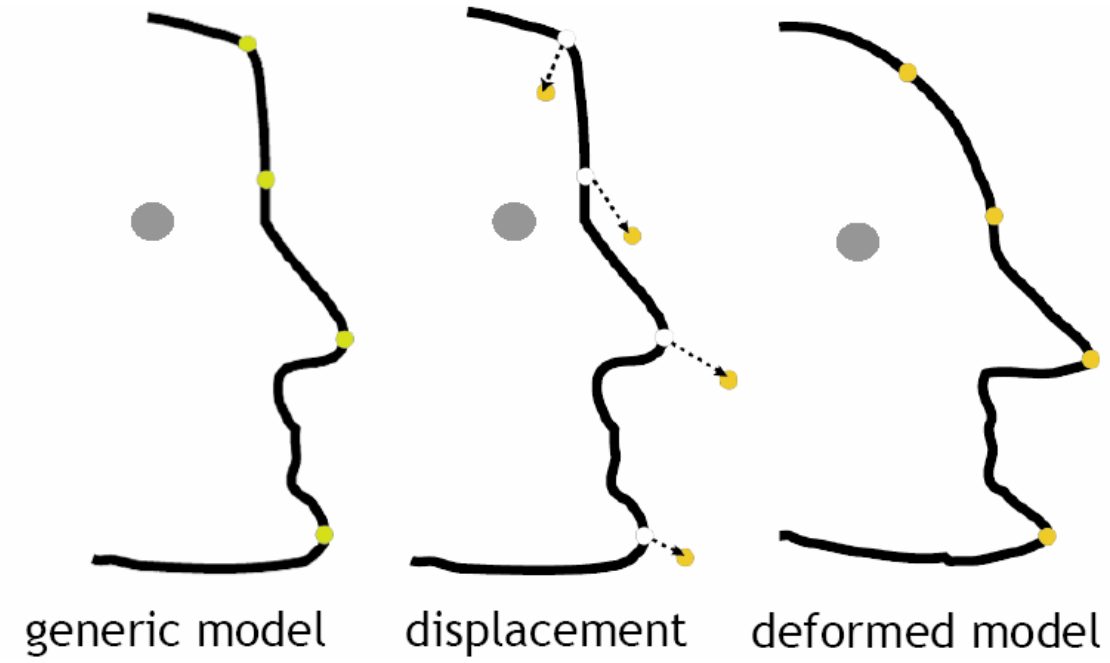
Step4: 用更複雜的 feature points 把 model 再更進一步 deform，使能更 match 臉上的一些 feature

Step5: 最後將 texture 貼上。

Mesh deformation

1. 計算 feature point 該移動到哪裡去。
2. 決定那些並非對應到 feature point 的 vertex 必須對應到哪邊去。

Scattered data interpolation



黃色點為 feature point，灰色點為整個 model 重心。

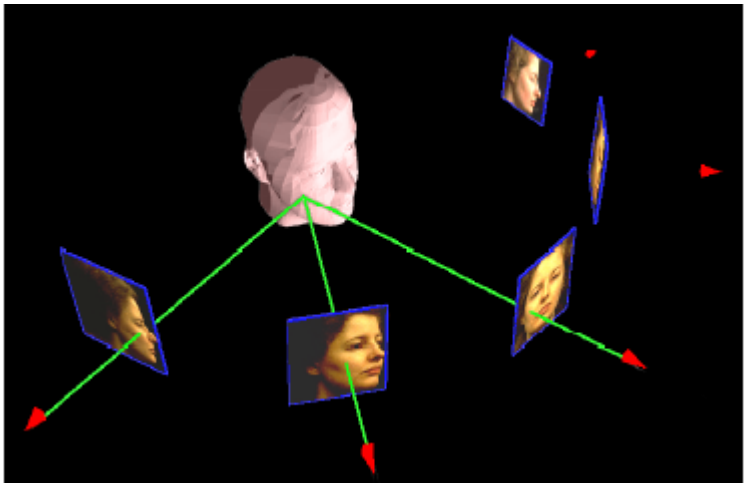
中間圖白色位置為 feature point 原來位置，黃色為白點將被移動到的地方。欲決定黑邊上的點要移動到哪去，先去看週圍的點移動到哪去，再用一個 smooth 的 kernel 來做一個 weighted average.

Texture extraction

分為 view independent and view-dependent，但 view dependent 通常提供較好的 texture。

做 weighting 前必須要考慮到以下幾點:

1. Occlusion:（某些部位只有在特定幾張 image 出現）
2. Smoothess: 轉動時不要跳動
3. Positional certainty: normal 越朝像中間越準確，越朝邊邊越不準確。
4. View similarity: 越靠近此 view 的 weighting 會越高。



View-independent VS. View-dependent

View-independent: 快，但會有 blur 的現象。

View-dependent: 慢，但 feature 會比較像。

Model reconstruction:

Input: 給不同時間點五張照片，一個以及 generic 3D model

Output: 該時間點的 3D model 以及 view-dependent texture。

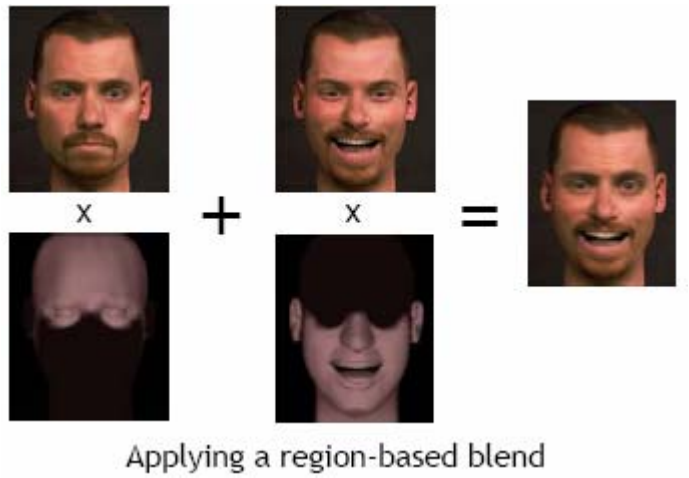
Creating new expression:

Global blending:（每個點都 apply 相同的 weighting）

下面是一個簡單的例子

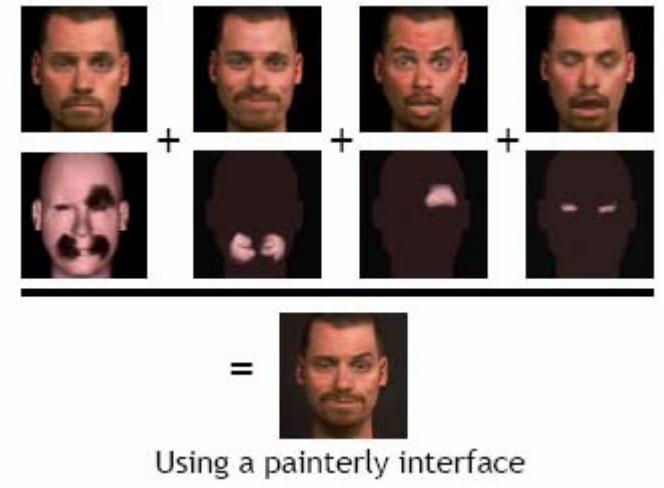
左圖跟中間的圖都各用一半的 weighting，最後再做 blending，可得到右圖這張新的 Expression。

Region-based blend: 下面是一個簡單的例子



我們取左邊的上半部份臉部，和中間的下半部份臉部，可得到右圖新的 expression

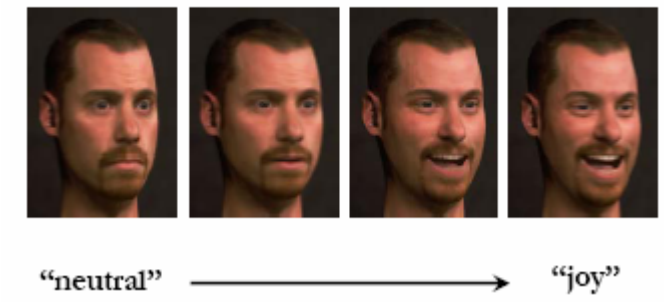
Painterly interface:



取各張 image 我們所要的部份，合成出一張新的 expression。

Animating between expressions

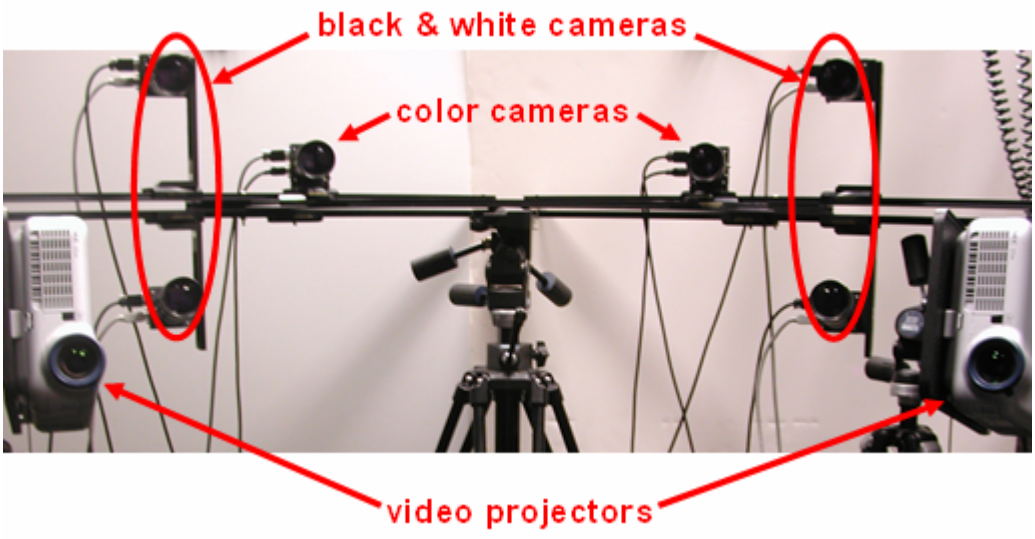
以下面的例子說明:
Input: 一張自然的表情，以及一張開心的表情
我們可以利用 morphing 的方式將中間的表情建構出來。
Morphing over time creates animation:



Spacetime faces:

建立 3Dmodel 的另一種方法（發表於 2003 年），能表現出更 detail 的部份(如皺紋)。(有別於前述發表於 1997 年的方法)

會用到下圖設備，使用 Stereo 的方式建立 3D model。



其中:

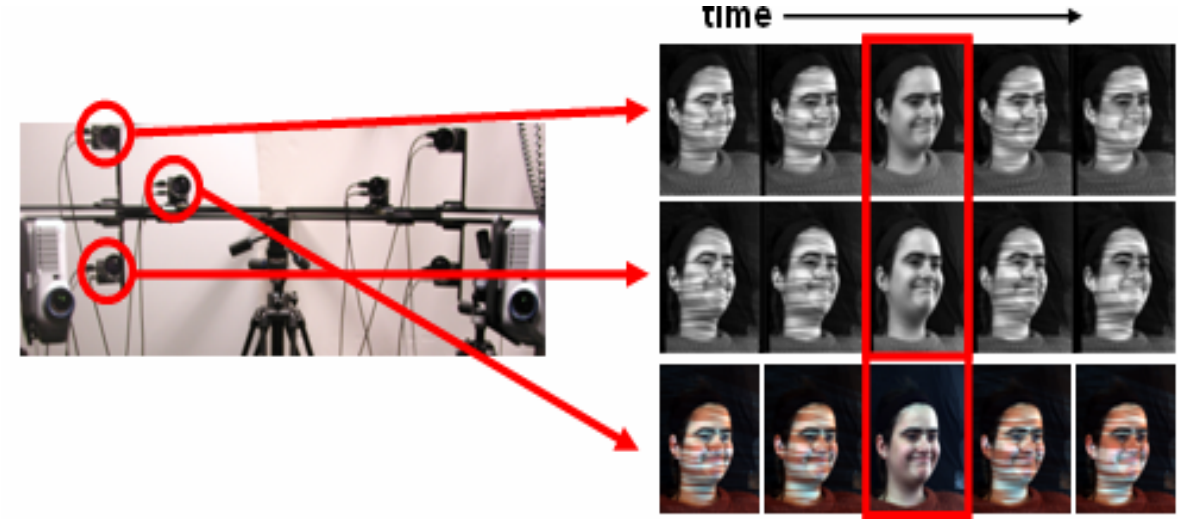
Black & white camera 用來補捉 range image。

Color Camera 用來補捉 texture

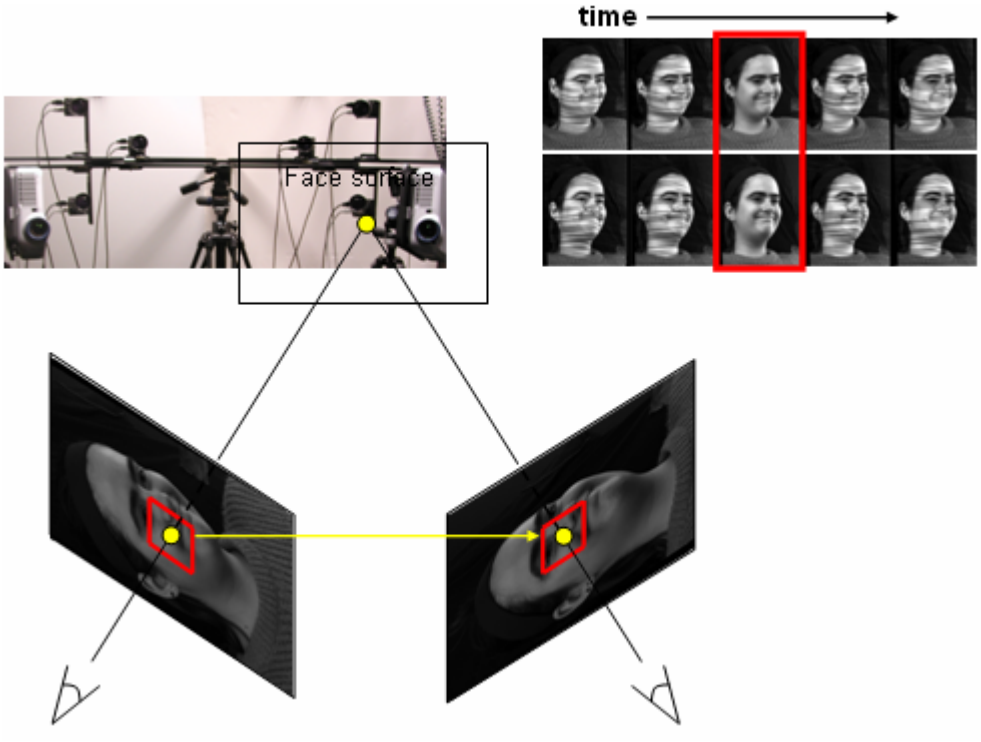
Video projectors 會投影出一些 patter 到人的臉上。

這些 Cameras 均為 60 frames/sec、3CCD、640*480 的 Resolution。

下圖為各個 camera 在不同時間點所得到的不同影像。



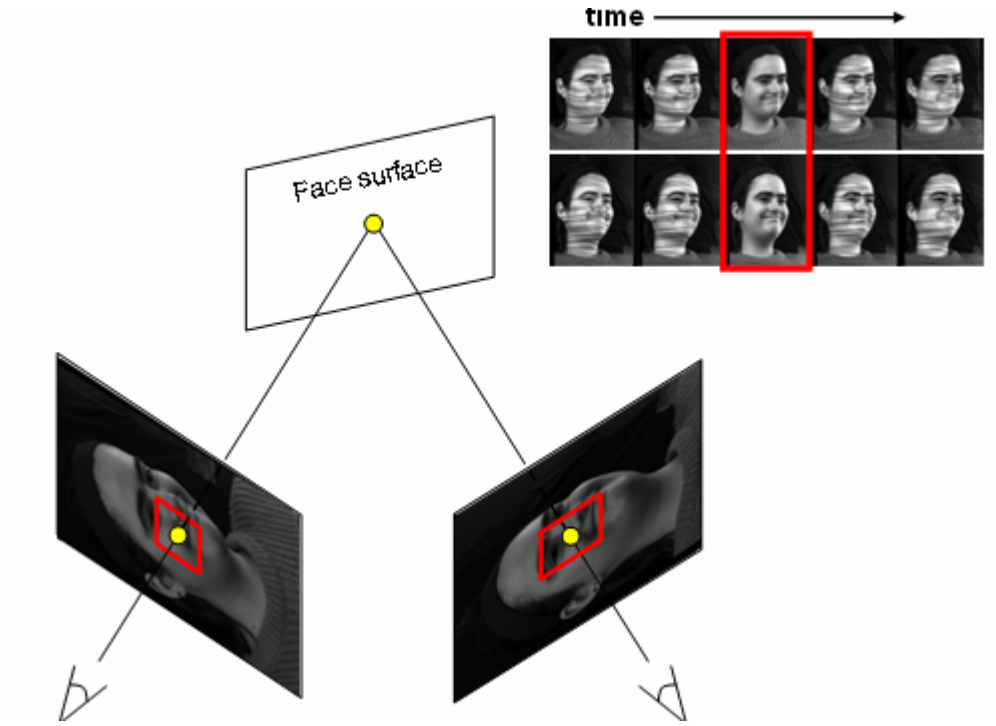
每三個 frames 為一個單位，兩個有投影 pattern 到臉上，一個沒有投影 pattern 到臉上。



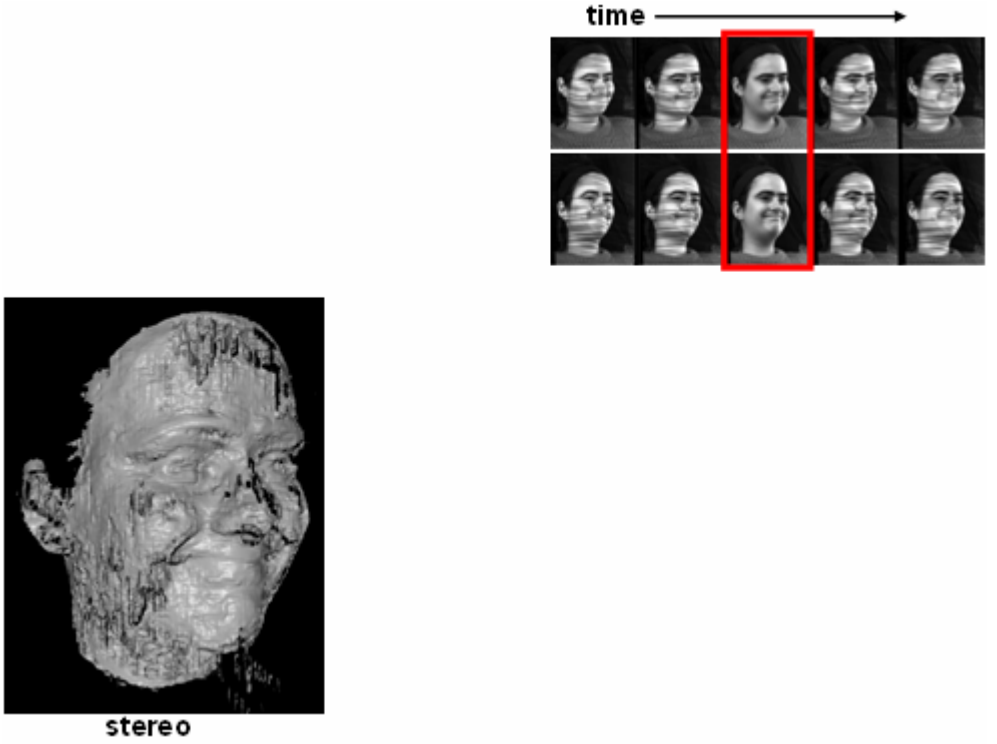
而我們的目的是決定出與沒有投影的那個 frame match 的 3D model。

Stereo:

單純地直接對該 frame 做 Stereo analysis:



找出該 frame 中每個點會形成一個 block，利用 block match 的方式找到另一台像機該 frame 的對應點。 如此一來便可由 2D image 利用 stereo 的方式找出 3D mode。 如下圖所示。

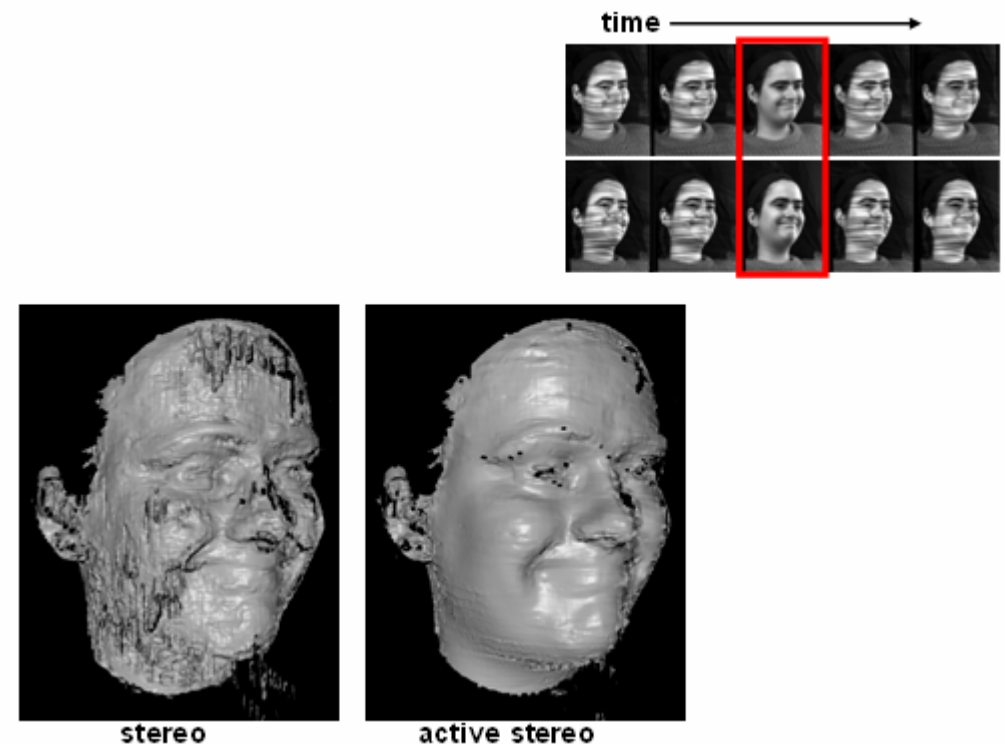


由於在做 block matching 時會有一些 ambiguous 的情況發生，所以結果不會太

好，
我們發現臉部會有許多 noise 的現象， 因此我們需要用投影 pattern 的方式來改善。如下圖所示。

Active Stereo:

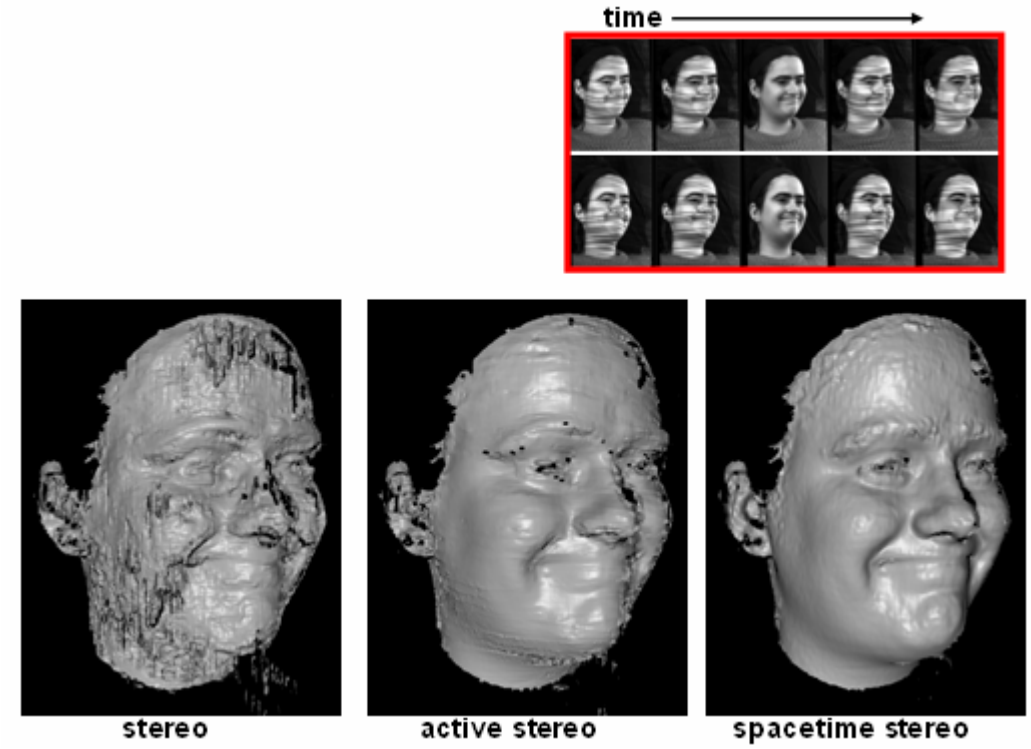
我們會利用投影的 pattern 來找 block 的 match，以減輕誤差。



我們發現結果會好很多。

但結果仍不太滿意，因此我們用 spacetime stereo，方法如下:

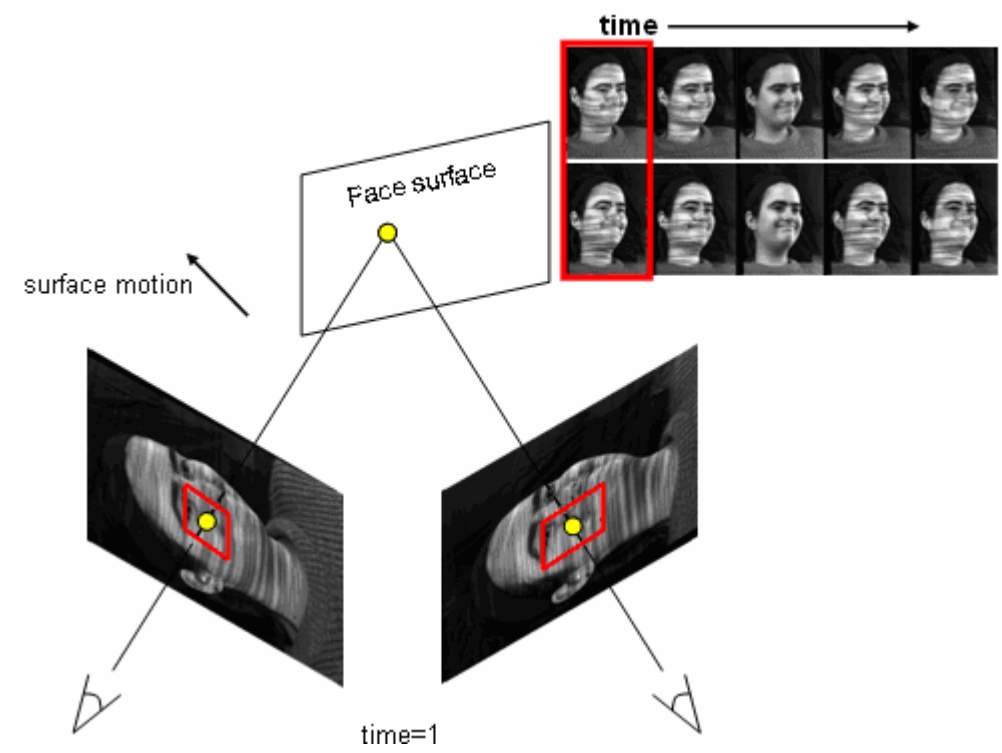
Spacetime stereo:



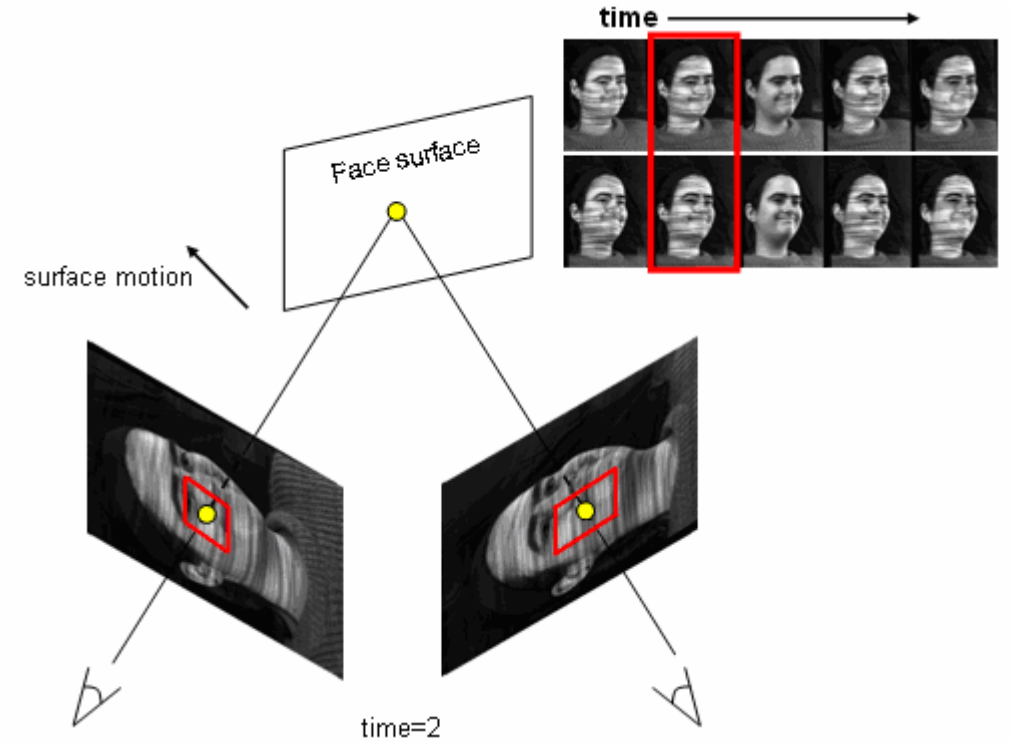
一次考慮 5 個 frame，使效果更好。Temporally 來看，五個 frame 中的 block 都要 match。Ex. 8*8 的 block，則是 8*8*5 的 volume 在 5 個 frame 都要 match。但我們必須要克服掉 motion 的問題。

我們會假定這一個 volume 只是一個 ”orthogonal volume 做一個 affine transformation”
(亦即臉的 motion 為 linear 移動)

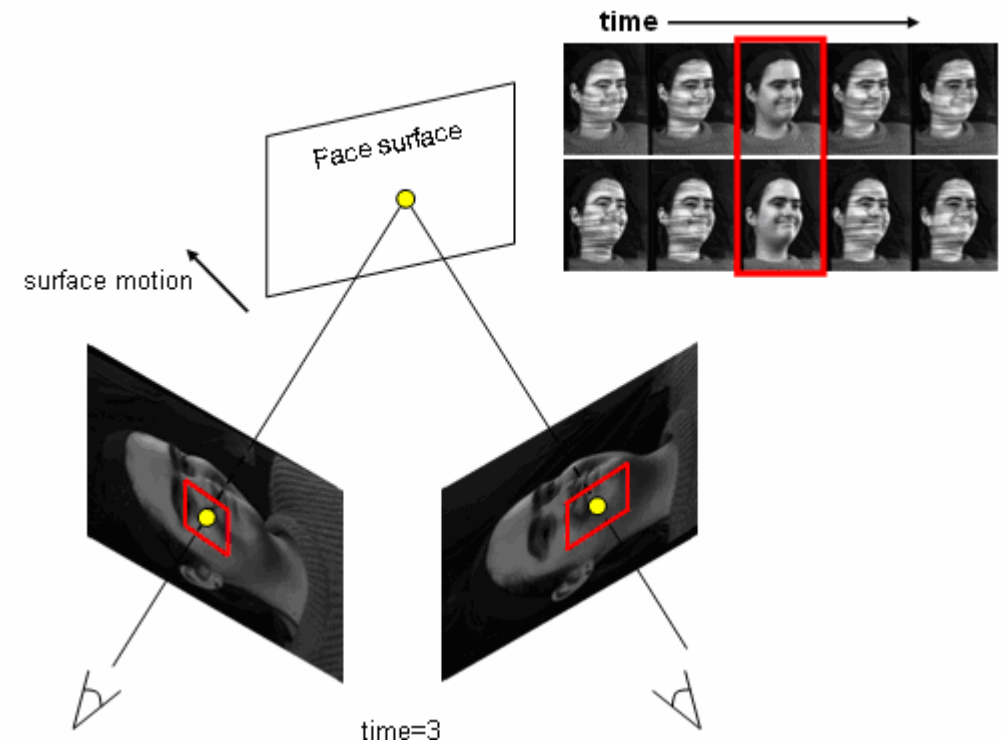
第一個 frame 的對應:



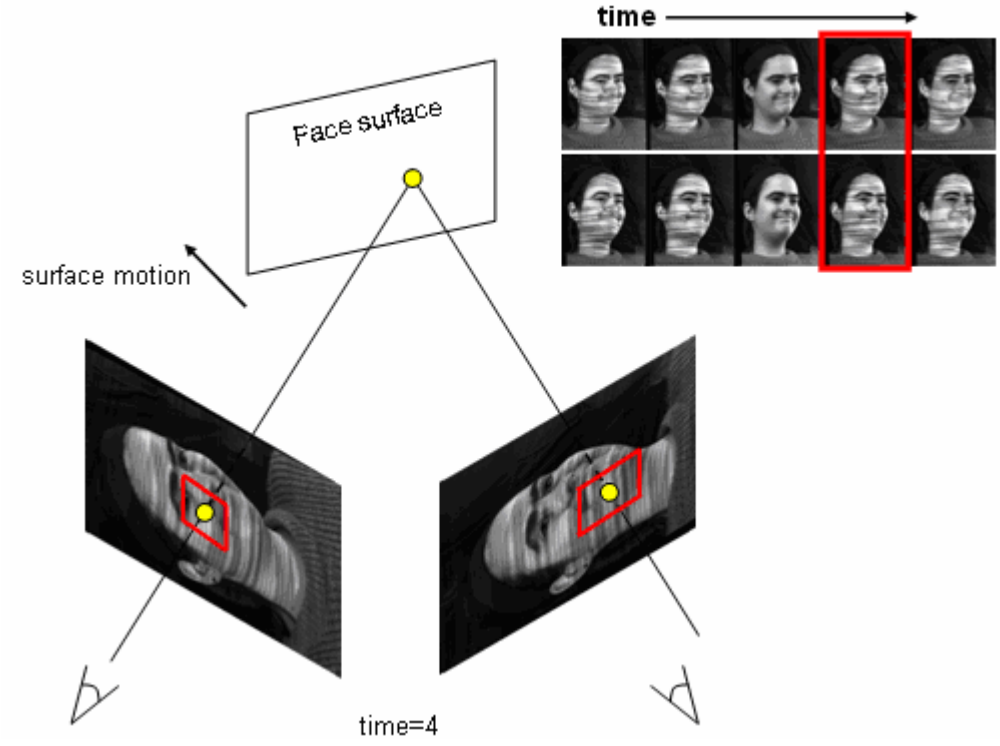
第二個 frame 的對應:



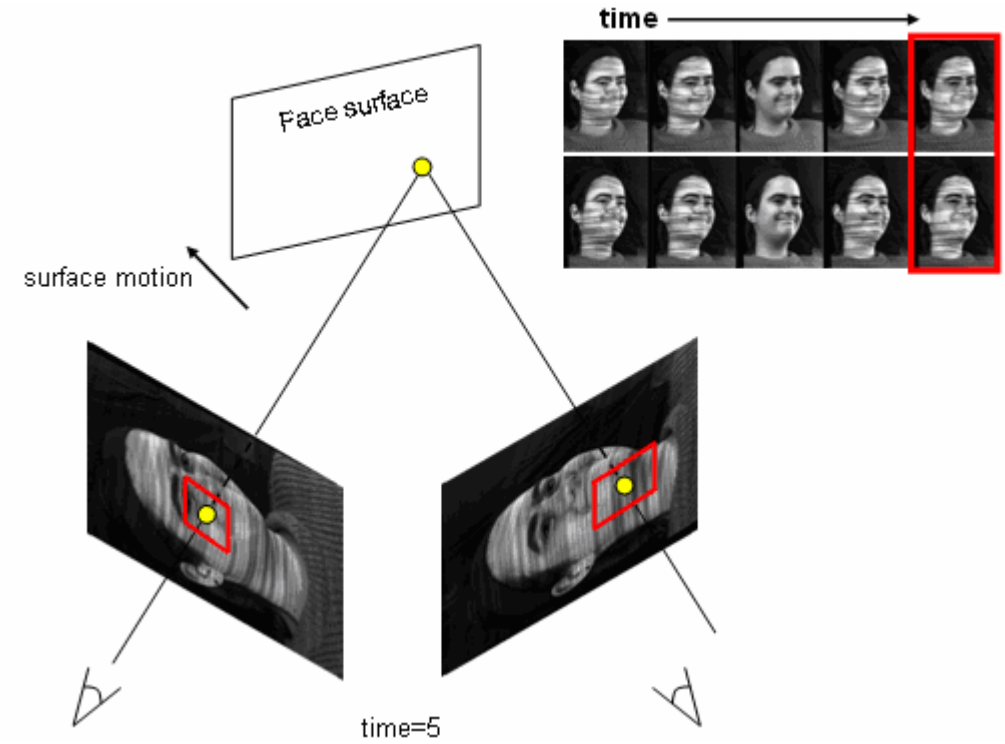
第三個 frame 的對應:



第四個 frame 的對應:



第五個 frame 的對應:



找出對於整個 volume 最 match 的點

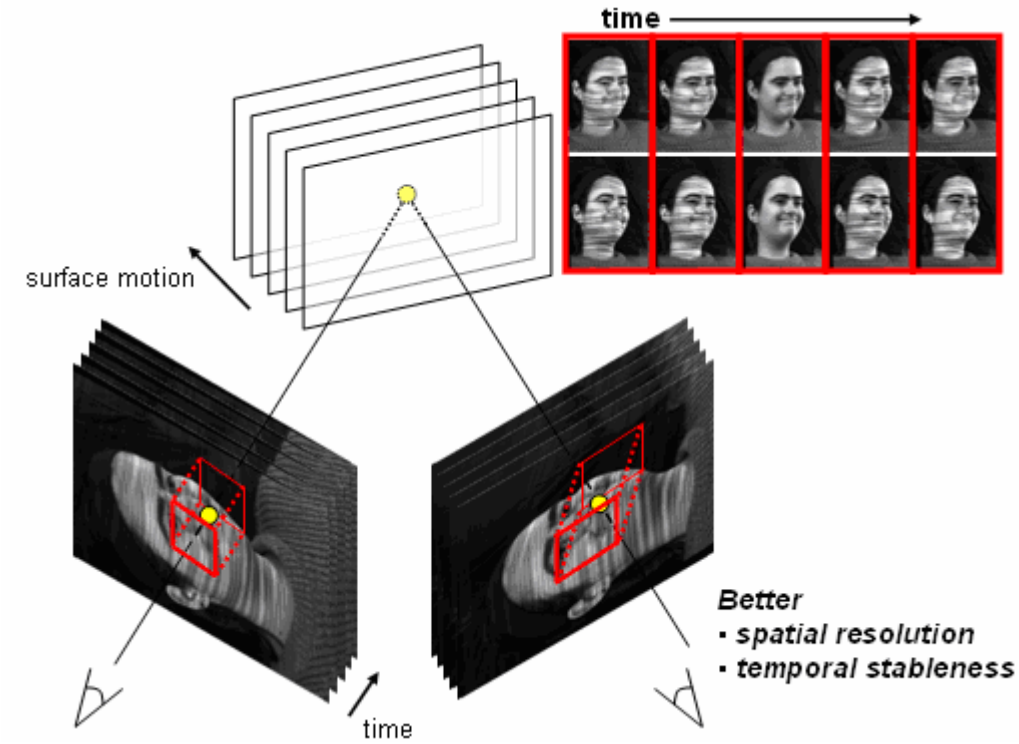
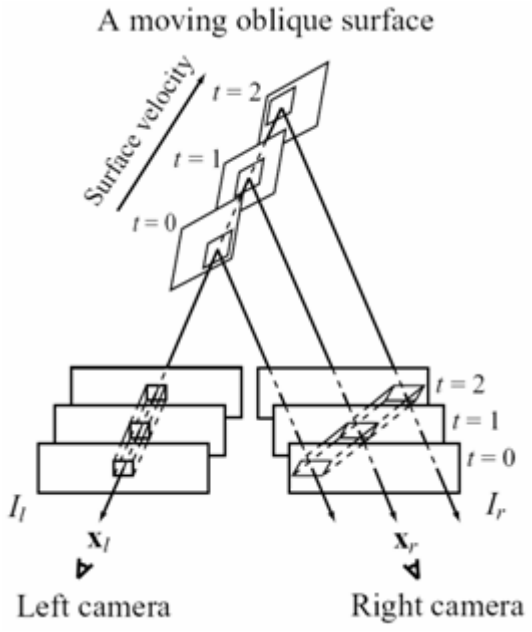


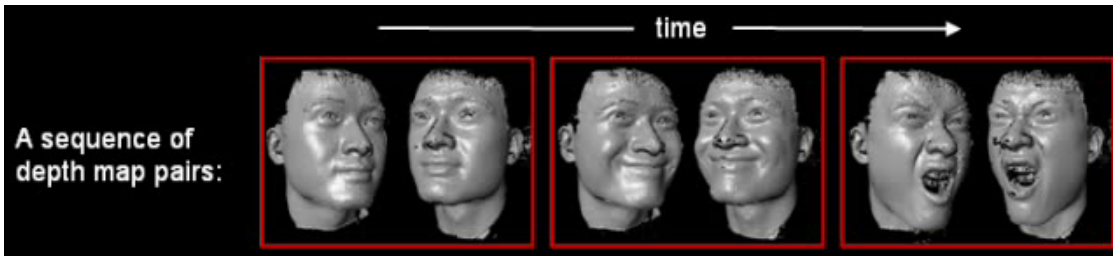
示意圖:



Spacetime Stereo 的缺點: 在這樣的 device 下，人通常不太願意把眼睛張開，因為兩個 projector 直射在臉上。

Fitting:

將左邊與右邊的 range image 合起來



3D face applications :

以下為三部電影用到 3D face 的例子:

1. The One（救世主）

劇情為兩個相同的人對打，左圖為替身，袋上綠色面具，我們將 3D model 覆上即可。

2. Gladiator（拍攝此電影時，該角色人物死亡，後來的部份均以 3D face 解決。）

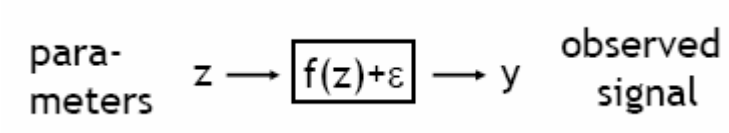
3. Spiderman 2

這整個場景均為 CG 所產生，缺點在於此人的臉部完全無表情變化。
(CYY 按: 因為他快死了)

Statistical methods

是用來解決 ill-posed 的問題。

Ill-posed：



給定一些parameter z,帶入function,再加上noise,得到observed signal y。

要解決的問題是：如何從所得到的 y，recover 回到 z（f 為未知）。

例子：

(1)super-resolution：

input image (parameter)為一個 high resolution image ，例如為 128*128 的 image，經過 down-sampling(使用一個 4*4 的 box，將 box 內的值取平均作 sample 的值)，則我們所觀察到的 image 為 64*64 的 image。所以這裡的 ill-pose 是如何從 64*64 的 observed signal 來 recover 回 128*128 的 parameter。雖然我們知道 forward process，但是我們所要的 parameter 個數比我們所觀察到的個數還要多。

(2)de-nosing：

input 為原來的 image，f 為 jpeg compression operation (DCT、quantization)，得到一張 compressed image，我們所觀察到的就是一張 compressed image，雖然這邊二邊的 image 的 dimension 一樣，但是經過 compressed 後很多 detail 也都不見了，所以這邊也算是 ill-pose 的問題；我們如何從一張 high compressed image 來 recover 回原來的 image

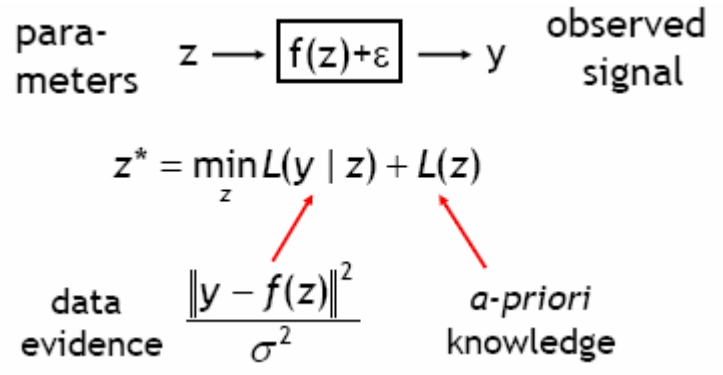
所以 statistical method 是用來解決，如何從 observed signal 來 recover 回原來的 parameter。

$$\begin{aligned} z^* &= \max_z P(z \mid y) \\ &= \max_z \frac{P(y \mid z)P(z)}{P(y)} \\ &= \min_z L(y \mid z) + L(z) \end{aligned}$$

找一個 optimal parameter z，使得 $P(z \mid y)$ 最高。以 super-resolution 為例，給定一張 64*64 的 image y，要找出一張 128 個 128 的 image z，使得 $P(z \mid y)$ 為最高。根據貝式定理知道 $P(z \mid y) = \frac{P(y \mid z)P(z)}{P(y)}$ ，因為對 z 做

quantization，所以 $P(y)$ 跟 optimalize 無關，可以把它當作 constant 忽略掉，再取 log 變成 likelihood，則乘法變加法，因此得到 minimize $L(y \mid z)+L(z)$。所以整個方程式可以寫成 $z^* = \min_z L(y \mid z)+L(z)$。

Data evidence and prior knowledge：



我們要猜一組 z，代入 f，使得 $f(z)$和 y 最接近。

以 super solution 為例，如果觀察到的 y 是 64*64 的灰色的圖(intensity level 都是 128)，則我們會猜 z 可能是 128*128 的灰色 image，也可能會猜 z 是 128*128 的 chess board pattern(黑白相間)的 image。所以我們必須使用 prior knowledge 讓我們的 ill-posed problem 變成 well condition problem。所以如果我們知道原來的 image 是 smooth image，則我們就可以知道 z 是 128*128 的灰色 image 而不是 chess board pattern 的 image。

There are approximately 10^{240} possible 10×10 gray-level images. Even human being has not seen them all yet. There must be a strong statistical bias.

Takeo Kanade

Approximately $8X10^{11}$ blocks per day per person.

如果給一個 10*10 的灰階的 block image，每個 pixel 有 2^8 種顏色，則這樣的 10*10 的 block 有 10^{240} 種變化，如果每一個人一天大約看 $8*10^{11}$ blocks，全球有 60 億人，則全球每天看的 image block 約 10^{22}，所以要多少人多少年才能把所有的 image block 看完。意思是說，image 出現的頻率並不是 equal 的，有些 image 可能一輩子都看不到，有些 image 出現頻率會比其他高很多，所以我們傾向於相信有些 image 是好的 image(就是我們想要的 image)，有些是壞的 image(大致都不會出現的 image)，所以我們才用 statistical method。

Generic priors:

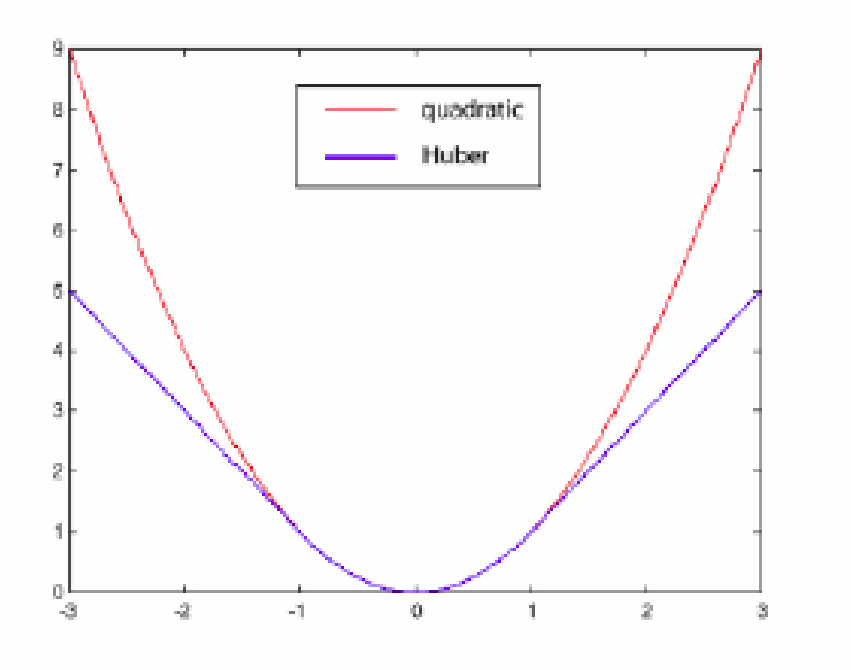
"Smooth images are good images."

$$L(z) = \sum_x \rho(V(x))$$

Gaussian MRF $\rho(d) = d^2$

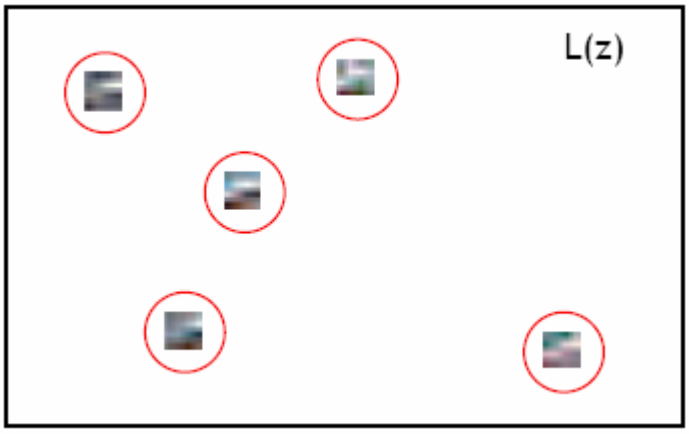
Huber MRF $\rho(d) = \begin{cases} d^2 & |d| \le T \\ T^2 + 2T(|d| - T) & d > T \end{cases}$

所以我們知道現在有好的 image 和壞的 image，則我們如何來 encode 好的 image，最常用的就是 generic priors。因為什麼都不知道，所以先假設 smooth 的 image 就是好的 image，想辦法去 encode z，$V(x)$去考慮和鄰居間的 frequency，通常我們會用 Gaussian MRF 和 Huber MRF(比較 robust)。但是 generic prior 得到的結果不是很好，比較模糊。

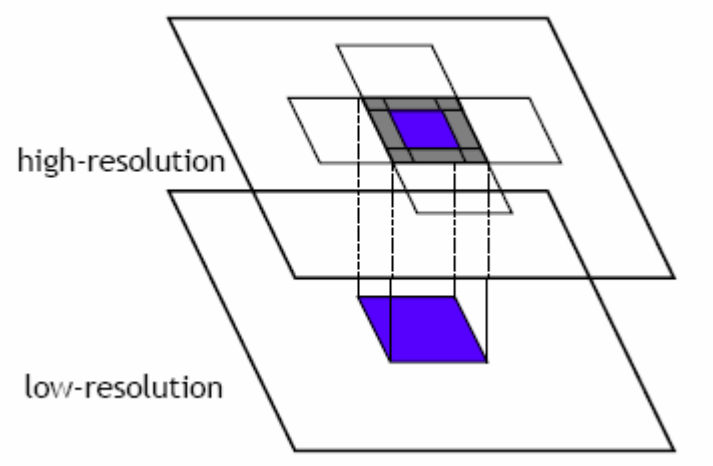


Example-based priors:

假設有出現過的 image 就是好的 image，則怎樣 encode 一張曾經出現過的 image 就是好的 image，在 super-resolution 的應用：去收集 6 張 200*200images，當中只要出現過的 image block 就是好的 image block，所以去生成 image 的時候，那些 block 最好是出現過的。



以 super-resolution 為例子，先給 low resolution 的 blocks，利用六張的 200*200 的 images 我都可以 down-sampling 得到 low-resolution，所以我們得到 high-resolution 對 low-resolution 的 care，我們就去找跟我們 low resolution 的 block 比較像的，然後將所對應的 high-resolution 當做這塊的 high-resolution，如果是一塊一塊去生成，必須要考慮 overlap 的問題，其 overlap 也要越相近越好。



Model-based priors:

如果現在要處理人臉的 image，那人臉的 image 就是好的 image，狗的貓的樹的 image 就不是好的 images。Model-based prior 基本上說你有一個 club，將所有屬於這個 club 的 image 產生一個 model，來 summarize 這些 image，最常用的就是 Gaussian model。Gaussian model 基本上是對這些 images 做 PCA。

"Face images are good images when working on face images ..."

Parametric model $Z=WX+\mu$ $L(X)$

$$z^* = \min_z L(y \mid z) + L(z)$$

$$\begin{cases} X^* = \min_x L(y \mid WX + \mu) + L(X) \\ z^* = WX^* + \mu \end{cases}$$

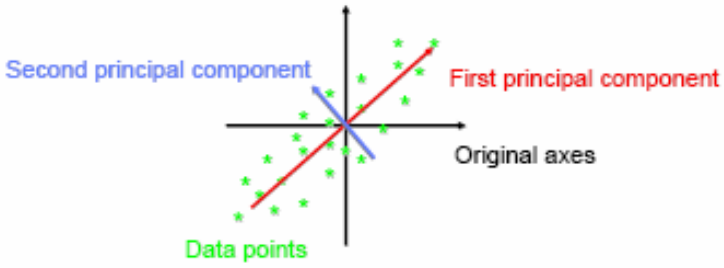
如果我們知道這張是人臉的照片，$L(z)$ 是 encode 怎樣的 image 才算是人臉的 image，基本上會從一堆人臉的 training data 求出 parametric model，然後利用 parametric model 來 evaluate 任何的 input image 是否為人臉的 image。我們用 PCA 來做這件事情，我們可以想像所有的 image 都是 high dimensional vector，然後 PCA 會把它 project 到 low dimensional 的 space 上去，可是這 low dimensional space 還是可以很真實的表現出我原來的 data，所以可以像是我用比較 compact 的技術來表現比較複雜的 image，所以每個人臉的 image 都可以用比較少量的技術 S 去生成，Z 是人臉的 image，S 就是這些技術(low dimensional)，例如我們將 1000 張的 128*128 的 image (Z)作 PCA，可以 reduce 到 20 dimensional 的 space，則 S 就是 20 dimensional 的 vector，每個 20 dimensional 的 image 就可以生成類似人臉的 image，所

以只要有 S就可以生出 Z，在作 PCA 的過程，同時也把這些 20 個 dimension 的每一個 component 可能出現的機率都有 prior model，所以現在換成我的 prior model 是對技術 S的 prior model，一但有 S就可以生出 Z，所以我們基本上把 Z換成 S，所以整個是對 S來做。Z 可以寫成 S 的一些 combination。

PCA:

Principal Components Analysis (PCA)

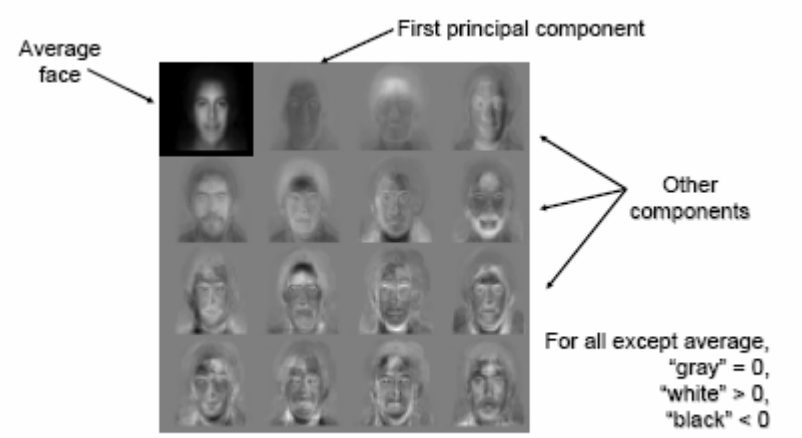
- Principal Components Analysis (PCA): approximating a high-dimensional data set with a lower-dimensional subspace



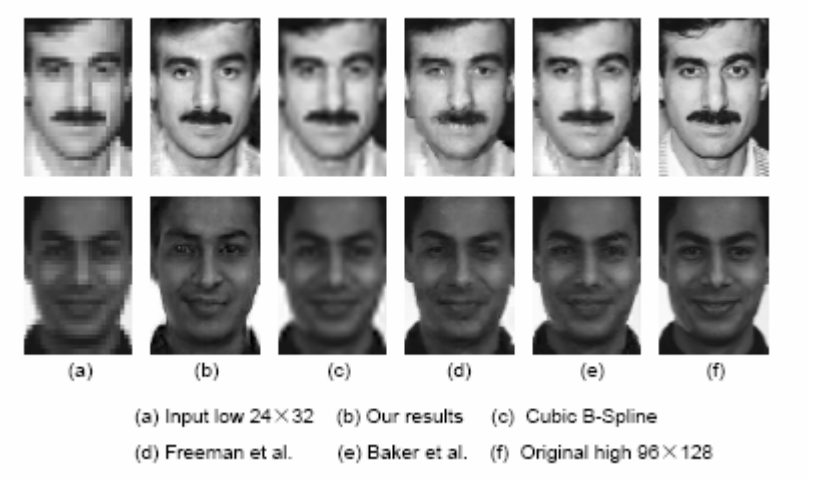
以 2d 為例子，給一堆綠色的 data points，如果我們用原來的座標軸來描述他，會發現不是 exactly 的描述方式，我們想用更 compact 的方式來描述他。這 2d 的 data，我們希望用 1d 的技術來描述，把這些點投射到 1d 上，取任何一軸都無法很真實的呈現這些 data。那我們應該怎麼找座標軸，經過 projection 後還能很真實的呈現原來的 data。最簡單的方式是先找一個軸，使得其他點投影到這軸的 variance 最大，接下來找 variance 第二大的且跟原來的軸垂直，然後我們就將綠色的點轉到 first principal component 和 second principal component 所描述的座標系裡面，我們可以發現這座標系能更真實的呈現這些 datas。所以如果在 n-dimension 系統，做過 PCA 之後，我們只要把 variance 小的軸都拿掉，就可以將 data 做到 dimension reduction 的動作。可以把它想成是一個 Gaussian function，把這些點算出一個 Gaussian function 來描述這些 2d 的點，所以作 PCA 跟做 Gaussian function 基本上是一樣的。

下面是作 PCA 的步驟：

- Given n k-d points
- Calculate the mean
- Calculate the covariance matrix
- SVD (eigen-analysis) on the covariance matrix



現在要做人臉的 image，那怎樣的 image 是好的 image，先給 training data，我們求出這些 image 的 mean vector，就是上圖的 Average face，我們用 PCA 算出 principal component，我們取 15 個 principal component，所以對所有 128*128 的 image 我們都可以用這 15 個技術來表示，所以一張圖如果是 A1~A15 的技術所組成，則此張圖的表示方式是 mean + A1*(第 1 個 component) + A2*(第 2 個 component) + … + A15*(第 15 個 component)，這這張 image 就可以很接近原來輸入的 image。為什麼取 15，是因為我們觀察 energy decade，發現到某個值之後 variance 都很小，意思是說這幾個 feature component 沒有什麼太大的影響，也就是說本來 128*128 的 image，本身 dimension 就沒那麼高，因為之間有強烈的 coherence，所以用 15 張就可以 cover 所有的 variance，所以 PCA 的意思就是這樣。

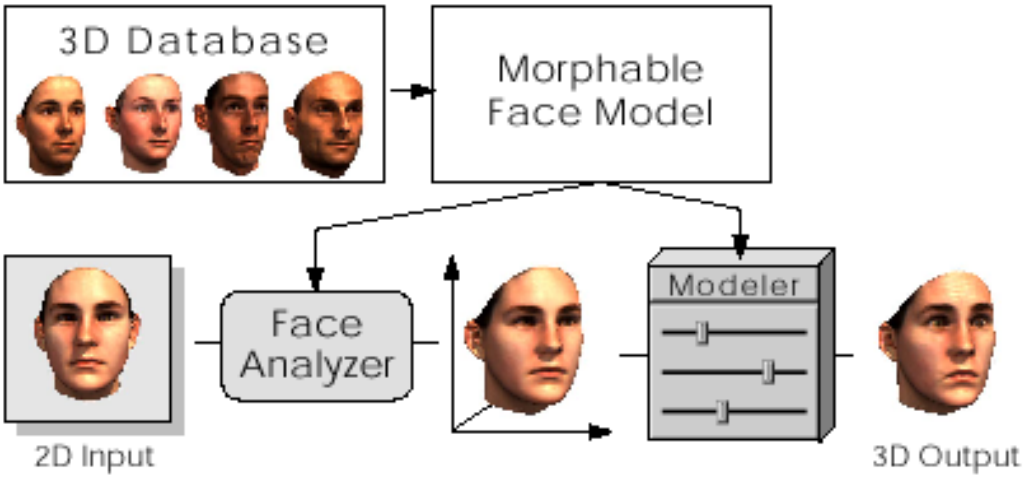


上圖是對人臉做 super resolution，然後使用三種 priors 的結果比較，(a)為 input low resolution image，(f)為真正的 high resolution image，基本上是每個方法將 low resolution 求出自己認為的 high resolution image。(c)是用 generic prior 做的結果，所以得到的結果比較模糊，(d)(e)是 example-based prior 得到的結果，但是其結果，有的時候人臉的地方不

像人臉,例如鼻子的地方有很奇怪的鼻線之類的,因為它只考慮 local 的 face 沒有考慮 global 的部份,(b)是 model-based prior,基本上它是 model-based 加上 example-based 在 global 部分，他用 PCA 算出人臉大概應該長什麼樣子，所以產生出來的 image 一定要長的像人臉，不能脫離人臉太遠，可是在 local 的 face 他用 example-face，所以結果比其他三張的結果更像人臉。

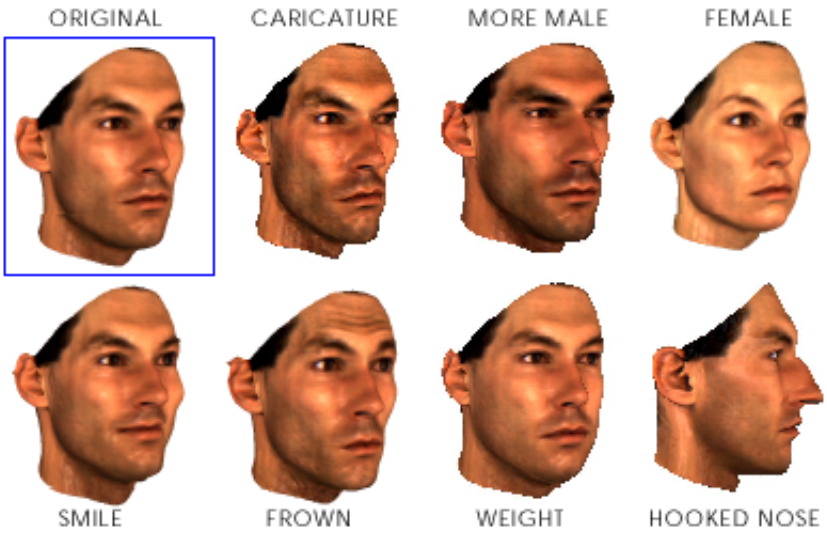
Face models from single images

用 ranging scanner 去 scan 200 個 ranging image，100 個男生 100 個女生，同時取得其 texture map，以此 200 個 ranging images 建立一個 prior model，每一張 ranging image 都是一堆 features 的集合，例如眼角,嘴角這些明顯的地方，此 model 即為 morphable 的 face model。



接著 input 一個 2D 的 image，我們就利用這個 morphable face model，用上面那兩百個 model 以 2D features 可以找得到對應點，以及可以產生出一個完整的 mesh 的情況下，即可產生一個 face model，其可以 match 我們的 input image，且是一個完整的 mesh。

然後我們可以對此 face mesh 做一些 morphing 的變化,使用此 model 的一些 gain, ranging image, image texture，用 PCA 去求出一個該 face shape 的 prior model，我們即可用此方法來調整一些參數，使得我們的 face model 可以做出一些表情，甚至可以讓這個人變胖變瘦等像是整型的改變。



接下來講到有關數學理論的部分

使用我們的 200 張 ranging images，將每張 image 都 align 好，每一個相對應的 feature 都互相對應好，每一個 ranging image 就變成了 vertex 所形成的集合，有 n 個 vertex 就是 n dimensional or 3n dimensional(彩色的話)的 vector，所以總共就有 200 個 images，每個 image 的 dimension 是 3n，所以我們可以將他看成是有 200 個 3n dimensional 的 points，我們即可用 PCA 的方式去算出這 200 個 points 的 mean 以及 principal component，假設我們取其中的 m 個 components，我們有了 mean face shape 和 m 個 face shape 的 principal component，所以每一個 face mesh 都可以投影到這個 m dimension 的 vector 上面去，所以只要給你一個 m dimensional 的 vector，我們都可以求出其對應的 ranging image。

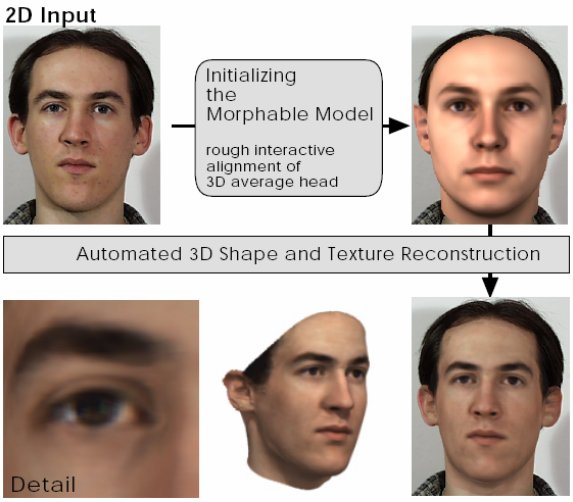
$$S_{model} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i , \quad T_{model} = \overline{T} + \sum_{i=1}^{m-1} \beta_i t_i , \qquad (1)$$

$\vec{\alpha}, \vec{\beta} \in \Re^{m-1}$. The probability for coefficients $\vec{\alpha}$ is given by

$$p(\vec{\alpha}) \sim exp[-\frac{1}{2} \sum_{i=1}^{m-1} (\alpha_i / \sigma_i)^2], \qquad (2)$$

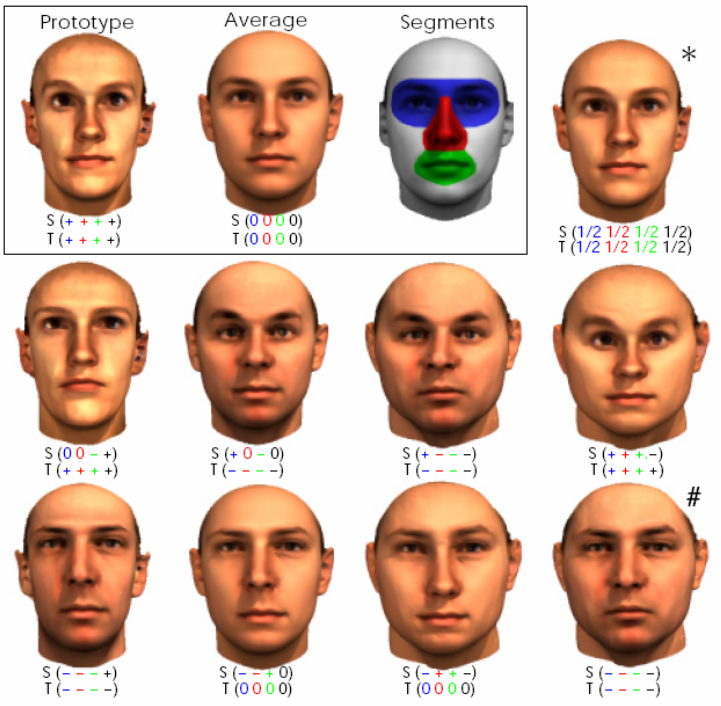
每一個face image都是一個mD的vector，係數即為α_1到α_m，s_i 就是第i個shape的principal component，α_i 就是對應到該component的coefficient，而β_i代表其相對應principal component的texture的coefficient，而且α有他自己的probability function，之前我們有用SVD算出其singular value，而singular value所對應的就是他的covariance，所以對每個component我們知道他的係數α

$_i$的分佈可已有多廣，對於每一個α，我們都可以得到其prior model，這就是我們要的morphable model，以此便能對於我們所input的face model加以變化。

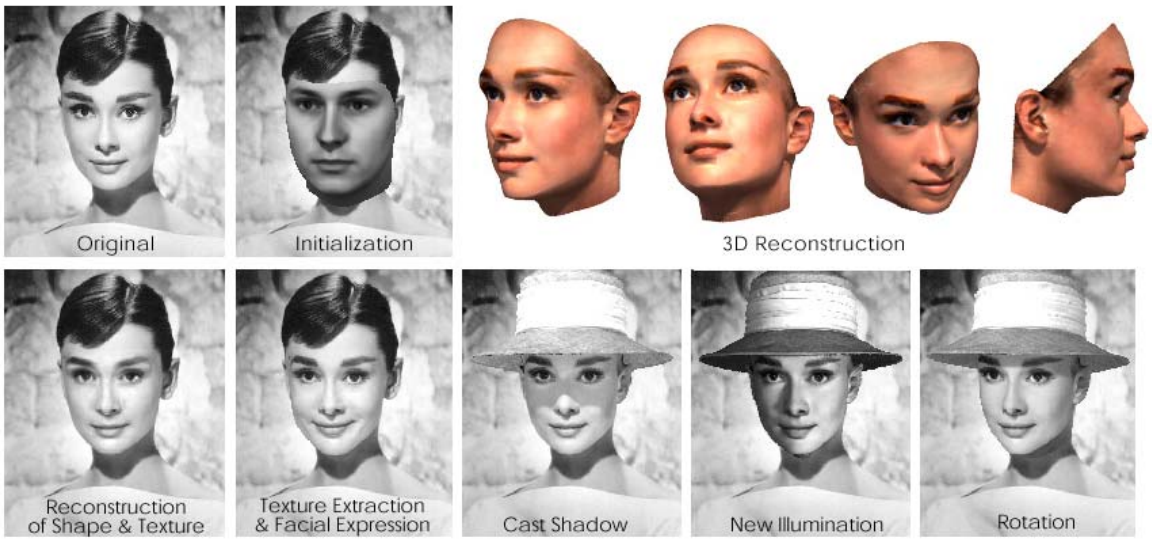


如果我們先以四個 features 為例的話，Divide face into 4 regions (eyes, nose, mouth, head)，For each new *prototype*, find amount of deviation from the reference shape and texture。

我們可以在這四個 features 上標上一些特性，再以 PCA 的方法來產生 face model。



下面這個奧黛麗赫本是另外一個例子，input 一張她的 2D image，就可以重建她的 3D model，然後我們就可以任意的對這個 3D model 做一些變化，像是加上帽子，或是笑得更開一些。



下面這個例子是將兩個真實世界裡的人臉，套用在電影海報上面或是畫像上面，讓人們真的可已變成畫裡面的男女主角。

再來這個應用是用在髮型上的變化，如果你不知道自己在什麼樣的髮型下面會變

成什麼樣子，或是想要預覽一下自己剪完頭髮之後的樣子，就可以使用下面這個技巧。

最後一個應用是在人的體型上面的應用，我們可以用相同的方法，對於人的體型也建立出一個 morphable 的 body model，來模擬我們所要呈現的 body model，如此一來，我們便能用一些簡單的參數來控制高矮胖瘦以及一些身體的特徵，或是做出一些特別的動作。

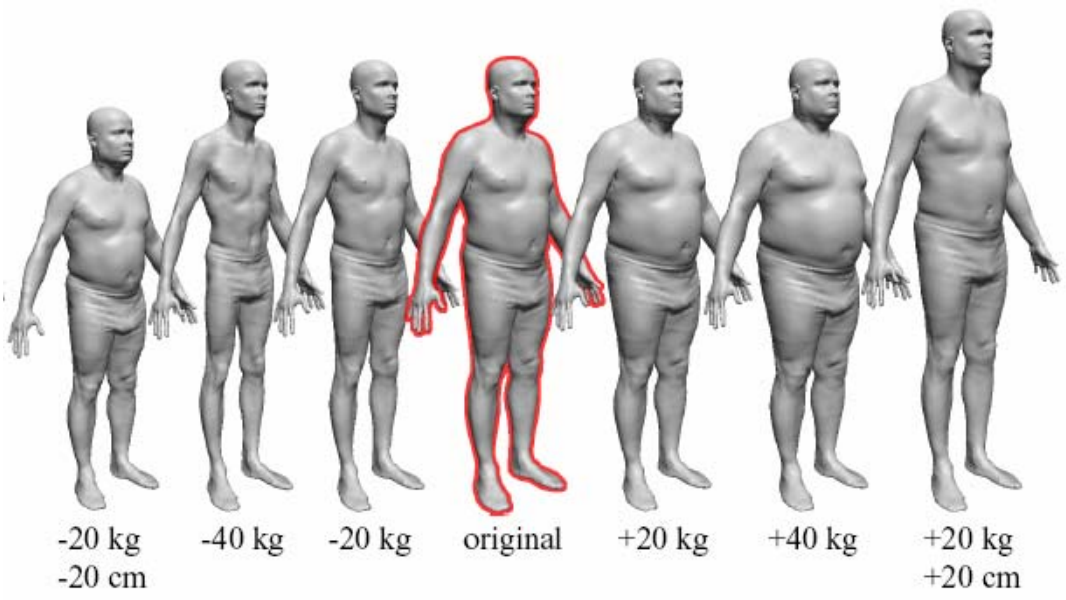
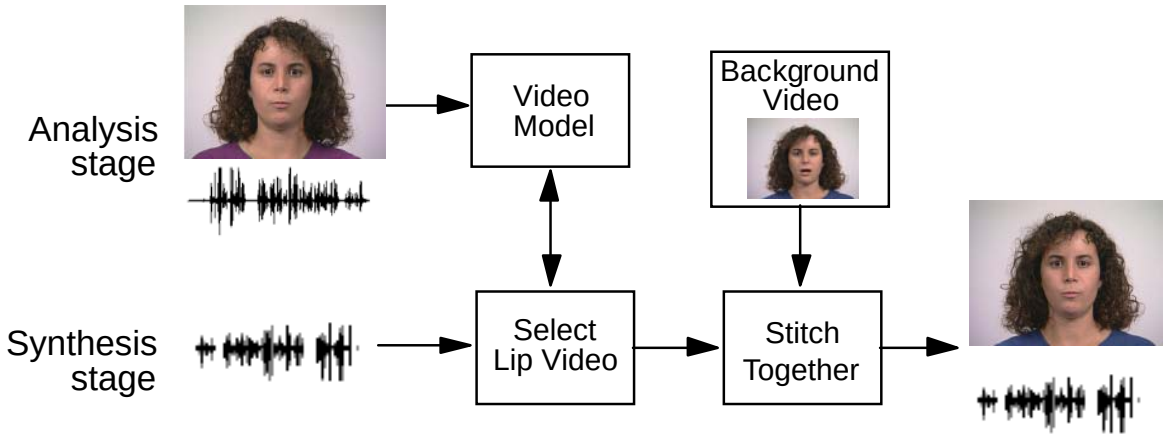


Image-based faces （lip sync.）

有了之前我們討論的 3D mesh 之後，我們想要讓一個人講出他沒有說過的話。首先 input 一個 training video，將此 video 分成影像跟聲音兩部分，對語音做 segmentation 將語音分成很多個音節，並且將其音節跟其對應的 video 記錄下來，接著 input 一段新的語音，也將此新的語音分成多個音節，對於每個新的音節，我們到之前建立的 database 裡去找出最像的音節，然後將此音節所對應到的那一小段 video sequence 找出來，將這些小的 video 連接起來，就可以得到我們的 output sequence。

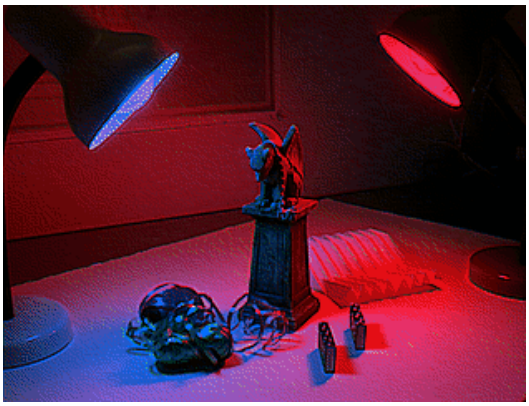


因為人是會動的，所以還要加上一些 global motion estimation，像是頭整個會動嘴巴也要跟著動，整個看起來會更自然一些。而且人們對於語音跟嘴巴的對應其實容忍度還蠻高的，像是有些韓劇用中文配音我們也覺得還蠻搭的，所以這個 research 有些時候 input video sequence 不是很長，sample 沒有很多，不過整體看起來會覺得也都還蠻順的。

Relighting faces

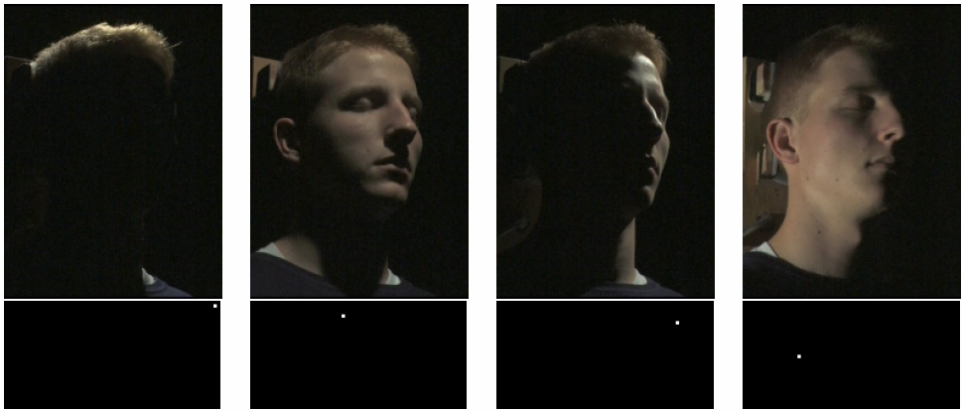
face relighting 的原理很簡單，因為光是可以相加的，像是下圖，如果你將左邊的燈亮起來會得到第一張照片，如果將有邊的燈亮起來會得到第二張照片，於是如果要同時將兩盞燈都亮起來，其實很簡單，就是直接將這兩張照片相加即可，如果光源是有顏色的光，其加成的方法也是一樣的。



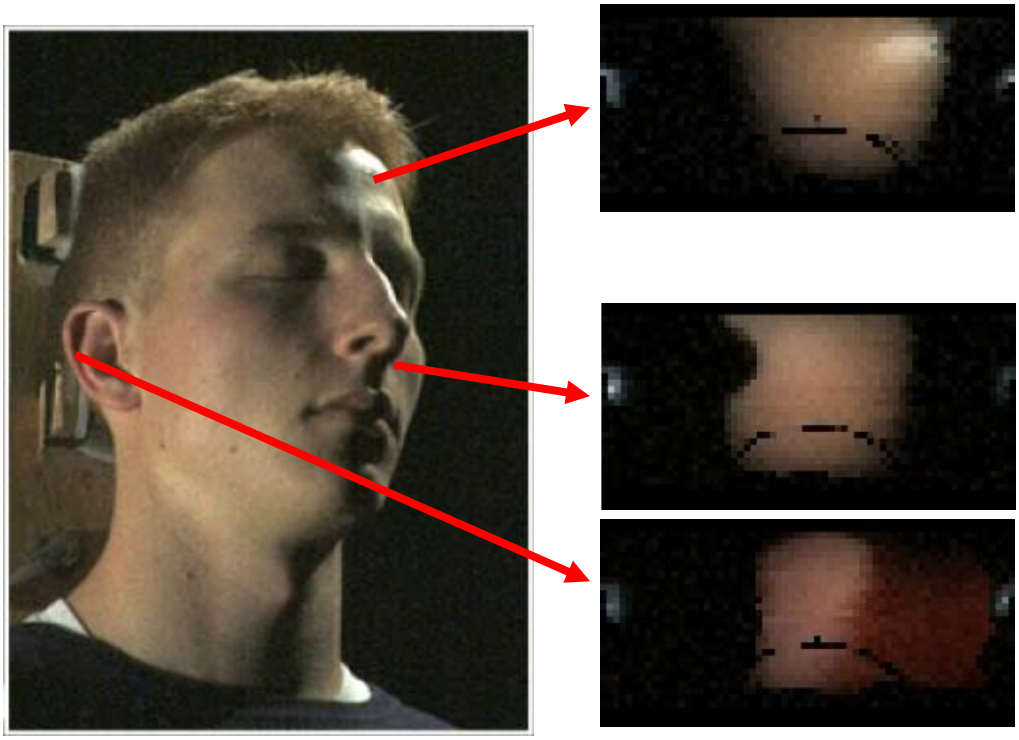



接下來這個方法就是建立在上述的性質上面，我們可以將人臉或是人，放到一個我們所想要的場景裡面。操作的過程中，我們會用到一個 device 叫做 light stage，在 light stage 上面有三個 cameras，人要坐在裡面一段時間靜止不動，然後 camera 會依照不同的光源做各個方向的旋轉，拍下兩千多張照片當作 database。


利用這兩千多張的照片，我們就可以對任何的場景做 relighting，下圖為這兩千多張照片的其中幾張，照片下方的白點代表的是光源的方向。所以我們記錄了當燈在某個方向的時候，整個 model 的每個位置的顏色是什麼，



下面這張圖我們可以看到會有一些黑影(occlusion)，這是 light stage 本身的鐵欄杆的影子，以及白影(flare)，這是當光源正射到 camera 是所造成的現象，所以我們要將上述兩個現象給除去。



於是我們就可以模擬我們的 model 在任何一種光源的場景下面所呈現的樣子，即使是多個不同且複雜的光源下也沒問題。

最後我們要談的是有關如何測量出現在我們的場景中有哪些光源，該如何模擬這些資料，我們所會用到的是一個鏡面球形的東西，它可以幫我們收集環境光源，好讓我們方便模擬這整個場景。