EE 604 : DIGITAL IMAGE PROCESSING
Extreme Point Based Image Segmentation & Gaussian Red Eye Correction

Shubh Gupta
Roll No. 14670

Abhinav Agrawal
Roll No. 14011
(Collaborator)

We explore the problem of inducing motion blur into static natural images and producing red eye corrected outputs for frontal human facial images. Although all the alternative problems were equally interesting, we choose focus on these two and attempt to improve existing algorithms for both the techniques while restricting ourselves to classical image processing and computer vision domain.

Red Eye Correction

We develop a novel approach for red eye correction which is based on two assumptions, (i) Shape of eye in natural images can be approximated by Gaussian distribution (ii) Red Eye Error is around the pupil.

Algorithm has been detailed in [1]. We would like to mention that m_1 and m_2 can be any general redness metrics. We found that $m_1 = b + g$ and $m_2 = \frac{r^2}{b^2+g^2+k'}$ with $k^{'} = 14$ worked best in our limited trials[1]. We used $m_3 = \frac{b+g}{2}$ and this too can be changed any other metric for replacing/updating the red values. k was empirically set to 5 with B being the standard 5×5 filter.

Motion Blur

Motion Blur requires segmentation of the object of interest. This usually requires input from an annotator in the form of a bounding box or full boundary specification. We develop upon one of the recent techniques[1] and segment using just extreme points of the object(usually 4 or 5)[1]. We make certain changes to their algorithm and hopefully develop a smoother pipeline. The details of the improved algorithm have been given in [2].

Our refined algorithm is able to retain able to extract out the information for the objects which can not be enclosed in bounding box. Also our method is adaptive and makes strides based on the euclidean distance between the points[3]. This allows our method to easily extend to very intricate detail objects with odd shapes by simply increasing the number of annotation points. The results for some chosen images have been displayed in [2].

Future Work

We feel both the techniques can be refinement to give better results. In particular, modeling of $p_{redpoint}$ and p with different distributions, segmentation technique can be improvements so the standard GrabCut algorithm runs faster. Red eye detection approach can also be applied to different problems such as over-saturation correction,color correction etc.

Algorithm 1 Red Eye Correction

procedure EYE DETECTOR(I) ▷ I is the input image
  $D_{hr} \leftarrow$ Harr Eye Detector(I)
  $D_{fc} \leftarrow$ Dlib Face and Eye Detector(I)
  $M_{detection} \leftarrow D_{fc} \cap D_{hr}$
  $I_{masked} \leftarrow I \cap M_{detection}$
  return I_{masked}
end procedure

procedure RED POINT DETECTOR(I_{masked}, t_l, k) ▷ I_{masked} is the masked input image
  $M_{redpoints} \leftarrow \emptyset$
  for $\forall p \in I_{masked}$ **do** ▷ This can be vectorized
    if $r > t_1$ & $r > m_1$ **then**
      $M_{redpoints} \leftarrow p \cup M_{redpoints}$
    end if
  end for
  $M_{redpoints} \leftarrow (...(((M_{redpoints} \circ B) \oplus B) \oplus B)...k$ times...)
  return $M_{redpoints}$
end procedure

procedure INTENSITY WEIGHT GENERATOR($M_{redpoints}$)
  $I_{redpoints} \leftarrow I_{masked} \cap M_{redpoints}$
  $M_{intensity} \leftarrow \emptyset$
  for $\forall p \in I_{redpoints}$ **do** ▷ This can be vectorized
    $p_{redpoint} \leftarrow m_2$
    $M_{intensity} \leftarrow p_{redpoint} \cup M_{intensity}$
  end for
  $\mu_{redpoints}, \sigma_{redpoint} \leftarrow$ STATS FINDER($M_{intensity}$)
  return $I_{redpoints}, \mu_{redpoints}, \sigma_{redpoint}$
end procedure

procedure LOCATION WEIGHT GENERATOR($I_{redpoints}, \mu_{redpoints}, \sigma_{redpoint}$)
  $M_{location} \leftarrow \emptyset$
  for $\forall p \in I_{redpoints}$ **do** ▷ This can be vectorized
    $p_{redpoint} \leftarrow m_2$
    $w_{intensity} \leftarrow \mathcal{N}(p_{redpoint} | \mu_{redpoints}, \sigma_{redpoint})$
    if $w_{intensity} > t_2$ **then**
      $M_{location} \leftarrow p \cup M_{location}$
    end if
  end for
  $\mu_{locpoints}, \sigma_{locpoint} \leftarrow$ STATS FINDER($M_{location}$)
  return $\mu_{locpoints}, \sigma_{locpoint}$
end procedure

procedure CORRECT RED EYE($I_{redpoints}, \mu_{locpoints}, \sigma_{locpoint}$)
  for $\forall p \in I_{redpoints}$ **do** ▷ This can be vectorized
    $w_{location} \leftarrow \mathcal{N}(p | \mu_{redpoints}, \sigma_{redpoint})$
    $r \leftarrow w_{location}(m_3) + (1 - w_{location})r$
  end for
end procedure

procedure STATS FINDER(S)
  return Mean(S), Standard Deviation(S)
end procedure

Algorithm 2 Extreme Point based Segmentation

procedure MODIFIED GRAB CUT(I, L_{points}, k) ▷ L_{points} is the list of pair of points
  $E_{base} \leftarrow$ Canny Edge Detector(I)
  $S_{boundary} \leftarrow \emptyset$
  for $\forall u, v \in L_{points}$ **do**
    $D_{uv} \leftarrow$ Euclidean Distance(u, v)
    $\sigma_{uv} \leftarrow \frac{D_{uv}}{2}$
    $I_u \leftarrow \mathcal{N}(XY_{mesh}|u, \Sigma_{uv})$
    $I_v \leftarrow \mathcal{N}(XY_{mesh}|v, \Sigma_{uv})$
    $I_{uv} \leftarrow \mathbf{1} - E_{base} \odot I_u \odot I_v$
    $B_{uv} \leftarrow$ Minimum Weight Path(I_{uv}, u, v)
    $S_{boundary} \leftarrow B_{uv} \cup S_{boundary}$
  end for
  $S_{crude} \leftarrow$ Flood Fill($S_{boundary}$)
  $S_{skeleton} \leftarrow$ Skelitonize(S_{crude})
  $S_{crude\,dilated} \leftarrow (...((S_{crude} \oplus B) \oplus B)...k$ times...)
  $R \leftarrow$ Rectange Finder($S_{crude\,dilated}$)
  $S_{rectangle} \leftarrow$ Flood Fill(R)
  $S_{foreground} \leftarrow S_{skeleton}$
  $S_{background} \leftarrow S^C_{rectangle}$
  $S_{probable\ foreground} \leftarrow S_{crude\,dilated}$
  $S_{probable\,background} \leftarrow R \cup S^C_{crude\,dilated}$
  $S_{final} \leftarrow$ GrabCut($S_{foreground}, S_{background}, S_{probable\ foreground}, S_{probable\,background}$)
  return $S_{final} \cap S_{crude}$
end procedure

References

[1] Papadopoulos, Dim P., et al. "Extreme clicking for efficient object annotation." *arXiv preprint arXiv:1708.02750 (2017).*

[2] Gaubatz, Matthew, and Robert Ulichney. "Automatic red-eye detection and correction." *Image Processing. 2002. Proceedings. 2002 International Conference on. Vol. 1. IEEE, 2002.*

[3] Mallick, S. "Automatic Red Eye Remover using OpenCV (C++ / Python)" *www.learnopencv.com, March, 2017*

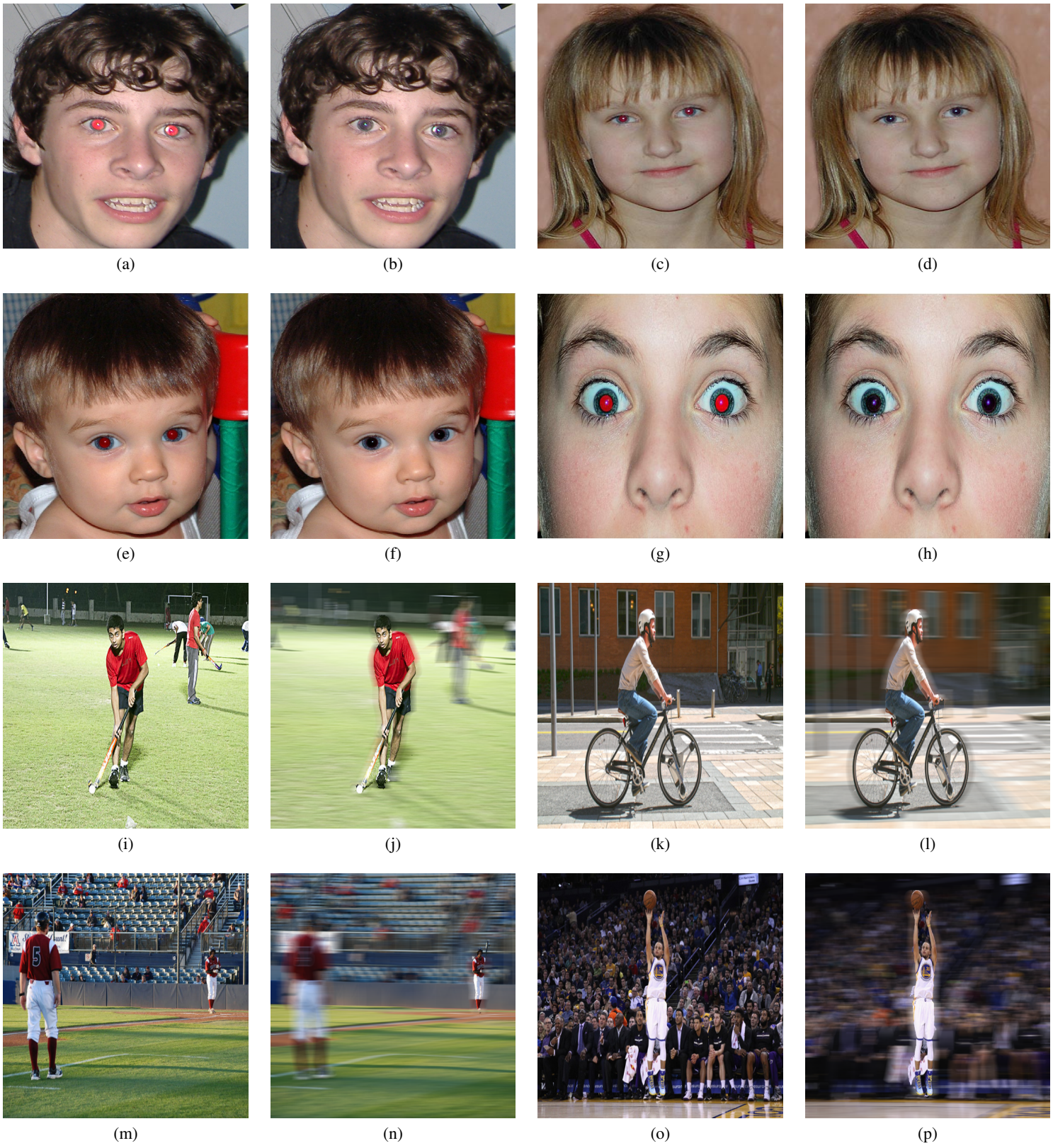
[4] Canny, J.,"A Computational Approach To Edge Detection" *IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679698, 1986*

[5] Dijkstra, E. W., "A note on two problems in connexion with graphs" *Numerische Mathematik1(1959)*

[6] Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake., "Grabcut: Interactive foreground extraction using iterated graph cuts." *ACM transactions on graphics (TOG). Vol. 23. No. 3. ACM, 2004.*

[7] King, Davis E. , "Dlib-ml: A machine learning toolkit." *Journal of Machine Learning Research 10.Jul (2009): 1755-1758.*

[8] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.*



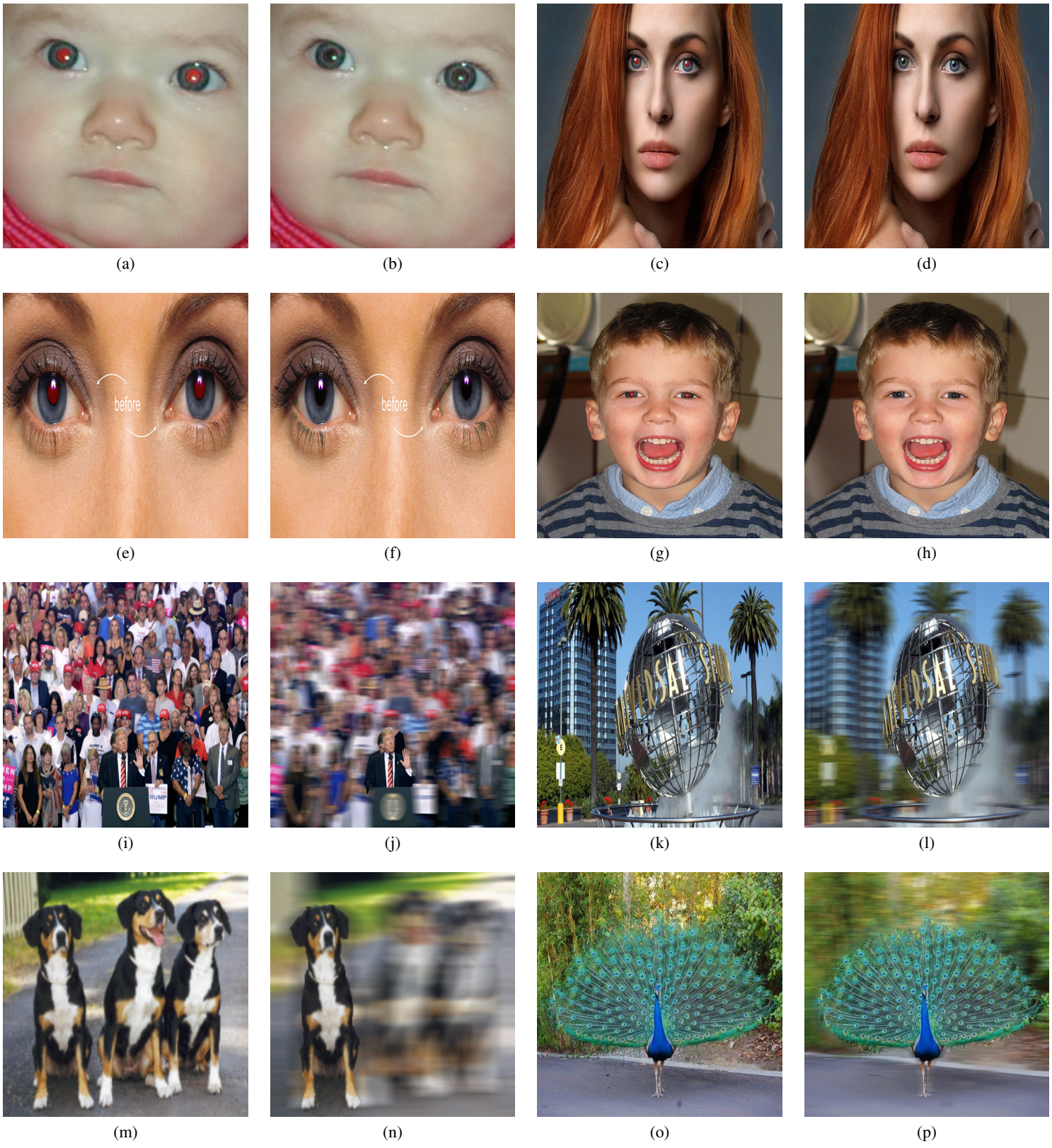
(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

Figure 1: Results for all the images provided in the assignment



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

Figure 2: Some extra output images for both the techniques.

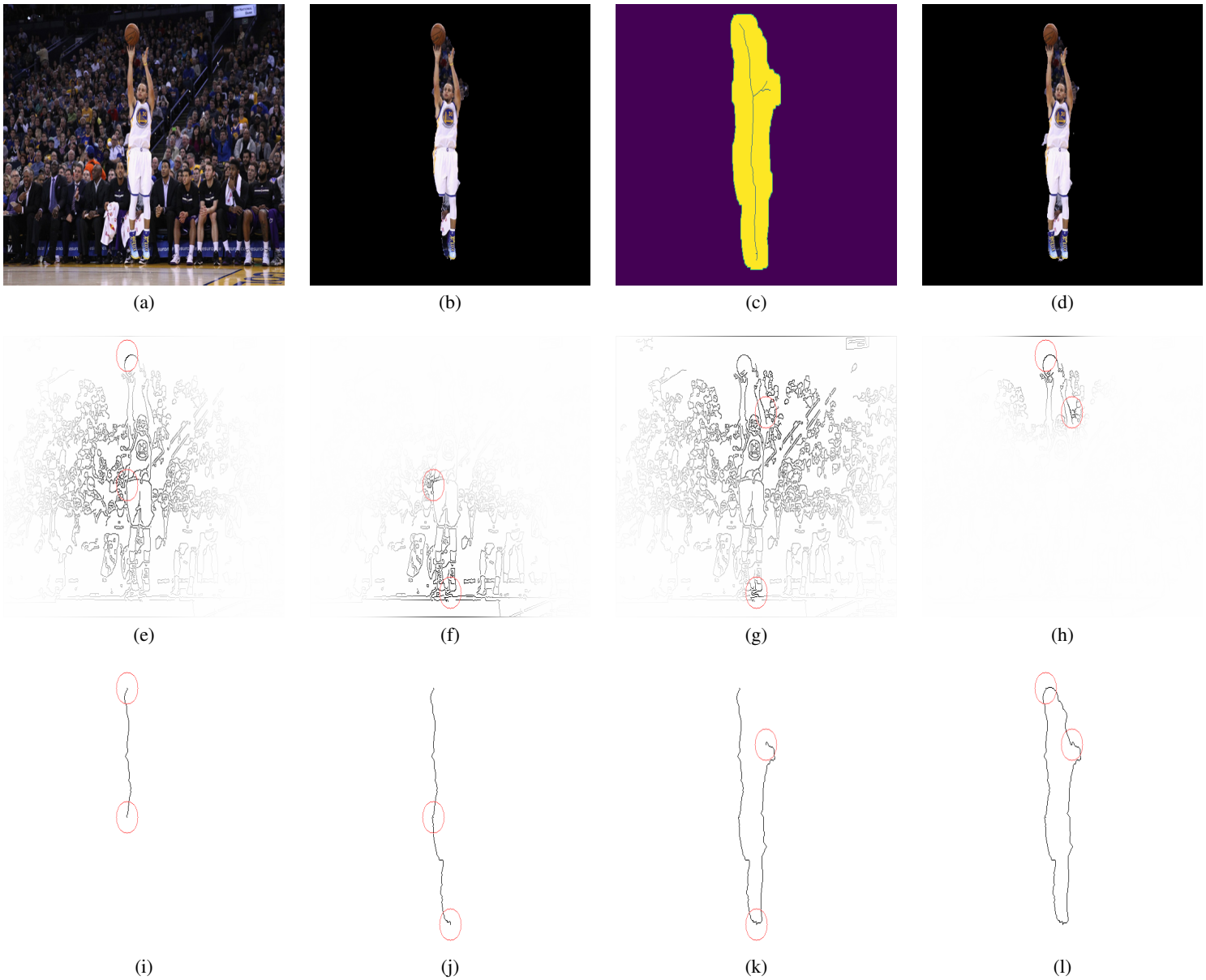


Figure 3: (a) Original image (b) Crude Segmentation using the user input (c) Skeleton generated from (b) to use as foreground points for GrabCut input (d) Improved segmentation obtained using GrabCut output and (b). (e-h) Gaussian Weighted Edge maps at different stages of the input points. (i-l) Boundary found using Dijkstra's Min Weighted Path Algorithm. Center of the red circles denote the two points under consideration