Ceva's Theorem

Mathias Schack Rabing

April 18, 2024

Abstract

This entry contains a definition of the area the triangle constructed by three points. Building on this, some basic geometric properties about the area of a triangle are derived. These properties are used to prove Ceva's theorem.

Contents

```
theory Ceva
imports
  Triangle.Triangle
begin

definition Triangle-area :: 'a::real-inner ⇒ 'a ⇒ 'a ⇒ real
  where Triangle-area x y z = abs(sin (angle x y z)) ∗ dist x y ∗ dist y z

lemma Triangle-area-per1 : Triangle-area a b c = Triangle-area b c a
proof −
  have H : abs(sin (angle a b c)) ∗ dist b c = abs(sin (angle b a c)) ∗ dist a c
    using sine-law-triangle
    by (metis (mono-tags, opaque-lifting) abs-mult real-abs-dist)
  show ?thesis
    apply(simp add: Triangle-area-def)
    using H
   by (metis abs-of-nonneg angle-commute dist-commute sin-angle-nonneg sine-law-triangle)
qed

lemma Triangle-area-per2 : Triangle-area a b c = Triangle-area b a c
proof −
  have H : abs(sin (angle a b c)) ∗ dist b c = abs(sin (angle b a c)) ∗ dist a c
    using sine-law-triangle
    by (metis (mono-tags, opaque-lifting) abs-mult real-abs-dist)
  show ?thesis
    using H
    by (simp add: Triangle-area-def dist-commute[of a b])
```

```
qed

lemma collinear-angle:
  fixes a b c :: 'a::euclidean-space
  shows collinear {a, b, c} ⟹ a ≠ b ⟹ b ≠ c ⟹ angle a b c ∈ {0, pi}
proof (cases a = c)
  case True
  assume Col : collinear {a, b, c}
  assume H1 : a ≠ b
  assume H2 : b ≠ c
  assume H3 : a = c
  show ?thesis
    using H1 H3 angle-refl-mid
    by auto
next
  case False
  assume Col : collinear {a, b, c}
  assume H1 : a ≠ b
  assume H2 : b ≠ c
  assume H3 : a ≠ c
  consider (bet1) between (b, c) a | (bet2) between (c, a) b | (bet3) between (a, b)
c
    using Col collinear-between-cases
    by auto
  then show ?thesis
  proof cases
    case bet1
    assume B1: between (b, c) a
    have H: angle c a b = pi
      apply(rule strictly-between-implies-angle-eq-pi)
      using B1 H3 H1
      by (auto simp: between-commute)
    show ?thesis
      by (smt (verit) H angle-nonneg angle-sum-triangle insert-iff)
  next
    case bet2
    assume B1: between (c, a) b
    show ?thesis
     by (metis H1 H2 bet2 between-commute sin-angle-zero-iff sin-pi strictly-between-implies-angle-eq-pi)
  next
    case bet3
    assume B1: between (a, b) c
    have H: angle b c a = pi
      apply(rule strictly-between-implies-angle-eq-pi)
      using B1 H3 H2 H1
      by (auto simp: between-commute)
    show ?thesis
      by (smt (verit) H angle-nonneg angle-sum-triangle insert-iff)
    qed
```

qed

lemma *Triangle-area-0* :
fixes *c* :: *'a::euclidean-space*
shows *Triangle-area a b c* = *0* $\longleftrightarrow$ *collinear* {*a*,*b*,*c*}
proof −
show *?thesis*
apply(*simp add*: *Triangle-area-def*)
using *collinear-angle*
by (*metis* (*no-types*, *lifting*) *Angles.angle-collinear collinear-2 insertCI insert-absorb sin-angle-zero-iff*)
qed

lemma *Angle-longer-side* :
fixes *a* :: *'a* :: *euclidean-space*
assumes *Col* : *between* (*b*,*d*) *c*
assumes *NeqBC* : $b \neq c$
shows *angle a b c* = *angle a b d*
proof (*cases* $a = b \vee b = d \vee c = d$)
case *True*
then show *?thesis*
using *Col*
by *auto*
next
case *False*
assume *H* : $\neg (a = b \vee b = d \vee c = d)$
have *NeqAB* : $a \neq b$
using *H*
by *auto*
have *NeqBD* : $b \neq d$
using *H*
by *auto*
have *NeqCD* : $c \neq d$
using *H*
by *auto*
have *Goal1* : *norm* $(d - b) *_R (c - b) =$ *norm* $(c - b) *_R (d - b)$
apply(*rule vangle-eq-0D*)
using *Col*
by (*metis Groups.add-ac*(*2*) *NeqBC NeqCD add-le-same-cancel1 angle-def angle-nonneg angle-sum-triangle eq-add-iff order.eq-iff strictly-between-implies-angle-eq-pi*)
have *Goal2* : $(a - b) \cdot (c - b) *$ *norm* $(d - b) \neq$
$(a - b) \cdot (d - b) *$ *norm* $(c - b) \Longrightarrow a = b$
apply(*simp only*: *mult.commute*[**where** *b*=*norm* (*d* − *b*)])
apply(*simp only*: *mult.commute*[**where** *b*=*norm* (*c* − *b*)])
apply(*simp only*: *real-inner-class.inner-scaleR-right*[*THEN sym*])
using *Goal1*
by *auto*
have *Goal* : $(a - b) \cdot (c - b) * ($*norm* $(a - b) *$ *norm* $(d - b)) =$

```
   (a − b) · (d − b) ∗ (norm (a − b) ∗ norm (c − b))
    using Goal2
    by auto
  show ?thesis
    apply(simp add: angle-def)
    using NeqAB NeqBD NeqCD NeqBC
    apply(simp only: vangle-def)
    using Goal
    by (smt (verit, best) eq-iff-diff-eq-0 frac-eq-eq no-zero-divisors norm-eq-zero)
qed

lemma Triangle-area-comb :
  fixes c :: 'a::euclidean-space
  assumes Col : between (b,c) m
  shows Triangle-area a b m + Triangle-area a c m = Triangle-area a b c
proof (cases b = m ∨ c = m)
  case True
  then
  have Eq : b = m ∨ c = m
    by auto
  have Tri0 : Triangle-area a m m = 0
    by (auto simp: Triangle-area-0)
  show ?thesis
    using Eq Tri0
    using Triangle-area-per1 Triangle-area-per2
    by (metis add.right-neutral add-0)
next
  case False
  then
  have Neq : ¬(b = m ∨ c = m)
    by auto
  have NeqBM : b ≠ m
    using Neq
    by auto
  have NeqCM : c ≠ m
    using Neq
    by auto
  have Angle1 : angle a b m = angle a b c
    using Col Angle-longer-side NeqBM NeqCM
    by auto
  have Angle2 : angle a c m = angle a c b
    using Col Angle-longer-side NeqBM NeqCM between-commute
    by metis
  have |sin (angle a b m)| ∗ dist a b ∗ dist b m +
       |sin (angle a c m)| ∗ dist a c ∗ dist c m =
       |sin (angle a b c)| ∗ dist a b ∗ dist b m +
       |sin (angle a c b)| ∗ dist a c ∗ dist c m
    using Angle1 Angle2
    by simp
```

```
  also have |sin (angle a b c)| * dist a b * dist b m +
        |sin (angle a c b)| * dist a c * dist c m =
        |sin (angle a b c)| * dist a b * dist b m +
        |sin (angle a b c)| * dist a b * dist c m
    using sine-law-triangle
    by (smt (verit) congruent-triangle-sss(17) dist-commute sin-angle-nonneg)
  also have
       |sin (angle a b c)| * dist a b * dist b m +
        |sin (angle a b c)| * dist a b * dist c m =
        |sin (angle a b c)| * dist a b * (dist b m + dist c m)
    by (metis inner-add(2) inner-real-def)
  also have |sin (angle a b c)| * dist a b * (dist b m + dist c m) =
        |sin (angle a b c)| * dist a b * dist b c
    by (metis assms between dist-commute)
  finally have Goal : |sin (angle a b m)| * dist a b * dist b m +
        |sin (angle a c m)| * dist a c * dist c m =
        |sin (angle a b c)| * dist a b * dist b c
    by simp
  show ?thesis
    apply(simp add: Triangle-area-def)
    using Goal
    by blast
qed

lemma Triangle-area-cal :
  fixes a :: 'a::euclidean-space
  assumes Col : collinear {a,m,b}
  shows ∃ k. dist a m * k = Triangle-area a c m ∧ dist b m * k = Triangle-area
b c m
proof (cases b = m ∨ a = m)
  case True
  then
  have Eq :(a ≠ m ∧ b = m) ∨ (a = m ∧ b ≠ m) ∨ (a = m ∧ b = m)
    by auto
  show ?thesis
    using Eq
   by(auto simp: Triangle-area-0 collinear-3-eq-affine-dependent exI[where x=Triangle-area
a c m / dist a m]
                    exI[where x=Triangle-area b c m / dist b m])
next
  case False
  then
  have H : ¬ (b = m ∨ a = m)
    by simp
  have NeqBM : b ≠ m and NeqMA : m ≠ a
    using H
    by auto
  have H1: dist a m * |sin (angle a m c)| * dist c m =
    |sin (angle a c m)| * dist a c * dist c m
```

```
    using sine-law-triangle
    by (smt (verit) angle-commute dist-commute mult.commute sin-angle-nonneg)
  have dist b m * |sin (angle a m c)| * dist c m =
    dist b m * |sin (pi − angle a m c)| * dist c m
    by auto
  also have dist b m * |sin (pi − angle a m c)| * dist c m =
    dist b m * |sin (angle b m c)| * dist c m
    using angle-inverse[THEN sym] Col NeqBM NeqMA
     by (smt (verit, ccfv-SIG) Angle-longer-side angle-commute between-commute
collinear-between-cases sin-pi-minus)
  also have dist b m * |sin (angle b m c)| * dist c m =
    |sin (angle b c m)| * dist b c * dist c m
    using sine-law-triangle
   by (metis abs-of-nonneg angle-commute dist-commute mult.commute sin-angle-nonneg)
  finally have H2: dist b m * |sin (angle a m c)| * dist c m =
    |sin (angle b c m)| * dist b c * dist c m
    by simp
  show ?thesis
    apply(simp add: Triangle-area-def)
    apply(rule exI[where x=|sin (angle a m c)| * dist c m])
    using H1 H2
    by auto
qed

lemma Triangle-area-comb-alt :
  fixes a :: 'a::euclidean-space
  assumes Col1 : collinear {a,m,b}
  assumes Col2 : collinear {c,k,m}
   shows Goal : ∃ h. dist a m * h = Triangle-area a c k ∧ dist b m * h =
Triangle-area b c k
proof −
  obtain H where TriB : dist a m * H = Triangle-area a c m ∧ dist b m * H =
Triangle-area b c m
    using Col1 Triangle-area-cal by blast
  obtain h where TriS : dist a m * h = Triangle-area a k m ∧ dist b m * h =
Triangle-area b k m
    using Col1 Triangle-area-cal by blast
  consider (bet1) between (k, m) c | (bet2) between (m, c) k | (bet3) between (c,
k) m
    using Col2 collinear-between-cases
    by auto
  then show ?thesis
  proof cases
    case bet1
    have AreaAC : dist a m * H = Triangle-area a c m and AreaBC : dist b m *
H = Triangle-area b c m
      using TriB
      by auto
    have AreaAM : dist a m * h = Triangle-area a k m and AreaBM : dist b m *
```

$h = \mathit{Triangle\text{-}area}\ b\ k\ m$
  using *TriS*
  by *auto*
assume *Bet* : $\mathit{between}\ (k, m)\ c$
have $\mathit{dist}\ a\ m * (h - H) = \mathit{dist}\ a\ m * h - \mathit{dist}\ a\ m * H$
  by (*simp add*: *right-diff-distrib*)
also have $\mathit{dist}\ a\ m * h - \mathit{dist}\ a\ m * H = \mathit{Triangle\text{-}area}\ a\ k\ m - \mathit{Triangle\text{-}area}\ a\ c\ m$
  using *AreaAC AreaAM*
  by *auto*
also have $\mathit{Triangle\text{-}area}\ a\ k\ m - \mathit{Triangle\text{-}area}\ a\ c\ m = \mathit{Triangle\text{-}area}\ a\ c\ k$
  using *Bet Triangle-area-comb*
  by (*metis Triangle-area-per1 Triangle-area-per2 diff-eq-eq*)
finally have *Goal1* : $\mathit{dist}\ a\ m * (h - H) = \mathit{Triangle\text{-}area}\ a\ c\ k$
  by *simp*
have $\mathit{dist}\ b\ m * (h - H) = \mathit{dist}\ b\ m * h - \mathit{dist}\ b\ m * H$
  by (*simp add*: *right-diff-distrib*)
also have $\mathit{dist}\ b\ m * h - \mathit{dist}\ b\ m * H = \mathit{Triangle\text{-}area}\ b\ k\ m - \mathit{Triangle\text{-}area}\ b\ c\ m$
  using *AreaBC AreaBM*
  by *auto*
also have $\mathit{Triangle\text{-}area}\ b\ k\ m - \mathit{Triangle\text{-}area}\ b\ c\ m = \mathit{Triangle\text{-}area}\ b\ c\ k$
  using *Bet Triangle-area-comb*
  by (*metis Triangle-area-per1 Triangle-area-per2 diff-eq-eq*)
finally have *Goal2* : $\mathit{dist}\ b\ m * (h - H) = \mathit{Triangle\text{-}area}\ b\ c\ k$
  by *simp*
show *?thesis*
  using *Goal1 Goal2* **by** *blast*
next
case *bet2*
have *AreaAC* : $\mathit{dist}\ a\ m * H = \mathit{Triangle\text{-}area}\ a\ c\ m$ **and** *AreaBC* : $\mathit{dist}\ b\ m * H = \mathit{Triangle\text{-}area}\ b\ c\ m$
  using *TriB*
  by *auto*
have *AreaAM* : $\mathit{dist}\ a\ m * h = \mathit{Triangle\text{-}area}\ a\ k\ m$ **and** *AreaBM* : $\mathit{dist}\ b\ m * h = \mathit{Triangle\text{-}area}\ b\ k\ m$
  using *TriS*
  by *auto*
assume *Bet* : $\mathit{between}\ (m, c)\ k$
have $\mathit{dist}\ a\ m * (H - h) = \mathit{dist}\ a\ m * H - \mathit{dist}\ a\ m * h$
  by (*simp add*: *right-diff-distrib*)
also have $\mathit{dist}\ a\ m * H - \mathit{dist}\ a\ m * h = \mathit{Triangle\text{-}area}\ a\ c\ m - \mathit{Triangle\text{-}area}\ a\ k\ m$
  using *AreaAC AreaAM*
  by *auto*
also have $\mathit{Triangle\text{-}area}\ a\ c\ m - \mathit{Triangle\text{-}area}\ a\ k\ m = \mathit{Triangle\text{-}area}\ a\ c\ k$
  using *Bet Triangle-area-comb*
  by (*smt* (*verit*) *between-triv1*)
finally have *Goal1* : $\mathit{dist}\ a\ m * (H - h) = \mathit{Triangle\text{-}area}\ a\ c\ k$

```
      by simp
    have dist b m * (H − h) = dist b m * H − dist b m * h
      by (simp add: right-diff-distrib)
    also have dist b m * H − dist b m * h = Triangle-area b c m − Triangle-area
b k m
      using AreaBC AreaBM
      by auto
    also have Triangle-area b c m − Triangle-area b k m = Triangle-area b c k
      using Bet Triangle-area-comb
      by (smt (verit) between-triv1)
    finally have Goal2 : dist b m * (H − h) = Triangle-area b c k
      by simp
    show ?thesis
      using Goal1 Goal2 by blast
  next
    case bet3
    have AreaAC : dist a m * H = Triangle-area a c m and AreaBC : dist b m *
H = Triangle-area b c m
      using TriB
      by auto
    have AreaAM : dist a m * h = Triangle-area a k m and AreaBM : dist b m *
h = Triangle-area b k m
      using TriS
      by auto
    assume Bet : between (c, k) m
    have dist a m * (H + h) = Triangle-area a c k
      by (simp add: AreaAC TriS Triangle-area-comb bet3 distrib-left)
    moreover have dist b m * (H + h) = Triangle-area b c k
      by (simp add: AreaBC TriS Triangle-area-comb bet3 distrib-left)
    ultimately show ?thesis
      by blast
    qed
qed

lemma Cevas :
  fixes a :: 'a::euclidean-space
  assumes MidCol : collinear {a,k,d} ∧ collinear {b,k,e} ∧ collinear {c,k,f}
  assumes TriCol : collinear {a,f,b} ∧ collinear {a,e,c} ∧ collinear {b,d,c}
  assumes Triangle : ¬ collinear {a,b,c}
  shows dist a f * dist b d * dist c e = dist f b * dist d c * dist e a
proof −
  obtain n1 where Tri1: dist a f * n1 = Triangle-area a c k ∧ dist b f * n1 =
Triangle-area b c k
    by (meson MidCol TriCol Triangle-area-comb-alt)
  obtain n2 where Tri2 : dist a e * n2 = Triangle-area a b k ∧ dist c e * n2 =
Triangle-area c b k
    by (meson MidCol TriCol Triangle-area-comb-alt)
  obtain n3 where Tri3 : dist b d * n3 = Triangle-area b a k ∧ dist c d * n3 =
Triangle-area c a k
```

by (*meson MidCol TriCol Triangle-area-comb-alt*)
have *Tri1'1* : *dist a f* ∗ *n1* = *Triangle-area a c k* **and** *Tri1'2* : *dist b f* ∗ *n1* = *Triangle-area b c k*
using *assms*
by (*auto simp*: *Tri1*)
have *Tri2'1* : *dist c e* ∗ *n2* = *Triangle-area c b k* **and** *Tri2'2* : *dist a e* ∗ *n2* = *Triangle-area a b k*
using *assms*
by (*auto simp*: *Tri2*)
have *Tri3'1* : *dist c d* ∗ *n3* = *Triangle-area c a k* **and** *Tri3'2* : *dist b d* ∗ *n3* = *Triangle-area b a k*
using *assms*
by (*auto simp*: *Tri3*)
have *dist a f* ∗ *n1* ∗ *dist b d* ∗ *n3* ∗ *dist c e* ∗ *n2* =
Triangle-area a c k ∗ *Triangle-area b a k* ∗ *Triangle-area c b k*
using *Tri1'1 Tri2'1 Tri3'2*
by *simp*
also have *Triangle-area a c k* ∗ *Triangle-area b a k* ∗ *Triangle-area c b k* =
Triangle-area c a k ∗ *Triangle-area a b k* ∗ *Triangle-area b c k*
using *Triangle-area-per2*
by *metis*
also have *Triangle-area c a k* ∗ *Triangle-area a b k* ∗ *Triangle-area b c k* =
dist b f ∗ *n1* ∗ *dist c d* ∗ *n3* ∗ *dist a e* ∗ *n2*
using *Tri1'2 Tri2'2 Tri3'1*
by *simp*
also have *dist b f* ∗ *n1* ∗ *dist c d* ∗ *n3* ∗ *dist a e* ∗ *n2* =
dist f b ∗ *n1* ∗ *dist d c* ∗ *n3* ∗ *dist e a* ∗ *n2*
using *dist-commute*
by *metis*
finally have *Goal*: *dist a f* ∗ *n1* ∗ *dist b d* ∗ *n3* ∗ *dist c e* ∗ *n2* =
dist f b ∗ *n1* ∗ *dist d c* ∗ *n3* ∗ *dist e a* ∗ *n2*
by *simp*
then consider (*n2*) *n2* = *0* | (*n1*) *n1* = *0* | (*n3*) *n3* = *0* |
(*dist*) *dist a f* ∗ (*dist b d* ∗ *dist c e*) = *dist f b* ∗ (*dist d c* ∗ *dist e a*)
by *auto*
then show *?thesis*
proof *cases*
case *n2*
then show *?thesis*
proof −
assume *n0* : *n2* = *0*
have *H1* : *Triangle-area c b k* = *0*
using *Tri2'1 n0*
by *auto*
have *H1'* : *collinear* {*c*,*b*,*k*}
using *H1 Triangle-area-0*
by *auto*
have *H1* : *Triangle-area a b k* = *0*
using *Tri2'2 n0*

```
        by auto
      have H2' : collinear {a,b,k}
        using H1 Triangle-area-0
        by auto
      have H : b = k
        using H1' H2' collinear-3-trans Triangle collinear-3-trans
        by (metis Triangle-area-0 Triangle-area-per1)
      have H1 : b = f
        using H Triangle collinear-3-trans MidCol TriCol
        by (metis doubleton-eq-iff)
      have H2 : b = d
        using H H1 Triangle collinear-3-trans MidCol TriCol
        by blast
      show ?thesis
        using H H1 H2
        by simp
    qed
  next
    case n1
    then show ?thesis
    proof -
      assume n0 : n1 = 0
      have H1 : Triangle-area a c k = 0
        using Tri1'1 n0
        by auto
      have H1' : collinear {a,c,k}
        using H1 Triangle-area-0
        by auto
      have H1 : Triangle-area b c k = 0
        using Tri1'2 n0
        by auto
      have H2' : collinear {b,c,k}
        using H1 Triangle-area-0
        by auto
      have H : c = k
        using H1' H2' collinear-3-trans Triangle collinear-3-trans
        by (smt (verit) insert-commute)
      have H1 : c = d
        using H H1' H2' Triangle
          by (metis Tri3'1 Tri3'2 Triangle-area-0 Triangle-area-per2 dist-eq-0-iff
mult-eq-0-iff)
      have H2 : c = e
        using H H1 H1' H2' Triangle
          by (metis Tri2'1 Tri2'2 Triangle-area-0 Triangle-area-per2 dist-eq-0-iff
mult-eq-0-iff)
      show ?thesis
        using H H1 H2
        by simp
    qed
```

```
  next
    case n3
    then show ?thesis
    proof -
      assume n0 : n3 = 0
      have H1 : Triangle-area c a k = 0
        using Tri3'1 n0
        by auto
      have H1' : collinear {c,a,k}
        using H1 Triangle-area-0
        by auto
      have H1 : Triangle-area b a k = 0
        using Tri3'2 n0
        by auto
      have H2' : collinear {b,a,k}
        using H1 Triangle-area-0
        by auto
      have H : a = k
        using H1' H2' collinear-3-trans Triangle
        by (metis (full-types) insert-commute)
      have H1 : a = f
        using H H1' H2' Triangle
          by (metis Tri1'1 Tri1'2 Triangle-area-0 Triangle-area-per1 dist-eq-0-iff
mult-eq-0-iff)
      have H2 : a = e
        using H H1 H1' H2' collinear-3-trans Triangle
        by (metis MidCol TriCol collinear-3-eq-affine-dependent)
      show ?thesis
        using H H1 H2
        by simp
        qed
  next
    case dist
    then show ?thesis
      by auto
    qed
qed


end
```