Template UPDATE:

It avoid the tracker to break down on the parts when the motion is so fast or mixed motions happens such as choosing the wheel of car while it has two motion one is rotation and other is veritically so it cause break in tracking.

Here is my implementation algorithm :

The Inverse Compositional Algorithm

Pre-compute:

(3) Evaluate the gradient ∇T of the template $T(\mathbf{x})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{0})$
(5) Compute the steepest descent images $\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
(6) Compute the Hessian matrix using Equation (38)

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
(2) Compute the error image $I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})$
(7) Compute $\sum_{\mathbf{x}} [\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]$
(8) Compute $\Delta \mathbf{p}$ using Equation (37)
(9) Update the warp $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta \mathbf{p})^{-1}$

until $\|\Delta \mathbf{p}\| \leq \epsilon$

My code does run but does not do the tracking if only change below part in

```
initAffineLKTracker
[row,col]=find(mask);%my code problem
template= img(row(1):row(end),col(1):col(end));
%figure;
%imshow(template);

%for i=1:size(img,1)
 % for j=1:size(img,2)
  % template(i,j)=img(i,j)*mask(i,j);
  %end
%end
%this cause the G to be zero
```

I am sure that I should use cropped template area from the image and then do obtain the gradient.

If use only masked image then the gradient is zero. But if use the template cropped I get the gradient but unfortunately it is wrong. I even used the manual way but can not get the desired answer.

Here is my implementation for derivatives :

```
function [Ix,Iy]=derivatives(I,sigma)

[x,y]=ndgrid(floor(-3*sigma):ceil(3*sigma),floor(-
3*sigma):ceil(3*sigma));
Gauss_x=-(x./(2*pi*sigma^4)).*exp(-
(x.^2+y.^2)/(2*sigma^2));
Gauss_y=-(y./(2*pi*sigma^4)).*exp(-
(x.^2+y.^2)/(2*sigma^2));
%%Filter the images
Ix = imfilter(I,Gauss_x,'conv');
Iy = imfilter(I,Gauss_y,'conv');

end
```

and here is my theory for computing new W :

the parameters of $\mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})$ are:

$$\begin{pmatrix} p_1 + \Delta p_1 + p_1 \cdot \Delta p_1 + p_3 \cdot \Delta p_2 \\ p_2 + \Delta p_2 + p_2 \cdot \Delta p_1 + p_4 \cdot \Delta p_2 \\ p_3 + \Delta p_3 + p_1 \cdot \Delta p_3 + p_3 \cdot \Delta p_4 \\ p_4 + \Delta p_4 + p_2 \cdot \Delta p_3 + p_4 \cdot \Delta p_4 \\ p_5 + \Delta p_5 + p_1 \cdot \Delta p_5 + p_3 \cdot \Delta p_6 \\ p_6 + \Delta p_6 + p_2 \cdot \Delta p_5 + p_4 \cdot \Delta p_6 \end{pmatrix},$$

I used deltap inversed instead of deltap to get the answer

$$\frac{1}{(1+p_1)\cdot(1+p_4) - p_2 \cdot p_3} \begin{pmatrix} -p_1 - p_1 \cdot p_4 + p_2 \cdot p_3 \\ -p_2 \\ -p_3 \\ -p_4 - p_1 \cdot p_4 + p_2 \cdot p_3 \\ -p_5 - p_4 \cdot p_5 + p_3 \cdot p_6 \\ -p_6 - p_1 \cdot p_6 + p_2 \cdot p_5 \end{pmatrix}.$$