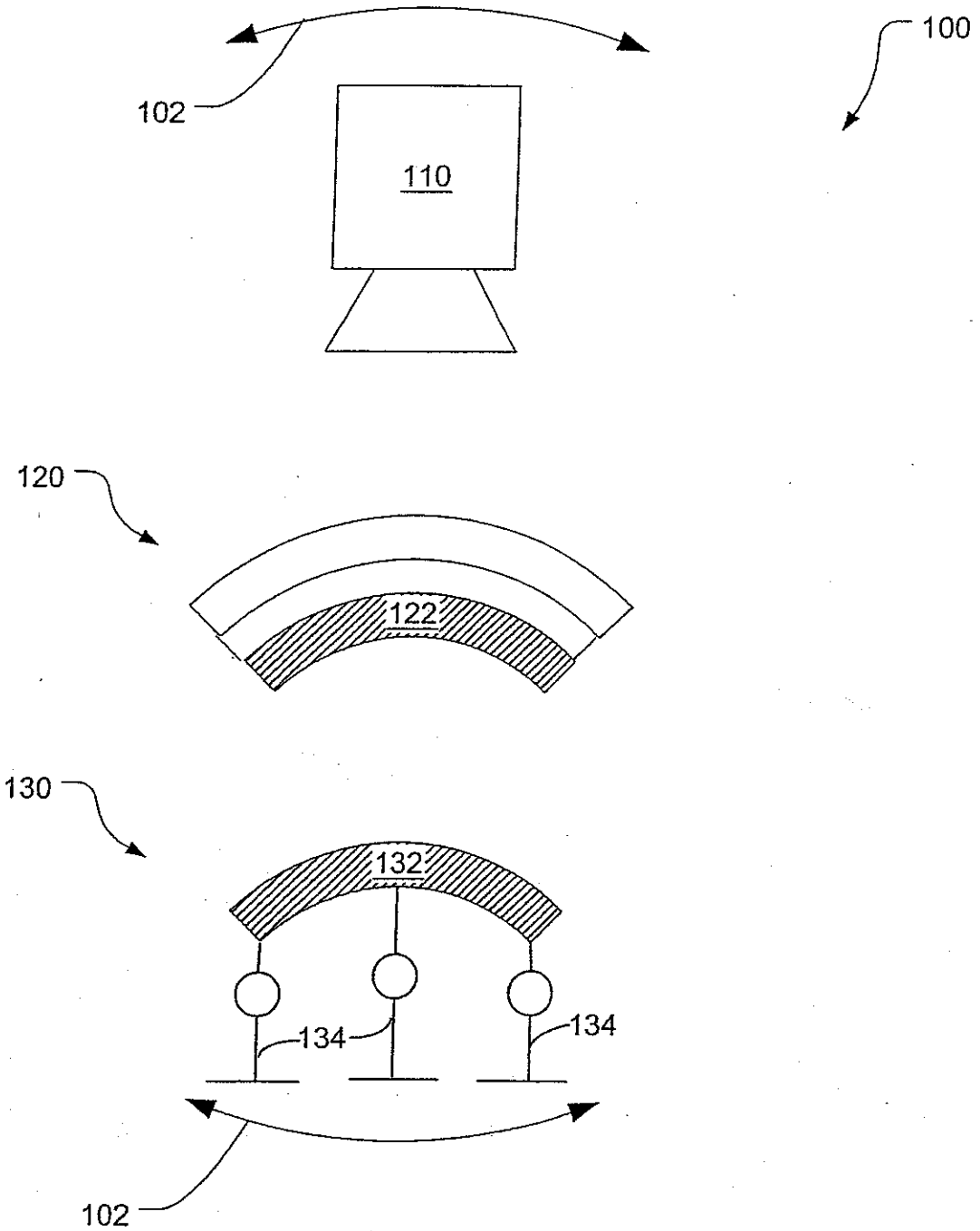


FIG. 1

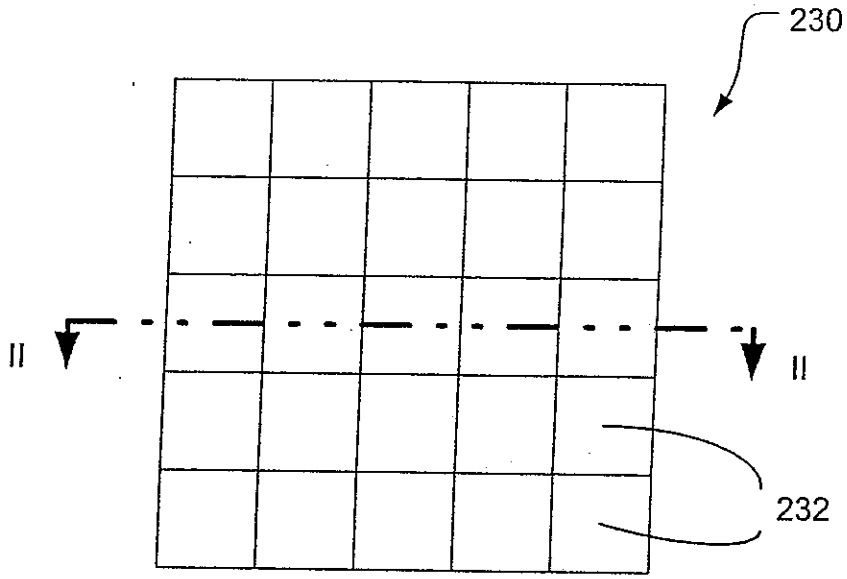


FIG. 2A

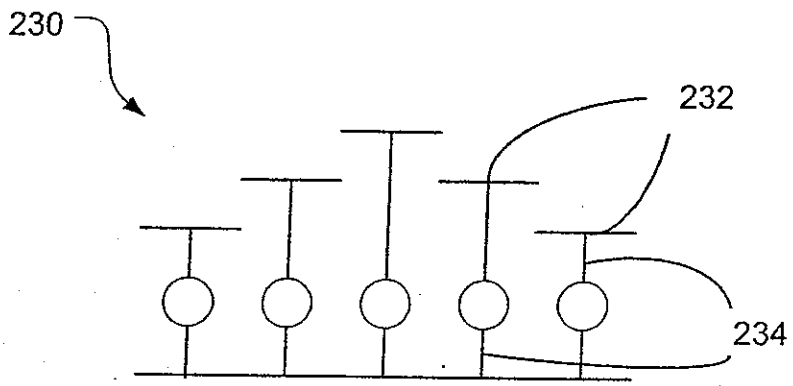


FIG. 2B

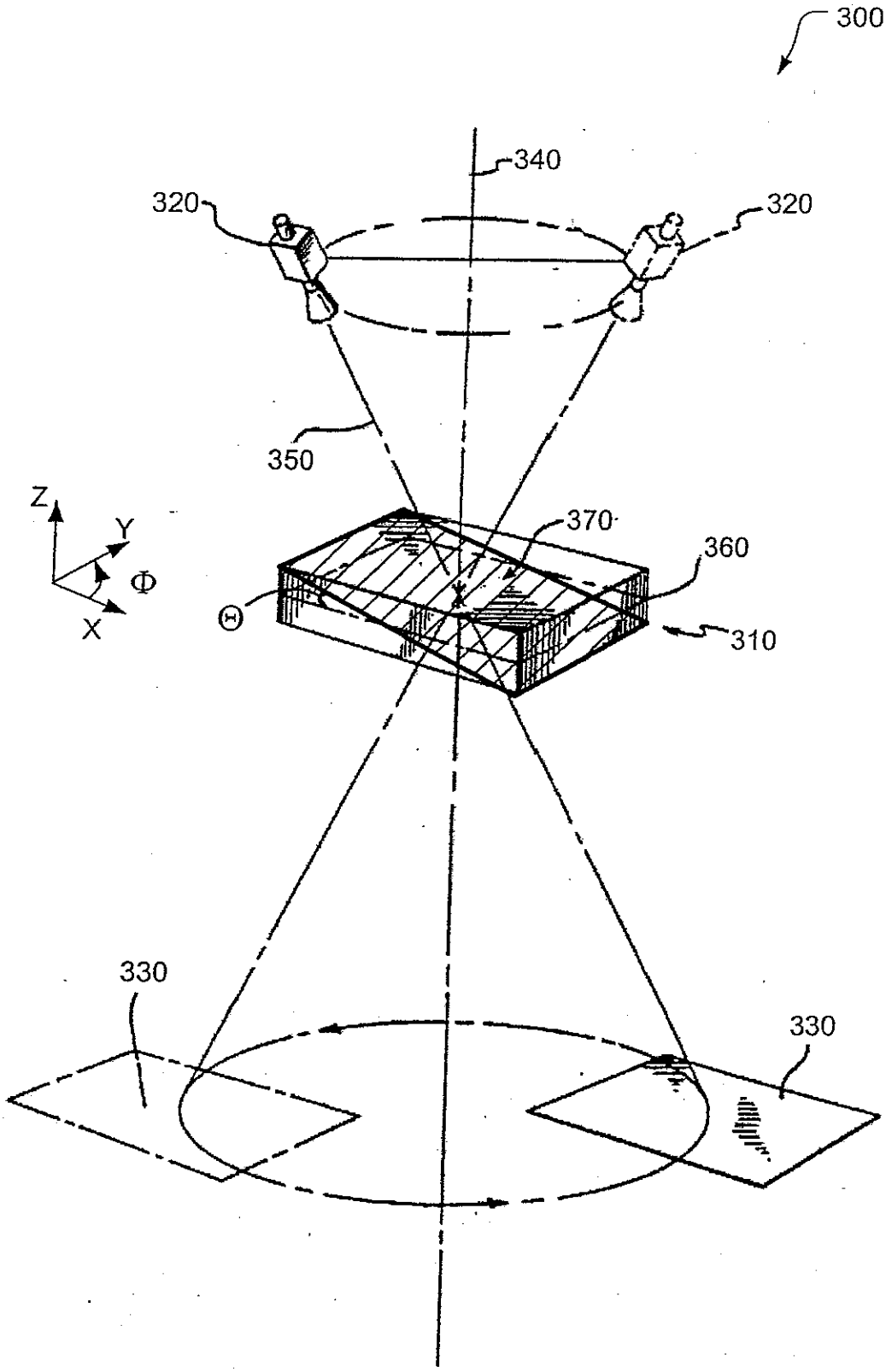


FIG. 3

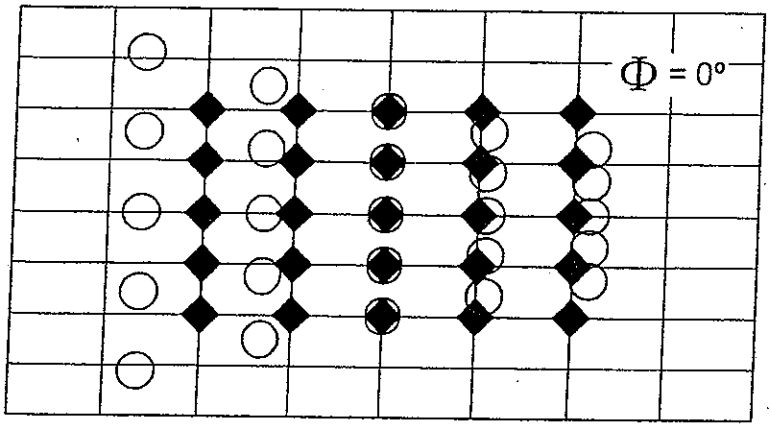


FIG. 4A

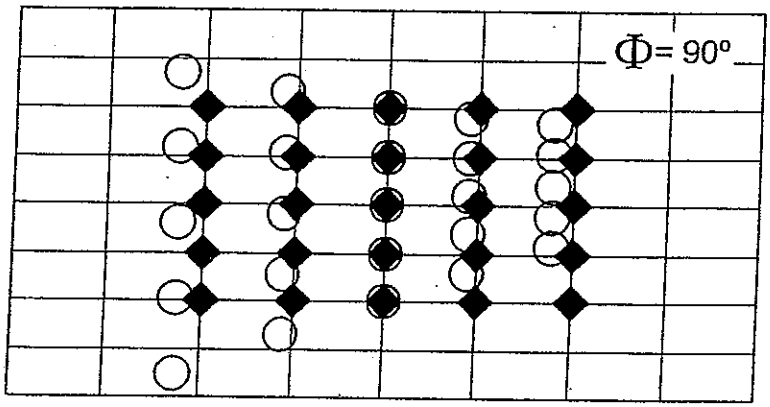


FIG. 4B

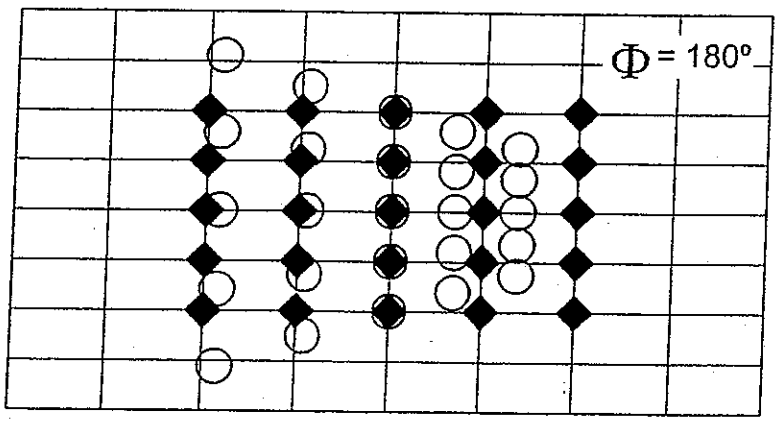


FIG. 4C

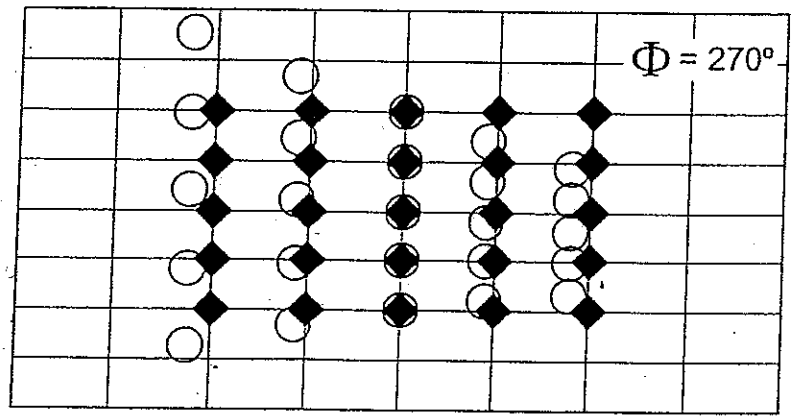


FIG. 4D

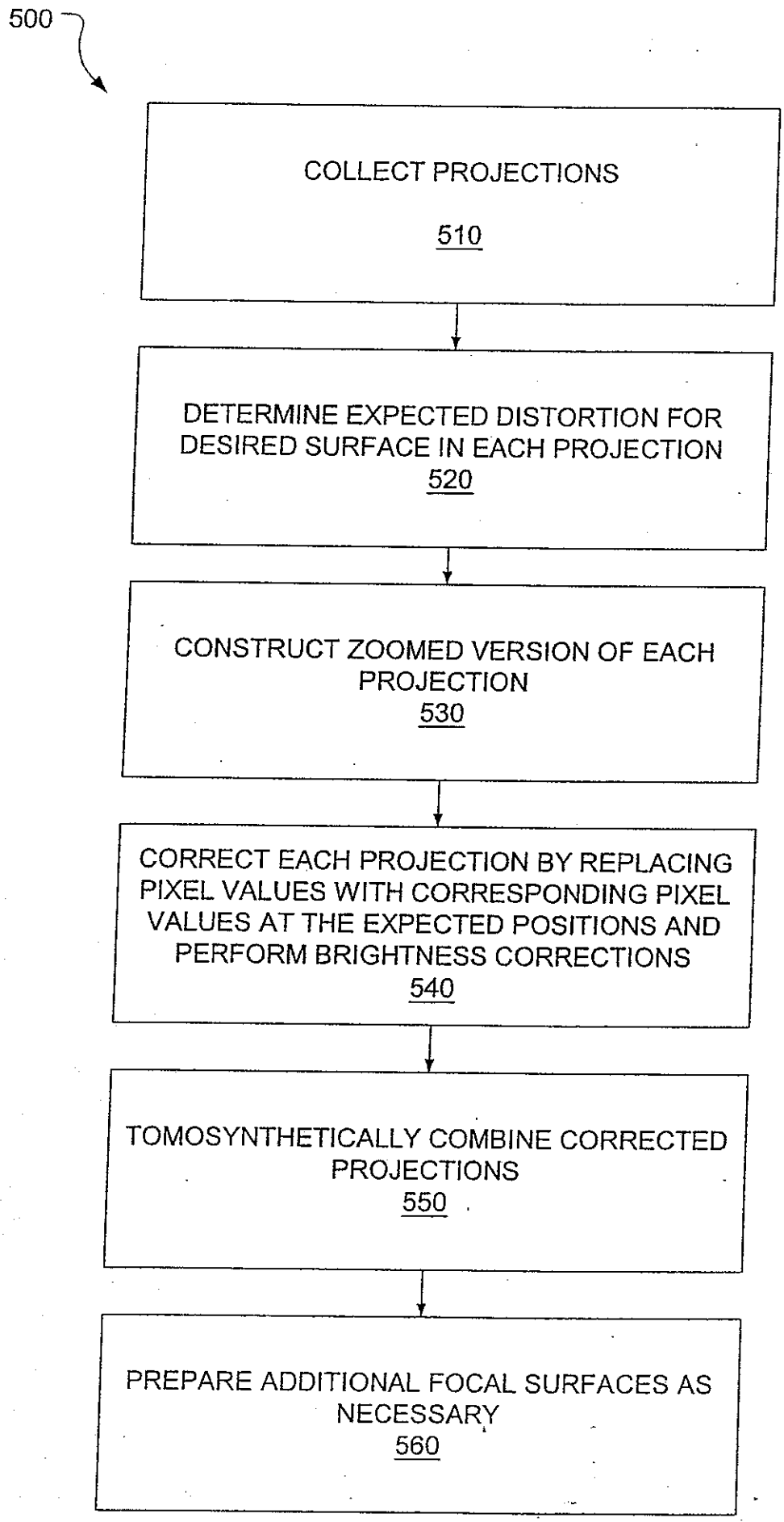

500
COLLECT PROJECTIONS
510
DETERMINE EXPECTED DISTORTION FOR DESIRED SURFACE IN EACH PROJECTION
520
CONSTRUCT ZOOMED VERSION OF EACH PROJECTION
530
CORRECT EACH PROJECTION BY REPLACING PIXEL VALUES WITH CORRESPONDING PIXEL VALUES AT THE EXPECTED POSITIONS AND PERFORM BRIGHTNESS CORRECTIONS
540
TOMOSYNTHETICALLY COMBINE CORRECTED PROJECTIONS
550
PREPARE ADDITIONAL FOCAL SURFACES AS NECESSARY
560

FIG. 5

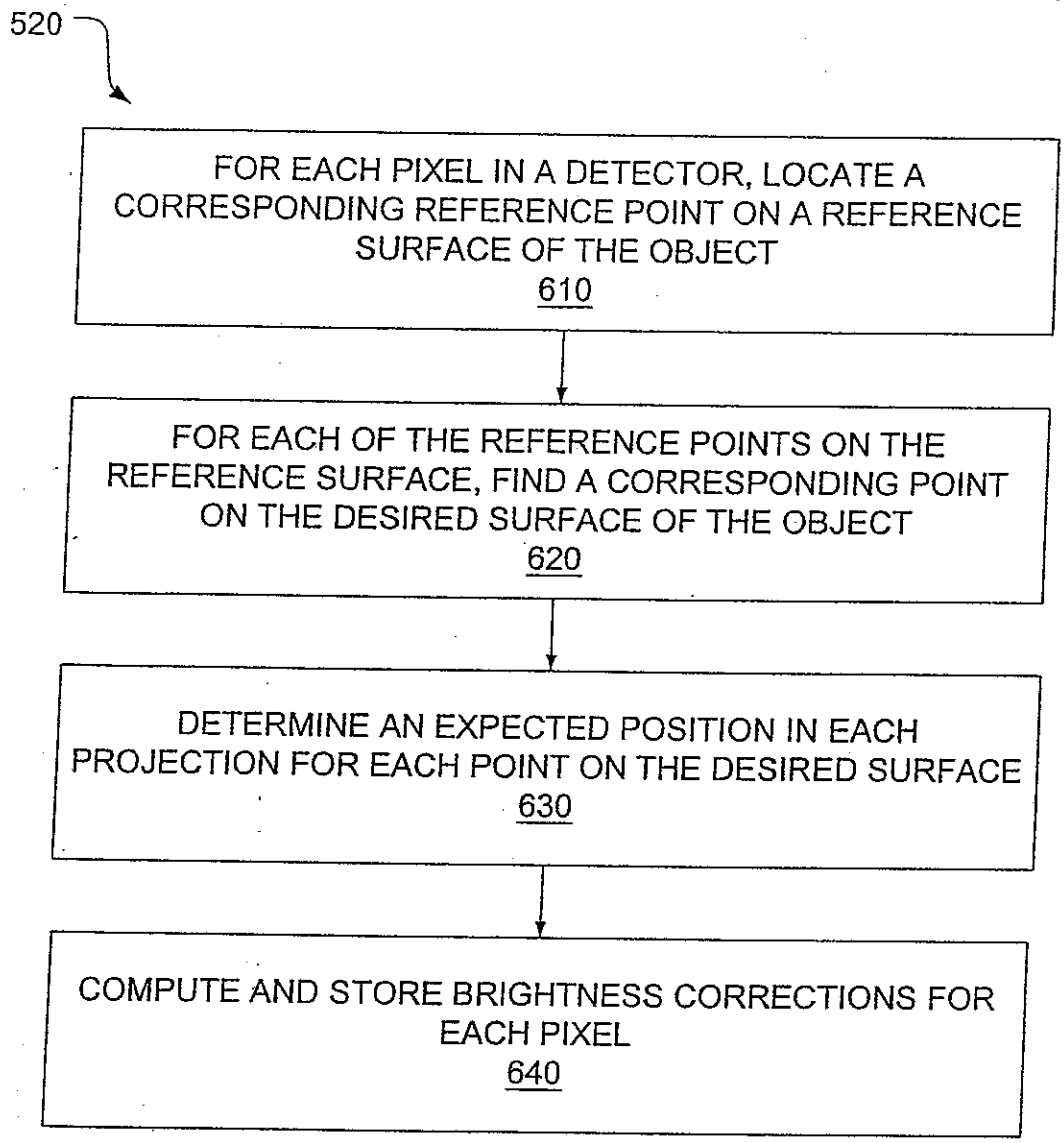

520
FOR EACH PIXEL IN A DETECTOR, LOCATE A CORRESPONDING REFERENCE POINT ON A REFERENCE SURFACE OF THE OBJECT
610
FOR EACH OF THE REFERENCE POINTS ON THE REFERENCE SURFACE, FIND A CORRESPONDING POINT ON THE DESIRED SURFACE OF THE OBJECT
620
DETERMINE AN EXPECTED POSITION IN EACH PROJECTION FOR EACH POINT ON THE DESIRED SURFACE
630
COMPUTE AND STORE BRIGHTNESS CORRECTIONS FOR EACH PIXEL
640

FIG. 6

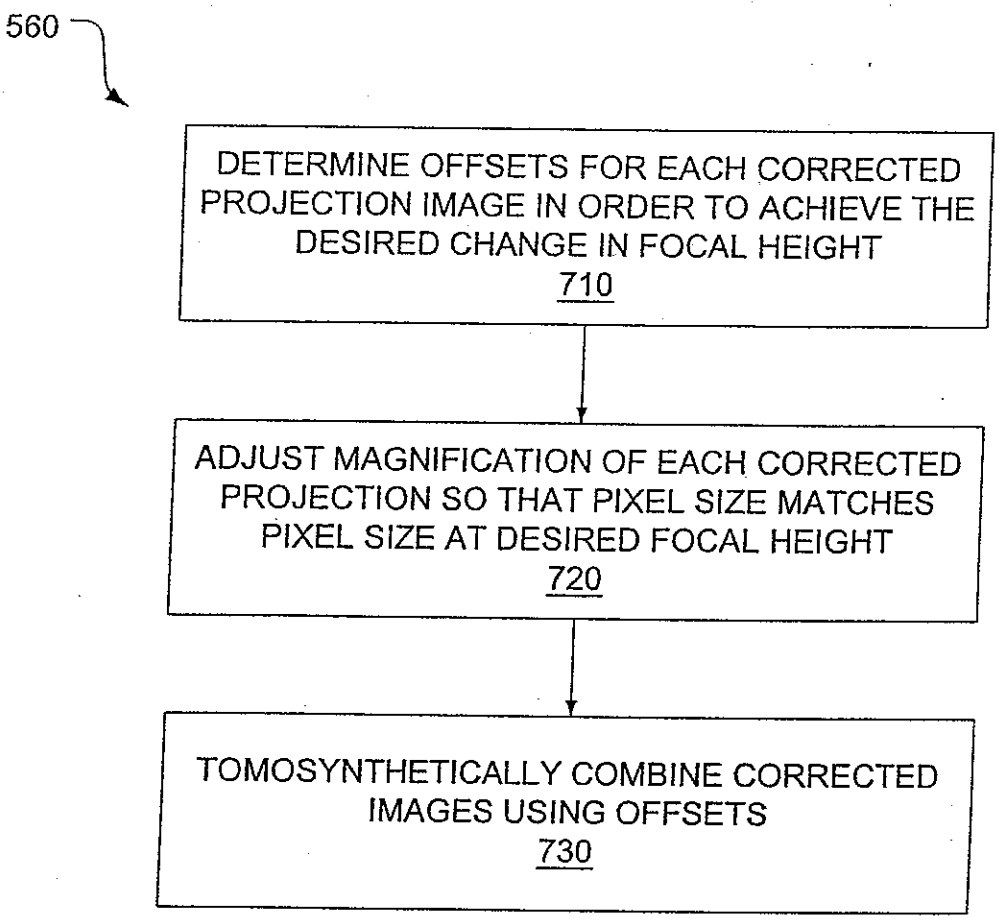

560
DETERMINE OFFSETS FOR EACH CORRECTED PROJECTION IMAGE IN ORDER TO ACHIEVE THE DESIRED CHANGE IN FOCAL HEIGHT
710
ADJUST MAGNIFICATION OF EACH CORRECTED PROJECTION SO THAT PIXEL SIZE MATCHES PIXEL SIZE AT DESIRED FOCAL HEIGHT
720
TOMOSYNTHETICALLY COMBINE CORRECTED IMAGES USING OFFSETS
730

FIG. 7

```
pro warp

n = 256;          N MUST BE EVEN, >= 16

while (!d.window ne -1) do wdelete;

; define warped surface at origin
step = fix(n/16);
zo = shift(dist(n, n), n/2, n/2) * 8 * !pi / (n * sqrt(2));
zo = 25*sin(zo)/zo;
window, 0, xsize=n, ysize=n, title='Warped Surface'
shade_surf, zo;

xo = dblarr(n, n);
yo = dblarr(n, n);
obj = bytarr(n, n);
for i = 0, n - 1 do begin
   xo[i, *] = i - n/2 + 0.5;
   yo[*, i] = i - n/2 + 0.5;
endfor;
for i = step, n - step, step do begin
   obj[i, step:n-step] = 255;
   obj[step:n-step, i] = 255;
endfor;
ndx = where(obj ne 0);   Remember non-zero positions

; mesh representation of original surface
zm=dblarr(n/16, n/16);
xm=dblarr(n/16, n/16);
ym=dblarr(n/16, n/16);
for i = 0, n / 16 - 1 do begin
   for j = 0, n / 16 - 1 do begin
      xm[i, j] = xo[16 * i, 16 * j];
      ym[i, j] = yo[16 * i, 16 * j];
      zm[i, j] = zo[16 * i, 16 * j];
   endfor;
endfor;
window, 1, xsize=n, ysize=n, title='Warped Surface'
surface, zm, xm, ym;

; display the original object
window, 2, xsize=n, ysize=n, title='Flat Object'
tv, obj;

; define source position
zs = 1000;
xs = 707;
ys = 707;

; define flat detector pixel array
zd0 = dblarr(n, n) - 9000;
mag0 = (-9000 - zs) / (-zs);  Flat object to flat detector
x0 = xs + mag0 * (xo - xs);
xc0 = xs - mag0 * xs;
y0 = ys + mag0 * (yo - ys);
yc0 = ys - mag0 * ys;
ix = floor((x0 - xc0) / mag0 + n / 2);
```

FIG. 8A

```
iy = floor((y0 - yc0) / mag0 + n / 2);
img = bytarr(n, n);
img[ix[ndx], iy[ndx]] = obj[ndx];
window, 3, xsize=n, ysize=n, title='Flat Obj/ Flat Det'
tv, img;

; image warped object on flat detector
mag = (zd0 - zs) / (zo - zs);  Warped object to flat detector
x1 = xs + mag * (xo - xs);
y1 = ys + mag * (yo - ys);
ix = floor((x1 - xc0) / mag0 + n / 2);
iy = floor((y1 - yc0) / mag0 + n / 2);
img = bytarr(n, n);
img[ix[ndx], iy[ndx]] = obj[ndx];
window, 4, xsize=n, ysize=n, title='Warped Obj/ Flat Det'
tv, img;

; define warped detector pixel array
zd = zd0 + mag * zo;
magw = (zd - zs) / (zo -zs);
x2 = xs + magw * (xo - xs);
y2 = ys + magw * (yo - ys);
ix = floor((x2 - xc0) / mag0 + n / 2);
iy = floor((y2 - yc0) / mag0 + n / 2);
img = bytarr(n, n);
img[ix[ndx], iy[ndx]] = obj[ndx];
window, 5, xsize=n, ysize=n, title='Warped Obj/Warped Det'
tv, img;

end;
```

FIG. 8B

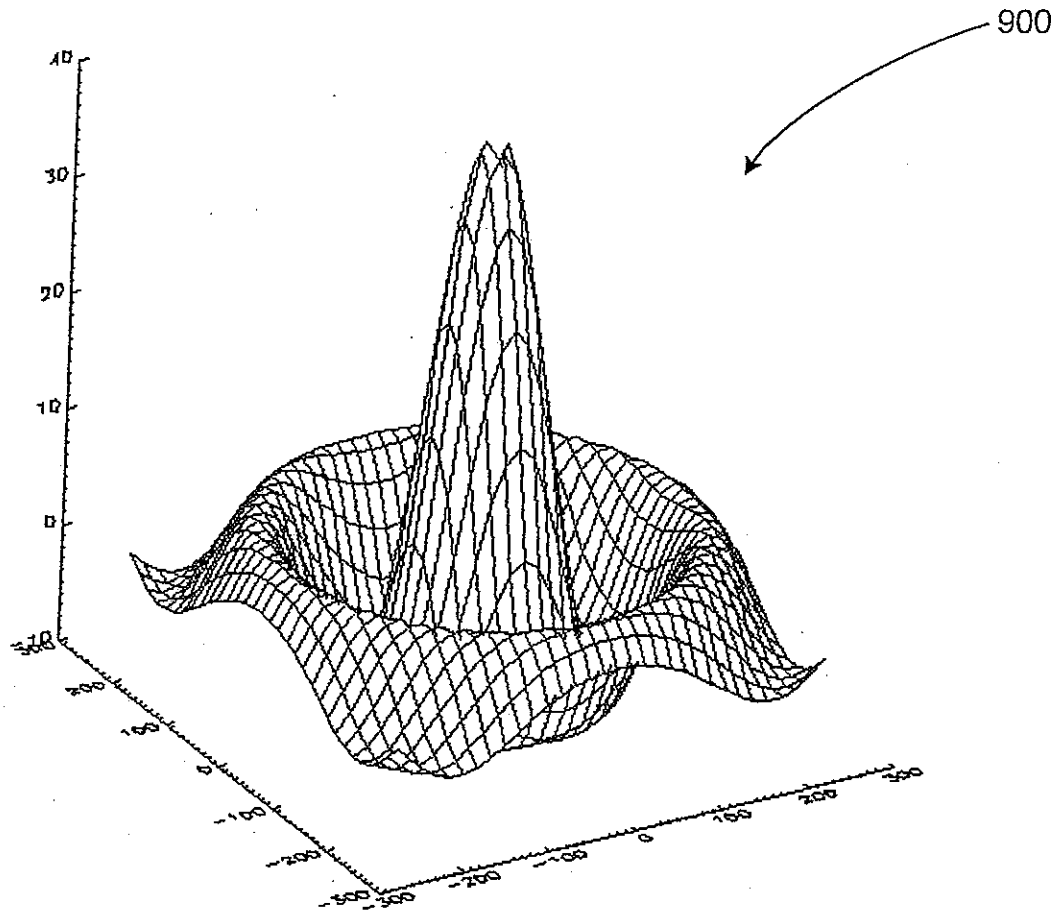


FIG. 9

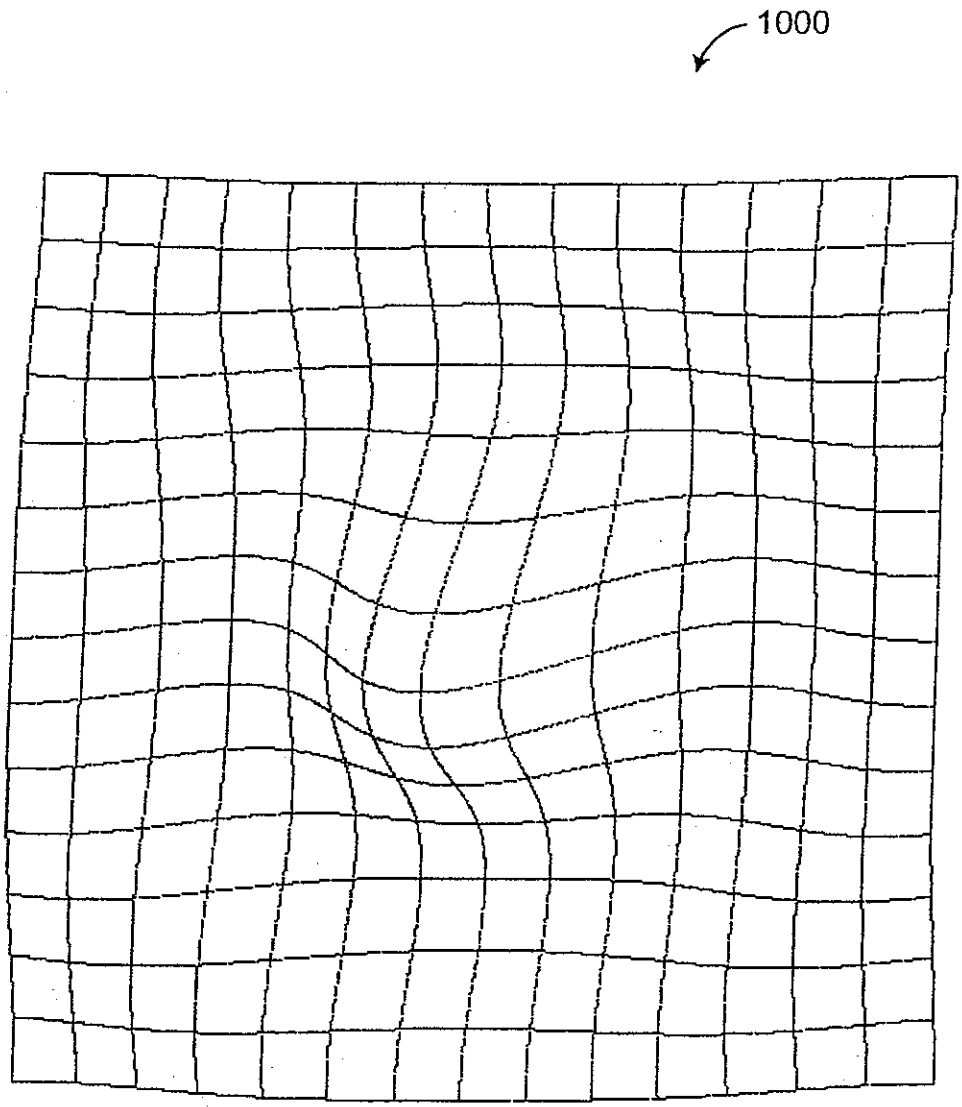


FIG. 10

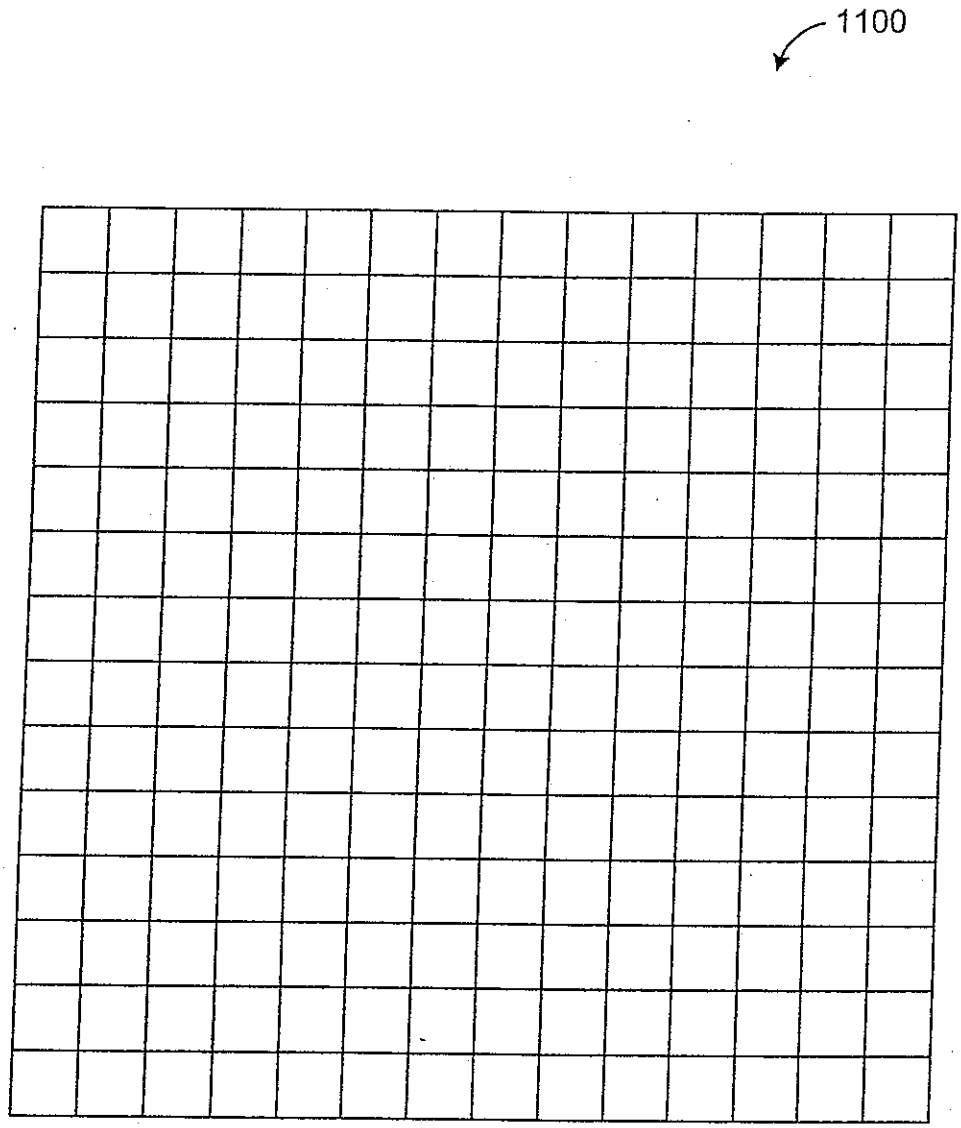


FIG. 11